QUERY DOCUMENTATION PACK

Query Documentation

Q1–Q8 Business Questions, Logic, and Annotated SQL

Reporting Period	January 2023 – December 2024
Document Version	1.0
Date Prepared	April 2026
Classification	Internal — Analytics Team
Prepared By	Data & Analytics Team

DISTRIBUTION

Analytics · Data Engineering · Onboarding · Knowledge Base

PREPARED BY	REVIEWED BY	APPROVED BY
_____	_____	_____
Signature / Date	*Signature / Date*	*Signature / Date*

Introduction

This pack documents the eight analytical queries (Q1 through Q8) that produced the findings in the Executive Memo (Document 1). Each query is presented as a standalone reference: the business question it answers, the assumptions it makes, the tables and joins involved, the filter logic, the output schema, edge cases, sample output, and the full annotated SQL.

All queries operate on the cleaned schema described in the Technical Methodology Report (Document 2, §4). Run order is independent — each query stands alone.

Conventions used throughout

Convention	Meaning
cleaned.<table>	Tables in the cleaned schema (see Document 2, §4 for definitions).
100.0 (with decimal)	Forces floating-point division; bare 100 truncates.
LIMIT N	Result set capped to N rows for readability.
::NUMERIC	Explicit cast to NUMERIC for ROUND() to avoid implicit cast warnings.
Sample output	Illustrative figures; product/customer/seller names genericized.[1]

Table 1. SQL conventions used in this pack.

Q1 — Customer Acquisition and 30-Day Conversion

Business question

Which states have the highest new-customer sign-ups in 2024, and what share of those new sign-ups place at least one order within 30 calendar days of registration?

Why it matters

Sign-up volume measures top-of-funnel marketing reach. 30-day conversion measures bottom-of-funnel commercial intent. The gap between the two is the leakage in the new-customer pipeline — and is the single most actionable acquisition metric the platform has.

Tables and joins

Table	Role	Join condition
cleaned.customers	Source of new sign-ups	Filter: signup_date in 2024.
cleaned.orders	Detect first purchase	INNER JOIN on customer_id; order_date within signup_date + 30 days.

Table 2. Q1 join structure.

Logic outline

- CTE 1 (new_customers_2024): All customers with signup_date in 2024.
- CTE 2 (state_signups): Top 5 states by total sign-ups.
- CTE 3 (converted_customers): Customers from CTE 1 who placed at least one order within 30 days. INNER JOIN on customer_id with date range predicate. DISTINCT to avoid double-counting customers who placed multiple orders.
- Final SELECT: LEFT JOIN states to converted customers and compute conversion percentage.

Output schema

Column	Type	Meaning
state	TEXT	State name (top 5 by sign-up count).
total_signups	INTEGER	Total customers who signed up in 2024 from this state.
converted_within_30_days	INTEGER	Distinct customers who placed at least one order within 30 days.
conversion_rate_pct	NUMERIC(5,2)	Conversion rate as a percentage to 2 decimal places.

Table 3. Q1 output schema.

Sample output

state	total_signups	converted_within_30_days	conversion_rate_pct
Lagos	146	72	**49.32**
FCT	92	38	41.30
Rivers	66	28	42.42
Oyo	63	21	**33.33**
Kano	58	18	**31.03**

Table 4. Q1 result set.

Edge cases

- Customers signing up in late December 2024 had less than 30 days to convert before the data period ended — they may appear in total_signups but not in converted_within_30_days. This is acknowledged in Document 1, footnote 2.
- DISTINCT in CTE 3 is essential — without it, customers who placed multiple early orders would be counted multiple times, inflating the conversion rate.

INTERVAL syntax: PostgreSQL requires `INTERVAL '30 days'` with the value as a string[1].

Related queries

Q6 (payment methods by state) provides the explanatory context — the two states with the lowest Q1 conversion are also the only two where Cash on Delivery dominates.

Annotated SQL

```
WITH new_customers_2024 AS (
    SELECT customer_id, state, signup_date
    FROM cleaned.customers
    WHERE signup_date >= '2024-01-01'
      AND signup_date <= '2024-12-31'
),
state_signups AS (
    SELECT state, COUNT(*) AS total_signups
    FROM new_customers_2024
    GROUP BY state
    ORDER BY total_signups DESC
    LIMIT 5
),
converted_customers AS (
```

[1]PostgreSQL's INTERVAL type requires the value to be quoted as a string literal: INTERVAL '30 days'. Passing an integer (e.g. 30) raises a type error.

```
    -- Customers who placed at least one order within 30 days of signup
    SELECT DISTINCT nc.customer_id, nc.state
    FROM new_customers_2024 nc
    INNER JOIN cleaned.orders o
        ON o.customer_id = nc.customer_id
       AND o.order_date >= nc.signup_date
       AND o.order_date <= nc.signup_date + INTERVAL '30 days'
)
SELECT
    ss.state,
    ss.total_signups,
    COUNT(cc.customer_id) AS converted_within_30_days,
    ROUND(
        COUNT(cc.customer_id) * 100.0 / ss.total_signups, 2
    ) AS conversion_rate_pct
FROM state_signups ss
LEFT JOIN converted_customers cc ON ss.state = cc.state
GROUP BY ss.state, ss.total_signups
ORDER BY ss.total_signups DESC;
```

Listing 1. Q1 — full annotated query.

Q2 — Product Performance

Business question

Which products generated the most revenue in 2024? Include category, total revenue, and number of distinct orders that contained the product.

Tables and joins

Table	Role	Join condition
cleaned.order_items	Revenue source	Driver of the join.
cleaned.products	Product name and category	INNER JOIN on product_id.
cleaned.orders	Date filter	INNER JOIN on order_id; filter to 2024.

Table 5. Q2 join structure.

Logic outline

- Inner-join order_items to products and orders. The order_items table is the granular ground truth for revenue (each row is one product unit at one price).
- Filter: order_date in 2024 and line_total IS NOT NULL (excludes the 4 products with NULL prices).
- Group by product_name and category; aggregate SUM(line_total) for revenue and COUNT(DISTINCT order_id) for order count.
- ORDER BY total_revenue DESC LIMIT 10.

Sample output

Rank	product_name (genericized)	category	revenue (₦)	orders
1	Product A	Electronics	26,712,440	73
2	Product B	Electronics	24,089,150	68
3	Product C	Electronics	23,415,890	65
4	Product D	Electronics	22,508,720	61
5	Product E	Electronics	21,772,340	58
...				
10	Product J	Electronics	17,802,510	44

Table 6. Q2 result set (top 10, all Electronics).

Edge cases

- Products with NULL line_total are excluded by the IS NOT NULL filter. This affects the 4 unrecoverable-price products documented in Document 3, §4.
- Products with no 2024 orders simply do not appear in the result. This is the desired behaviour — the question asks about products that generated revenue, not all products.
- If two products tie on revenue, the order between them is non-deterministic. For this dataset, no ties were observed.

Related queries

Q4 provides the period-level revenue context. Q7 explores why mid-rated products produce more revenue than high-rated ones — at the category level rather than the product level.

Annotated SQL

```
SELECT
    p.product_name,
    p.category,
    ROUND(SUM(oi.line_total)::NUMERIC, 2) AS total_revenue,
    COUNT(DISTINCT o.order_id) AS total_orders
FROM cleaned.order_items oi
JOIN cleaned.products p ON oi.product_id = p.product_id
JOIN cleaned.orders o ON oi.order_id = o.order_id
WHERE o.order_date >= '2024-01-01'
  AND o.order_date <= '2024-12-31'
  AND oi.line_total IS NOT NULL
GROUP BY p.product_name, p.category
ORDER BY total_revenue DESC
LIMIT 10;
```

Listing 2. Q2 — full annotated query.

Q3 — Seller Fulfilment Efficiency

Business question

Which sellers have the fastest average fulfilment time, and how does fulfilment speed correlate with customer rating?

Tables and joins

Table	Role	Join condition
cleaned.orders	Compute fulfilment hours	Filter: order_status = 'Delivered' AND delivery_date IS NOT NULL.
cleaned.reviews	Compute average rating	Joined via orders to attribute reviews to sellers.
cleaned.sellers	Display seller_name	JOIN on seller_id.

Table 7. Q3 join structure.

Logic outline

- CTE 1 (seller_fulfilment): For each seller, calculate average fulfilment hours from delivered orders. HAVING COUNT >= 20 restricts to sellers with statistically meaningful sample size.
- CTE 2 (seller_ratings): For each seller, average rating across reviews tied to that seller's orders.
- Final SELECT: LEFT JOIN ratings (some sellers may have no reviews).
- Order by avg_fulfilment_hours ASC; limit 20.

Fulfilment hour calculation

Hours are computed as `EXTRACT(EPOCH FROM (delivery_date - order_date)) / 3600`. Both columns are TIMESTAMP types, so the subtraction yields an INTERVAL. EPOCH extracts the duration in seconds; dividing by 3600 converts to hours[2].

Sample output

seller_name (genericized)	completed	avg hours	avg rating	Note
Seller A	147	63.4	4.6	Top qualifier

[2]EXTRACT(EPOCH FROM ...) returns the duration in seconds as a numeric. Dividing by 3600 converts to hours. Both delivery_date and order_date are TIMESTAMP types, so the subtraction yields an INTERVAL.

seller_name (genericized)	completed	avg hours	avg rating	Note
Seller B	82	91.0	3.25	Fast but mid-rated
Seller C	118	94.2	4.4	Bonus-qualifying
Seller D	104	98.7	4.3	Bonus-qualifying
Seller E	98	102.4	4.2	Bonus-qualifying
...				

Table 8. Q3 result set (top 5 of 20 fastest sellers).

Edge cases

- Sellers with fewer than 20 completed orders are excluded by HAVING — this prevents new sellers from gaming the ranking with one or two atypically fast deliveries.
- Sellers with no reviews appear with NULL avg_customer_rating — captured by the LEFT JOIN.
- Negative time intervals would arise from data corruption (delivery_date < order_date). The EXTRACT calculation would produce a negative number; AVG would still complete but the result would be misleading. None observed in the cleaned dataset.

Related queries

Q8 imposes both volume (≥10 delivered orders) and rating (≥4.0) thresholds — a stricter version of the Q3 quality bar. Sellers fast in Q3 but rated below 4.0 do not qualify in Q8.

Annotated SQL

```
WITH seller_fulfilment AS (
    SELECT
        o.seller_id,
        COUNT(o.order_id) AS total_completed_orders,
        ROUND(
            AVG(EXTRACT(EPOCH FROM
                (o.delivery_date - o.order_date)) / 3600
            ), 2
        ) AS avg_fulfilment_hours
    FROM cleaned.orders o
    WHERE o.order_status = 'Delivered'
      AND o.delivery_date IS NOT NULL
      AND o.order_date IS NOT NULL
    GROUP BY o.seller_id
```

```
    HAVING COUNT(o.order_id) >= 20
),
seller_ratings AS (
    SELECT
        o.seller_id,
        ROUND(AVG(r.rating), 2) AS avg_customer_rating
    FROM cleaned.reviews r
    JOIN cleaned.orders o ON r.order_id = o.order_id
    WHERE r.rating IS NOT NULL
    GROUP BY o.seller_id
)
SELECT
    sf.seller_id,
    s.seller_name,
    sf.total_completed_orders,
    sf.avg_fulfilment_hours,
    sr.avg_customer_rating
FROM seller_fulfilment sf
JOIN cleaned.sellers s ON sf.seller_id = s.seller_id
LEFT JOIN seller_ratings sr ON sf.seller_id = sr.seller_id
ORDER BY sf.avg_fulfilment_hours ASC
LIMIT 20;
```

Listing 3. Q3 — full annotated query.

Q4 — Quarterly Revenue Trends

Business question

How did quarterly revenue, average order value, and order count change between 2023 and 2024? Which single quarter showed the strongest year-over-year revenue growth?

Logic outline

Two-part query. Part 1 produces the full quarterly breakdown for both years. Part 2 self-joins the 2024 quarters to their 2023 counterparts to compute year-over-year growth percentages.

- CTE quarterly_metrics: GROUP BY year and quarter; aggregate revenue, AOV, and order count.
- Part 1: SELECT from CTE, ordered by year then quarter.
- Part 2: Self-join the CTE on quarter; filter to 2024 vs 2023; compute growth_pct as ((2024 − 2023) / 2023) × 100.0; order DESC; LIMIT 1.

Sample output — Part 1 (quarterly breakdown)

yr	qtr	total_revenue (₦)	avg_order_value (₦)	orders	yoy %
2023	1	7,800,000	354,545	22	—
2023	2	24,600,000	342,857	72	—
2023	3	48,900,000	330,405	148	—
2023	4	73,700,000	311,433	237	—
2024	1	116,500,000	342,647	340	**+1,393%**
2024	2	162,300,000	350,540	463	+560%
2024	3	226,300,000	356,141	636	+363%
2024	4	365,900,000	344,718	1,061	+396%

Table 9. Q4 Part 1 — quarterly breakdown across both years.

Sample output — Part 2 (strongest YoY quarter)

qtr	revenue 2023 (₦)	revenue 2024 (₦)	growth_pct
Q1	7,800,000	116,500,000	**+1,393.59**

Table 10. Q4 Part 2 — strongest YoY growth quarter.

> **READ Q1 2024 WITH CARE**

> The 1,395% YoY figure is mathematically correct but reflects a tiny 2023 baseline (22 orders, ₦7.8M). Q4 2024 (₦365.9M) is the largest absolute quarter and a more meaningful indicator of platform scale.

Edge cases

- Division by zero: if any 2023 quarter had ₦0 revenue, the growth_pct calculation would error. Not encountered in this dataset because all four 2023 quarters had non-zero revenue.
- Quarter-on-quarter growth (within a year) is not computed by this query — only year-over-year for the same quarter.

Annotated SQL

```
-- Part 1: Quarterly breakdown
WITH quarterly_metrics AS (
    SELECT
        EXTRACT(YEAR FROM order_date)::INTEGER AS yr,
        EXTRACT(QUARTER FROM order_date)::INTEGER AS qtr,
        ROUND(SUM(total_amount)::NUMERIC, 2) AS total_revenue,
        ROUND(AVG(total_amount)::NUMERIC, 2) AS avg_order_value,
        COUNT(order_id) AS total_orders
    FROM cleaned.orders
    WHERE order_date >= '2023-01-01'
      AND order_date <= '2024-12-31'
      AND total_amount IS NOT NULL
    GROUP BY EXTRACT(YEAR FROM order_date),
             EXTRACT(QUARTER FROM order_date)
)
SELECT yr, qtr, total_revenue, avg_order_value, total_orders
FROM quarterly_metrics
ORDER BY yr, qtr;

-- Part 2: Quarter with strongest YoY revenue growth
WITH quarterly_metrics AS ( /* same CTE as above */ ),
yoy_growth AS (
    SELECT
        q2024.qtr,
        q2023.total_revenue AS revenue_2023,
        q2024.total_revenue AS revenue_2024,
        ROUND(
            (q2024.total_revenue - q2023.total_revenue) * 100.0
            / q2023.total_revenue, 2
        ) AS growth_pct
    FROM quarterly_metrics q2024
    JOIN quarterly_metrics q2023 ON q2024.qtr = q2023.qtr
    WHERE q2024.yr = 2024 AND q2023.yr = 2023
)
SELECT qtr AS strongest_growth_quarter,
       revenue_2023, revenue_2024, growth_pct
```

```
FROM yoy_growth
ORDER BY growth_pct DESC
LIMIT 1;
```

Listing 4. Q4 — full annotated query (Part 1 and Part 2).

Q5 — Customer Spend Segmentation

Business question

Segment 2024 customers by total annual spend into High (≥ ₦100K), Medium (₦50K–₦99K), and Low (< ₦50K). For each segment, report customer count, average spend per customer, and total revenue contribution.

Logic outline

- CTE customer_spend_2024: SUM total_amount per customer for 2024.
- CTE segmented: Bucket each customer using CASE.
- Final SELECT: Aggregate by segment.

Sample output

segment	customers	avg spend (₦)	total revenue (₦)
High Spenders	**603**	1,440,298	**868,500,000**
Medium Spenders	52	38,461	2,000,000
Low Spenders	22	27,272	600,000
Total	**677**	**—**	**871,100,000**

Table 11. Q5 result set.

> **89% CONCENTRATION**
> 603 of 677 active customers are High Spenders. The Medium and Low tiers combined contribute under ₦3M — barely 0.3% of total revenue. The platform's revenue is overwhelmingly generated by a single customer segment, with little contribution from casual or occasional buyers.

Edge cases

- A customer with NULL total_amount on every order would not appear in the result (the IS NOT NULL filter excludes them). Affected: 19 orders tied to 4 unpriced products. Realistically, no customer is purely affected — they will have other valid orders.
- Boundary handling: a customer with exactly ₦100,000 spend falls in High Spenders; exactly ₦50,000 falls in Medium. CASE evaluates top-down.

Annotated SQL

```
WITH customer_spend_2024 AS (
    SELECT o.customer_id, SUM(o.total_amount) AS total_spend
    FROM cleaned.orders o
```

```
    WHERE o.order_date >= '2024-01-01'
      AND o.order_date <= '2024-12-31'
      AND o.total_amount IS NOT NULL
    GROUP BY o.customer_id
),
segmented AS (
    SELECT
        customer_id,
        total_spend,
        CASE
            WHEN total_spend >= 100000 THEN 'High Spenders'
            WHEN total_spend >= 50000  THEN 'Medium Spenders'
            ELSE 'Low Spenders'
        END AS segment
    FROM customer_spend_2024
)
SELECT
    segment,
    COUNT(*) AS customer_count,
    ROUND(AVG(total_spend)::NUMERIC, 2) AS avg_spend_per_customer,
    ROUND(SUM(total_spend)::NUMERIC, 2) AS total_revenue_contribution
FROM segmented
GROUP BY segment
ORDER BY total_revenue_contribution DESC;
```

Listing 5. Q5 — full annotated query.

Q6 — Payment Method Preferences by State

Business question

For each state, show the transaction count and total amount for each payment method. Identify the most popular payment method per state.

Tables and joins

Table	Role	Join condition
cleaned.payments	Payment method and amount	Driver.
cleaned.orders	Bridge to customer	INNER JOIN on order_id.
cleaned.customers	State attribution	INNER JOIN on customer_id.

Table 12. Q6 join structure.

Logic outline

- CTE payment_by_state: GROUP BY state, payment_method; count transactions and sum amounts.
- CTE ranked: ROW_NUMBER() OVER (PARTITION BY state ORDER BY transaction_count DESC) — tags the most popular method per state.
- Final SELECT: present all rows with a 'Most Popular' indicator on the rank=1 rows.

Why ROW_NUMBER and not RANK

ROW_NUMBER assigns a unique sequential number even when two payment methods tie on transaction_count. RANK would assign the same rank to ties, which would mark multiple methods as 'Most Popular' — not what the question asks. If the desired behaviour is 'all tied methods are most popular', switch to RANK.

Sample output

state	payment_method	transactions	amount (₦)	popularity
Lagos	Card	412	186,400,000	**Most Popular**
Lagos	Mobile Money	175	78,200,000	
Lagos	Bank Transfer	126	55,400,000	
Lagos	Cash on Delivery	79	32,100,000	
Kano	Cash on Delivery	44	12,300,000	**Most Popular**

state	payment_method	transactions	amount (₦)	popularity
Kano	Bank Transfer	17	4,800,000	
...				

Table 13. Q6 result excerpt. Card dominates Lagos, FCT, Rivers; Cash on Delivery dominates Kano and Oyo.

Related queries

Q6 is the explanatory companion to Q1 — the two states where Cash on Delivery dominates (Kano, Oyo) are the two states with the lowest 30-day conversion rates.

Annotated SQL

```
WITH payment_by_state AS (
    SELECT
        c.state,
        p.payment_method,
        COUNT(*) AS transaction_count,
        ROUND(SUM(p.amount)::NUMERIC, 2) AS total_amount
    FROM cleaned.payments p
    JOIN cleaned.orders o ON p.order_id = o.order_id
    JOIN cleaned.customers c ON o.customer_id = c.customer_id
    GROUP BY c.state, p.payment_method
),
ranked AS (
    SELECT
        state, payment_method, transaction_count, total_amount,
        ROW_NUMBER() OVER (
            PARTITION BY state
            ORDER BY transaction_count DESC
        ) AS rank
    FROM payment_by_state
)
SELECT
    state, payment_method, transaction_count, total_amount,
    CASE WHEN rank = 1 THEN 'Most Popular' ELSE '' END AS popularity
FROM ranked
ORDER BY state, transaction_count DESC;
```

Listing 6. Q6 — full annotated query.

Q7 — Review Ratings and Sales Performance

Business question

Group products by average review rating into High Rated (≥ 4.0), Mid Rated (3.0–3.99), and Low Rated (< 3.0). For each category, report product count, total revenue, and average unit price.

Logic outline

- CTE product_avg_rating: AVG(rating) per product, excluding NULL ratings (the 5 coerced records from cleaning).
- CTE rating_categories: Bucket each product using CASE.
- Final SELECT: Join to products (for unit_price) and order_items (for revenue). LEFT JOIN to order_items to keep products that may have ratings but no 2024 orders.
- Custom ORDER BY using a CASE expression to enforce High → Mid → Low ordering rather than alphabetical.

Sample output

rating_category	products	total_revenue (₦)	avg_unit_price (₦)
High Rated	124	357,500,000	**46,400**
Mid Rated	98	**498,600,000**	58,200
Low Rated	43	59,400,000	41,200

Table 14. Q7 result set. 15 products excluded for lack of any review.

> **COUNTER-INTUITIVE FINDING**
> Mid-rated products produce ₦141M more revenue than high-rated products. High-rated products carry the lowest average unit price (₦46.4K). Customers may rate cheaper, simpler products favourably while spending on more expensive mid-rated SKUs.

Edge cases

- 15 products have no reviews — they are excluded by the INNER JOIN to product_avg_rating. The Executive Memo (footnote 3) acknowledges this exclusion.
- Products with reviews but no 2024 orders appear with NULL revenue — captured by the LEFT JOIN to order_items. SUM(NULL) is treated as zero in this context.
- Boundary: a product with exactly 4.0 average rating falls in High Rated; exactly 3.0 falls in Mid Rated.

Annotated SQL

```
WITH product_avg_rating AS (
    SELECT r.product_id, AVG(r.rating) AS avg_rating
    FROM cleaned.reviews r
    WHERE r.rating IS NOT NULL
    GROUP BY r.product_id
),
rating_categories AS (
    SELECT
        product_id, avg_rating,
        CASE
            WHEN avg_rating >= 4.0 THEN 'High Rated'
            WHEN avg_rating >= 3.0 THEN 'Mid Rated'
            ELSE 'Low Rated'
        END AS rating_category
    FROM product_avg_rating
)
SELECT
    rc.rating_category,
    COUNT(DISTINCT rc.product_id) AS product_count,
    ROUND(SUM(oi.line_total)::NUMERIC, 2) AS total_revenue,
    ROUND(AVG(p.unit_price)::NUMERIC, 2) AS avg_unit_price
FROM rating_categories rc
JOIN cleaned.products p ON rc.product_id = p.product_id
LEFT JOIN cleaned.order_items oi
    ON oi.product_id = rc.product_id
   AND oi.line_total IS NOT NULL
GROUP BY rc.rating_category
ORDER BY
    CASE rc.rating_category
        WHEN 'High Rated' THEN 1
        WHEN 'Mid Rated' THEN 2
        ELSE 3
    END;
```

Listing 7. Q7 — full annotated query.

Q8 — Top Seller Bonus Qualification

Business question

Identify the top 10 sellers in 2024 by total revenue who completed at least 10 delivered orders AND have an average customer rating of 4.0 or above.

Tables and joins

Table	Role	Join condition
cleaned.orders	Volume + revenue threshold	Filter: 2024, Delivered, total_amount IS NOT NULL.
cleaned.reviews	Rating threshold	Joined via orders to attribute reviews to sellers.
cleaned.sellers	Display seller_name	JOIN on seller_id.

Table 15. Q8 join structure.

Logic outline

- CTE seller_performance_2024: COUNT(orders) and SUM(revenue) per seller. HAVING COUNT >= 10 enforces the volume threshold.
- CTE seller_ratings_2024: AVG(rating) per seller, scoped to 2024 reviews (via orders.order_date).
- Final SELECT: INNER JOIN both CTEs. WHERE avg_customer_rating >= 4.0 enforces the rating threshold. ORDER BY revenue DESC; LIMIT 10.

Sample output

seller_name	orders	rating	revenue (₦M)	Status
Seller A	147	4.6	**10.98**	**Qualified**
Seller C	126	4.5	9.72	**Qualified**
Seller D	118	4.4	8.91	**Qualified**
Seller E	104	4.3	8.27	**Qualified**
Seller F	98	4.2	7.85	**Qualified**
...				
Seller J	62	4.0	5.12	**Qualified**

Table 16. Q8 result set (10 of 90 sellers qualify).

Why only 10 of 90

The two thresholds work together. 10 delivered orders is a low bar (most sellers exceed it). But a 4.0-or-above average rating filters heavily — most sellers receive a mix of ratings that average somewhere in the 3.0–3.9 band. The intersection of both conditions narrows the bonus pool sharply.

Edge cases

- Sellers with no reviews appear in seller_performance_2024 but not in seller_ratings_2024 — the INNER JOIN excludes them. This is intentional: a seller with no rating data has not demonstrated the quality bar.
- Reviews scoped to 2024: a seller with mostly 2023 reviews and no 2024 reviews would not qualify, even if their 2023 reviews were excellent. Bonus criteria are explicitly 2024-only.

Annotated SQL

```
WITH seller_performance_2024 AS (
    SELECT
        o.seller_id,
        COUNT(o.order_id) AS total_orders,
        ROUND(SUM(o.total_amount)::NUMERIC, 2) AS total_revenue
    FROM cleaned.orders o
    WHERE o.order_date >= '2024-01-01'
      AND o.order_date <= '2024-12-31'
      AND o.order_status = 'Delivered'
      AND o.total_amount IS NOT NULL
    GROUP BY o.seller_id
    HAVING COUNT(o.order_id) >= 10
),
seller_ratings_2024 AS (
    SELECT
        o.seller_id,
        ROUND(AVG(r.rating)::NUMERIC, 2) AS avg_customer_rating
    FROM cleaned.reviews r
    JOIN cleaned.orders o ON r.order_id = o.order_id
    WHERE r.rating IS NOT NULL
      AND o.order_date >= '2024-01-01'
      AND o.order_date <= '2024-12-31'
    GROUP BY o.seller_id
)
SELECT
    sp.seller_id, s.seller_name,
    sp.total_orders, sr.avg_customer_rating, sp.total_revenue
FROM seller_performance_2024 sp
JOIN cleaned.sellers s ON sp.seller_id = s.seller_id
JOIN seller_ratings_2024 sr ON sp.seller_id = sr.seller_id
WHERE sr.avg_customer_rating >= 4.0
ORDER BY sp.total_revenue DESC
LIMIT 10;
```

Listing 8. Q8 — full annotated query.

Cross-Query Reference

Query	Depends on / paired with	Purpose of the pairing
Q1	Q6 (payment by state)	Q6 explains why Kano and Oyo (low conversion in Q1) underperform — both are COD-dominant.
Q2	Q7 (rating vs revenue)	Q7 explains why all top revenue products are Electronics — mid-rated products lead, and Electronics dominates that band.
Q3	Q8 (bonus qualification)	Q8 is a stricter version of Q3: same seller universe, additional rating threshold.
Q4	Q5 (spend segmentation)	Q4 explains period-level revenue; Q5 explains who within that revenue is contributing.
Q6	Q1 (state conversion)	Q6 supplies the causal hypothesis for the conversion gap surfaced in Q1.
Q7	Q2 (top products)	Q7 contextualises Q2 — top products are not necessarily highest-rated.
Q8	Q3 (fastest sellers)	Q8 narrows to high-quality sellers; Q3 is the speed-only ranking.

Table 17. Cross-query reference. Each query's findings amplify or contextualise others.

Document Control

Version	Date	Change Summary	Author
0.1	2026-03-15	All 8 queries written and tested	Analytics
0.5	2026-03-29	Per-query documentation drafted	Analytics
0.9	2026-04-08	Cross-query reference; sample outputs added	Analytics
1.0	2026-04-15	Approved as final reference	Analytics

Table 18. Revision history.