Ecommerce Target SQL with Python

Target Corporation Overview

Target Corporation is one of the largest retailers in the U.S., headquartered in Minneapolis, Minnesota. With approximately 1,963 stores and over 400,000 employees, Target recorded a revenue of $107 billion in 2023.

Data Sources

You worked with several datasets in CSV format, such as:

- Customers
- Orders
- Sellers
- Products
- Payments
- Geolocation
- Order Items

These datasets were loaded into a MySQL database for analysis.

SQL Queries and Analysis

Basic Queries:

1. **List all unique cities where customers are located:**
   - **Query:**

     ```
     sql
     Copy code
     SELECT DISTINCT customer_city FROM customers;
     ```

   - **Output:** Major cities include São Paulo, Campinas, and Mogi das Cruzes.
2. **Count the number of orders placed in 2017:**
   - **Query:**

     ```
     sql
     Copy code
     SELECT COUNT(order_id) FROM orders WHERE YEAR(order_purchase_timestamp) =
     2017;
     ```

   - **Output:** 45,101 orders were placed in 2017.

3. **Find the total sales per category:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT products.product_category, ROUND(SUM(payments.payment_value), 2) AS
      sales
      FROM products
      JOIN order_items ON products.product_id = order_items.product_id
      JOIN payments ON payments.order_id = order_items.order_id
      GROUP BY products.product_category;
      ```

    - **Output:** Categories like Furniture and Decoration ($1,430,176.39), and Bed Table Bath ($1,712,553.67) were among the highest contributors.
4. **Calculate the percentage of orders that were paid in installments:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT (SUM(CASE WHEN payment_installments >= 1 THEN 1 ELSE 0 END) /
      COUNT(*)) * 100 FROM payments;
      ```

    - **Output:** 99.9981% of orders were paid in installments.
5. **Count the number of customers from each state:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT customer_state, COUNT(customer_id) AS customer_count FROM customers
      GROUP BY customer_state;
      ```

    - **Output:** The highest number of customers come from São Paulo, followed by other states.

Intermediate Queries:

1. **Calculate the number of orders per month in 2018:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT MONTHNAME(order_purchase_timestamp) AS months, COUNT(order_id) AS
      order_count
      FROM orders
      WHERE YEAR(order_purchase_timestamp) = 2018
      GROUP BY months;
      ```

    - **Output:** Orders peaked in August and May of 2018.

2. **Find the average number of products per order, grouped by customer city:**
    - **Query:**

      ```
      sql
      Copy code
      WITH count_per_order AS (
         SELECT orders.order_id, COUNT(order_items.product_id) AS
      products_per_order
         FROM orders
         JOIN order_items ON orders.order_id = order_items.order_id
         GROUP BY orders.order_id
      )
      SELECT customers.customer_city, AVG(count_per_order.products_per_order)
      FROM customers
      JOIN count_per_order ON customers.customer_id = orders.customer_id
      GROUP BY customers.customer_city;
      ```

    - **Output:** Padre Carvalho had the highest average products per order (7.00).
3. **Calculate the percentage of total revenue contributed by each product category:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT product_category, ROUND((SUM(payment_value) / (SELECT
      SUM(payment_value) FROM payments)) * 100, 2) AS sales_percentage
      FROM products
      JOIN order_items ON products.product_id = order_items.product_id
      JOIN payments ON payments.order_id = order_items.order_id
      GROUP BY product_category;
      ```

    - **Output:** Categories like Bed Table Bath contributed 10.70%, followed by Health Beauty (10.35%).
4. **Identify the correlation between product price and the number of times a product has been purchased:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT CORR(product_price, num_purchases) FROM products;
      ```

    - **Output:** The correlation was -0.10, indicating a weak inverse relationship between price and purchase frequency.
5. **Calculate the total revenue generated by each seller, and rank them by revenue:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT seller_id, SUM(payment_value) AS revenue, RANK() OVER(ORDER BY
      SUM(payment_value) DESC) AS rank
      FROM order_items
      JOIN payments ON order_items.order_id = payments.order_id
      GROUP BY seller_id;
      ```

    - **Output:** The top seller generated the highest revenue, and sellers were ranked accordingly.

Advanced Queries:

1. **Calculate the moving average of order values for each customer over their order history:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT customer_id, order_purchase_timestamp, AVG(payment_value) OVER
      (PARTITION BY customer_id ORDER BY order_purchase_timestamp ROWS BETWEEN 2
      PRECEDING AND CURRENT ROW) AS moving_avg_order_value
      FROM orders
      JOIN payments ON orders.order_id = payments.order_id;
      ```

    - **Output:** The moving average was calculated for each customer, helping identify trends in their spending.
2. **Calculate the cumulative sales per month for each year:**
    - **Query:**

      ```
      sql
      Copy code
      SELECT year(order_purchase_timestamp) AS years,
      month(order_purchase_timestamp) AS months, SUM(payment_value) OVER (ORDER
      BY years, months) AS cumulative_sales
      FROM orders
      JOIN payments ON orders.order_id = payments.order_id;
      ```

    - **Output:** The cumulative sales increased steadily over each month from 2016 to 2018.
3. **Calculate the year-over-year growth rate of total sales:**
    - **Query:**

      ```
      sql
      Copy code
      WITH sales_data AS (
          SELECT year(order_purchase_timestamp) AS years, SUM(payment_value) AS
      total_sales
          FROM orders
          JOIN payments ON orders.order_id = payments.order_id
          GROUP BY years
      )
      SELECT years, (total_sales - LAG(total_sales) OVER (ORDER BY years)) /
      LAG(total_sales) OVER (ORDER BY years) * 100 AS yoy_growth_rate
      FROM sales_data;
      ```

    - **Output:** The year-over-year growth rate was 20.0% in 2018.
4. **Calculate the retention rate of customers:**
    - **Query:**

      ```
      sql
      Copy code
      WITH first_purchase AS (
          SELECT customer_id, MIN(order_purchase_timestamp) AS first_order
          FROM orders
          GROUP BY customer_id
      )
      SELECT COUNT(DISTINCT a.customer_id) * 100.0 / COUNT(DISTINCT
      b.customer_id) AS retention_rate
      FROM first_purchase a
      LEFT JOIN orders b ON a.customer_id = b.customer_id AND
      b.order_purchase_timestamp > a.first_order AND b.order_purchase_timestamp
      <= DATE_ADD(a.first_order, INTERVAL 6 MONTH);
      ```

    - **Output:** The retention rate was calculated based on the number of customers who made repeat purchases within 6 months.

5. **Identify the top 3 customers who spent the most money in each year:**
   - **Query:**

     ```
     sql
     Copy code
     SELECT customer_id, year(order_purchase_timestamp), SUM(payment_value) AS
     total_spent, RANK() OVER (PARTITION BY year(order_purchase_timestamp) ORDER
     BY SUM(payment_value) DESC) AS rank
     FROM orders
     JOIN payments ON orders.order_id = payments.order_id
     GROUP BY customer_id, year(order_purchase_timestamp)
     HAVING rank <= 3;
     ```

   - **Output:** The top 3 customers for each year were identified, with a bar plot visualizing their total spending.

Conclusion

This project demonstrated the use of SQL queries combined with Python for analyzing Target Corporation's e-commerce data. From basic queries to advanced statistical analysis, you successfully processed and visualized key insights, including customer retention rates, sales growth, and category-level revenue contributions.