E-Commerce Sales Analysis SQL Project

Structured SQL Questions and Solutions

Basic Level Questions

1. List all customer names and their region

```sql
SELECT name, region FROM customers;
```

2. Show all products with their category and price

```sql
SELECT product_name, category, ROUND(price, 2)
FROM products;
```

3. How many orders were placed in total

```sql
SELECT COUNT(order_id) AS total_order FROM orders;
```

4. Display total sales amount

```sql
SELECT ROUND(SUM(total_amount), 2) AS total_sales FROM orders;
```

5. Find all unique payment methods used

```sql
SELECT DISTINCT payment_method FROM payments;
```

6. Get product names and quantity sold

```sql
SELECT p.product_name, SUM(ot.quantity) AS total_quantity
FROM products AS p
JOIN order_items AS ot ON ot.product_id = p.product_id
GROUP BY p.product_name;
```

7. Show all orders made by customer named ”Alice”

```sql
SELECT * FROM customers AS c
JOIN orders AS o ON o.customer_id = c.customer_id
WHERE c.name = 'alice';
```

Intermediate Level Questions

1. Which product category has the highest total quantity sold

```sql
SELECT TOP 1 p.category AS product_category, SUM(ot.quantity) AS
    total_quantity
FROM products AS p
JOIN order_items AS ot ON ot.product_id = p.product_id
GROUP BY p.category;
```

2. Find top 3 customers by total amount spent

```sql
SELECT TOP 3 c.name, c.customer_id, ROUND(SUM(o.total_amount), 2) AS
    total_amount_spent
FROM customers AS c
JOIN orders AS o ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
ORDER BY SUM(o.total_amount) DESC;
```

3. List all orders along with product names and their quantity

```sql
SELECT p.product_name, o.order_id, oi.quantity
FROM order_items AS oi
JOIN orders AS o ON o.order_id = oi.order_id
JOIN products AS p ON p.product_id = oi.product_id;
```

4. Show how many orders were paid using each payment method

```sql
SELECT p.payment_method, COUNT(p.payment_method) AS order_paid
FROM payments AS p
GROUP BY p.payment_method, p.payment_status
HAVING p.payment_status = 'completed';
```

5. Find average price of products in each category

```sql
SELECT p.product_name, p.category, ROUND(AVG(p.price), 2) AS avg_price
FROM products AS p
GROUP BY p.category, p.product_name
ORDER BY p.category;
```

6. Show total revenue per region

```sql
SELECT c.region, ROUND(SUM(o.total_amount), 2) AS total_revenue
FROM customers AS c
JOIN orders AS o ON o.customer_id = c.customer_id
GROUP BY c.region;
```

7. Display customer failed payments

```sql
SELECT c.name, c.customer_id, p.payment_method, p.payment_status
FROM customers AS c
JOIN orders AS o ON o.customer_id = c.customer_id
JOIN payments AS p ON p.order_id = o.order_id
WHERE p.payment_status = 'failed';
```

Advanced Level Questions

1. For each customer show their most expensive order

```sql
WITH temp AS (
    SELECT customer_id, ROUND(MAX(total_amount), 2) AS most_expensive
    FROM orders
    GROUP BY customer_id
)
SELECT c.name, c.customer_id, temp.most_expensive
FROM temp
JOIN customers AS c ON c.customer_id = temp.customer_id;
```

2. Identify day with highest number of orders

```sql
SELECT TOP 1 order_date AS highest_sales_day
FROM orders
GROUP BY order_date
ORDER BY COUNT(customer_id) DESC;
```

3. Identify day with highest sales

```sql
SELECT TOP 1 order_date, ROUND(SUM(total_amount), 2) AS highest_sales
FROM orders
GROUP BY order_date
ORDER BY highest_sales DESC;
```

4. List all customers who bought more than 2 different product categories

```sql
WITH pro AS (
    SELECT c.customer_id, c.name, p.category
    FROM customers AS c
    JOIN orders AS o ON o.customer_id = c.customer_id
    JOIN order_items AS oi ON oi.order_id = o.order_id
    JOIN products AS p ON p.product_id = oi.product_id
)
SELECT pro.name, COUNT(DISTINCT pro.category) AS total_categories
FROM pro
GROUP BY pro.name
HAVING COUNT(DISTINCT pro.category) > 2;
```

5. Report with total orders, revenues and failed payments per customer

```sql
WITH pro AS (
    SELECT c.customer_id, c.name, o.total_amount, oi.quantity, ps.
   payment_status
    FROM customers AS c
    JOIN orders AS o ON o.customer_id = c.customer_id
    JOIN order_items AS oi ON oi.order_id = o.order_id
    JOIN products AS p ON p.product_id = oi.product_id
    JOIN payments AS ps ON ps.order_id = o.order_id
)
SELECT pro.name,
       SUM(pro.quantity) AS total_quantity,
       SUM(pro.total_amount) AS revenue,
       COUNT(CASE WHEN pro.payment_status = 'failed' THEN 1 END) AS
   failed_transaction
FROM pro
GROUP BY pro.name;
```

6. Find customers who placed more than one order and never had a failed payment

```sql
WITH pro AS (
    SELECT c.customer_id, c.name, ps.payment_status
    FROM customers AS c
    JOIN orders AS o ON o.customer_id = c.customer_id
    JOIN payments AS ps ON ps.order_id = o.order_id
)
SELECT pro.name
FROM pro
GROUP BY pro.name
HAVING COUNT(CASE WHEN pro.payment_status = 'failed' THEN 1 END) < 1
   AND COUNT(pro.name) > 1;
```

7. Rank products by total quantity sold within each category

```sql
WITH sales AS (
    SELECT p.category, p.product_name, SUM(oi.quantity) AS total_sales
    FROM products AS p
    JOIN order_items AS oi ON oi.product_id = p.product_id
    GROUP BY p.category, p.product_name
)
SELECT *, RANK() OVER (PARTITION BY category ORDER BY total_sales DESC) AS
   rank_category
FROM sales;
```

Conclusion

This project on **E-Commerce Sales Analysis using SQL** provided me with valuable hands-on experience in applying SQL to analyze real-world business data. Here are the key learnings from the project:

1. **SQL Functions:** I learned how to effectively use various SQL functions such as `COUNT()`, `SUM()`, `ROUND()`, `AVG()`, and `DISTINCT()` to perform calculations, format data, and summarize insights from large datasets.
2. **Gained Data Analysis Skills:** I developed a strong understanding of how to extract meaningful insights from raw data by writing complex queries, filtering data using conditions, and aggregating results using `GROUP BY` and `HAVING` clauses.
3. **Understood Relational Databases:** I enhanced my ability to work with multiple related tables by using different types of joins (`INNER JOIN`, `LEFT JOIN`) and subqueries, helping me understand how data is connected across different entities in a database.

Overall, this project has strengthened my foundational SQL skills and given me confidence to tackle more advanced data analysis problems in the future.