Segmentation and Targeting Analytics

Naming the Humps: Discovering, Verifying, and Targeting the Customer Groups Inside One Company

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026

Chapter Information

ABSTRACT

This chapter opens Part II with customer segmentation and the guide's first sustained application of unsupervised learning: the move from describing a customer base to dividing it into groups a marketing organization can act on. It places segmentation inside the segmentation-targeting-positioning framework, surveys segmentation bases and the data each requires, and separates a priori schemes declared before analysis from post hoc schemes estimated from data. Its two working methods are RFM scoring on an exhaustive named grid and k-means clustering, developed from distance and standardization through the choice of k and the profiling, stability, and holdout checks that separate a defensible segmentation from a plausible-looking accident. Useful-segment criteria and a worked attractiveness matrix turn verified segments into targeting decisions. The labs build StyleCraft's customer-feature table, score the grid, cluster at full scale, and test the recovery of four designed segments.

KEYWORDS

market segmentation; targeting; stp framework; rfm analysis; customer features; k-means clustering; standardization; silhouette score; cluster validation; unsupervised learning

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Segmentation and targeting analytics. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 6, Version 1.0) [Open educational resource]. CC BY 4.0.

identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY
Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

Chapter Learning Objectives

By the end of this chapter, students should be able to:

- Explain the strategic role of segmentation in marketing decision-making, situate it within the segmentation-targeting-positioning framework, and distinguish segmentation as a strategic act from segmentation as a computation.
- Compare the major segmentation bases — geographic, demographic, psychographic, and behavioral — and state the data each requires, including which bases StyleCraft's tables can and cannot support.
- Distinguish a priori from post hoc segmentation, and state precisely what a post hoc method estimates from data and what the analyst must still supply.
- Conduct and interpret an RFM analysis: define recency, frequency, and monetary value at the customer grain, score them on a declared rule, resolve tied values defensibly, and read an exhaustive named grid.
- Build a customer-grain feature table from transaction data, declaring the population, the analysis date, the window, and the derivation rule for every feature, and reconcile the table to certified totals.
- Explain why distance-based methods require standardization, demonstrate with hand arithmetic how unscaled units let one feature dominate a distance, and explain why standardization does not address redundancy among features.
- Apply k-means clustering for exploratory segmentation: sketch the algorithm, explain initialization sensitivity, and choose k using the elbow method, silhouette scores, and business judgment together.
- Profile clusters at declared grains, measure stability with a permutation-invariant index across seeds and subsamples, profile withheld attributes, and state precisely what each check does and does not establish.
- Evaluate whether segments are useful against the five criteria — measurable, substantial, accessible, differentiable, actionable — and build a segment-attractiveness matrix that marks evidence separately from judgment.
- Identify where segmentation crosses ethical lines: sensitive attributes and their proxies, differential pricing and offers, and treatment decisions that punish tenure artifacts rather than customer potential.

Chapter 5 closed Part I with a description that kept insisting on one thing: StyleCraft's customers are not one population. The decomposed average-order-value gap, the two-grain profile, and the cohort tables all pointed to distinct customer worlds living inside a single company —

description found the humps. This chapter, which opens Part II, names them. It develops segmentation as the discipline of dividing a customer base into groups that can be measured, verified, and — the part that makes it strategy rather than arithmetic — treated differently. The route runs from strategic foundations (why segment, on what bases, decided in advance or estimated from data) through the two working methods of the analyst (RFM scoring and k-means clustering), and ends where segmentation earns its keep: deciding which segments to target and with what. Because the groups in this chapter are estimated by an algorithm rather than declared by managers, the chapter also installs a new verification obligation — the audit of a clustering solution — that the AI section, the labs, and the exercises all rehearse.

> **CONCEPT**
>
> **What This Chapter Is Really About**
>
> Chapter 5 argued that summarizing is the second analysis. This chapter makes the argument that follows from it: grouping is the third, and it is the first analysis in this guide whose output is produced by an algorithm rather than by arithmetic the analyst fully controls. A frequency table cannot be wrong about which rows exist; a clustering can — it will return as many groups as it was asked for, drawn with perfect confidence, whether or not those groups exist in any sense a marketer should care about. The discipline of Part I was reconciliation: check every summary against known totals. The discipline this chapter adds is validation: an algorithmic grouping is a hypothesis about structure, and it earns trust only by surviving tests it was not built to pass — refits from new starting points and new samples, profiling on variables it never saw, and the blunt managerial question of whether the groups can be treated differently at all. A segmentation that survives those tests changes the marketing calendar. One that is merely admired in a deck changes nothing, however elegant its geometry.

6.1 Marketing Decision Context: Four Slots on the Fall Calendar

Chapter 5 ended with the expansion review in possession of a defensible account of its one seductive number: the suburban average order value decomposed into basket size and occasionwear mix, a two-grain profile showing two customer worlds, and cohort tables comparing those worlds at the same age. The review's capital-allocation question moves to later chapters. But the meeting produced a second decision, quieter and nearer, and it lands on the CRM manager — the same CRM manager whose worry about declining repeat purchase opened this guide in Chapter 1.

The decision is the fall lifecycle-marketing program. StyleCraft's email and app platform can operationally sustain four new differentiated customer journeys this fall — four sequences of messages, offers, and triggers, each with its own creative, its own cadence, and its own budget line. Not seven, not nine: four is what the CRM team can build, test, and maintain alongside the weekly drops. Two clarifications of that constraint matter later and are stated now. The always-on welcome program that greets a new signup and works toward a first purchase is existing business as usual; it continues, and it does not consume one of the four new slots. And a slot

buys one journey, not one message: a journey may carry an internal branch keyed to a declared rule, at the cost of the extra creative that branch requires, but a fifth creative-and-measurement frame is what the platform cannot carry. Every one of StyleCraft's purchasing customers must be assigned to exactly one of the four journeys by a rule that lives in the database rather than in someone's judgment, every signed-up customer who has not purchased must be accounted for somewhere, and the program locks in twenty-one days, when creative production begins. The stake is the retention budget: after Chapter 5's cohort evidence, no one in the building still believes one message serves every customer, and the money will follow whatever map of the customer base the CRM manager brings to the sign-off meeting.

The problem is that three maps have already arrived, and they disagree. The brand agency's pitch deck contains seven personas — vivid, named, photographed characters ("The Weekend Curator," "The Campus Tastemaker") built from two focus groups and considerable craft, none of them connected to a single row of StyleCraft's data. The head of retail wants a simpler map: two segments, urban and suburban, straight from Chapter 5's profile, with a "suburban VIP" push to court the high-basket occasion shopper. And a data-science-inclined intern, working overnight with an AI assistant, has produced the third map: a nine-cluster solution on customer data, each cluster already named by the assistant with unsettling fluency, presented in a notebook nobody has audited. Nine clusters, seven personas, two metros — and four slots on the calendar.

The CRM manager's task is therefore not to pick the most persuasive deck. It is to answer, with evidence, the question all three maps claim to answer: which customer groups does StyleCraft's data actually contain? That question decomposes into the questions this chapter teaches. First, the strategy question: what should the groups be based on — who customers are, where they live, or what they do — given that the fall program can only act on differences it can reach? Second, the discovery question: should the groups be declared in advance, the way the head of retail's two-metro scheme declares them, or estimated from the data, the way the intern's clustering claims to have discovered them — and if estimated, by what method and with what checks? Third, the verification question, which the intern's notebook makes urgent: an algorithm asked for nine clusters produced nine clusters, as it would have produced six or thirteen — what would it take to establish that any clustering describes real structure rather than an arbitrary partition wearing confident names? And fourth, the targeting question: even among real segments, four slots force choices — which groups justify a differentiated journey, and by what criteria?

The deadline is fixed, the audience includes the head of retail and the agency, and the deliverable is concrete: at most four segments, each defined by a written rule on StyleCraft's own columns, each profiled honestly per Chapter 5's disciplines, each verified to be more than an artifact, and each attached to a recommended treatment. The chapter builds toward that deliverable in order. Sections 6.2 through 6.4 establish the strategic frame: what segmentation is for, what it can be based on, and the a priori versus post hoc distinction that separates the head of retail's map from the intern's. Sections 6.5 and 6.6 build the analyst's first method and its raw material: RFM analysis and the customer-grain feature table it requires. Sections 6.7 through

6.10 develop clustering — distance, scaling, k-means, the choice of k, and the profiling and stability checks that make a clustering defensible. Section 6.11 converts segments into targeting decisions. Section 6.12 examines how AI assistants help and fail at exactly this work, and the labs in Section 6.13 run the full sequence on StyleCraft's certified data — where, by design, the right answer is known: the dataset was built to contain four segments, and the analysis must find them.

6.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The agency's seven personas, the head of retail's two metros, and the intern's nine clusters are all segmentations of the same customer base. For each map, state what it is based on and what data, if any, could prove it wrong.
- The platform supports four journeys. Suppose the data genuinely contained six well-separated customer groups. What are the CRM manager's options, and what does each cost?
- The intern's assistant named a cluster “Affluent Suburban Loyalists.” StyleCraft's customers table contains no income column, and Chapter 5 showed suburban shoppers rarely enroll in loyalty. What has the assistant done, and which section of this chapter is designed to catch it?
- The head of retail's two-metro scheme requires no algorithm at all: every customer's metro is already a column. What does that scheme gain by being so simple, and what did Chapter 5's profile suggest it loses?

6.2 Segmentation as Strategy and Segmentation as Computation

Everything in this chapter serves one strategic idea, and it is older than any algorithm in it. A market is not a homogeneous mass of interchangeable buyers; it is a collection of groups whose wants, behaviors, and responses to marketing differ, and a firm that recognizes those groups can serve chosen ones deliberately rather than serving an imaginary average customer badly (Smith, 1956). Chapter 5 supplied this guide's empirical version of the same idea: heterogeneity is a finding, and StyleCraft's two-humped order values were the question “is this one group?” in statistical clothes. Segmentation is the affirmative program built on the answer no.

DEFINITION

Market Segmentation

Market segmentation is the division of a market into distinct groups of buyers — segments — that are internally similar and externally different on characteristics relevant to how the firm will serve them, undertaken so that marketing effort can be differentiated across groups rather than uniform over the whole market.

Source: Adapted from Smith (1956) and Kotler and Keller (2016).

Segmentation is the first move of a three-part framework that organizes marketing strategy: segmentation, targeting, and positioning. The firm first divides the market into segments; it then targets — selects which segments to serve, with what priority and resources; and it finally positions — designs an offering and message intended to occupy a distinct place in the targeted segment's mind (Kotler & Keller, 2016). This guide is an analytics guide, so it engages the framework from the analyst's seat: segmentation is where the analytics is densest and is this chapter's technical core; targeting is a decision this chapter equips with evidence in Section 6.11; and positioning is a creative and strategic act the analyst informs — with segment profiles — but does not compute. When the CRM manager assigns thousands of customers to four journeys, all three moves happen: the base is segmented, four groups are targeted with differentiated treatment, and each journey's creative positions StyleCraft for its group.

DEFINITION

Segmentation, Targeting, and Positioning (STP)

STP is the sequential strategy framework in which a firm divides a market into segments, selects target segments to serve, and positions its offering distinctively for each target. In analytics practice, segmentation is computed, targeting is decided on computed evidence, and positioning is informed by segment profiles.

Source: Adapted from Kotler and Keller (2016).

The working distinction the rest of the chapter depends on is between segmentation as a strategic act and segmentation as a computation, because the two are routinely confused and the confusion is expensive in both directions. As a strategic act, segmentation is a commitment: it declares which customer differences the organization will organize itself around — whose journeys get built, whose offers differ, whose names appear on the marketing calendar. As a computation, segmentation is a procedure: a scoring rule or a clustering algorithm partitions rows of a customer table. The computation is the easy part, and it has never been easier — Section 6.12 will show that a fluent cluster solution is now one prompt away. The strategy is the binding part: a partition of rows becomes a segmentation only when each group is attached to a difference in treatment the organization can actually execute. The intern's nine clusters may be excellent geometry; against a four-journey calendar, at least five of them are, strategically, not segments at all. The reverse failure is just as real: a segmentation declared purely from the executive floor — two metros, because the org chart has an urban lead and a suburban lead — commits the organization around a difference the data may show is not the difference that matters.

CONCEPT

Segmentation Is a Treatment Decision

A partition of customers is arithmetic. A segmentation is a partition plus a commitment: for each group, a way the organization will treat it differently — different journeys, offers, budgets, service levels, or products. Before admiring any segmentation, ask the treatment question: what will we do differently to each of these groups that we could not do to the whole base? Groups without an answer are analysis; groups with an answer are strategy. The question also runs backward, and Section 6.11 will formalize it: the number of treatments the organization can execute is a design constraint on the segmentation itself — a constraint-as-design-input in exactly the sense of Section 2.4.

Source: Course concept developed for this guide, informed by Kotler and Keller (2016) and Yankelovich and Meer (2006).

The distinction settles the opening case's first argument before any data is touched. The agency's personas are positioning aids — vivid characters that help creative teams write to someone rather than to no one — and nothing in this chapter retires them from that job. But they are not, as delivered, a segmentation: no rule assigns a StyleCraft customer to "The Weekend Curator," so no journey, budget, or measurement can attach to it. The head of retail's two-metro scheme is a genuine segmentation — every customer is assigned by rule, and treatments can differ by metro — but whether it is a good one depends on whether metro is the difference that best predicts differential response, which is an empirical question the scheme itself cannot settle. And the intern's clusters are, so far, a computation awaiting the strategic test. The chapter's job is to give the CRM manager the machinery to run all three maps through the same discipline.

6.3 Segmentation Bases and the Data They Require

If segmentation divides customers by their differences, the first design decision is which differences. The candidate characteristics are called segmentation bases, and marketing practice organizes them into four families (Kotler & Keller, 2016; Wedel & Kamakura, 2000). Geographic bases divide by location: region, metro, urban versus suburban, climate. Demographic bases divide by who customers are in census terms: age, generation, gender, income, household composition. Psychographic bases divide by how customers think and live: values, lifestyles, attitudes, interests. Behavioral bases divide by what customers do with the category and the brand: purchase recency and frequency, spend, product mix, channel use, loyalty status, response to promotions.

DEFINITION

Segmentation Base

A segmentation base is the characteristic or set of characteristics on which customers are divided into segments — geographic, demographic, psychographic, or behavioral. The choice of base determines what data the segmentation requires, what differences it can express, and what treatments it can support.

Source: Adapted from Wedel and Kamakura (2000) and Kotler and Keller (2016).

For the analyst, the base decision is inseparable from a data decision, because each family makes different demands on the tables of Chapter 3. Table 6.1 walks the four families through StyleCraft's actual columns, and the exercise generalizes: before endorsing any base, locate the columns that would operationalize it, at the grain the segmentation needs, with the validity Chapter 3 taught you to interrogate.

Table 6.1

Segmentation bases and their data requirements at StyleCraft

Base family	Example variables	StyleCraft columns	Data notes for the analyst
Geographic	Region, metro, store proximity	home_region, home_metro, primary_store_id (customers)	Available for every customer; coarse; stable; a proxy for much it does not name (Section 6.16)
Demographic	Age, generation, gender	age_band, generation, gender (customers)	Self-reported at signup; age_band optional, so partly missing; no income column exists — "affluent" is inferred, not measured
Psychographic	Values, lifestyle, style identity	none	Not in the schema; requires surveys or licensed data; the agency's personas live here, unmeasured
Behavioral	Recency, frequency, spend, discount use, channel, loyalty	derived from transactions; loyalty_tier, app_user, email_opt_in (customers)	Richest family; requires the grain change of Section 6.6 — behavior lives in transactions at line grain and must be rolled up to customers

Two lessons from the table organize what follows. The first is that behavioral bases dominate database marketing practice, and for a principled reason: they are computed from what customers verifiably did rather than from what customers say or from what their zip code implies, and differential response to marketing — the thing targeting cares about — is itself a behavior. The empirical segmentation literature reaches a similar verdict: bases closest to purchase behavior

tend to be the most actionable for customer-level marketing, while general characteristics like demographics travel better for media buying and product design than for predicting response (Wedel & Kamakura, 2000; Haley, 1968). The fall program is customer-level marketing on StyleCraft's own base, which is why this chapter's two working methods — RFM in Section 6.5 and clustering on behavioral features in Sections 6.7 through 6.10 — are both behavioral.

The second lesson is that the base decision is also an honesty decision. StyleCraft's schema can support geographic, demographic, and behavioral segmentation at declared levels of quality; it cannot support psychographic segmentation at all, and it cannot support “affluent” as anything but an inference from geography. A segmentation deliverable that names segments in psychographic or income language — as the intern's assistant did — is asserting bases the data does not contain, a validity failure in exactly Section 3.6's sense. The discipline is to name segments in the vocabulary of the base that produced them, a rule Section 6.10 will make operational.

Well-chosen bases are also frequently combined. Nothing restricts a segmentation to one family: the scheme the labs build clusters on behavioral features and then profiles the resulting groups geographically and demographically — using the other families as description and as a check rather than as inputs. That division of labor, behavioral bases in, other bases held out for profiling, is deliberate, and Section 6.10 will show both what it can establish and what it cannot.

6.4 Two Roads to Segments: A Priori and Post Hoc

With a base chosen, there remain two roads to actual segments, and the segmentation literature has named them for half a century (Wind, 1978). In a priori segmentation, the analyst declares the segments in advance: the number of groups and the rule that assigns customers to them are decided before the data are consulted, and the data's job is to size and profile groups whose definitions were never in question. Urban versus suburban metro is a priori; so is loyalty tier, so is “customers acquired through TikTok,” and so is every RFM grid cell in Section 6.5, because the scoring rule and the named grid are fixed before any customer is scored. In post hoc segmentation, the segments are estimated from the data: the analyst chooses the features, the method, and the candidate numbers of groups, and the data determine which customers belong together and where the boundaries fall. Clustering is the canonical post hoc method, and Sections 6.7 through 6.10 are its treatment.

DEFINITION

A Priori and Post Hoc Segmentation

A segmentation is a priori when the number of segments and the assignment rule are declared before analysis, so that data serve to size and profile predefined groups. A segmentation is post hoc when group membership and boundaries are estimated from the data after the analyst specifies the features, the method, and the candidate values of k. The post hoc road removes the analyst's prior about who belongs with whom; it does not remove the analyst's choices about what to measure, how to scale it, and how many groups to ask for. In k-means, k is selected using evidence and judgment (Section 6.9); it is not produced by the algorithm.

Source: Adapted from Wind (1978); k-means characterization per Pedregosa et al. (2011).

Neither road is superior; they answer different questions and fail in different ways, and Table 6.2 sets the comparison the CRM manager needs. The a priori road is transparent, cheap, instantly explainable, and guaranteed actionable if the defining variable is one the organization can act on — a journey keyed to loyalty_tier can be built this afternoon. Its risk is exactly its virtue: the segments encode a prior belief, and if the belief is wrong — if the difference that matters is not the one declared — the data will obligingly size and profile groups that miss the structure. Chapter 5's profile hinted at precisely this: the urban-suburban split is real, but the spread within each metro was wide, which is the descriptive signature of structure the two-metro scheme cannot express. The post hoc road can find structure nobody declared — that is its entire value — but it trades away transparency: it requires the feature table of Section 6.6, the scaling discipline of Section 6.7, an algorithm with sensitivities of its own, and, above all, the validation of Section 6.10, because a method that returns groups will return them whether or not they exist.

Table 6.2

A priori versus post hoc segmentation

Dimension	A priori	Post hoc
Segments defined	Before analysis, by declared rule	By the analysis, from the data, given the analyst's inputs
Number of segments	Chosen by the analyst or given by the variable	Supplied by the analyst as k and defended on evidence (Section 6.9)
Typical methods	Cross-tabs, profiles, RFM scoring	Cluster analysis (k-means in this guide)
Chief strength	Transparent, explainable, immediately actionable	Can find structure no one declared
Chief risk	Encodes a possibly wrong prior belief	Returns a partition whether or not structure exists; requires validation (Section 6.10)
Verification burden	Sizing and profiling done honestly (Chapter 5)	Everything in Chapter 5, plus stability and holdout checks
Opening-case example	Head of retail's two metros	Intern's nine clusters

The two roads also combine, and the combination is this chapter's recommended practice rather than a compromise. A common professional sequence runs post hoc discovery first, a priori operation second: clustering estimates candidate structure; the analyst translates the resulting groups into explicit written rules on named columns and retains the fitted model that produced them (a step Section 6.10 formalizes); and the organization then operates the segmentation a priori — every new customer assigned by a stated procedure, every journey keyed to that assignment, every assignment auditable. The discovery is statistical; the deployment is governance. This sequence is what the labs perform, and it is what the CRM manager's deliverable — "each segment defined by a written rule on StyleCraft's own columns" — actually requires.

One more distinction keeps the roads honest, and it echoes Chapter 5's insistence that a designed answer key is not a reason to skip the work. StyleCraft's dataset was built, per its specification, to contain four latent customer segments. That fact makes the labs gradable: the post hoc analysis has a known truth to recover. It does not make the analysis circular, because the analyst's procedure — features, scaling, method, k, validation — must earn the recovery; a bad procedure will happily fail to find structure that is genuinely there, and the exercises include exactly such failures to diagnose. Real customer bases do not ship with answer keys, which is why the validation discipline of Section 6.10, rehearsed here where the truth is known, is the portable skill.

6.5 RFM Analysis: The Workhorse of Behavioral Segmentation

The analyst's first working method predates every algorithm in this chapter and remains one of the most operationally used segmentation techniques in direct and database marketing: RFM analysis, the scoring of customers on recency, frequency, and monetary value (Hughes, 1994). The three quantities themselves are not new to this guide — recency, frequency, and monetary value are metrics, defined per the four-element discipline of Section 3.5.1 and cataloged in Appendix B; Chapter 5 used all three descriptively. What this chapter owns is the method built on them: scoring each quantity into ranked bands, combining the scores into a grid, and naming the grid's regions as operable segments.

The three definitions must be fixed before any scoring, because each hides the measurement decisions Chapter 3 taught you to write down. Recency is the elapsed time since the customer's most recent purchase, computed against a declared analysis date — this guide's labs use June 30, 2026, the same analysis date as Lab 3.1 — and stated in days. Frequency is the count of the customer's distinct orders in a declared window; this chapter uses the full 24-month file and counts orders, not lines, which is a grain commitment (an order of five lines is one act of purchasing, not five). Monetary value is the customer's total revenue in the same window; a common variant uses average order value instead, and the choice matters — total revenue rewards frequent modest baskets, average order value rewards rare large ones, and on StyleCraft that is precisely the urban-suburban contrast — so the choice is declared, not defaulted. Every one of these is a numerator, denominator or window, and filter decision, and the standing rule of Section 3.5.1 applies: write the definition before computing the number.

DEFINITION

RFM Analysis

RFM analysis is a behavioral segmentation method that scores each customer on recency of last purchase, frequency of purchases, and monetary value in a declared window — classically by quintile — and groups customers by their combination of scores. The scoring tradition rests on the practitioner premise that recent, frequent, high-value buyers are the most responsive to direct marketing; treated analytically, each grid region's treatment logic is a hypothesis about response, to be tested by campaigns rather than assumed from the score.

Source: Adapted from Hughes (1994). For the formal link between RFM measures and customer value, see Fader et al. (2005).

The scoring step converts each raw quantity into a rank-based score, classically by quintiles: customers are sorted on the quantity and divided into five equal-count bins, scored 5 for the best bin through 1 for the worst — where “best” means most recent, most frequent, and highest value, so recency is scored in reverse of its raw days. Quintile scoring has two properties that explain its persistence. It is distribution-free: Chapter 5 showed that spend and frequency are strongly right-skewed and concentration-heavy, and ranking neutralizes the skew that would make raw-value cutoffs arbitrary. And it is self-calibrating within its population: the bins move with the base, so

“a 5 on monetary” means the top fifth of this base in this window rather than a fixed dollar threshold that inflation and growth quietly retire.

That second property is regularly overstated, and the overstatement is expensive, so state the limit plainly. Quintile scores preserve relative rank within a stated population and window. They do not preserve absolute performance: the top monetary quintile exists even when spending has fallen for every customer in the base, and a customer whose score rises from 3 to 4 may have spent less than last year while others spent less still. Scores computed on different windows or different populations are therefore not comparable unless the changing cut points are reported alongside them — which makes the cut points part of the deliverable, exactly as Section 3.5.1 would predict.

DEFINITION

Quintile Scoring

Quintile scoring converts a numeric quantity into a five-level ordinal score by ranking all customers on the quantity and assigning 5 to the top fifth through 1 to the bottom fifth, with the direction chosen so that 5 is always the marketing-favorable end. Scores are relative to the scored population and window: they preserve rank, not level, and are not comparable across periods or populations unless the cut points are reported with them. The handling of tied values is declared as part of the scoring rule.

Source: Adapted from Hughes (1994).

Ties deserve the paragraph most treatments skip, because on real marketing data they are not an edge case. Frequency at StyleCraft is lumpy — a large share of customers have exactly one order, a designed fact of the New/At-Risk population — and five equal-count bins cannot be cut through a value that thousands of customers share. Software resolves this either by complaining (pandas' qcut raises an error on duplicate bin edges) or by an arbitrary-but-deterministic rule. The error is the more useful outcome, because it is the data telling you the quantity has too few distinct values to support five honest bins.

Four repairs are available, and choosing among them is a scoring-rule decision that belongs in the written definitions. Use fewer frequency bands, with cut points at values customers actually take. Keep tied values together and accept unequal group sizes, by ranking densely or cutting on declared thresholds. Break ties on a second, genuinely behavioral quantity, so that the distinction means something. Or rank with a row-order tiebreaker — pandas' rank(method="first") — and label it, in the deliverable, as an arbitrary classroom convention. The last option is the one AI assistants reach for by default, and it is the weakest: it assigns different scores to customers with identical frequency on the basis of where they happen to sit in the file, so it is reproducible only while the sort order is, and the distinction it manufactures has no behavioral content whatsoever.

This guide takes the first repair. StyleCraft's operating frequency rule uses three declared bands rather than five quintiles: exactly one order in the window scores F1, two or three orders

score F3, and four or more orders score F5. The bands are cut at values the distribution actually contains, every customer with the same order count receives the same score, and the mapping onto the five-point grid axis is stated rather than implied. Recency and monetary value keep their quintiles, because both are near-continuous and support five bins without ties. Code 6.3 in Lab 6.1 implements the diagnosis and the repair together.

With three scores per customer, the classical presentation collapses to the two most operational dimensions — recency and frequency — and arranges them as a grid whose regions carry conventional names. Table 6.3 is this guide's operational grid, and its most important property is structural rather than nominal: it is a complete lookup over all twenty-five recency-by-frequency cells, so every scored customer falls in exactly one region and no customer falls in two. That property is not decoration. The opening case requires every purchasing customer to receive exactly one journey, and a published grid with gaps or overlaps lets two analysts apply the same rules and reach different assignments — which is a governance failure before it is an analytic one. The names are operational shorthand, not discoveries: “Champions” is a label for the recent-and-frequent corner declared by the scoring rule, which is what makes RFM an a priori method in Section 6.4's sense — the grid exists before any customer is scored, and the data's job is to fill it.

Table 6.3

The named RFM grid: a complete lookup over recency and frequency scores

Recency score	**F1**	**F2**	**F3**	**F4**	**F5**
R5 (most recent)	New Customers	Potential Loyalists	Loyal Customers	Champions	Champions
R4	Potential Loyalists	Potential Loyalists	Loyal Customers	Champions	Champions
R3	Needs Attention	Needs Attention	Loyal Customers	Loyal Customers	Loyal Customers
R2	Hibernating	Needs Attention	At Risk	At Risk	At Risk
R1 (least recent)	Hibernating	Hibernating	At Risk	At Risk	At Risk

Two notes complete the grid's operating definition. First, under StyleCraft's three-band frequency rule the scored population occupies only the F1, F3, and F5 columns; F2 and F4 stay empty. They are retained in the published grid because the grid is the general lookup — a population whose frequency distribution supports five bands maps onto it without amendment — and because a lookup with missing columns is the defect this table exists to eliminate. Second, region membership is determined by the pair of scores and nothing else: monetary value remains attached to every customer and returns as a sizing and prioritization variable in Section 6.11, but

it does not move a customer between regions. Table 6.4 gives each region its reading and its conventional treatment logic — and treats that logic, per the Definition box above, as a hypothesis about response rather than an established response rule.

Table 6.4

The seven grid regions: reading and conventional treatment logic

Region	**Cells**	**Reading**	**Conventional treatment logic (a hypothesis to test)**
Champions	R4–R5 × F4–F5	Bought recently, buy often	Reward and early access; protect rather than discount
Loyal Customers	R5×F3, R4×F3, R3×F3–F5	Steady repeaters, or recent moderate repeaters	Nurture; cross-category offers
Potential Loyalists	R5×F2, R4×F1–F2	Recent, not yet habitual	Onboard toward the second and third purchase
New Customers	R5×F1	Just arrived, one order	Welcome journey; measure, do not judge
Needs Attention	R3×F1–F2, R2×F2	Middling on both, drifting	Low-cost re-engagement test before recency falls further
At Risk	R1–R2 × F3–F5	Were frequent, have gone quiet	Win-back triggers, tested incentives
Hibernating	R2×F1, R1×F1–F2	Long gone, rarely bought	Low-cost reactivation or suppression

Read against the opening case, the grid is already a serious candidate map: it is behavioral, rule-defined, exhaustive, reproducible in the database, and its regions come with treatment logic attached — several of its rows are recognizably journeys. Its limits are equally visible and motivate the rest of the chapter. The grid sees exactly three behaviors, and its regions are drawn on two of them; StyleCraft's designed structure involves channel, discount dependence, and loyalty enrollment, none of which R, F, and M can express — the Discount Trend Follower and the Urban Loyal Core can occupy the same grid cell while requiring opposite treatments, and discounting is precisely the lever on which the “reward, do not discount” logic would misfire against a discount-dependent customer. The grid's boundaries are declared, not estimated, so it cannot surprise you. Widening the behavioral lens requires more features and a method that can weigh all of them at once — which is the business of Sections 6.6 through 6.10.

6.6 Building the Customer-Grain Feature Table

Both of this chapter's methods consume the same raw material: a table with one row per customer and one column per behavioral quantity. StyleCraft does not ship behavior at that grain — behavior lives in transactions, at order-line grain — so the table must be built, and the build is a grain change in exactly Section 3.3's sense, executed with the groupby mechanics established in Chapter 4's lab. This guide calls the result the customer-grain feature table, and the companion repository ships a completed version, customer_features, as the answer key the labs reconcile against; the columns are features in Section 3.5's role vocabulary because their destiny is to be model inputs.

> **CONCEPT**
>
> **The Customer-Grain Feature Table**
>
> The customer-grain feature table is the analytic asset that powers customer-level modeling: one row per customer in a declared population, one column per feature, every feature computed from certified data by a written derivation rule at a declared analysis date and window. Building it is a grain change (Section 3.3), so it carries grain-change obligations: declare the source grain and target grain, declare which rows the population admits and which it excludes, state each feature's aggregation rule, and reconcile the result — row counts, sums, and distinct counts — against the certified source before anything downstream consumes it. The table is an asset, not a byproduct: later chapters in Part II model from this same table, so an error built into it now propagates through the rest of the book.
>
> Source: Course concept developed for this guide, informed by the grain discipline of Section 3.3.

Before any feature is derived, the population must be settled, because the opening case contains a constraint the naive build cannot satisfy. The CRM manager must account for every signed-up customer, and behavioral features are undefined for a customer who has never purchased: recency has no last order to count back from, average order value divides by zero, and every purchase-derived share has an empty numerator over an empty denominator. Two operating designs resolve this honestly. A full-base segmentation includes every signed-up customer and declares zero-purchase behavior explicitly — frequency 0, monetary 0, average order value missing rather than zero, a separate recency rule keyed to signup, and purchase-derived shares recorded as not applicable. A purchasing-customer segmentation restricts the clustered population to customers with at least one purchase in the window and routes signups with no purchase in observed history to a separate activation program that sits outside the clustering.

This guide takes the second design, and states it as a rule rather than as a filter buried in a query. The customer_features table contains customers with at least one transaction in the declared window; its row count is therefore the certified distinct-purchaser count, not the customers table's row count. Customers outside it are not discarded, and naming them precisely matters more than it looks. The complement of the clustered population is not “never purchased.” It is “no purchase in the analysis window,” and it holds two quite different groups: customers

with no purchase anywhere in the observed history, and customers who bought before the window opened and have since gone quiet. With twenty-four months of history on file, even the first group is properly described as having no purchase in observed history rather than none ever — the file cannot see further back than it goes. Code 6.5 separates the three populations and reports all three counts.

The destinations differ accordingly, and they fit the opening case's constraint rather than straining it. Customers with no purchase in observed history belong to the always-on welcome and activation program StyleCraft already runs — business as usual, continuing unchanged, and consuming none of the four new fall slots. Customers who purchased only before the window are lapsed rather than new, and they are candidates for win-back treatment inside the fall program; if a later release shortens the window, they re-enter the clustered population and the population rule has to be restated with it. Stating the population this way keeps the opening case's promise — every customer has exactly one destination — without pretending that a customer with no purchases can be described by purchase behavior, and without quietly inventing a fifth journey.

Table 6.5 specifies the features the labs build, their derivation rules, and — a distinction the rest of the chapter depends on — their role in the segmentation. Clustering inputs are the behavioral features the algorithm will see. Profiling holdouts are attributes deliberately withheld from the algorithm and reserved for Section 6.10's checks. And one column, exposure_months, is neither: it is a denominator, computed so that other features can be expressed as rates, and it never enters the distance calculation itself.

Table 6.5

The customer_features build: derivation rules and roles

Feature	**Derivation rule (analysis date June 30, 2026; 24-month window)**	**Source**	**Role**
recency_days	Days from the customer’s most recent order_date to the analysis date	transactions	Clustering input (state)
frequency	Count of distinct order_id in the window	transactions	Clustering input (level)
monetary	Sum of line_revenue in the window	transactions	Clustering input (level)
exposure_months	Days from the later of signup_date and the window start to the analysis date, divided by 30.44, then floored at 1.0 month	customers	Denominator only — never clustered
orders_per_month	frequency ÷ exposure_months	derived	Clustering input (intensity)

Feature	Derivation rule (analysis date June 30, 2026; 24-month window)	Source	Role
revenue_per_month	monetary ÷ exposure_months	derived	Clustering input (intensity)
aov	monetary ÷ frequency	derived	Clustering input (shape)
discount_share	Revenue on lines with discount_pct > 0 ÷ monetary	transactions	Clustering input (shape)
store_share	Distinct orders with channel = Store ÷ frequency	transactions	Clustering input (shape)
tenure_days	Days from signup_date to the analysis date	customers	Profiling holdout
loyalty_tier	As stored (ordinal per Table 3.5)	customers	Profiling holdout
app_user	As stored	customers	Profiling holdout
email_opt_in	As stored	customers	Profiling holdout
home_metro	As stored	customers	Profiling holdout
acquisition_channel	As stored	customers	Profiling holdout
age_band	As stored; partly missing, per Table 6.1	customers	Profiling holdout

Three build decisions deserve their reasons stated, because each is a judgment call an AI assistant will make silently. The first is the population rule just settled, which must be written down and reflected in every reconciliation. The second is the ratio features: aov, discount_share, store_share, and the two per-month rates are ratios, so each carries the full four-element definition of Section 3.5.1 and each has an edge-case story — a one-order customer's aov equals that order's value; a customer with no discounted lines has discount_share exactly 0, which is data rather than missingness; and exposure_months carries a declared floor of one month, because a customer observed for nine days cannot support a stable monthly rate.

The third decision is the one this chapter cares about most, and the draft version of this section got it half right. Holding tenure_days out of the clustering inputs stops the algorithm from using signup date directly. It does not stop the algorithm from rediscovering it. Frequency and monetary value computed over a 24-month file are mechanically bounded by how long the customer has existed: a customer who signed up nine weeks ago cannot have accumulated two years of orders or two years of spend even if her purchasing intensity is identical to a two-year customer's. Lifetime totals in a fixed window are, in part, a measure of exposure wearing the

costume of value — and a segmentation that clusters on them alone will separate customers by age and then let a manager read the separation as quality.

The repair is to give the algorithm both readings and let the profile distinguish them. The level features (frequency, monetary) say what the customer has done in total; the intensity features (orders_per_month, revenue_per_month) say what the customer does per month of exposure, with exposure defined as observed time in the window rather than calendar tenure. A short-tenured customer buying briskly and a long-tenured customer buying rarely can share a lifetime total and sit far apart on intensity, which is exactly the distinction a treatment map needs. Note the deliberate asymmetry: tenure is excluded as a clustering coordinate and simultaneously used as a denominator. Be exact about what that buys. Expressing behavior per month of exposure reduces the mechanical effect of tenure and puts it on screen beside the lifetime totals; it does not eliminate the effect from a specification that still contains those totals, and the per-month rates are themselves least stable for the customers observed shortest, which is why exposure carries a floor. Clustering on tenure directly, by contrast, is how the effect is rediscovered and mistaken for a finding.

Two consequences follow for the sections ahead. First, the fourth segment the labs recover is a lifecycle position rather than a quality verdict — it collects customers whose observed value is low chiefly because their observed time is short, together with longer-tenured customers who have genuinely gone quiet; Section 6.14 shows how to read it and how not to. Second, the input set now contains features that are mathematically dependent on one another — aov is monetary divided by frequency, and each per-month rate is a level feature divided by exposure. Standardization will equalize their scales, and Section 6.7 will show that equalizing scale is not the same as removing redundancy. The response is not to guess but to measure: the feature-set sensitivity analysis of Lab 6.2 refits without aov, without the level features, and without the intensity features, and reports how far the assignments move.

The build closes the way every grain change in this guide closes, with reconciliation to known totals. The certified file's verification log (Chapter 4) fixed the total revenue, the distinct order count, and the distinct purchasing-customer count; the feature table must return them exactly — the sum of monetary equals certified revenue, the sum of frequency equals certified distinct orders, and the row count equals certified distinct purchasers. Lab 6.1 performs the build twice, first on a hand-checkable miniature and then at full scale, and its Verification Checks are these three equalities plus the population statement that explains the gap between the feature table's row count and the customers table's.

6.7 Distance, Similarity, and the Tyranny of Units

Post hoc segmentation needs what a priori segmentation never did: a mathematical answer to the question “which customers are alike?” Clustering methods group customers by similarity, and the standard move is to define similarity as the inverse of distance: represent each customer as a point whose coordinates are the feature values, and call two customers similar when the points

are close. The workhorse distance is Euclidean — the straight-line distance of geometry, extended to as many dimensions as there are features (Everitt et al., 2011).

DEFINITION

Euclidean Distance

The Euclidean distance between two customers represented as feature vectors is the square root of the sum of squared differences across their features. It is the default distance of k-means clustering; its value depends on the units and spread of every feature that enters it, which is why feature scaling is part of the method rather than an optional refinement.

Source: Adapted from Everitt et al. (2011).

The definition's second sentence is where segmentation projects quietly die, so it earns a worked example at hand scale. Take three customers from Lab 6.1's miniature, described on three features — recency in days, frequency in orders, monetary in dollars. C002, the occasion shopper: recency 120, frequency 2, monetary 400.00. C005, the frequent urban regular: recency 3, frequency 3, monetary 127.00. C006, the lapsed one-time big basket: recency 198, frequency 1, monetary 242.00. Intuition and marketing logic say C002 resembles C006 — infrequent, high-basket, fading — far more than either resembles C005. The raw arithmetic agrees, but examine why. The C002-C006 distance is the square root of $78^2 + 1^2 + 158^2$, which is the square root of 31,049, about 176.2 — and 24,964 of the 31,049, roughly 80 percent, is contributed by the monetary difference alone. The C002-C005 distance is the square root of $117^2 + 1^2 + 273^2$, about 297.0, with monetary contributing 84 percent. Frequency, the feature Chapter 5 suggested separates StyleCraft's worlds most sharply, contributes one part in tens of thousands to both distances. The dollars are not more important; they are merely measured in bigger numbers. Recency in days spans a couple of hundred; monetary in dollars spans hundreds and, at full scale, thousands; frequency spans single digits. Unscaled Euclidean distance is, on this table, monetary value with a rounding error — the clustering would dutifully return spend bands and a manager would call them segments.

The remedy is standardization: convert every feature to a common scale before any distance is computed. The standard choice is the z-score — subtract the feature's mean, divide by its standard deviation — after which every feature has mean 0 and standard deviation 1, and a difference of 1.0 means “one standard deviation apart” regardless of the feature's native units (James et al., 2021). On the standardized miniature, frequency's voice returns: C005's three orders against C006's one order becomes a difference of about two standard deviations, fully comparable to the monetary gap, and the recovered geometry matches the marketing intuition instead of the payroll.

DEFINITION

Standardization (z-Score Scaling)

Standardization rescales a numeric feature by subtracting its mean and dividing by its standard deviation, giving every standardized feature mean 0 and standard deviation 1 so that distance computations weight features by their variation in standard-deviation units rather than by their native units. The scaling parameters are computed from a declared population, are part of the segmentation's written definition, and must be retained with the fitted model if new customers are ever to be assigned consistently (Section 6.10).

Source: Adapted from James et al. (2021) and Pedregosa et al. (2011).

CONCEPT

Scaling Is a Modeling Decision, Not Plumbing

Standardization looks like preprocessing and is actually a declaration of what "alike" means: it asserts that one standard deviation of recency should count the same as one standard deviation of discount dependence. That is usually the right default and it is never neutral — an analyst who wanted monetary value to matter more could legitimately weight it, provided the weighting is declared and defended. The indefensible version is the silent one: skipping scaling entirely, which does not avoid the weighting decision but makes it by accident, handing the definition of customer similarity to whichever feature happens to be denominated in the largest numbers. When auditing any clustering — your own, a colleague's, or an assistant's — the first question is Chapter 3's question in new clothing: what, exactly, was measured? Here: what, exactly, was the distance computed on?

Source: Course concept developed for this guide, informed by James et al. (2021).

Three boundary notes complete the toolkit at working level. Standardization applies to numeric features; categorical attributes like home_metro do not subtract and divide, which is the technical half of the reason Table 6.5 holds them out for profiling. Z-scores do not remove skew — Chapter 5 showed monetary value is strongly right-skewed, and it remains right-skewed after standardization, so a heavily skewed feature can still exert outsized pull through its extreme values, and standardization computed from the mean and standard deviation is itself sensitive to those extremes (Pedregosa et al., 2011). Working practice ranges from accepting this, to log-transforming skewed monetary features, to the rank-based scoring RFM already performs; the labs use standardization and let the stability checks of Section 6.10 report whether extreme customers are steering the solution.

The third note is the one most often missed, and Section 6.6 has already set it up. Standardization equalizes scale. It does nothing about redundancy. If two features carry the same information — or if one is a function of two others, as aov is of monetary and frequency — standardizing them does not collapse them into one voice; it gives each an equally loud voice, so the underlying quantity speaks several times in the distance calculation and is weighted accordingly. This is not automatically an error, and it does not automatically demand that a

feature be removed. It demands that the analyst know it is happening and test it: fit the proposed specification, refit with the composite dropped, refit with a component dropped, and compare the assignments and the profiles. Feature selection is part of the definition of distance, which means it is part of the definition of "alike," which means it is a modeling decision that belongs in the written specification rather than in a default. Lab 6.2 runs exactly that comparison.

6.8 k-Means Clustering: Intuition, Algorithm, and Sensitivities

With customers as scaled points and similarity as distance, clustering is the task of partitioning the points into groups that are compact within and separated between. This guide's method is k-means, the most widely used clustering algorithm in marketing practice — chosen here for the same reasons it is chosen in industry: it is fast at customer-base scale, its output is easy to profile, and its failure modes are well understood, which for a verification-centered course is a feature (MacQueen, 1967; Lloyd, 1982; James et al., 2021). It also belongs to a family the rest of Part II does not: k-means is unsupervised: there is no outcome variable, nothing is predicted, and the algorithm is given no answer to fit toward (Pedregosa et al., 2011). That is precisely why validation has to be constructed rather than measured.

The intuition first. Suppose the number of groups, k, is given — Section 6.9 takes up where it comes from. k-means seeks k cluster centers, called centroids, such that every customer is assigned to the nearest centroid and the total of squared distances from customers to their assigned centroids — a quantity called inertia, or within-cluster sum of squares — is as small as possible. A good solution is one where customers sit close to their centers: compact clusters, and a centroid that genuinely typifies its members, because each centroid is simply the average of its cluster on every feature. That last fact is the profiler's gift — a centroid is a readable customer sketch in standardized units — and it is also a hand-verifiable claim Section 6.12's audit routine will exploit.

DEFINITION

k-Means Clustering

k-means clustering partitions observations into a chosen number k of clusters by locating k centroids and assigning each observation to its nearest centroid so as to minimize inertia, the total within-cluster sum of squared distances. It is fitted by alternating assignment and centroid-update steps from an initial placement of centroids, and its solution can depend on that initialization. It is an unsupervised method: it uses no outcome variable and predicts nothing.

Source: Adapted from MacQueen (1967), Lloyd (1982), and Pedregosa et al. (2011).

The algorithm sketch is short enough to hold in your head, and holding it there is what demystifies the output. Step zero: place k centroids somewhere — classically at random customer positions. Step one, assignment: give every customer to its nearest centroid. Step two, update: move each centroid to the mean of the customers just assigned to it. Repeat steps one and two; each round can only lower inertia, so the alternation settles — converges — when

assignments stop changing. The procedure is a loop a student can trace on paper with a dozen points, and Lab 6.2's first Verification Check does exactly that in miniature.

Convergence, however, is not correctness, and the algorithm's two sensitivities are the chapter's verification theme in mathematical form. The first is initialization sensitivity: the loop settles into a locally best solution reachable from wherever the centroids started, and different random starts can settle into different partitions with different inertias (Pedregosa et al., 2011). Practice answers with better starts and more of them — the k-means++ initialization spreads the initial centroids apart, and standard implementations rerun the whole procedure from multiple starts, keeping the lowest-inertia result (Arthur & Vassilvitskii, 2007) — but "the software reruns internally" does not discharge the analyst's obligation: a solution worth deploying should be demonstrated stable across independently seeded fits, which is Section 6.10's stability check. The second sensitivity is the one the opening case turns on, and it deserves its own box.

CONCEPT

The Algorithm Returns the k You Asked For

k-means answers the question it is asked, which is not the question the marketer is asking. Asked for nine clusters, it partitions the customers into nine — optimally arranged, confidently bounded — whether the data contain nine groups, four, or none at all; a continuous but unstructured customer cloud can be sliced into k tidy regions with the same serene output. (The one exception is degenerate data with fewer distinct points than requested clusters, where an implementation may return fewer clusters and warn.) The algorithm cannot report "there are no segments here," and it will never volunteer that a different k fits better. Every claim of the form "the analysis found nine segments" is therefore, uninspected, a claim about the prompt, not about the customers. What converts a partition into evidence of structure is everything around the algorithm: the choice of k defended on multiple instruments (Section 6.9), stability across refits and samples, and separation on variables the algorithm never saw (Section 6.10). The intern's overnight notebook is not wrong; it is unexamined — and this box is the reason the distinction matters.

Source: Course concept developed for this guide, informed by Pedregosa et al. (2011).

Two further properties round out the working-level picture, both consequences of the squared-Euclidean objective. k-means prefers compact, roughly round clusters of broadly similar spread; genuinely elongated, unevenly sized, or nested structures get carved into pieces, which at customer-base resolution is usually tolerable and occasionally the finding (Pedregosa et al., 2011). And because means are sensitive to extremes, a handful of outlying customers — Chapter 5's top-decile whales — can drag a centroid toward themselves; the stability checks and the concentration awareness of Section 5.9 are the standing defenses. Neither property demands a more exotic algorithm for this course's purposes; both demand the profiling and validation that the next two sections install.

6.9 Choosing k: Elbow, Silhouette, and Business Judgment

Post hoc segmentation's defining freedom — that group membership is estimated rather than declared — arrives with a question the algorithm cannot answer for itself: what should k be? The honest answer is that k is chosen, not discovered, and it is chosen with instruments plus judgment. This section supplies the two standard instruments and then insists on the third leg, because on real customer data the instruments alone are routinely indecisive and occasionally misleading.

The first instrument reads inertia. Fit k-means across a range of candidate k values — the labs use 2 through 8 — and plot each solution's inertia. At the optimum, inertia is non-increasing as k rises (more centers, shorter distances; at the absurd limit of one cluster per customer it reaches zero), so the plot's level is uninformative; its shape is the signal. Where adding a cluster stops buying much compactness, the curve bends, and the bend — the elbow — marks the k beyond which additional clusters are subdividing structure rather than finding it (Thorndike, 1953).

> **DEFINITION**
>
> **Elbow Method**
>
> The elbow method selects candidate values of k by plotting inertia against k and looking for the bend where further increases in k yield sharply diminishing reductions in inertia, on the reasoning that k values past the bend split real groups rather than separate them. Its verdicts are visual and frequently ambiguous, so it nominates candidates rather than deciding.
>
> Source: Adapted from Thorndike (1953).

Two properties of that curve are worth fixing before Lab 6.2 reads one. At the optimum inertia is non-increasing rather than strictly decreasing: an added centroid that buys no improvement leaves it flat, and a flat step is information rather than a defect. And because each candidate k is a separately initialized optimization, a higher-k fit can settle into a worse local minimum than a lower-k fit and report a small rise. A rise is therefore a warning to inspect initialization and convergence and to refit with more starts — not, by itself, proof that the code is broken.

The second instrument reads separation per customer. The silhouette score asks, for each customer, how close it sits to its own cluster compared with the nearest other cluster, scaled to lie between −1 and +1: near +1 means snugly placed, near 0 means on a boundary, negative means arguably misassigned (Rousseeuw, 1987). Averaged over all customers, the silhouette gives each candidate k a single separation grade, and the k with the highest average is the instrument's nominee. The silhouette is more decisive than the elbow and correspondingly more dangerous to obey blindly: on customer bases with one dominant divide — StyleCraft's urban-suburban split is exactly such a divide — the silhouette often crowns k = 2, because the coarsest cut is the cleanest, even when finer structure is real, designed, and strategically the entire point.

DEFINITION

Silhouette Score

A customer's silhouette compares its average distance to members of its own cluster with its average distance to members of the nearest other cluster, yielding a value between −1 and +1 where higher means better placed. The mean silhouette across customers summarizes a clustering's separation and is compared across candidate values of k. It grades geometry only: a high silhouette says the partition is clean, not that its groups are marketing segments.

Source: Adapted from Rousseeuw (1987).

The third leg is business judgment, and calling it a leg rather than a tiebreaker is deliberate: the instruments measure geometry, and geometry is not the objective — a segmentation is chosen for treatment, and Section 6.2's treatment question bears directly on k. The platform sustains four journeys; a k of 9 is undeployable this fall regardless of its silhouette, and a k of 2 discards distinctions — discount dependence, lifecycle stage — that the treatments exist to exploit. Judgment also carries domain knowledge the geometry cannot see: Chapter 5's cohort and concentration evidence, the designed logic of the expansion story, and the practical requirement that every segment be large enough to justify a journey (Section 6.11's substantial criterion, previewed). Table 6.6 summarizes the three legs and, importantly, what each cannot do.

Table 6.6

The three legs of choosing k

Leg	What it measures	What it cannot do
Elbow (inertia)	Diminishing returns in compactness as k grows	The bend is often gradual; nominates a range, rarely a value
Silhouette	Average quality of each customer’s geometric placement	Favors the coarsest clean cut; blind to strategy and treatment capacity
Business judgment	Deployability, treatment capacity, domain structure, segment sizes	Cannot certify that the preferred k matches structure — that is the instruments’ and the validation checks’ job

The working procedure, which Lab 6.2 executes, braids the legs rather than ranking them: compute elbow and silhouette across the candidate range; shortlist the k values that either instrument nominates; profile the solutions on the shortlist (Section 6.10); and choose on treatment grounds among solutions the instruments and profiles both support — documenting the choice and the rejected candidates, because “why four?” is the first question the sign-off meeting will ask, and “the silhouette said so” is not an answer that survives this section's own caveats. On StyleCraft the braid has a designed resolution: the instruments shortlist a small range, the profiles

at k = 4 align with recognizable and treatable customer stories, and — uniquely to a designed dataset — the answer key confirms it. The documentation habit, not the confirmation, is what transfers to data without keys.

6.10 Profiling, Validating, and Naming Segments

A converged, k-chosen clustering is still only a partition. This section converts it into a segmentation the CRM manager can sign: first by profiling — describing each cluster honestly enough to see what it is — then by validating — testing what the partition can and cannot be shown to reflect — and finally by naming, ruling, and preserving — translating clusters into words, into written assignment rules, and into a saved model the organization can actually run. All three steps consume Chapter 5's toolkit; what this chapter owns is their application to algorithmic groups and the two checks, stability and holdout profiling, that algorithmic groups uniquely require.

Cluster profiling is the customer profile of Section 5.6 with clusters as the grouping variable, and every discipline transfers: sizes first, centers with spreads, declared grains, distributions shown where shapes are mixed. The centroid table is the natural spine — k rows, one per cluster, features as columns, values as within-cluster means — read first in standardized units (which features define each cluster: “recency 1.4 standard deviations better than average”) and then translated back to native units for the deck (“median 11 days since last purchase”), because standardized units are for analysts and days and dollars are for decisions. Two Chapter 5 cautions bite with extra force here. Cluster means are means: a cluster's monetary centroid can be propped by a few whales, so spreads and medians travel with it. And cluster sizes are findings, not footnotes: size is the first row of the profile and the first input to the substantiality judgment of Section 6.11.

Validation is the new obligation, and it has two instruments that answer two different questions. The first is the stability check, and its question is whether the partition is a property of the data or of the fitting run: a segmentation worth deploying should not depend on the random seed, the particular sample, or the day the notebook ran. The operational tests are refits under perturbation — refit from several independent seeds, refit on random subsamples of, say, 80 percent of customers — followed by a comparison of the assignments. The comparison is where practitioner treatments usually go wrong. Cluster labels are arbitrary from run to run, so the naive procedure — cross-tabulate and count the customers outside each row's dominant cell — quietly assumes the runs line up one to one, and they need not: two clusters in the baseline can both have their largest overlap with the same cluster in the refit, and the arithmetic will report stability that does not exist.

Use a measure built for the problem instead. The adjusted Rand index compares two partitions by asking, over all pairs of customers, how often the two solutions agree about whether the pair belongs together — a quantity that is invariant to relabeling and corrected for the agreement expected by chance (Hubert & Arabie, 1985). It reaches 1.0 for identical partitions and sits near 0.0 for unrelated ones, and because the chance correction can be exceeded

downward it also goes negative, to a lower bound of −0.5 for especially discordant partitions; a negative value means the two solutions agree less than chance would predict (Pedregosa et al., 2011). Where a customer-level reassignment rate is genuinely wanted for the deck, align the labels first with an explicit one-to-one assignment, then count. Precision of language matters here too: a scikit-learn fit with n_init = 10 is already the best of ten internal starts, so five seeded fits are "five independently seeded fits, each retaining the best of ten initializations," not "five random starting points."

DEFINITION

Stability Check

A stability check assesses whether a clustering solution reflects structure in the data rather than an accident of fitting, by refitting under perturbation — independently seeded initializations, random subsamples — and measuring agreement between the resulting partitions with a relabeling-invariant index such as the adjusted Rand index such as the adjusted Rand index, which is 1.0 for identical partitions, near 0.0 for unrelated ones, and negative for partitions agreeing less than chance would predict. Low agreement indicates a partition the data do not support at the chosen k. High agreement establishes that the partition is reproducible; it does not establish that the groups are meaningful.

Source: Adapted from Everitt et al. (2011); index per Hubert and Arabie (1985).

The second instrument is holdout profiling, and it answers a different question: not "is this partition reproducible?" but "does it line up with anything outside its own inputs?" Recall Table 6.5's division: the algorithm clustered on behavior only, while metro, loyalty tier, app use, acquisition channel, age band, and tenure were withheld. Now profile the clusters on the withheld attributes. If a cluster built purely from purchasing rhythm turns out to be overwhelmingly app-using, loyalty-enrolled, and urban — attributes the algorithm never saw — the clusters are at least describing customers who differ in ways beyond the features that made them. Holdout profiling is predict-then-verify (Section 1.7) at the level of an entire analysis: before running the holdout profile, the analyst writes down what each cluster should look like on the withheld attributes if it is the kind of customer the behavioral profile suggests — and then checks.

Be exact about what this establishes, because the temptation to overclaim here is strong and the overclaim is the kind a sign-off meeting repeats. The withheld attributes are not statistically independent of the clustering inputs — metro, app use, loyalty enrollment, acquisition channel, and purchasing behavior are correlated with one another — so agreement between them is convergent, interpretive evidence rather than independent proof. It raises confidence that the clusters are interpretable and correspond to describable kinds of customers; it does not demonstrate that they are natural types, and it emphatically does not demonstrate that they will respond differently to treatment, which is a causal claim only a test can support. The converse is equally worth stating: a behaviorally useful segmentation may show little demographic separation and remain entirely valid for CRM purposes. Failed predictions are findings rather than embarrassments — a "loyal regulars" cluster that turns out to be half discount-dependent

has just told you the behavioral features were incomplete, one section before the naming step would have shipped the error.

> **CONCEPT**
>
> **Check Clusters on Variables They Never Saw**
>
> Withhold meaningful attributes from the clustering inputs; predict, from each cluster's behavioral profile, how it should differ on the withheld attributes; then profile and compare. Confirmed predictions are one layer of external-variable evidence that no inertia curve or silhouette can supply. They are not proof of natural kinds: the holdout attributes are correlated with the inputs, so agreement is convergent rather than independent, and nothing in the exercise speaks to how the groups will respond to being treated differently. Report holdout agreement as what it is — supporting evidence for interpretability — and leave response claims to the experiments of later chapters.
>
> Source: Course concept developed for this guide, informed by Everitt et al. (2011) and Pedregosa et al. (2011).

Naming comes next, deliberately late, because names are the most contagious artifact a segmentation produces: the profile stays in the appendix, the name goes in the calendar, the deck, and eventually the org chart. Two rules keep names honest. Name from evidence: every word in a segment name must be redeemable against the profile — a cluster may be called “Store-Only Occasion Shoppers” if store_share and aov say so, and may not be called “Affluent” on any evidence StyleCraft's schema contains, which retires the intern's assistant's christening and enforces Section 6.3's base-vocabulary discipline. And name the behavior, not the person, where possible: behavioral names (“Discount-Dependent,” “One-Purchase Recent”) describe what the data measured; person names (“Bargain Hunters”) drift toward the psychographic claims the data cannot support and invite the gap-instinct misreadings Section 5.14 warned about.

The last step is the one most chapters omit, and omitting it is how segmentations die in production. A clustering is not a set of thresholds; it is a fitted object. Assigning a customer to a cluster requires the feature definitions, the scaler's means and standard deviations, the fitted centroids, and a nearest-centroid rule applied in standardized space. Simplified business rules — a WHERE clause on two or three columns — are useful and often necessary, because campaign platforms speak SQL rather than Python, but a threshold rule written by eye from a centroid table does not reproduce the fitted assignments and should never be assumed to. The professional practice is to do both and reconcile: retain and version the scaler, the model, and the feature definitions as the assignment system of record; publish the simplified rule as the operational approximation; and report the agreement between them, naming the customers on which they disagree. Discovery was statistical; deployment is governance, and governance runs on artifacts that can be rerun.

CONCEPT

Preserve the Model, Not Just the Rules

A deployed segmentation needs five things retained together and versioned as one release: the feature definitions and the analysis window; the fitted scaler's means and standard deviations; the fitted centroids; the nearest-centroid assignment procedure in standardized space; and the date and population on which all of it was fitted. New customers are then assigned by transforming their features with the saved scaler and predicting with the saved model — not by re-fitting, which would silently redraw the segments, and not by eyeballed thresholds, which approximate the model without reproducing it. Where a simplified rule is published for a campaign platform, measure its agreement with the model's assignments and document the exceptions. A segmentation nobody can rerun is a slide, not a system.

Source: Course concept developed for this guide, informed by Pedregosa et al. (2011).

6.11 From Segments to Targets: Useful-Segment Criteria and Segment Attractiveness

Verified, profiled, named segments still leave the strategic decision standing: which segments get which of the four journeys — and does every segment deserve one at all? This section supplies the two evaluation frames marketing strategy uses: the usefulness screen, which every segment must pass to be operable, and the attractiveness comparison, which ranks the segments that pass.

The usefulness screen is the classical five-criterion test (Kotler & Keller, 2016). A segment is measurable if its size and characteristics can be computed from available data; substantial if it is large and valuable enough to justify differentiated treatment; accessible if the firm can actually reach it through channels it operates; differentiable if it responds differently to marketing than other segments; and actionable if the firm can design and execute distinct treatment for it within real constraints.

DEFINITION

Useful-Segment Criteria

The useful-segment criteria hold that a segment justifies differentiated marketing only if it is measurable (size and traits computable from data), substantial (large and valuable enough to serve profitably), accessible (reachable through operable channels), differentiable (distinct in its response to marketing), and actionable (treatable distinctly within the firm's actual capabilities).

Source: Adapted from Kotler and Keller (2016).

Two of the five criteria are routinely mis-scored in practice, and both errors are worth naming before the screen is applied. The first is differentiability. Clustering establishes that groups differ on the features that made them, and profiling establishes that they differ on some attributes that did not. Neither establishes differential response, which is a statement about what

happens when treatment changes and therefore a causal claim. A descriptive profile nominates a treatment hypothesis; a campaign or an experiment validates it. Scored honestly, the differentiable cell of a fresh segmentation reads “distinct behavioral profile; differential response untested,” and the treatment map's first year is partly an apparatus for testing it.

The second is substantiality, where the usual shorthand — a small segment is a curiosity — is too categorical to be useful. A segment of ninety customers may be exceptionally valuable per customer, strategically important as a beachhead in a new market, cheap to serve through an existing channel, legally or reputationally consequential, or valuable as a test cell that de-risks a larger rollout. What size actually determines is the burden of proof: a small segment must justify its journey on value, growth, strategic importance, and treatment economics rather than on headcount, and the smaller it is, the more of that case has to be made explicitly. Substantiality is a threshold question about whether differentiated execution pays, not a headcount test.

The criteria do their best work applied concretely, and Table 6.7 applies them to the four segments the labs recover — profiled per Section 6.10 and matching, by design, the dataset's planted structure. Reading the table row by row is a rehearsal for the sign-off meeting, and three cells deserve narration. Accessibility is where the suburban segment's weakness concentrates: a store-only, low-opt-in, no-app population is expensive to reach with a CRM program that lives in email and app — a finding that reframes “high AOV” considerably. Actionability is where the platform constraint rules: four journeys is the actionability budget, and it is why the agency's seven personas and the intern's nine clusters fail this screen before any statistics are consulted. And the New/At-Risk row carries the chapter's standing caveat in two cells at once — the segment is perfectly measurable, but what it measures is lifecycle position as much as value, and its response is not merely undifferentiated but unobserved.

Table 6.7

The usefulness screen applied to the four recovered segments

Criterion	Urban Loyal Core	Suburban Occasion	Discount-Dependent	New / At-Risk
Measurable	Yes — fitted behavioral assignment procedure	Yes — fitted behavioral assignment procedure	Yes — fitted behavioral assignment procedure	Yes — but measures lifecycle position as much as value (Section 6.14)
Substantial	Large share of customers and revenue	Smaller count; high revenue per order; margin caveat	Meaningful count; thin margin	Large count; low observed value largely by construction
Accessible	High — email and app reachability, with strong loyalty enrollment	Low — store-only, weak opt-in, no app	High — reachable on promotional channels	Moderate — recent signups, opt-in fresh
Differentiable	Distinct behavioral profile; differential response untested	Distinct rhythm and basket; response untested	Purchases concentrate on discounted lines; response to discount and brand treatments untested	Profile confounded with exposure; response unobserved
Actionable	Yes — reward and early-access journey	Partially — occasion triggers possible; channel limits bite	Yes, carefully — Section 6.16's trap lives here	Only after a tenure-based branch separates recent signups from established customers who have lapsed; the branch sits inside the segment's one journey slot and is declared as part of the assignment rule

Applied strictly, that qualification means none of the four segments has yet passed all five criteria, and the chapter will not pretend otherwise. Among segments that provisionally pass the measurable, substantial, accessible, and actionable portions of the screen, attractiveness analysis supports an initial targeting decision while differential response remains a treatment hypothesis to test. Attractiveness is that decision's evaluation frame: segments are compared on current size and value, growth trajectory, margin quality, reachability cost, strategic fit with the firm's direction, and fit with the firm's capabilities (Kotler & Keller, 2016; Wedel & Kamakura, 2000). The analyst's contribution is to put evidence in every cell of that comparison — sizes and values from the profile, growth from Chapter 5's cohort tables, margin from the unit-cost story, reach

cost from the accessibility facts — and then to mark, honestly, which cells are evidence and which are judgment.

DEFINITION

Targeting and Segment Attractiveness

Targeting is the selection of which segments to serve and with what priority and resources. Segment attractiveness is the comparative evaluation that informs it, weighing each qualifying segment's size, value, growth, margin, cost to reach, and fit with the firm's strategy and capabilities — an evidence-plus-judgment ranking, not a computation. Cells sourced from the analysis and cells sourced from management judgment are marked separately, because the two carry different warranties.

Source: Adapted from Kotler and Keller (2016) and Wedel and Kamakura (2000).

Table 6.8 gives the frame a worked shape, and building it is the deliverable that connects the clustering to the four slots. Its rows are the qualifying segments; its columns run from the arithmetic the profile supplies to the judgment the room supplies, in that order, with an explicit evidence-versus-judgment declaration so that no reader has to guess which is which. Code 6.15 produces the profile-arithmetic rows directly. The same-age trajectory row is evidence rather than judgment but it comes from elsewhere — it is merged in from the Chapter 5 cohort table, because a cohort view needs a period axis the feature table does not carry. The remaining rows, cost to serve and strategic fit and the declaration and the recommended priority, are written by the analyst and defended in the meeting.

Table 6.8

The segment-attractiveness matrix (structure; values computed in Lab 6.2)

Dimension (source)	Urban Loyal Core	Suburban Occasion	Discount-Dependent	New / At-Risk
Customers, and share of base (profile)	—	—	—	—
Revenue in window, and share (profile)	—	—	—	—
Revenue per customer (profile)	—	—	—	—
Revenue per exposure-month (profile)	—	—	—	—
Same-age trajectory (Chapter 5 cohort view)	—	—	—	—
Discount share of revenue, a margin proxy (profile)	—	—	—	—

Dimension (source)	Urban Loyal Core	Suburban Occasion	Discount-Dependent	New / At-Risk
Reachability: share app or email reachable (holdout profile)	—	—	—	—
Cost to serve and to treat (judgment, with finance)	—	—	—	—
Strategic fit with the expansion (judgment)	—	—	—	—
Evidence versus judgment (declared)	—	—	—	—
Recommended priority and journey (decision)	—	—	—	—

Attractiveness analysis is where descriptive language most wants to become causal, and the insight-statement discipline of Section 5.9 patrols the line. The data support “the Urban Loyal Core holds the largest share of repeat revenue and is the cheapest segment to reach”; they do not yet support any claim about what targeting it will cause. Every recommended priority in the last row of Table 6.8 is a decision made under that limitation, which is an argument for making the limitation visible rather than for postponing the decision.

The section's output, and the chapter's strategic deliverable, is the treatment map: each retained segment paired with a journey, a priority, and the one-line evidentiary basis; each excluded grouping paired with the criterion it failed; and the customers with no purchase in the window split between the always-on activation program and the win-back treatment, per Section 6.6. That map is what four slots on the fall calendar actually purchase — and producing it defensibly required every section behind it.

6.12 AI as a Segmentation Assistant

Segmentation is the point in this guide where AI assistance changes character. In Chapters 4 and 5, the assistant drafted code whose outputs the analyst could check against arithmetic — totals, counts, recomputed cells. In this chapter the assistant can draft the entire analytic act: prompted with a customer file, a current assistant will build features, scale them, choose a k, fit the clustering, name the segments, and write the strategy narrative, delivering in one response what the chapter just spent nine sections disciplining. The intern's overnight notebook is not a hypothetical; it is the new default state of the world. The division of labor therefore has to be stated with more care here than anywhere in Part I.

Where the assistant genuinely helps, use it, and the list is substantial. Drafting the feature-table build — the groupby scaffolds, the ratio features with their edge cases, the reconciliation printout — is legitimate delegation on mechanics Chapter 4 established, provided the derivation

rules come from your written definitions rather than the assistant's defaults. Generating the evaluation loop — fit across k = 2 through 8, collect inertias and silhouettes, plot both — is boilerplate the assistant writes faster than you do. Proposing candidate features is a useful brainstorm ("what behavioral ratios might separate promo-driven customers?") provided proposals enter the specification as candidates, not conclusions. And drafting profile narration from a finished, verified centroid table can accelerate the deck, provided every sentence survives Section 6.10's naming rules and Section 5.9's insight-statement discipline.

What must never be delegated are the judgments, and this chapter's failure modes are characteristic enough to name individually. The confident k: asked to "segment these customers," assistants pick a k — often whatever a single silhouette sweep nominates, often simply a plausible-sounding number — and present it as found rather than chosen, with none of Section 6.9's braid. The skipped scaling: assistants sometimes cluster raw features, and Section 6.7 showed exactly what that produces — spend bands in segment costume; the error is invisible in the output, which is what makes it dangerous. The invented persona: asked to name or describe clusters, assistants reliably enrich them with attributes no input contained — incomes, life stages, attitudes — because fluent persona language saturates their training; "Affluent Suburban Loyalists" was generated by exactly this reflex, asserting a base the schema lacks (Section 6.3) and a loyalty relationship the holdout profile contradicts. The single-run solution: assistants fit once and report; nothing in a default workflow refits, perturbs, or compares partitions, so initialization luck ships as structure. The arbitrary tie rule: asked to score RFM quintiles on lumpy frequency, assistants reach for a row-order tiebreaker and never mention that the resulting distinctions have no behavioral content (Section 6.5). The narrated geometry: asked to interpret clusters, assistants convert arithmetic into story — "this segment is disengaging because…" — attaching motives and causes to what is, so far, a partition of a feature table. And the overfit taxonomy: more clusters always fit the fitted file more snugly, and an assistant asked for "detailed segments" will happily oblige with nine, geometry's version of the actionability failure in Section 6.11.

AI IN PRACTICE

The Cluster Solution That Must Survive a Refit

A productive pattern for delegated segmentation, in two prompts and an audit. Prompt one, the specification: paste the feature list with derivation rules (Table 6.5), the population rule, the analysis date, the certified totals, and the platform's treatment capacity, then ask for the build and the evaluation sweep only — "Build customer_features per these rules; print the three reconciliation figures; fit k-means for k = 2 through 8 on standardized inputs with a fixed random seed; report inertias and mean silhouettes; recommend nothing yet."

Prompt two, after you have chosen k per Section 6.9's braid: request the fit, the centroid table in standardized and native units, and the validation artifacts — adjusted Rand indices across five independently seeded fits and two 80 percent subsamples, and the holdout profile on the withheld attributes. Require that the fitted scaler and model be returned as objects, not as prose thresholds.

Then audit in four moves before any name or narrative is written: verify one centroid by hand (filter the feature table to one cluster, average one feature, compare); read the stability indices and reject the solution if the partitions do not reproduce; check the holdout profile against the predictions you wrote down before running it; and strike every noun in every segment name that no input feature or holdout column can redeem. Predictions first, per Section 1.7's predict-then-verify; every exchange documented per Appendix D — the assistant fitted the clustering, but the analyst of record certified that it is structure and not luck.

The deeper point extends the guide's standing division of labor to its first genuinely algorithmic output. In Chapter 5 the assistant drafted summaries and the analyst audited the claims; here the assistant drafts a model of the customer base and the analyst audits the model — scaling, k, stability, holdout agreement, and every noun in every name. Appendix C provides prompt templates for the feature build and the audit routine; Appendix D's documentation requirement applies to every exchange; and the labs that follow are built so the audit has teeth, because on StyleCraft's designed data the truth is known: four segments were planted, and an analysis that cannot recover them — or that confidently reports nine — has failed in a way the answer key will name.

6.13 Hands-On Application in Python and Google Colab

The preceding sections built the method; this section runs it against the opening case, in two labs that mirror the chapter's two halves. Lab 6.1 builds the raw material: the customer-grain feature table, first on a ten-customer miniature where every number can be verified by hand, then at full scale with reconciliation to certified totals, closing with the RFM grid. Lab 6.2 performs the discovery: scaling, k-means, the choice of k, profiling, and the validation checks — ending with the comparison this dataset uniquely permits, recovered clusters against the four designed segments, and with the two artifacts the CRM manager actually deploys: the attractiveness matrix and the saved assignment model. Throughout, follow the course division of labor: AI assistants may draft the code (Appendix C has prompting templates; Appendix A covers Colab

mechanics), but every output is predicted before it is computed, every build is reconciled to certified totals, and every AI exchange is documented per Appendix D.

6.13.1 Lab 6.1, Part A: The Feature Table in Miniature

The miniature is a twenty-order extract for ten StyleCraft customers, constructed in the notebook at order grain — Chapter 5's labs already practiced the line-to-order roll-up, so this lab starts one step later and points there rather than repeating it. The analysis date is June 30, 2026, matching Lab 3.1.

Code 6.1. Create the miniature order extract

```
import numpy as np
import pandas as pd

analysis_date = pd.Timestamp("2026-06-30")

orders_mini = pd.DataFrame({
    "customer_id": [
        "C001","C001","C001","C001","C002","C002","C003","C003",
        "C003","C004","C005","C005","C005","C006","C007","C008",
        "C008","C009","C010","C010"],
    "order_id": [
        "O01","O02","O03","O04","O05","O06","O07","O08","O09",
        "O10","O11","O12","O13","O14","O15","O16","O17","O18",
        "O19","O20"],
    "order_date": pd.to_datetime([
        "2026-06-21","2026-05-30","2026-04-18","2026-02-07",
        "2026-03-02","2025-11-19","2026-06-05","2026-04-24",
        "2026-01-16","2026-06-12","2026-06-27","2026-06-10",
        "2026-05-02","2025-12-14","2026-02-27","2026-06-18",
        "2026-03-29","2026-06-24","2026-05-16","2026-01-03"]),
    "order_revenue": [
        58.00, 47.00, 39.00, 52.00, 214.00, 186.00, 21.60,
        28.80, 19.20, 44.00, 36.00, 62.00, 29.00, 242.00,
        24.00, 71.00, 55.00, 38.00, 167.00, 145.00]
})

# The miniature's certified totals, written as checks rather than prose.
assert len(orders_mini) == 20
assert orders_mini["order_id"].is_unique
assert orders_mini["customer_id"].nunique() == 10
assert np.isclose(orders_mini["order_revenue"].sum(), 1578.60)
assert (orders_mini["order_date"] <= analysis_date).all()

print(len(orders_mini), "orders |",
      orders_mini["order_revenue"].sum().round(2), "total revenue")
```

Expected output: 20 orders and a total of 1578.6, with five assertions passing silently.
Input: twenty literal orders for ten customers. Transformation: none yet. Output: the extract's certified totals, recorded now because every build below must return them. The assertions are the point of the cell: Chapter 4 argued that a prediction belongs in the log before the step runs, and an assertion is that prediction written where the notebook itself will enforce it. The last one

guards the recency calculation that follows, since an order dated after the analysis date would produce a negative recency — a Chapter 4-grade date defect rather than a customer behavior. The grain change is one grouped aggregation, exactly Chapter 4's mechanics with this chapter's derivation rules:

Code 6.2. Build and reconcile the miniature feature table

```
cf = (orders_mini
      .groupby("customer_id")
      .agg(last_order=("order_date", "max"),
           frequency=("order_id", "nunique"),
           monetary=("order_revenue", "sum"))
      .assign(recency_days=lambda d: (analysis_date
                                      - d["last_order"]).dt.days,
              aov=lambda d: (d["monetary"] / d["frequency"]).round(2))
      .drop(columns="last_order")
      .round(2))

# The grain change must preserve the extract's certified totals, and the
# one-order edge case must behave the way the definition says it will.
assert len(cf) == 10
assert cf["frequency"].sum() == 20
assert np.isclose(cf["monetary"].sum(), 1578.60)
assert (cf["recency_days"] >= 0).all()
assert np.allclose(cf.loc[cf["frequency"] == 1, "aov"],
                   cf.loc[cf["frequency"] == 1, "monetary"])

print(cf.sort_values("recency_days"))
```

Expected output: ten rows sorted from C005 at 3 days to C006 at 198 days; C002 with frequency 2, monetary 400.00, aov 200.00; C006 with frequency 1 and monetary equal to aov at 242.00; and five assertions passing.

Input: the twenty certified orders. Transformation: rows are collected by customer, and two features are derived from the aggregates. Output: a ten-row customer-grain table. The three reconciliation assertions are the grain-change identities of Section 6.6 in executable form — row count is the distinct-customer count, frequency total is the order count, monetary total is the extract's certified revenue — and if any of them fails, the build has silently dropped or double-counted something. Finding that on ten customers is the entire reason the miniature exists.

VERIFICATION CHECK

Before you run: compute two rows entirely by hand from Code 6.1's extract. C002's row: last order March 2, 2026, so recency_days = 120 (29 remaining March days + 30 + 31 + 30); frequency = 2; monetary = 400.00; aov = 200.00. C006's row: recency_days = 198, frequency = 1, monetary = aov = 242.00. Then predict the two structural facts the assertions encode — ten rows, twenty orders, 1578.60 in revenue — and predict that C005 is the most recent customer at 3 days.

After you run: all five assertions pass silently and the printed table matches both hand-computed rows and the sort order.

Investigate if: a reconciliation assertion raises. A failed row count means the grouping key contained a missing value; a failed frequency total means orders were counted at line grain somewhere upstream; a failed monetary total means a filter ran that nobody declared.

Scoring is next, and the miniature was designed to make it teachable: ten customers cut into five bins means exactly two per bin, fully checkable by hand — and frequency was designed to break.

Code 6.3. Diagnose and resolve tied frequency scores

```
cf["R"] = pd.qcut(cf["recency_days"], 5, labels=[5,4,3,2,1]).astype(int)
cf["M"] = pd.qcut(cf["monetary"],     5, labels=[1,2,3,4,5]).astype(int)

# Frequency: predict the result before running this block.
try:
    cf["F"] = pd.qcut(cf["frequency"], 5, labels=[1,2,3,4,5]).astype(int)
except ValueError as e:
    print("qcut refused to cut frequency:", e)

# The declared repair (Section 6.5): three frequency bands cut at values
# the distribution actually takes, mapped onto the grid's 1-5 axis.
# Customers with the same order count always receive the same score.
freq_bands = pd.IntervalIndex.from_tuples(
    [(0, 1), (1, 3), (3, np.inf)], closed="right")
band_score = {freq_bands[0]: 1, freq_bands[1]: 3, freq_bands[2]: 5}
cf["F"] = (pd.cut(cf["frequency"], bins=freq_bands)
             .map(band_score).astype(int))

assert cf[["R", "F", "M"]].notna().all().all()
assert set(cf["F"]).issubset({1, 3, 5})
assert (cf.groupby("frequency")["F"].nunique() == 1).all()

print(cf[["recency_days", "frequency", "monetary", "R", "F", "M"]]
      .sort_values(["R", "F"], ascending=False))
```

Expected output: a ValueError message reading "Bin edges must be unique," then a ten-row table in which C005 scores R5 F3 M3, C009 scores R5 F1 M1, C001 scores R4 F5 M4, and C006 scores R1 F1 M4.

Input: the miniature feature table. Transformation: recency and monetary value are cut into quintiles, frequency is banded, and the scores are attached. Output: three ordinal columns and

one instructive failure. The ten frequency values are 4, 3, 3, 2, 2, 2, 1, 1, 1, 1: four distinct values cannot supply five distinct bin edges, so qcut raises rather than inventing a cut. That error is the data telling you the quantity cannot support five honest bins, and the repair is a scoring-rule decision rather than a coding workaround.

The third assertion distinguishes this repair from the row-order tiebreaker an assistant would have reached for: it states that any two customers with the same frequency receive the same score, which is false under rank(method="first"). The bands are a declared rule and belong in the deliverable's written definitions — one order scores F1, two or three score F3, four or more score F5.

With scores in hand, the last step of Part A is the assignment that the opening case actually requires: every customer in exactly one named region, by a rule two analysts would apply identically.

Code 6.4. Assign every customer to a named grid region

```
# Table 6.3 as code: a complete lookup over all twenty-five cells.
RFM_GRID = {
    (5, 1): "New Customers",       (5, 2): "Potential Loyalists",
    (5, 3): "Loyal Customers",     (5, 4): "Champions",
    (5, 5): "Champions",
    (4, 1): "Potential Loyalists", (4, 2): "Potential Loyalists",
    (4, 3): "Loyal Customers",     (4, 4): "Champions",
    (4, 5): "Champions",
    (3, 1): "Needs Attention",     (3, 2): "Needs Attention",
    (3, 3): "Loyal Customers",     (3, 4): "Loyal Customers",
    (3, 5): "Loyal Customers",
    (2, 1): "Hibernating",         (2, 2): "Needs Attention",
    (2, 3): "At Risk",             (2, 4): "At Risk",
    (2, 5): "At Risk",
    (1, 1): "Hibernating",         (1, 2): "Hibernating",
    (1, 3): "At Risk",             (1, 4): "At Risk",
    (1, 5): "At Risk",
}

# Exhaustive and mutually exclusive, checked rather than asserted in prose.
assert len(RFM_GRID) == 25
assert set(RFM_GRID) == {(r, f) for r in range(1, 6)
                                for f in range(1, 6)}

cf["rfm_region"] = [RFM_GRID[(r, f)] for r, f in zip(cf["R"], cf["F"])]

assert cf["rfm_region"].notna().all()
assert len(cf["rfm_region"]) == len(cf)

print(cf[["R", "F", "M", "rfm_region"]]
      .sort_values(["R", "F"], ascending=False))
print(cf["rfm_region"].value_counts())
```

Expected output: C001 in Champions; C005, C008, and C003 in Loyal Customers; C009 in New Customers; C004 in Needs Attention; C002 and C010 At Risk; C006 and C007 Hibernating —

three Loyal Customers, two At Risk, two Hibernating, and one each in Champions, Needs Attention, and New Customers.

Input: the scored miniature. Transformation: a dictionary lookup on the pair of scores. Output: one region per customer, and two assertions that make the grid's structural promise executable — twenty-five cells defined, and every customer assigned. A lookup written this way cannot develop the gaps and overlaps a prose rule set develops, because the second assertion fails the moment a cell goes missing.

Read the result against the raw extract before moving on. C002, the high-basket occasion shopper, carries the largest monetary total in the table and lands in At Risk, because the regions are drawn on recency and frequency alone — Section 6.5's closing argument made visible on ten rows, and exactly the limitation that sends the chapter to clustering.

6.13.2 Lab 6.1, Part B: The Build at Full Scale

Part B repeats the build on the certified transactions file and the customers table, per Table 6.5's full derivation rules. The window and the analysis date are declared in the first cell, the population rule is declared with them, the ratio and intensity features join the core three, and the customer attributes join as profiling holdouts. An AI assistant may draft this build using the specification prompt in Section 6.12's AI in Practice box; the reconciliation, not the drafting, is the graded discipline. The cells assume transactions_clean and customers are loaded and that order_date has been parsed to a true date.

Code 6.5. Build the full customer-feature table

```
analysis_date = pd.Timestamp("2026-06-30")
window_start  = analysis_date - pd.DateOffset(months=24)

# Certified figures for the DECLARED WINDOW, copied from the Chapter 4
# verification log before anything is computed.
CERT_REVENUE    = ...   # total revenue in the window
CERT_ORDERS     = ...   # distinct orders in the window
CERT_PURCHASERS = ...   # distinct customers with an order in the window

window = transactions_clean[
    (transactions_clean["order_date"] > window_start)
    & (transactions_clean["order_date"] <= analysis_date)]

# Key integrity FIRST. groupby drops rows whose key is missing and
# nunique() ignores missing values, so a missing customer_id could pass
# a "<= 1 customer per order" check by counting zero.
required = ["order_id", "customer_id", "order_date",
            "line_revenue", "channel"]
assert window[required].notna().all().all(),     "required transaction fields
contain missing values"
assert customers["customer_id"].is_unique
assert window["customer_id"].isin(customers["customer_id"]).all()

# Each order must carry exactly one customer, one date, one channel --
# otherwise the "first" and "min" below would silently pick a value.
for col, msg in [("customer_id", "an order has zero or multiple customers"),
                 ("order_date", "an order spans multiple dates"),
                 ("channel", "an order spans multiple channels")]:
    assert (window.groupby("order_id")[col]
            .nunique(dropna=False).eq(1).all()), msg

orders = (window
          .groupby(["order_id", "customer_id"], as_index=False)
          .agg(order_date=("order_date", "min"),
               order_revenue=("line_revenue", "sum"),
               channel=("channel", "first")))

assert len(orders) == window["order_id"].nunique()
assert np.isclose(orders["order_revenue"].sum(),
                  window["line_revenue"].sum())

# Discounted revenue stays at line grain: an order can mix discounted
# and full-price lines, so this cannot be computed from order totals.
disc = (window
        .assign(discounted=lambda d: d["line_revenue"]
                .where(d["discount_pct"] > 0, 0.0))
        .groupby("customer_id")
        .agg(discounted_revenue=("discounted", "sum")))

# POPULATION RULE (Section 6.6): one row per customer with at least one
# order in the window. Everyone else is counted, named, and routed --
# not clustered, and not silently dropped.
cf_full = (orders
           .groupby("customer_id")
           .agg(last_order=("order_date", "max"),
                frequency=("order_id", "nunique"),
                monetary=("order_revenue", "sum"),
                store_orders=("channel", lambda s: (s == "Store").sum()))
           .join(disc)
```

```
            .join(customers.set_index("customer_id")[["signup_date"]]))

# The complement of the clustered population is NOT "never purchased".
window_purchasers  = set(window["customer_id"])
observed_purchasers = set(transactions_clean["customer_id"].dropna())

never_observed = customers[
    ~customers["customer_id"].isin(observed_purchasers)]
inactive_outside_window = customers[
    customers["customer_id"].isin(observed_purchasers)
    & ~customers["customer_id"].isin(window_purchasers)]

# The three populations partition the customers table exactly.
assert (len(cf_full) + len(never_observed)
        + len(inactive_outside_window) == len(customers))

print("purchasing customers (clustered):", len(cf_full))
print("customers with no purchase in the analysis window:",
      len(customers) - len(cf_full))
print("  no purchase in observed history -> activation program:",
      len(never_observed))
print("  purchased only before the window -> lapsed, win-back:",
      len(inactive_outside_window))
```

Expected output: a clustered row count equal to the certified distinct-purchaser count; three population counts that add back to the customers table; and every assertion passing.
Input: the certified line-grain file and the customers dimension. Transformation: the declared window is applied, keys are validated, lines are collected to orders, discounted revenue is summed at line grain, and orders are collected to customers. Output: the level features, the raw material for the ratios, and a three-way population count. The window filter is stated as an interval, open at the start and closed at the end, so that boundary orders belong to exactly one side; and the discount roll-up runs on lines rather than orders, because an order that mixes a discounted dress with a full-price scarf is one order and two different facts.

The key checks come before the grouping for a reason worth knowing. pandas drops rows whose grouping key is missing, and nunique() ignores missing values by default, so an order whose customer_id is missing would disappear from the grouped frame while a check written as nunique().le(1) counted zero and passed. Running notna() first, then asserting eq(1) with dropna=False, closes both doors. The date and channel checks close a third: the aggregation below takes the minimum date and the first channel, which would quietly manufacture a consistent order out of an inconsistent one.

The population arithmetic is the correction the opening case needs. Subtracting the feature table's row count from the customers table counts everyone with no purchase in this window, which is not the same population as customers who have never purchased: it also includes customers whose only orders predate the window. The two go to different places -- the first to the always-on activation program, the second to win-back inside the fall program -- so the cell separates them and asserts that the three counts partition the base. On a file holding only twenty-four months, even never_observed means no purchase in observed history rather than none ever, and the deliverable says so.

Code 6.6. Derive the ratio and intensity features, and reconcile

```
# Exposure is observed time inside the window, not calendar tenure, and
# it carries a declared floor: a customer observed for nine days cannot
# support a stable monthly rate.
cf_full["recency_days"] = (analysis_date
                           - cf_full["last_order"]).dt.days
observed_days = (analysis_date
                 - cf_full["signup_date"].clip(lower=window_start)).dt.days
cf_full["exposure_months"] = np.maximum(observed_days / 30.44, 1.0)

cf_full["orders_per_month"]  = (cf_full["frequency"]
                                / cf_full["exposure_months"])
cf_full["revenue_per_month"] = (cf_full["monetary"]
                                / cf_full["exposure_months"])
cf_full["aov"]            = cf_full["monetary"] / cf_full["frequency"]
cf_full["discount_share"] = (cf_full["discounted_revenue"]
                             / cf_full["monetary"])
cf_full["store_share"]    = (cf_full["store_orders"]
                             / cf_full["frequency"])
cf_full = cf_full.drop(columns=["last_order", "signup_date",
                                "store_orders", "discounted_revenue"])

# Profiling holdouts join last, and are never clustered on.
attrs = customers.assign(
    tenure_days=lambda d: (analysis_date - d["signup_date"]).dt.days)
holdouts = ["tenure_days", "loyalty_tier", "app_user", "email_opt_in",
            "home_metro", "acquisition_channel", "age_band"]
cf_full = cf_full.join(attrs.set_index("customer_id")[holdouts])

# The three certified equalities of Section 6.6, plus the edge cases.
assert len(cf_full) == CERT_PURCHASERS
assert cf_full["frequency"].sum() == CERT_ORDERS
assert np.isclose(cf_full["monetary"].sum(), CERT_REVENUE, atol=0.01)
assert (cf_full["recency_days"] >= 0).all()
assert cf_full[["aov", "discount_share", "store_share",
                "orders_per_month",
                "revenue_per_month"]].notna().all().all()
# Shares summed from rounded line revenue need a tolerance, not a bare 1.
tol = 1e-9
assert cf_full["discount_share"].between(-tol, 1 + tol).all()
assert cf_full["store_share"].between(-tol, 1 + tol).all()
assert (cf_full["exposure_months"] >= 1.0).all()
assert np.allclose(cf_full.loc[cf_full["frequency"] == 1, "aov"],
                   cf_full.loc[cf_full["frequency"] == 1, "monetary"])

print(cf_full[["recency_days", "frequency", "monetary", "aov",
               "orders_per_month", "revenue_per_month",
               "discount_share", "store_share"]].describe().round(2))
```

Expected output: an eight-column summary with no missing values, minimum recency of 0 or more, shares bounded by 0 and 1, and nine assertions passing.

Input: the aggregated customer table. Transformation: recency from the declared analysis date, exposure from the later of signup and the window start, the two intensity rates, and the three shape ratios. Output: the completed clustering inputs plus the holdouts. Note what exposure_months does and does not do. It uses signup date — the very column Table 6.5 holds

out — as a denominator, deliberately: dividing by exposure reduces tenure's mechanical effect on lifetime totals and puts it on screen beside them, while clustering on tenure is how that same effect gets rediscovered and mistaken for a finding (Section 6.6). It does not remove the effect, because frequency and monetary remain in the specification and remain bounded by exposure. The floor is applied to the months, not to the days, so the minimum exposure is exactly 1.0 month and the assertion says so — an earlier draft floored the day count at 30 and produced a minimum of 0.986, which is the kind of quiet discrepancy between a stated rule and its implementation this chapter exists to catch.

The three certified equalities are the chapter's known-totals check in feature-table form, and the row-count equality is also the population statement: the feature table is smaller than the customers table by exactly the no-purchase-in-window count printed by Code 6.5, and that difference is reported, split, and named rather than absorbed.

VERIFICATION CHECK

Before you run: copy the three certified figures for the declared window from your Chapter 4 verification log into the notebook as constants, and predict in writing that the feature table's row count equals the certified purchaser count and is smaller than the customers table's row count — then state why in one sentence. Predict two more structural facts: recency_days has no negative values, and every customer with frequency = 1 has aov equal to monetary exactly.

After you run: all nine assertions pass, Code 6.5's three population counts add back to the customers table's row count, the minimum exposure_months is exactly 1.0, and the summary shows no missing values in any clustering input.

Investigate if: your finished table disagrees with the shipped customer_features answer key on any shared column. A merged comparison should agree everywhere; a disagreement is a derivation-rule difference to locate and name, not to average away. Check the window boundary first — it is the most common source.

Part B closes with the full-scale grid, which reuses Code 6.4's lookup unchanged. That reuse is the point: the assignment rule is the same object at ten customers and at eight thousand.

Code 6.7. Score and read the full-scale RFM grid

```
# Recency and monetary value support quintiles at scale; frequency
# still does not, so the banded rule of Code 6.3 carries over unchanged.
cf_full["R"] = pd.qcut(cf_full["recency_days"].rank(method="average"),
                       5, labels=[5, 4, 3, 2, 1]).astype(int)
cf_full["M"] = pd.qcut(cf_full["monetary"].rank(method="average"),
                       5, labels=[1, 2, 3, 4, 5]).astype(int)
cf_full["F"] = (pd.cut(cf_full["frequency"], bins=freq_bands)
                  .map(band_score).astype(int))

cf_full["rfm_region"] = [RFM_GRID[(r, f)]
                         for r, f in zip(cf_full["R"], cf_full["F"])]

# Every customer assigned, exactly once.
assert cf_full["rfm_region"].notna().all()
assert len(cf_full["rfm_region"]) == len(cf_full)

grid = (cf_full.groupby("rfm_region")
        .agg(customers=("monetary", "size"),
             revenue=("monetary", "sum"))
        .assign(customer_share=lambda d: d["customers"]
                / d["customers"].sum(),
                revenue_share=lambda d: d["revenue"]
                / d["revenue"].sum()))

assert np.isclose(grid["revenue"].sum(), CERT_REVENUE, atol=0.01)
assert grid["customers"].sum() == len(cf_full)

print(grid.round(3))
```

Expected output: one row per occupied region, customer and revenue shares each summing to 1.000, and the region revenues reconciling to the certified window total.

Input: the full feature table. Transformation: the declared scoring rule and the twenty-five-cell lookup. Output: the named grid with sizes and revenue shares. The two assertions restate the exhaustiveness promise at scale and add the reconciliation that makes the revenue shares readable: a grid whose revenue does not add back to the certified window total has lost customers somewhere between the feature table and the summary.

Close Part B by writing three insight statements per Section 5.9 from the grid — at least one of which must name something the grid cannot see, rehearsing Section 6.5's closing argument and setting up Lab 6.2.

6.13.3 Lab 6.2: Discovery, Validation, and the Answer Key

Lab 6.2 clusters. The inputs are the eight behavioral features of Table 6.5; the holdouts stay held out; and the lab's structure is the chapter's argument in code: scale, sweep, choose, profile, validate, test the specification, compare, and deploy.

Code 6.8. Standardize the clustering features

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, adjusted_rand_score

features = ["recency_days", "frequency", "monetary",
            "orders_per_month", "revenue_per_month",
            "aov", "discount_share", "store_share"]

# The scaler is fitted once and KEPT. Section 6.10 needs this object.
scaler = StandardScaler()
X = scaler.fit_transform(cf_full[features])

assert np.allclose(X.mean(axis=0), 0, atol=1e-9)
assert np.allclose(X.std(axis=0), 1, atol=1e-9)

print(np.round(X.mean(axis=0), 6))   # eight zeros
print(np.round(X.std(axis=0), 6))    # eight ones
print(pd.DataFrame(X, columns=features).corr().round(2))
```

Expected output: eight zeros, eight ones, and a correlation matrix in which frequency and monetary are strongly related to each other and to their per-month counterparts.
Input: the eight clustering features. Transformation: z-scores. Output: the scaled matrix, plus the two facts that define a z-score, printed rather than assumed. The correlation matrix is not decoration. It is the visible form of Section 6.7's last argument: standardization has equalized the eight features' scales and done nothing whatever about the fact that several of them carry the same information. Code 6.13 measures what that costs.

A useful side computation, echoing Section 6.7: pick one high-monetary and one high-frequency customer and confirm that scaling changed which of their differences dominates the distance between them.

Code 6.9. Evaluate candidate values of k

```
sweep = []
for k in range(2, 9):
    km = KMeans(n_clusters=k, n_init=10, random_state=42).fit(X)
    sweep.append((k, km.inertia_, silhouette_score(X, km.labels_)))

sweep = pd.DataFrame(sweep,
                     columns=["k", "inertia", "mean_silhouette"])

# Inertia should be non-increasing in a well-optimized sweep. A rise is a
# WARNING to inspect, not proof of a bug: each fit is a separate
# optimization and can settle into a worse local minimum.
if not sweep["inertia"].is_monotonic_decreasing:
    print("WARNING: inertia rose; refit with more starts before "
          "reading the elbow.")

print(sweep.round(3))
```

Expected output: seven rows; inertia falling steeply from k = 2 and then flattening; mean silhouette values that do not necessarily peak where the elbow bends.

Input: the scaled matrix. Transformation: seven independent fits. Output: the two instruments of Section 6.9, side by side. Note the diagnostic rather than the assertion. An earlier draft of this lab asserted that inertia must strictly decrease; that is too strong on both counts. At the optimum, inertia is non-increasing rather than strictly decreasing — an added centroid that buys no improvement leaves it flat — and in practice a higher-k fit can land in a worse local minimum than a lower-k fit, which produces a rise that means "inspect and refit," not "the code is broken."

Then apply the braid. Shortlist the elbow's neighborhood and the silhouette's nominees, and carry the shortlist — not a single winner — into profiling. The choice of k = 4 for the cells that follow must be argued in a short markdown cell citing all three legs of Table 6.6, including the four-journey actionability constraint. "The answer key has four" is not one of the legs, and the cell should read as if no answer key existed.

VERIFICATION CHECK

Before you run: predict that inertia will be non-increasing across the sweep and say why in one sentence. Then predict which k the silhouette will crown, knowing Section 6.9's warning and Chapter 5's evidence of one dominant urban-suburban divide: the defensible prediction is that a small k scores at or near the top even though the designed structure is finer. Record the prediction before running, because the point of the exercise is not to be right but to notice when the instrument disagrees with the strategy.

After you run: the inertia column falls and flattens; the silhouette column has a maximum you can name; and the two instruments nominate either the same k or a shortlist of two or three.

Investigate if: the silhouette's winner is k = 2 and you were about to deploy it. Two segments cannot carry the discount-dependence and lifecycle distinctions the treatments exist to exploit, so the silhouette's verdict is evidence about geometry that judgment is entitled to overrule — in writing, with the reason stated.

Code 6.10. Fit and profile the selected solution

```
K = 4
model = KMeans(n_clusters=K, n_init=10, random_state=42)
labels = model.fit_predict(X)
cf_full["cluster"] = labels

assert len(np.unique(labels)) == K

centroids_std = pd.DataFrame(model.cluster_centers_, columns=features)
centroids_std.index.name = "cluster"

# A centroid IS the mean of its cluster, up to the optimizer's
# convergence tolerance. This is the audit of Section 6.12, executed.
assert np.allclose(centroids_std.values,
                   scaler.transform(
                       cf_full.groupby("cluster")[features].mean()),
                   atol=0.01)

native = (cf_full.groupby("cluster")[features]
          .agg(["median", "mean"]).round(2))
sizes = cf_full["cluster"].value_counts().sort_index()

assert sizes.sum() == len(cf_full)

print(sizes)                      # sizes first, per Section 5.6
print(centroids_std.round(2))     # what defines each cluster
print(native)                     # what to say in the meeting
```

Expected output: four cluster sizes summing to the feature table's row count; a centroid table in standard-deviation units; and a native table carrying medians beside means for every feature.
Input: the scaled matrix. Transformation: one fit at the chosen k. Output: the profile in both vocabularies. Read the standardized table first to see what defines each cluster — a value of −0.8 on recency_days means the cluster's members bought more recently than average by four fifths of a standard deviation — then read the native table for the deck, because standardized units are for analysts and days and dollars are for decisions.

Medians travel beside means for the reason Section 5.4 established: a cluster's monetary centroid can be propped by a handful of whales, and a mean its own median contradicts is a warning that the cluster is not one story. Before touching the holdouts, write the predictions — for each cluster, one sentence on how it should differ on loyalty_tier, app_user, home_metro, acquisition_channel, and tenure_days if its behavioral profile means what it appears to mean.

Code 6.11. Test seeded and subsample stability

```
# Each fit below is the best of ten internal initializations, so these
# are five INDEPENDENTLY SEEDED FITS, not five random starting points.
seed_ari = {}
for seed in [7, 21, 99, 2026, 31337]:
    alt = KMeans(n_clusters=K, n_init=10,
                 random_state=seed).fit_predict(X)
    seed_ari[seed] = adjusted_rand_score(labels, alt)

# Subsample refits: 80 percent of customers, compared on the customers
# both solutions saw. The index is relabeling-invariant, so no alignment
# step is needed and no dominant-cell counting can mislead.
sub_ari = {}
rng = np.random.default_rng(42)
for i in range(2):
    idx = rng.choice(len(X), size=int(0.8 * len(X)), replace=False)
    sub = KMeans(n_clusters=K, n_init=10,
                 random_state=42).fit_predict(X[idx])
    sub_ari[f"subsample_{i + 1}"] = adjusted_rand_score(labels[idx], sub)

print("seeded ARI:  ", {s: round(v, 3) for s, v in seed_ari.items()})
print("subsample ARI:", {s: round(v, 3) for s, v in sub_ari.items()})
print("minimum observed:",
      round(min(list(seed_ari.values()) + list(sub_ari.values())), 3))
```

Expected output: seven indices no greater than 1.0, with a stable solution close to 1.0 throughout; values near 0 would indicate chance-level agreement and negative values worse-than-chance agreement.

Input: the scaled matrix and the baseline labels. Transformation: seven refits under perturbation. Output: seven agreement scores. The adjusted Rand index asks, over all pairs of customers, how often two solutions agree about whether the pair belongs together, corrected for the agreement chance would produce (Hubert & Arabie, 1985). It is 1.0 for identical partitions and near 0.0 for unrelated ones, and it can go negative — to a lower bound of −0.5 — when two partitions agree less than chance would predict, which is why the expected output states a ceiling rather than a range (Pedregosa et al., 2011).

The index replaces the procedure most practitioner treatments use — cross-tabulate the two runs and count the customers outside each row's largest cell — and the replacement is not cosmetic: that procedure assumes the runs correspond one to one, and two baseline clusters can both have their largest overlap with the same cluster in the refit, reporting a stability that does not exist. Where a customer-level reassignment rate is wanted for the deck, align the labels first with an explicit one-to-one assignment, then count, and report the index alongside it. In the deck's plain language: “we refit from five independent seeds and on two random 80 percent subsets; the partitions agreed at an adjusted Rand index of at least [value].” A solution that holds its shape has earned Section 6.10's adjective; one that churns has not, whatever its silhouette said.

Code 6.12. Profile the withheld attributes

```
hold_num = (cf_full.groupby("cluster")
            .agg(customers=("tenure_days", "size"),
                 tenure_days_median=("tenure_days", "median"),
                 app_user=("app_user", "mean"),
                 email_opt_in=("email_opt_in", "mean")))

assert hold_num["customers"].sum() == len(cf_full)
print(hold_num.round(3))

# crosstab drops missing values silently and the rows still sum to 1.0,
# so the assertion would pass while hiding a real difference. age_band is
# partly missing by design (Table 6.1), and missingness can itself differ
# across clusters -- so it is profiled as a category, not discarded.
for col in ["home_metro", "loyalty_tier",
            "acquisition_channel", "age_band"]:
    profiled = cf_full[col].astype("string").fillna("Missing")
    tab = pd.crosstab(cf_full["cluster"], profiled, normalize="index")
    assert np.allclose(tab.sum(axis=1), 1.0)
    print(col)
    print(tab.round(3))
```

Expected output: a numeric holdout table with one row per cluster, then four row-normalized cross-tabulations whose rows each sum to 1.000, with an explicit Missing column wherever the attribute is incomplete.

Input: the clustered feature table. Transformation: group by cluster, summarize the columns the algorithm never saw. Output: the holdout profile. Row-normalize rather than cell-normalize, per Section 5.5's direction rule: the comparison being made is across clusters, so the percentages run within each cluster. Filling missing values with an explicit Missing category matters more here than anywhere else in the chapter. Dropping them leaves every row summing to 1.000 and the assertion passing while a cluster that is half unknown on age band looks exactly like one that is fully observed — and in a holdout check, systematically different missingness is itself a finding.

Now grade the predictions you wrote before Code 6.11 ran, and grade them honestly in both directions. Confirmed predictions are one layer of external-variable evidence, and they are worth a sentence in the deck. They are not proof that the clusters are natural kinds: the holdout attributes are correlated with the clustering inputs, so agreement is convergent rather than independent (Section 6.10). And nothing here speaks to differential response, which is a causal claim no clustering can support. Failed predictions are findings: a cluster you expected to be loyalty-enrolled and that turns out not to be has just told you the behavioral features were incomplete.

Code 6.13. Test feature-set sensitivity

```
# Standardization equalizes scale; it does not remove redundancy.
# aov is monetary / frequency, and each per-month rate is a level
# feature / exposure. This cell measures what those dependencies cost.
SPECS = {
    "base (8 features)": features,
    "without aov": [f for f in features if f != "aov"],
    "without level features":
        [f for f in features if f not in ("frequency", "monetary")],
    "without intensity features":
        [f for f in features if not f.endswith("_per_month")],
}

sens = {}
for name, cols in SPECS.items():
    Xs = StandardScaler().fit_transform(cf_full[cols])
    lab = KMeans(n_clusters=K, n_init=10,
                 random_state=42).fit_predict(Xs)
    sens[name] = adjusted_rand_score(labels, lab)

assert np.isclose(sens["base (8 features)"], 1.0)

for name, v in sens.items():
    print(f"{name:<28} ARI vs base: {v:.3f}")
```

Expected output: 1.000 for the base specification against itself, and three values at or below 1.000 for the alternatives; an alternative may also reach 1.000 if removing those features leaves every assignment unchanged.

Input: the feature table. Transformation: three refits on reduced feature sets, each compared with the base solution by adjusted Rand index. Output: a sensitivity report. The first line is a control: a specification compared with itself must return exactly 1.0, and if it does not, the comparison is broken before any conclusion is drawn.

The purpose is not to select a winner automatically but to make visible that feature selection is part of the definition of distance, and therefore of "alike." If dropping aov moves a large share of customers, aov was carrying real weight and its inclusion is a modeling decision to be defended rather than defaulted; if it moves almost nobody, the composite was redundant and the simpler specification is easier to govern. Either way the honest deliverable reports the comparison, not the winner.

Code 6.14. Compare with the designed answer key

```
designed = (customers.set_index("customer_id")["segment_designed"]
            .reindex(cf_full.index))
assert designed.notna().all(), "answer key misses some customers"

key = pd.crosstab(designed, cf_full["cluster"])
assert key.values.sum() == len(cf_full)

print(key)
print("ARI against the designed segments:",
      round(adjusted_rand_score(designed, labels), 3))
```

Expected output: a four-by-four cross-tabulation whose cells sum to the feature table's row count, and a single agreement index.

Input: the clustered table and the shipped answer key. Transformation: one cross-tabulation and one index. Output: the comparison only a designed dataset allows. Read the cross-tabulation the way Section 6.10 taught: because cluster numbering is arbitrary, the signature of strong recovery is a permutation pattern — each designed segment concentrated in one cluster column — rather than a diagonal, and the index summarizes the pattern without requiring the labels to line up.

The off-diagonal customers are the finding, not the failure. The specification planted four segments — Urban Loyal Core, Suburban Occasion, Discount Trend Follower, and New/At-Risk — separable on behavioral structure but with deliberate overlap, so recovery requires judgment rather than falling out automatically. Profile a handful of the misassigned customers individually: that is what "the boundaries between segments are judgment calls" looks like one customer at a time.

Code 6.15. Build the segment-attractiveness matrix

```
NAMES = {0: "...", 1: "...", 2: "...", 3: "..."}   # after profiling
cf_full["segment"] = cf_full["cluster"].map(NAMES)
assert cf_full["segment"].notna().all()

# Table 6.8's row is DISCOUNT SHARE OF REVENUE -- a pooled ratio. The
# mean of customer-level shares would weight every customer equally and
# answer a different question, so discounted revenue is rebuilt and the
# ratio is taken on the totals (the averaged-average rule of Section 5.3).
cf_full["discounted_revenue"] = (cf_full["monetary"]
                                 * cf_full["discount_share"])

attract = (cf_full.groupby("segment")
           .agg(customers=("monetary", "size"),
                revenue=("monetary", "sum"),
                discounted_revenue=("discounted_revenue", "sum"),
                revenue_per_customer=("monetary", "mean"),
                revenue_per_exposure_month=("revenue_per_month", "mean"),
                median_tenure_days=("tenure_days", "median"),
                app_reachable=("app_user", "mean"),
                email_reachable=("email_opt_in", "mean")))
attract["discount_share_of_revenue"] = (attract["discounted_revenue"]
                                        / attract["revenue"])
attract["customer_share"] = (attract["customers"]
                             / attract["customers"].sum())
attract["revenue_share"] = attract["revenue"] / attract["revenue"].sum()

# The matrix is a partition of the base and of the window's revenue.
assert attract["customers"].sum() == len(cf_full)
assert np.isclose(attract["revenue"].sum(), CERT_REVENUE, atol=0.01)
assert np.isclose(attract["customer_share"].sum(), 1.0)

print(attract.round(3))
```

Expected output: one row per named segment, customer and revenue shares each summing to 1.000, revenue reconciling to the certified window total, and a discount share of revenue that differs from the average of the segment's customer-level shares.

Input: the named, clustered table. Transformation: one grouped summary. Output: the profile-arithmetic rows of Table 6.8. Two figures deserve attention. revenue_per_exposure_month is the size-adjusted companion to revenue_per_customer, and comparing the two is how a lifecycle segment stops being mistaken for a weak one: a segment can trail badly on lifetime revenue and lead on revenue per month of exposure, which is a different sentence with a different budget attached. And discount_share_of_revenue is computed as a pooled ratio rather than as the mean of customer-level shares, because Table 6.8 names it a share of revenue: averaging the customer shares gives a thousand-dollar customer and a forty-dollar customer equal votes, which is Section 5.3's averaged average wearing a margin costume. It stands in for margin quality until finance supplies real unit costs, and it is labeled a proxy in the deck for exactly that reason.

Two rows of Table 6.8 are deliberately absent from this cell. The same-age trajectory is evidence rather than judgment, but it needs the period axis of a cohort table and must be merged in from the Chapter 5 analysis; leaving it blank here and filling it there is the honest arrangement,

and forgetting it is how a lifecycle segment gets judged on lifetime totals after all. Cost to serve, strategic fit, the evidence-versus-judgment declaration, and the recommended priority are not computable at all: write them by hand, mark them as judgment, and defend them in the meeting, because a matrix that silently mixes measured cells with asserted ones is the failure that declaration exists to prevent.

Code 6.16. Save and apply the operating assignment model

```
# Deployment needs the fitted objects, not thresholds read off a table.
SEGMENTATION = {
    "version": "2026-fall-v1",
    "analysis_date": analysis_date,
    "window_start": window_start,
    "population": "customers with >= 1 order in the window",
    "features": features,
    "scaler": scaler,          # the fitted means and standard deviations
    "model": model,            # the fitted centroids
    "names": NAMES,
}

def assign_segments(new_features, spec=SEGMENTATION):
    """Assign with the FITTED scaler and model. Never refit here:
    refitting would silently redraw the segments."""
    missing = [f for f in spec["features"]
               if f not in new_features.columns]
    assert not missing, f"missing features: {missing}"
    Xn = spec["scaler"].transform(new_features[spec["features"]])
    return pd.Series([spec["names"][c]
                      for c in spec["model"].predict(Xn)],
                     index=new_features.index, name="segment")

# Reproducing the fitted assignments is the deployment check.
assert (assign_segments(cf_full) == cf_full["segment"]).all()
sample = cf_full.sample(500, random_state=1)
assert (assign_segments(sample) == sample["segment"]).all()

# An ILLUSTRATIVE CANDIDATE rule for the campaign platform, written
# after profiling. Branches return SEGMENT NAMES, never NAMES[i]:
# cluster numbers are arbitrary, so a discount branch has no reason to
# land on cluster 0. Whether this rule is deployable is decided by the
# agreement it measures, not by the fact that it was written.
rule = np.where(cf_full["discount_share"] > 0.50, "Discount-Dependent",
        np.where(cf_full["store_share"] > 0.60, "Suburban Occasion",
         np.where(cf_full["orders_per_month"] > 0.50, "Urban Loyal Core",
                  "New / At-Risk")))
assert set(np.unique(rule)) <= set(NAMES.values())

agreement = (rule == cf_full["segment"]).mean()
print(f"candidate rule agrees with the model on {agreement:.1%} "
      "of customers")
print(pd.crosstab(cf_full["segment"], rule))
```

Expected output: both deployment assertions passing, one agreement percentage, and a cross-tabulation locating every customer on whom the rule and the model disagree.

Input: the fitted scaler, the fitted model, and the feature table. Transformation: a saved specification and an assignment function that transforms with the saved scaler and predicts with the saved centroids. Output: reproducible assignments and an honest account of the approximation the campaign platform will run.

The two deployment assertions are the whole point of the first half of the cell. The first says the saved objects reproduce the fitted assignments exactly; the second says they do so for an arbitrary subset, which is what assigning next month's new customers amounts to. Neither would pass if the function refitted — and refitting is exactly what a well-meaning engineer will do when the notebook is handed over without the objects.

The threshold rule in the second half is a candidate, not a conclusion, and two things about it are easy to get wrong. Its branches return segment names rather than NAMES[0] and NAMES[1], because cluster numbers carry no meaning: the cluster that turns out to be discount-dependent may be numbered 2, and a rule that maps a discount branch onto cluster 0 by construction is measuring its own indexing. And the agreement figure is the test, not the footnote. SQL is what the campaign platform speaks, so a simplified rule is often necessary; but if the measured agreement comes back low, the correct conclusion is that this rule is not an acceptable surrogate for the model, and the response is to find better thresholds, add a branch, or push the platform to consume scored assignments rather than recompute them. Publish the number beside the rule and keep the cross-tabulation: the customers in its off-diagonal cells are the ones who would receive the wrong journey.

The lab's final deliverable is the CRM manager's one-page treatment map per Section 6.11: four named segments with their written assignment rules and the saved model that those rules approximate, a journey and a priority for each, the customers with no purchase in the window split between activation and win-back per Code 6.5, the evidentiary basis in one line each, and the validation results — stability indices, feature-set sensitivity, and holdout agreement — as the map's footer. The sign-off meeting's first question is “why should we believe these four groups are real?”, and the footer is the answer. Deliverables for both labs: the notebook, the reconciliation log, the written scoring and population rules, the saved segmentation specification, the treatment map, and the AI-Use Appendix (Appendix D) documenting every assistant exchange, including at least one delegated step you corrected and why.

6.14 Marketing Interpretation and Managerial Insight

The lab's outputs are tables; the sign-off meeting runs on sentences. This section translates, and — per this guide's standing practice — it does so partly by exhibiting a wrong managerial reading and correcting it, because segmentation's characteristic misreadings are more expensive than description's: a misread summary misinforms a meeting, while a misread segmentation gets built into journeys, budgets, and triggers that run automatically for a year.

The wrong reading, and it will be offered in the meeting by someone reasonable: “Cluster 4 — the New/At-Risk group — has the lowest monetary value, the lowest frequency, and the weakest repeat behavior of the four segments. It is our worst-performing segment. The efficient

move is to zero out its journey and reallocate the fourth slot's budget to Champions.” Every clause before the conclusion is true, and the conclusion is a trap with two jaws. The first jaw is arithmetic: the segment's lifetime totals are bounded by exposure. Many of its members are StyleCraft's most recent signups, and a customer nine weeks old cannot have accumulated two years of orders or two years of spend, so “lowest monetary and frequency” is partly what recency of arrival looks like at the customer grain — the tenure trap that Section 6.6 mitigated by adding the intensity features beside the lifetime totals rather than removing it, and that Chapter 5's cohort discipline exists to catch. The lab's own attractiveness matrix supplies the test: compare revenue per exposure-month beside revenue per customer, and compare the segment's early trajectory against older cohorts at the same age, per a Table 5.7-style cohort view. Lifetime totals do not answer the performance question; they restate the tenure question.

The second jaw is compositional. The newest customers concentrate, mechanically, wherever acquisition is newest — including the Wave 4 stores of the expansion. A treatment decision that abandons “low-value” new customers is, in substantial part, a decision to abandon the newest markets for being new, reached without ever mentioning them. And there is a third thing the wrong reading misses, which the segment's name has been carrying since Section 6.6: this cluster is a lifecycle position rather than a customer type, and the position it names is occupied by two strategically opposite populations. Genuinely new customers have unknown future value and a clear treatment — onboard them. Long-tenured customers who have gone quiet have known past value and a different treatment — win them back. The corrected reading holds all three points at once: “Cluster 4 is a lifecycle stage, not a quality verdict. Its observed value is low partly because its observed time is short; its members split into a new population and a lapsed one, which the journey must separate on tenure before it sends a single message — a branch inside the segment's one slot, not a fifth program; and its performance metric is same-age cohort progression and revenue per exposure-month, not lifetime spend.”

The corrected reading generalizes into the section's positive discipline: a segment profile supports sentences about what each group is and does, and it disciplines sentences about what to do. The treatment map earns managerial language when each row follows the insight-statement form of Section 5.9 — comparison, magnitude in decision units, connection to the named decision, verbs that stop at the evidence. “The Urban Loyal Core is roughly [share] of purchasing customers and [share] of revenue in the window, reaches near-fully through app and email, and is the base's retention engine — its journey should reward and protect, and discounting into it spends margin on behavior we already receive” survives scrutiny. “Targeting Champions will grow revenue 15 percent” does not — no analysis in this chapter measures what any treatment causes, and crossing that boundary requires the experimental evidence Part II reaches later. Between that evidence and this chapter stands a nearer question the meeting will also ask: what actually moves these customers — which levers, at what strength? Segments describe who; the drivers question is the next chapter's, and the treatment map should say so rather than improvise.

One more translation deserves its paragraph, because it converts this chapter's most technical work into its most persuasive slide: the validation footer. “We refit the analysis from five

independent seeds and on two random 80 percent subsets of customers; the partitions agreed at an adjusted Rand index of at least [value], where 1.0 means identical partitions, 0 means chance agreement, and a negative value would mean worse than chance" is a sentence a non-technical executive can weigh, and so is "we built the groups from purchasing behavior only, then checked attributes the algorithm never saw." State the footer's limits in the same breath, because the limits are what make the rest of it credible: stability establishes that the partition is reproducible, not that it is meaningful; holdout agreement establishes that the groups differ on attributes beyond their inputs, not that they are natural types or that they will respond differently to treatment. The honest headline is therefore not "the customer base contains four kinds of customers" but "these four groups are reproducible, interpretable, and different enough on attributes we withheld to be worth treating differently — and the first year of treatment is how we find out whether they respond differently too." That sentence is longer, and it is the one that survives the follow-up question.

6.15 Business Analytics in Practice

This section turns from the fictional case to how segmentation operates in industry — where it is among the oldest analytic disciplines in marketing and among the most routinely fumbled, usually at the joint between the statistics and the organization. The three vignettes below are composite: they synthesize recurring patterns in retail, financial services, and consumer marketing teams rather than reporting on any one named organization, and they are offered as situations a new analyst will recognize rather than as claims about how often they occur.

6.15.1 Nine Clusters, Three Treatments

The first vignette concerns a large retail bank. The bank's analytics group, well staffed and technically strong, delivered a nine-cluster segmentation of its retail customer base: statistically defensible, stable under refit, beautifully profiled, each cluster with a name, a persona sheet, and a deck of its own. The deployment meeting killed it in an hour, for a reason no silhouette score could see. The bank's CRM and campaign infrastructure could execute exactly three differentiated treatment programs — three creative streams, three offer structures, three measurement frames — and no budget existed to change that within the planning year. Nine segments mapped onto three treatments is, operationally, three segments with six ghosts.

The rework that followed inverted the process. The treatment capacity was written into the analytic specification as a design input, the clustering was rerun and evaluated at small k, and the deliverable became three coarse, deployable segments each with genuinely different economics — a solution the first project's authors would have called statistically timid and the campaign calendar called usable. The practice lesson is Sections 6.2 and 6.11 fused: actionability is not the last criterion applied to a finished segmentation; in mature organizations it is the first constraint written into the specification, and the number of treatments the organization can execute is as binding as any property of the data.

6.15.2 RFM as a Production System

The second vignette is RFM as a production system rather than an analysis. Tiered loyalty programs in fashion, beauty, and specialty retail run on machinery any reader of Section 6.5 would recognize: customers are scored on recency, frequency, and spend on an automated cadence; tier assignment and tier movement are rule-driven consequences of the scores; and the scores fire triggers — a recency threshold crossed without a purchase launches the win-back sequence, a frequency milestone unlocks the reward, a lapsed high-monetary customer routes to the retention offer with the richest allowable incentive.

Practitioners who operate these systems describe the analytic work as largely governance rather than modeling. The scoring definitions are metric definitions in exactly Section 3.5.1's sense — owned, documented, and versioned — because a silent change to the recency window reassigns thousands of customers overnight and quietly rewrites the meaning of every downstream report. Threshold changes are proposed, impact-sized against a trailing window, and change-logged in the manner Section 5.13.3 described for funnel definitions. And the recurring operational failures are measurement failures — a timezone bug in last-purchase dates, a returns policy that made monetary value negative for edge-case customers — rather than statistical ones.

The same governance applies to the model-based segmentations of Sections 6.7 through 6.10, and the discipline there is worth spelling out because it is where new analysts are most often surprised. A production segmentation is a versioned release: the feature definitions, the fitted scaler, the fitted centroids, and the population rule ship together under a version number, and new customers are scored against that release rather than against a fresh fit. The release is refreshed on a declared cadence rather than continuously, because a segmentation that silently redraws itself every night makes every trend chart that uses it uninterpretable. Between refreshes the assignments are monitored for drift: the share of customers landing in each segment is tracked, and a material move triggers investigation before it triggers a refit, since the usual cause is an upstream data change rather than a change in customers. And a release is eventually retired — redrawn on new features, or replaced by a scheme keyed to a new strategy — with a documented mapping from old segments to new, because every dashboard, trigger, and target that referenced the old names has to be moved. The lesson reinforces this chapter's a priori road and extends Chapter 3's closing thread: an operating segmentation is a set of written definitions and fitted objects running on a schedule, and whoever governs them governs the program.

6.15.3 Personas and Statistical Segments

The third vignette concerns the reconciliation problem the opening case staged: personas versus statistical segments. Consumer organizations of any size frequently hold both — a brand or insights team maintains qualitative personas, vivid and motivating, while an analytics team maintains data-driven segments, measurable and operable — and the default relationship between the two artifacts is mutual disregard, with creative teams planning against characters no database can find and CRM teams targeting groups no storyteller can love. The strategy literature has documented the failure at the persona end: segmentations built for messaging drift toward

demographic and lifestyle portraits that neither predict behavior nor guide the decisions the segmentation was commissioned to serve (Yankelovich & Meer, 2006).

The reconciliation practice that mature organizations converge on runs through exactly this chapter's machinery: treat the personas as hypotheses, profile the statistical segments on every attribute the persona narratives invoke, and test the mapping. Sometimes a persona turns out to be a recognizable region of a real segment and inherits its size, value, and reachability; sometimes a beloved persona maps to nobody, which is a finding the focus group could never have produced. Where the mapping holds, the persona survives as the communication layer — the name and face the creative team writes to — wrapped around a measured membership the CRM team can actually target; where it fails, one artifact yields. The lesson is Section 6.10's naming discipline at organizational scale: stories and statistics are reconciled by profiling, not by seniority.

6.15.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into a single expectation: the segmentation work you will actually be asked to defend is rarely the clustering. It is the specification that wrote treatment capacity in as a constraint, the definitions document that lets the scoring run unattended for a year, the versioned release that lets next month's customers be assigned without redrawing last month's, the profile that told the brand team which of their personas the data could find, and the validation footer that let a vice president believe four groups were worth four budgets. The statistics in this chapter are learnable in a week; the habit that distinguishes careers is refusing to call any partition a segmentation until a treatment, a definition, and a validation are attached — because a segmentation succeeds when the marketing calendar changes, and by no other test.

6.16 Ethics, Proxy Discrimination and Differential Treatment

The Business Analytics in Practice section described segmentation as the analytics that changes calendars. This section examines the ethical weight of exactly that power, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), and honest summarization (Section 5.14) — to the act this chapter adds: sorting people into groups in order to treat them differently. Differential treatment is not a side effect of segmentation; Section 6.2 defined it as the point. That is why this chapter's ethics cannot be a caution appended to a technique. The technique is the ethical object.

Begin with proxies, because clustering manufactures them by construction. StyleCraft's schema contains no race, religion, health, or income column, and it is tempting to conclude that its segmentations therefore cannot discriminate on such attributes. The conclusion is false, and the mechanism deserves to be understood rather than feared. Attributes are correlated: geography encodes income and, in most American metros, race; age band encodes life stage; channel behavior and device ownership encode affluence and age at once. A segmentation built on

innocent columns can therefore sort customers by a sensitive attribute no column names — the disparate-impact mechanism that Barocas and Selbst (2016) show can arise unintentionally, through historical data, inherited patterns, and proxies, rather than only through malice, and that Section 1.13 introduced as a standing concern of responsible analytics. It is a well-documented risk in data-driven decision systems, and treating it as one is the difference between an audit and an alibi.

Two consequences follow for practice. Dropping a sensitive column is not a defense: the information usually survives in its correlates, so “we never used age” and “our treatment does not vary by age” are separate empirical claims, and only the second matters. And the first instrument of the audit is the one this chapter already built for interpretability: holdout profiling. Profile every deployed segmentation on the sensitive and proxy attributes available — age band, gender, metro, region — precisely because they were not inputs, and read the result as an impact statement: who, demographically, lands in the segment that gets the richest offers, and who lands in the one that gets suppression? Section 6.10's validation tool and this section's fairness tool are the same instrument pointed at different questions.

Being the same instrument, it inherits the same limits, and this is where an ethics section must be more careful than a methods section. A profile of who lands where is a description of exposure, not a determination of fairness. It does not establish that a difference in treatment is unjustified, does not measure the benefit or burden the treatment actually delivers, does not settle whether a practice is lawful in any particular jurisdiction, and does not survive the moment the segmentation changes. Passing a demographic profile is a minimum screen — the first thing an analyst does, not the last thing an organization needs. The fuller screen adds four things: treatment rates computed by demographic and proxy group, not merely segment membership; an explicit comparison of the benefits and the burdens each group receives, since a segment that gets fewer discounts and more service is not obviously disadvantaged; a review of the eligibility and exclusion rules themselves, because the sharpest harms usually live in who is suppressed rather than in who is targeted; monitoring after deployment, because assignment shares drift; and an escalation path — to legal, to policy, to whoever owns the risk — wherever differential treatment creates material harm or touches a regulated domain. An analyst who validates properly has built the first instrument of that screen and none of the rest.

Differential pricing and offers are where segment-based treatment meets its sharpest public line. StyleCraft's catalog prices are uniform — a designed fact — but a CRM program does not need price tags to price-discriminate: allocating discounts, early access, free shipping, and win-back incentives by segment sets different effective prices for different people, and Section 6.11's treatment logic openly recommends it (“reward Champions, do not discount them; incentivize At Risk”). Much of this is legitimate, long-practiced, and welfare-defensible; some of it curdles, and the line has recognizable markers. Treatment keyed to behavior the customer controls and can understand — buy more often, earn better treatment — reads as a bargain and survives disclosure. Treatment keyed to who the customer is, or to proxies for it — the store-only suburban shopper quietly excluded from every promotion because her segment's profile says she

pays full price — starts to price the person rather than the behavior, and its test is the one Section 2.12's framing ethics proposed: would the program survive being explained, in plain language, to the customers in its least-favored cell? A treatment map that cannot be read aloud is not a strategy; it is a secret.

The chapter's final caution is the one its own dataset was built to teach, and Section 6.14 has already sprung half of it. The New/At-Risk segment — short tenure, single purchases, unknown loyalty for many of its members — is measurably real, and it concentrates geographically where StyleCraft is newest, including the Wave 4 expansion markets. A naive value-based treatment policy deprioritizes that segment; a naive read of the early response data that follows goes further and concludes those markets underperform. The dataset's ground truth documents what the naive analysis misses: the apparent underperformance is an artifact of store age and marketing exposure, not of customer potential — new customers in under-marketed markets behave exactly as new customers in under-marketed markets should. This chapter is the first place an analyst can step into that trap, by letting a tenure artifact wear a value costume in a treatment map; the discipline that catches it here — same-age comparison, intensity reported beside level, tenure held out of the inputs and profiled afterward, and treatment decisions that name what a segment measures rather than what its averages resemble — is the discipline that catches its sharper versions when models start making the recommendations.

CONCEPT

Segments Treat People Differently — Audit Whom

Before any segmentation is deployed to vary treatment, run the minimum screen. Profile it on the sensitive and proxy attributes that were not inputs, and write down who receives the best and worst treatment in demographic terms. State, for each segment, whether its defining signal is behavior the customer controls or circumstance the customer occupies — tenure, geography, store age — and refuse value verdicts on circumstance. Apply the read-aloud test to the treatment map.

Then treat the screen as the floor it is. Failing the first check is disparate impact unexamined; failing the second is punishing people for when and where they arrived; failing the third is a pricing secret waiting for a screenshot. Passing all three establishes only that the obvious failures are absent. What a fairness assessment additionally requires — treatment rates by group, benefits weighed against burdens, exclusion rules reviewed, monitoring after launch, and escalation where harm or regulation is in play — is the organization's obligation, and the analyst's job is to say so rather than to let one profile table stand in for it.

Source: Course concept developed for this guide, informed by Barocas and Selbst (2016).

6.17 Chapter Summary

This chapter opened Part II by converting Chapter 5's insistence — these customers are not one population — into named, verified, targetable groups, and it equipped the CRM manager's four-slot decision with a defensible map. The main point is that segmentation is a treatment decision

before it is a computation: a partition of customers becomes a segmentation only when each group is attached to a difference in treatment the organization can execute, and the organization's treatment capacity is a design input to the analysis, not an inconvenience discovered at deployment.

The chapter's machinery arrived in two movements. The strategic movement placed segmentation inside the segmentation-targeting-positioning framework, surveyed the four base families and the data each requires — behavioral bases dominating customer-level work because they are computed from verified acts — and separated the two roads to segments: a priori schemes declared in advance, transparent and instantly operable, and post hoc schemes estimated from data, capable of surprise and correspondingly burdened with proof. The analytic movement walked both roads. RFM analysis scored recency, frequency, and monetary value — frequency on declared bands rather than quintiles, because real frequency is lumpy and a row-order tiebreaker manufactures distinctions with no behavioral content — and arranged an exhaustive twenty-five-cell grid whose regions are journeys in waiting. The customer-grain feature table was built as a disciplined grain change on a declared purchasing population — with the customers outside it named as no purchase in the window rather than never purchased, and split between activation and win-back — reconciled to certified totals, with level features joined by exposure-normalized intensity features so that lifetime totals stop standing in for tenure unchecked, and with profiling attributes deliberately held out.

Distance made “alike” mathematical, standardization kept the mathematics honest — unscaled clustering is spend banding by accident — and the section closed on the limit of standardization: equalizing scale does nothing about redundancy, which is why feature selection is a modeling decision and the labs test it. k-means was developed to working level as an unsupervised method, with the assign-and-update sketch, initialization sensitivity, and the box worth carrying for a career: the algorithm returns the k you asked for, so structure is established by everything around the algorithm, never by its output. The choice of k braided elbow, silhouette, and business judgment; profiling translated centroids into evidence; and validation supplied the chapter's verification theme in two instruments with two different warranties — stability under reseeding and subsampling, measured with a relabeling-invariant index, which establishes reproducibility and not meaning, and holdout profiling, which is convergent evidence and not proof of natural kinds or of differential response. The last step was the one production forgets: the fitted scaler, the fitted centroids, the feature definitions, and the population rule are retained and versioned together, because a segmentation nobody can rerun is a slide rather than a system.

The usefulness screen and a worked attractiveness matrix turned verified segments into targeting decisions, with differentiability scored as an untested hypothesis and substantiality restated as a burden of proof rather than a headcount test. The AI section named the assistant-era failure modes — the confident k, the skipped scaling, the invented persona, the single-run solution, the arbitrary tie rule — and installed the audit routine that answers them. The ethics section confronted the chapter's own purpose: proxies screened by the same holdout profiling

that supports interpretation, offer allocation tested by whether it prices behavior or persons, the demographic profile named as a minimum screen rather than a fairness certification, and the tenure trap named before it could cost the newest markets their journey. The labs ran the full sequence on StyleCraft's designed data, where four planted segments graded the work.

Looking ahead, the treatment map describes who: four segments, sized, profiled, validated, and paired with journeys. The first question the map cannot answer is already on the table from the sign-off meeting — what moves these customers? Which levers, at what strength, with what confidence? Segments describe who; drivers explain what moves them. The next chapter takes up that question with the analyst's next instrument, regression — measuring how spend and behavior relate to engagement, tenure, discounts, and segment membership on the very feature table this chapter built — and with a new verification discipline to match, because the era of auditing AI-generated models has only begun.

6.18 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: attach a treatment to every group, choose bases the data can support, declare scoring rules including ties, scale before measuring distance, defend k on three legs, and validate every estimated structure by refit and holdout before naming it. They are organized into two groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Assignment #2, Segmentation and Targeting, draws on the homework sets. In-class activities are prepared for discussion rather than submitted. Each exercise also carries its assignment label — required practice, homework submission, or in-class discussion — so that instructors can assign selectively.

6.18.1 Core Chapter Practice

Exercise 6.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- State the difference between segmentation as a strategic act and segmentation as a computation, and give the opening case's example of each.
- Why is a set of personas, however vivid, not a segmentation as delivered? What single addition would convert one?
- Define a priori and post hoc segmentation, then state precisely what a post hoc method estimates from the data and what the analyst must still supply. Classify: the RFM grid; the intern's nine clusters; loyalty tiers; the labs' final deliverable (careful — it travels both roads).
- Quintile scores are often called self-calibrating. State what they preserve and what they do not, and give one comparison they cannot support without additional information.

Why does k-means require standardized features? What does the clustering return if they are skipped, and why does standardization not solve the separate problem of redundant features?

The elbow and the silhouette are both computed from the same fitted models. What different question does each answer, and what can neither answer?

State the two validation instruments of Section 6.10, what each establishes, and — precisely — what each does not.

List the five useful-segment criteria, identify which one the platform's four-journey limit invokes, and explain why a fresh segmentation cannot honestly score the differentiable criterion as satisfied.

Name the five artifacts a deployed segmentation must retain, and explain what breaks if a team keeps only the written threshold rules.

Exercise 6.2 Pick the Base (Required Practice)

For each StyleCraft decision below, recommend a segmentation base family, name the specific columns that would operationalize it (or state that the schema cannot support it), and note one validity concern per Chapter 3.

Choosing which two metros receive a direct-mail catalog test.

Deciding which customers receive the win-back incentive.

Briefing the agency's creative team on the voice of the fall campaign.

Selecting customers for early access to the holiday occasionwear drop.

Reporting to the board on “the affluent customer opportunity.”

Exercise 6.3 Score the Grid by Hand (Required Practice)

Using only Lab 6.1 Part A's printed feature table (no code), assign every one of the ten miniature customers to a region of Table 6.3's grid, applying the declared frequency bands. Then answer three questions. Which customer's placement most misrepresents their likely value to StyleCraft, and which feature that the grid cannot see explains why? Table 6.3 has twenty-five cells and Table 6.4 lists seven regions — show that the mapping is exhaustive and mutually exclusive by counting the cells each region claims. Finally, write the frequency scoring rule you would document at full scale, in one sentence precise enough that two analysts applying it independently would produce identical scores, and state in one further sentence why rank(method="first") would not meet that standard.

Exercise 6.4 Audit the Scaling and the Feature Set (Required Practice)

A colleague clusters the full feature table without standardization and reports four segments. Using Section 6.7's worked trio (C002, C005, C006), show with hand arithmetic what fraction of the C002-C005 squared distance the monetary feature contributes, and state in one sentence what the colleague's four segments almost certainly are. Describe the one-line check — no refitting required — that would confirm your diagnosis from the colleague's own centroid table. Then extend the audit: the colleague standardizes and refits, and reports that the problem is solved.

Name the two features in Table 6.5 that are mathematically dependent on others in the set, explain in two sentences why standardization does not address that, and state which cell of Lab 6.2 would settle the question and what result would justify keeping the eight-feature specification.

Exercise 6.5 Build, Cluster, and Map End-to-End (Homework Submission)

Complete Lab 6.1 Part B and Lab 6.2 on the certified files. Submit: the reconciled feature table with its written population rule and the three population counts; the full-scale RFM grid with the documented frequency rule; the evaluation sweep with a written three-leg defense of your chosen k and the candidates you rejected; the profiled and named segments with written assignment rules; all three validation results — seeded and subsample stability, feature-set sensitivity, and the holdout profile with your pre-written predictions — stated in plain sentences; the answer-key comparison; the completed attractiveness matrix with its evidence-versus-judgment column; the saved segmentation specification and the measured agreement of your simplified rule with it; and the one-page treatment map with its validation footer. Submissions are graded on the written rules, the k defense, the honesty of the validation footer, and the treatment map as heavily as on the code.

Exercise 6.6 AI-Assisted Cluster Audit (Homework Submission)

Give an AI assistant the feature table and this deliberately underspecified prompt, verbatim: "Segment these customers and describe the segments." Then audit the response using Section 6.12's routine. Verify one centroid by hand. Refit its solution from three different seeds and report adjusted Rand indices rather than dominant-cell counts, explaining in one sentence why the index is the safer measure. Run the holdout profile. Check whether it scaled the features, how it handled tied frequency values if it scored RFM at all, and whether it retained the fitted scaler and model or only described thresholds. List every attribute its segment names or descriptions asserted that no input feature or holdout column contains. Report what it chose for k and whether it disclosed choosing. Document the full exchange per the AI-use documentation template in Appendix D, and conclude with two sentences on which audit step changed your assessment most.

6.18.2 In-Class Activities

Exercise 6.7 Find the Flaw in the Segmentation (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

An analyst clusters on recency_days, frequency, monetary, and tenure_days, and reports a "high-value veterans" segment and a "low-value newcomers" segment.

A deck reports "the silhouette identified two segments," and the fall program is built on urban versus suburban alone.

A segmentation's overall average order value is reported as the simple average of the four segments' AOVs.

An agency deliverable names a cluster “Eco-Conscious Millennials” from a table containing R, F, M, and channel shares.

A team refits last quarter's clustering on this quarter's data, gets visibly different groups, and concludes the customer base has changed.

The nine-cluster solution is defended in the meeting with “we validated it — the fit converged and the inertia is low.”

A stability report cross-tabulates two runs, counts the customers outside each row's largest cell, and reports 96 percent stability. Two of the baseline clusters have their largest overlap with the same cluster in the refit.

A campaign team is handed a deck listing four segments with threshold rules and asks how to assign the customers who signed up last week.

Exercise 6.8 Differential Treatment Mini-Cases (In-Class Discussion)

For each mini-case, apply Section 6.16's minimum screen — proxy profile, behavior versus circumstance, read-aloud test — then say which additional elements of the fuller screen the case requires, and come prepared to defend a verdict: proceed, modify, or refuse.

The treatment map excludes the Suburban Occasion segment from all promotional emails “because they pay full price anyway.”

A proposed “high-potential” journey is keyed to app ownership, and the holdout profile shows app ownership concentrates sharply in the two youngest age bands.

Finance proposes suppressing all marketing to the New/At-Risk segment for two quarters to “let the low performers wash out,” with the savings funding a Champions expansion.

A win-back incentive ladder offers the richest discounts to the highest-monetary lapsed customers, and the read-aloud test is applied from the perspective of a loyal modest-basket customer who receives the smallest offer.

A segmentation passes its demographic profile with no visible imbalance, and the team proposes to record it as “fairness verified” in the release documentation.

6.19 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

A priori segmentation. Segmentation in which the number of segments and the assignment rule are declared before analysis, so that data serve to size and profile predefined groups (adapted from Wind, 1978).

Adjusted Rand index. A relabeling-invariant measure of agreement between two partitions of the same observations, computed over all pairs and corrected for chance agreement; 1.0 indicates identical partitions, values near 0 indicate unrelated ones, and negative values

— bounded below at −0.5 — indicate partitions agreeing less than chance would predict. Used in this chapter to measure clustering stability and feature-set sensitivity (adapted from Hubert & Arabie, 1985).

Cluster profiling. The description of algorithmically estimated groups — sizes first, centroids read in standardized and native units, spreads and medians alongside means — performed with the profile disciplines of Section 5.6 and preceding any naming of segments.

Customer-grain feature table. The analytic asset with one row per customer in a declared population and one column per feature, built from certified transaction data by written derivation rules at a declared analysis date and window, and reconciled to certified totals before downstream use.

Elbow method. The selection heuristic that plots inertia against candidate values of k and nominates the bend beyond which additional clusters buy sharply diminishing compactness (adapted from Thorndike, 1953).

Euclidean distance. The straight-line distance between customers represented as feature vectors — the square root of the sum of squared feature differences — whose value depends on every feature's units and spread (adapted from Everitt et al., 2011).

Exposure normalization. The expression of a level feature as a rate over the time a customer was actually observed inside the analysis window, reported beside the level feature so that a lifetime total bounded by tenure is not read as a measure of customer value. It reduces the mechanical tenure effect rather than removing it, since the level features remain in the specification.

Inertia. The total within-cluster sum of squared distances from observations to their assigned centroids; the quantity k-means minimizes. At the optimum it is non-increasing as k rises, so it informs the choice of k only through the shape of the curve, and a reported increase is a signal to inspect initialization and convergence.

k-means clustering. The unsupervised partitioning algorithm that locates k centroids and assigns each observation to its nearest centroid to minimize inertia, fitted by alternating assignment and update steps from an initialization on which the solution can depend (adapted from MacQueen, 1967; Lloyd, 1982; Pedregosa et al., 2011).

Market segmentation. The division of a market into internally similar, externally different groups of buyers so that marketing effort can be differentiated across groups rather than uniform over the whole market (adapted from Smith, 1956; Kotler & Keller, 2016).

Post hoc segmentation. Segmentation in which group membership and boundaries are estimated from the data by an analytic method such as clustering, after the analyst specifies the features, the method, and the candidate values of k, and carrying a corresponding burden of validation (adapted from Wind, 1978).

Quintile scoring. The conversion of a numeric quantity into a five-level ordinal score by ranking customers and assigning 5 to the marketing-favorable fifth through 1 to the least; scores preserve rank within the scored population and window but not absolute level, and tie handling is declared as part of the rule (adapted from Hughes, 1994).

RFM analysis. The behavioral segmentation method that scores customers on recency, frequency, and monetary value in a declared window and groups them by score combinations on an exhaustive named grid, with each region's treatment logic treated as a hypothesis about response rather than an established rule (adapted from Hughes, 1994; Fader et al., 2005).

Segment attractiveness. The comparative evaluation of qualifying segments on size, value, growth, margin, cost to reach, and strategic and capability fit, used to inform targeting priority, with measured cells marked separately from judgment cells (adapted from Kotler & Keller, 2016; Wedel & Kamakura, 2000).

Segmentation base. The characteristic or set of characteristics — geographic, demographic, psychographic, or behavioral — on which customers are divided, determining the data required and the treatments supportable (adapted from Wedel & Kamakura, 2000; Kotler & Keller, 2016).

Silhouette score. A per-observation comparison of average within-cluster distance against average distance to the nearest other cluster, scaled to lie between −1 and +1 and averaged across observations to grade a clustering's geometric separation at each candidate k (adapted from Rousseeuw, 1987).

Stability check. The validation instrument that refits a clustering under perturbation — independently seeded initializations, subsamples — and measures agreement between the resulting partitions with a relabeling-invariant index; high agreement establishes reproducibility, not meaning (adapted from Everitt et al., 2011; Hubert & Arabie, 1985).

Standardization (z-score scaling). The rescaling of a numeric feature by subtracting its mean and dividing by its standard deviation, giving mean 0 and standard deviation 1 so that distances weight features in standard-deviation units rather than native units; it equalizes scale and does not remove redundancy (adapted from James et al., 2021; Pedregosa et al., 2011).

STP (segmentation, targeting, positioning). The sequential strategy framework of dividing a market into segments, selecting target segments, and positioning the offering distinctively for each target (adapted from Kotler & Keller, 2016).

Targeting. The selection of which segments to serve, with what priority and resources, decided on segment-attractiveness evidence within the firm's treatment capacity (adapted from Kotler & Keller, 2016).

Useful-segment criteria. The five-part screen — measurable, substantial, accessible, differentiable, actionable — that a segment must pass to justify differentiated marketing (adapted from Kotler & Keller, 2016).

6.20 Further Readings

Students who want additional background may begin with the following readings. Strategy and marketing foundations are listed first, methods second.

Smith (1956) for the founding statement of segmentation as strategy — short, readable, and the source of the idea every algorithm in this chapter serves.

Yankelovich and Meer (2006) for the sharpest practitioner critique of segmentation drift — how segmentations decay into demographic portraiture and how to key them to decisions instead; the extended companion to Sections 6.2 and 6.15.

Wedel and Kamakura (2000) for the comprehensive scholarly treatment of segmentation bases and methods; the reference behind Sections 6.3 and 6.11.

Fader et al. (2005) for the bridge from RFM measures to formal customer valuation — where Section 6.5's workhorse meets the customer-lifetime-value modeling of later chapters.

James et al. (2021), Chapter 12, for the standard accessible treatment of k-means and its relatives one level of formality above this chapter, read alongside Rousseeuw (1987) for the original silhouette paper — brief, geometric, and clarifying about what separation scores do and do not certify.

Barocas and Selbst (2016) for the definitive legal-analytic treatment of disparate impact in data-driven decisions; the foundation under Section 6.16, first cited in Section 1.13.

6.21 References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Cluster analysis* (5th ed.). Wiley.

Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). RFM and CLV: Using iso-value curves for customer base analysis. *Journal of Marketing Research, 42*(4), 415–430. https://doi.org/10.1509/jmkr.2005.42.4.415

Haley, R. I. (1968). Benefit segmentation: A decision-oriented research tool. *Journal of Marketing, 32*(3), 30–35. https://doi.org/10.1177/002224296803200306

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218. https://doi.org/10.1007/BF01908075

Hughes, A. M. (1994). *Strategic database marketing*. Probus Publishing.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer.

Kotler, P., & Keller, K. L. (2016). *Marketing management* (15th ed.). Pearson Education.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory, 28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). University of California Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Smith, W. R. (1956). Product differentiation and market segmentation as alternative marketing strategies. *Journal of Marketing, 21*(1), 3–8. https://doi.org/10.1177/002224295602100102

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika, 18*(4), 267–276. https://doi.org/10.1007/BF02289263

Wedel, M., & Kamakura, W. A. (2000). *Market segmentation: Conceptual and methodological foundations* (2nd ed.). Kluwer Academic Publishers.

Wind, Y. (1978). Issues and advances in segmentation research. *Journal of Marketing Research, 15*(3), 317–337. https://doi.org/10.1177/002224377801500302

Yankelovich, D., & Meer, D. (2006). Rediscovering market segmentation. *Harvard Business Review, 84*(2), 122–131.