Stage 5

Group Members: M Tausif Tajwar Bhuiyan
Luke Rodriguez
Dinh Nhat Nguyen

November 10, 2025

Contents

1 Market and Sales Discoveries

1.1 Query 1.1: High-Value Customer States

Explanation: Find the states that have the highest average order value (AOV), showing where customers spend the most per transaction.

```
SELECT
    g.geolocation_state AS state,
    s.state_name,
    COUNT(DISTINCT o.order_id) AS total_orders,
    AVG(oi.price + oi.freight_value) AS avg_order_value
FROM CUSTOMERS c
JOIN GEOLOCATION g ON c.customer_zip_code_prefix = g.zip_code_prefix
JOIN STATES s ON g.geolocation_state = s.state_code
JOIN ORDERS o ON c.customer_id = o.customer_id
JOIN ORDER_ITEMS oi ON o.order_id = oi.order_id
GROUP BY g.geolocation_state, s.state_name
ORDER BY avg_order_value DESC
LIMIT 10;
```

Interestingness to Analysts: This query shows geographic markets where customers have higher purchasing power or willingness to spend, that indicates targeted marketing campaigns, premium product launches, and strategic resource allocation to high-value regions. States with high AOV may require increased seller recruitment, premium inventory, or specialized customer service investments.

(returns top 10 states by default)

1.2 Query 1.2: Top Selling Categories by Revenue

Explanation: Determine which product categories account for the largest total revenue overall.

```
SELECT
    c.category_name_english,
    COUNT(DISTINCT oi.order_id) AS orders_count,
    COUNT(DISTINCT oi.product_id) AS products_count,
    SUM(oi.price) AS total_revenue,
    AVG(oi.price) AS avg_product_price
FROM CATEGORIES c
JOIN PRODUCTS p ON c.category_id = p.category_id
JOIN ORDER_ITEMS oi ON p.product_id = oi.product_id
GROUP BY c.category_id, c.category_name_english
ORDER BY total_revenue DESC
LIMIT 15;
```

Interestingness to Analysts: Understanding revenue distribution across categories helps prioritize inventory investments, identify growth opportunities, and allocate marketing budgets effectively.

(returns top 15 categories)

1.3 Query 1.3: States with Customers, No Orders

Explanation: Identify states that have registered customers but have zero recorded orders from those residents, highlighting untapped potential.

```
SELECT
    s.state_code,
    s.state_name,
    COUNT(DISTINCT c.customer_id) AS registered_customers
FROM STATES s
```

```
JOIN GEOLOCATION g ON s.state_code = g.geolocation_state
JOIN CUSTOMERS c ON g.zip_code_prefix = c.customer_zip_code_prefix
WHERE NOT EXISTS (
    SELECT 1
    FROM ORDERS o
    WHERE o.customer_id = c.customer_id
)
GROUP BY s.state_code, s.state_name
HAVING COUNT(DISTINCT c.customer_id) > 0
ORDER BY registered_customers DESC;
```

Why Interesting to Analysts: This identifies market product failures where customer acquisition succeeded but conversion did not, suggesting issues with product availability, delivery logistics, payment methods, or customer experience.

1.4 Query 1.4: Highest Sales Geolocation

Explanation: Pinpoint the single most active geolocation (specific zip code) associated with the highest number of customer orders.

```
SELECT
    g.zip_code_prefix,
    g.geolocation_city,
    g.geolocation_state,
    g.geolocation_lat,
    g.geolocation_lng,
    COUNT(DISTINCT o.order_id) AS total_orders,
    COUNT(DISTINCT c.customer_id) AS unique_customers,
    SUM(oi.price + oi.freight_value) AS total_revenue
FROM GEOLOCATION g
JOIN CUSTOMERS c ON g.zip_code_prefix = c.customer_zip_code_prefix
JOIN ORDERS o ON c.customer_id = o.customer_id
JOIN ORDER_ITEMS oi ON o.order_id = oi.order_id
GROUP BY g.zip_code_prefix, g.geolocation_city, g.geolocation_state,
         g.geolocation_lat, g.geolocation_lng
ORDER BY total_orders DESC
LIMIT 1;
```

Why Interesting to Analysts: Identifying the single hottest location showing local strategies like establishing pickup points, positioning warehouse inventory, partnering with local influencers, or studying customer demographics in that specific area to replicate success elsewhere.

2 Seller Performance and Network

2.1 Query 2.1: Seller Success Rate by Review Score

Explanation: Calculate the average review score for each seller, but only using orders that actually received a review to assess quality.

```sql
SELECT
    s.seller_id,
    g.geolocation_city AS seller_city,
    g.geolocation_state AS seller_state,
    COUNT(DISTINCT oi.order_id) AS orders_fulfilled,
    COUNT(DISTINCT r.review_id) AS reviews_received,
    AVG(r.review_score) AS avg_review_score,
    ROUND(COUNT(DISTINCT r.review_id) * 100.0 /
          COUNT(DISTINCT oi.order_id), 2) AS review_rate_percentage
FROM SELLERS s
JOIN GEOLOCATION g ON s.seller_zip_code_prefix = g.zip_code_prefix
JOIN ORDER_ITEMS oi ON s.seller_id = oi.seller_id
LEFT JOIN ORDER_REVIEWS r ON oi.order_id = r.order_id
GROUP BY s.seller_id, g.geolocation_city, g.geolocation_state
HAVING COUNT(DISTINCT r.review_id) > 0
ORDER BY avg_review_score DESC, orders_fulfilled DESC
LIMIT 20;
```

Why Interesting to Analysts: This shows top-performing sellers who could be featured prominently, rewarded with lower commission rates, or studied as best-practice examples. Low-scoring sellers may need quality improvement interventions, additional training, or potential removal from the platform.

(returns top 20 sellers)

2.2 Query 2.2: Unused Product Catalog

Explanation: List all existing products that have never been included in any order item, indicating dead inventory.

```sql
SELECT
    p.product_id,
    c.category_name_english AS category,
    p.product_weight_g,
    p.product_length_cm,
    p.product_height_cm,
    p.product_width_cm,
    p.product_photos_qty
FROM PRODUCTS p
JOIN CATEGORIES c ON p.category_id = c.category_id
WHERE NOT EXISTS (
    SELECT 1
    FROM ORDER_ITEMS oi
    WHERE oi.product_id = p.product_id
)
ORDER BY c.category_name_english, p.product_id;
```

Why Interesting to Analysts: Dead inventory represents wasted catalog space, missed opportunities, and potential issues with product listing quality, pricing, or categorization. This list helps to target actions like removing inactive products, repricing underperformers, improving product descriptions/photos, or investigating why certain product types fail to attract buyers.

2.3 Query 2.3: States with Customers, No Sellers

Explanation: Identify states that have customers but do not have any registered sellers, highlighting a potential fulfillment gap.

```
SELECT
    s.state_code,
    s.state_name,
    COUNT(DISTINCT c.customer_id) AS customer_count
FROM STATES s
JOIN GEOLOCATION g ON s.state_code = g.geolocation_state
JOIN CUSTOMERS c ON g.zip_code_prefix = c.customer_zip_code_prefix
WHERE NOT EXISTS (
    SELECT 1
    FROM SELLERS sel
    JOIN GEOLOCATION g2 ON sel.seller_zip_code_prefix = g2.zip_code_prefix
    WHERE g2.geolocation_state = s.state_code
)
GROUP BY s.state_code, s.state_name
ORDER BY customer_count DESC;
```

Why Interesting to Analysts: This identifies supply-side gaps where customer demand exists but local seller infrastructure is absent, leading to longer delivery times, higher shipping costs, and reduced customer satisfaction.

2.4 Query 2.4: Single-Product Sellers

English Explanation: Count how many sellers offer only one unique product in their entire portfolio.

```
SELECT
    COUNT(*) AS single_product_seller_count
FROM (
    SELECT
        oi.seller_id,
        COUNT(DISTINCT oi.product_id) AS unique_products
    FROM ORDER_ITEMS oi
    GROUP BY oi.seller_id
    HAVING COUNT(DISTINCT oi.product_id) = 1
) AS single_product_sellers;
```

Why Interesting to Analysts: Single-product sellers may represent specialists, or new sellers testing the platform. This metric helps understand seller diversity and platform maturity.

3 Order and Review Quality

3.1 Query 3.1: Order Review Rate

Explanation: Determine what percentage of total orders end up receiving at least one customer review.

```sql
SELECT
    COUNT(DISTINCT o.order_id) AS total_orders,
    COUNT(DISTINCT r.order_id) AS reviewed_orders,
    ROUND(COUNT(DISTINCT r.order_id) * 100.0 /
          COUNT(DISTINCT o.order_id), 2) AS review_rate_percentage
FROM ORDERS o
LEFT JOIN ORDER_REVIEWS r ON o.order_id = r.order_id;
```

Why Interesting to Analysts: Review rate is a engagement metric indicating customer satisfaction, platform trust, and feedback loop health.

3.2 Query 3.2: Worst Rated Product Category

Explanation: Find the product category that has the lowest average customer review score across all time.

```sql
SELECT
    c.category_name_english AS category,
    COUNT(DISTINCT p.product_id) AS products_in_category,
    COUNT(DISTINCT r.review_id) AS total_reviews,
    AVG(r.review_score) AS avg_review_score,
    SUM(CASE WHEN r.review_score <= 2 THEN 1 ELSE 0 END) AS negative_reviews,
    SUM(CASE WHEN r.review_score >= 4 THEN 1 ELSE 0 END) AS positive_reviews
FROM CATEGORIES c
JOIN PRODUCTS p ON c.category_id = p.category_id
JOIN ORDER_ITEMS oi ON p.product_id = oi.product_id
JOIN ORDER_REVIEWS r ON oi.order_id = r.order_id
GROUP BY c.category_id, c.category_name_english
HAVING COUNT(DISTINCT r.review_id) >= 10
ORDER BY avg_review_score ASC
LIMIT 1;
```

Why Interesting to Analysts: Identifying the worst-performing category reveals systemic quality issues, seller problems, or customer expectation mismatches requiring immediate attention.

3.3 Query 3.3: Orders Paid in Full (Single Payment)

Explanation: Count the number of orders that were covered by exactly one payment record, indicating straightforward transactions.

```sql
SELECT
    COUNT(*) AS single_payment_orders
FROM (
    SELECT
        order_id,
        COUNT(*) AS payment_count
    FROM ORDER_PAYMENTS
    GROUP BY order_id
    HAVING COUNT(*) = 1
) AS single_payments;
```

Why Interesting to Analysts: Single-payment orders represent simpler, lower-risk transactions with less administrative overhead, fewer failure points, and reduced customer confusion.

3.4 Query 3.4: Review Score Extremes (1-Star vs. 5-Star)

Explanation: Report the total count of 1-star reviews and 5-star reviews to quickly gauge the extremes of customer sentiment.

```
SELECT
    SUM(CASE WHEN review_score = 1 THEN 1 ELSE 0 END) AS one_star_reviews,
    SUM(CASE WHEN review_score = 5 THEN 1 ELSE 0 END) AS five_star_reviews,
    ROUND(SUM(CASE WHEN review_score = 1 THEN 1 ELSE 0 END) * 100.0 /
          COUNT(*), 2) AS one_star_percentage,
    ROUND(SUM(CASE WHEN review_score = 5 THEN 1 ELSE 0 END) * 100.0 /
          COUNT(*), 2) AS five_star_percentage,
    COUNT(*) AS total_reviews
FROM ORDER_REVIEWS;
```

Why Interesting to Analysts: Extreme ratings reveal polarized customer experiences that is high 5-star counts indicate strong satisfaction and product-market fit, while high 1-star counts signal serious problems requiring urgent intervention.

4 Payments and Transactions

4.1 Query 4.1: Most Common Payment Type by State

Explanation: Determine which payment method (credit card, boleto, etc.) is most frequently used by customers in each state.

```sql
WITH state_payment_counts AS (
    SELECT
        g.geolocation_state,
        s.state_name,
        op.payment_type,
        COUNT(*) AS payment_count
    FROM CUSTOMERS c
    JOIN GEOLOCATION g ON c.customer_zip_code_prefix = g.zip_code_prefix
    JOIN STATES s ON g.geolocation_state = s.state_code
    JOIN ORDERS o ON c.customer_id = o.customer_id
    JOIN ORDER_PAYMENTS op ON o.order_id = op.order_id
    GROUP BY g.geolocation_state, s.state_name, op.payment_type
),
ranked_payments AS (
    SELECT
        geolocation_state,
        state_name,
        payment_type,
        payment_count,
        ROW_NUMBER() OVER (PARTITION BY geolocation_state
                           ORDER BY payment_count DESC) AS rank
    FROM state_payment_counts
)
SELECT
    geolocation_state,
    state_name,
    payment_type AS most_common_payment_type,
    payment_count
FROM ranked_payments
WHERE rank = 1
ORDER BY geolocation_state;
```

Why Interesting to Analysts: Payment method preferences vary significantly by region due to banking infrastructure, credit systems, cultural preferences, and socioeconomic factors. Understanding state-level payment preferences enables optimized payment gateway integration, targeted financial partnership strategies.

4.2 Query 4.2: Average Number of Installments per Payment Type

English Explanation: Compute the average installment count grouped by payment type to understand financing behaviors.

```sql
SELECT
    payment_type,
    COUNT(*) AS total_payments,
    AVG(payment_installments) AS avg_installments,
    MIN(payment_installments) AS min_installments,
    MAX(payment_installments) AS max_installments,
    ROUND(AVG(payment_value), 2) AS avg_payment_value
FROM ORDER_PAYMENTS
GROUP BY payment_type
ORDER BY avg_installments DESC;
```

Why Interesting to Analysts: Installment behavior reveals customer financial constraints, purchasing power, and payment preferences.

5 Customer Behavior and Loyalty

5.1 Query 5.1: Repeat Purchase Customers

Explanation: Identify customers who placed more than one order, to measure customer loyalty and repeat business.

```sql
SELECT
    c.customer_id,
    c.customer_unique_id,
    g.geolocation_city,
    g.geolocation_state,
    COUNT(DISTINCT o.order_id) AS total_orders,
    MIN(o.order_purchase_timestamp) AS first_order_date,
    MAX(o.order_purchase_timestamp) AS last_order_date,
    SUM(oi.price + oi.freight_value) AS total_lifetime_value
FROM CUSTOMERS c
JOIN GEOLOCATION g ON c.customer_zip_code_prefix = g.zip_code_prefix
JOIN ORDERS o ON c.customer_id = o.customer_id
JOIN ORDER_ITEMS oi ON o.order_id = oi.order_id
GROUP BY c.customer_id, c.customer_unique_id,
         g.geolocation_city, g.geolocation_state
HAVING COUNT(DISTINCT o.order_id) > 1
ORDER BY total_orders DESC, total_lifetime_value DESC
LIMIT 50;
```

Why Interesting to Analysts: This query identifies the most valuable customer segment for VIP programs, exclusive offers, loyalty rewards, and retention campaigns.

(returns top 50 repeat customers)

5.2 Query 5.2: Average Time Between Orders (Returning Customers)

Explanation: Calculate the average number of days between consecutive orders for customers who made multiple purchases.

```sql
WITH customer_orders AS (
    SELECT
        customer_id,
        order_id,
        order_purchase_timestamp,
        LAG(order_purchase_timestamp) OVER
            (PARTITION BY customer_id ORDER BY order_purchase_timestamp)
            AS previous_order_date
    FROM ORDERS
),
order_gaps AS (
    SELECT
        customer_id,
        DATEDIFF(day, previous_order_date, order_purchase_timestamp)
            AS days_between_orders
    FROM customer_orders
    WHERE previous_order_date IS NOT NULL
)
SELECT
    COUNT(DISTINCT customer_id) AS returning_customers,
    AVG(days_between_orders) AS avg_days_between_orders,
    MIN(days_between_orders) AS min_days_between_orders,
    MAX(days_between_orders) AS max_days_between_orders,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY days_between_orders)
        AS median_days_between_orders
FROM order_gaps;
```

Why Interesting to Analysts: Purchase frequency is a retention metric where shorter intervals indicate higher engagement and loyalty, while longer gaps suggest weakening customer relationships.

6 Additional Queries

6.1 Query 6.1: Delivery Performance by State

Explanation: Calculate average delivery time (actual vs. estimated) for each state to identify logistics performance issues.

```
SELECT
    g.geolocation_state,
    s.state_name,
    COUNT(DISTINCT o.order_id) AS total_delivered_orders,
    AVG(DATEDIFF(day, o.order_purchase_timestamp,
        o.order_delivered_customer_date)) AS avg_actual_delivery_days,
    AVG(DATEDIFF(day, o.order_purchase_timestamp,
        o.order_estimated_delivery_date)) AS avg_estimated_delivery_days,
    AVG(DATEDIFF(day, o.order_estimated_delivery_date,
        o.order_delivered_customer_date)) AS avg_delay_days,
    SUM(CASE WHEN o.order_delivered_customer_date >
        o.order_estimated_delivery_date THEN 1 ELSE 0 END) AS late_deliveries,
    ROUND(SUM(CASE WHEN o.order_delivered_customer_date >
        o.order_estimated_delivery_date THEN 1 ELSE 0 END) * 100.0 /
        COUNT(*), 2) AS late_delivery_percentage
FROM ORDERS o
JOIN CUSTOMERS c ON o.customer_id = c.customer_id
JOIN GEOLOCATION g ON c.customer_zip_code_prefix = g.zip_code_prefix
JOIN STATES s ON g.geolocation_state = s.state_code
WHERE o.order_delivered_customer_date IS NOT NULL
GROUP BY g.geolocation_state, s.state_name
ORDER BY avg_delay_days DESC;
```

Why Interesting to Analysts: Delivery performance directly impacts customer satisfaction and review scores. States with high delay percentages (¿20%) or long average delays (¿3 days) indicate logistics problems requiring infrastructure investment, carrier partnerships, or warehouse positioning.

6.2 Query 6.2: Category Performance by Quarter

Explanation: Track quarterly revenue trends for each product category to identify seasonal patterns and growth trajectories.

```
SELECT
    c.category_name_english,
    YEAR(o.order_purchase_timestamp) AS order_year,
    QUARTER(o.order_purchase_timestamp) AS order_quarter,
    COUNT(DISTINCT oi.order_id) AS orders,
    SUM(oi.price) AS revenue,
    AVG(oi.price) AS avg_item_price
FROM CATEGORIES c
JOIN PRODUCTS p ON c.category_id = p.category_id
JOIN ORDER_ITEMS oi ON p.product_id = oi.product_id
JOIN ORDERS o ON oi.order_id = o.order_id
WHERE o.order_status = 'delivered'
GROUP BY c.category_name_english, YEAR(o.order_purchase_timestamp),
         QUARTER(o.order_purchase_timestamp)
ORDER BY c.category_name_english, order_year, order_quarter;
```

Why Interesting to Analysts: Quarterly trends reveal seasonal demand patterns essential for inventory planning, marketing budget allocation, and seller recruitment timing.

6.3 Query 6.3: Revenue by State and Year

Explanation: Calculates the total revenue and order count for a specific state for a user-specified year. Parameters * ? (State Code) * ? (Year)

```
SELECT
    COUNT(DISTINCT T1.order_id) AS Total_Orders,
    SUM(T2.payment_value) AS Total_Revenue
FROM ORDERS T1
JOIN ORDER_PAYMENTS T2 ON T1.order_id = T2.order_id
JOIN CUSTOMERS T3 ON T1.customer_id = T3.customer_id
WHERE
    T3.customer_state = ?
    AND T1.order_purchase_timestamp LIKE ? || '%'
GROUP BY
    T3.customer_state;
```

Why Interesting to Analysts: This query helps analysts uncover geographic and temporal revenue trends. By allowing users to specify both a state and a year

6.4 Query 6.4: Seller Inventory Check

Explanation: Lists all sellers who offer products whose descriptions contain specific user-defined keywords. Parameters * ? (Product Keyword)

```
SELECT DISTINCT
    T1.seller_id
FROM ORDER_ITEMS T1
JOIN PRODUCTS T2 ON T1.product_id = T2.product_id
WHERE
    T2.product_description_length IS NOT NULL
    AND T2.product_description_length LIKE '%' || ? || '%';
```

Why Interesting to Analysts: This query gives analysts visibility into the market supply side, specifically, which sellers are offering products that match certain keywords or product themes