DATA ANALYST INTERNSHIP PROJECT

SQL Data Analysis Task Submission

Date: 16 February 2026

Submitted as part of internship evaluation.

1. Objective

To perform data extraction and analysis using SQL queries on an e-commerce database. The task demonstrates understanding of filtering, aggregation, joins, subqueries, views, indexing, and query optimization techniques.

2. Database Schema

```
customers (customer_id, customer_name, city, signup_date)
products (product_id, product_name, category, price)
orders (order_id, customer_id, order_date, total_amount)
order_items (order_item_id, order_id, product_id, quantity)
```

3. SQL Queries Implemented

```
-- SELECT with WHERE and ORDER BY
SELECT * FROM customers
WHERE city = 'Hyderabad'
ORDER BY signup_date DESC;

-- GROUP BY with Aggregation
SELECT customer_id, SUM(total_amount) AS total_revenue
FROM orders
GROUP BY customer_id;

-- INNER JOIN
SELECT c.customer_name, o.order_id, o.total_amount
FROM customers c
INNER JOIN orders o
ON c.customer_id = o.customer_id;

-- LEFT JOIN
SELECT c.customer_name, o.order_id
FROM customers c
LEFT JOIN orders o
ON c.customer_id = o.customer_id;

-- Subquery
SELECT customer_id, total_amount
FROM orders
WHERE total_amount > (SELECT AVG(total_amount) FROM orders);

-- ARPU Calculation
SELECT SUM(total_amount) / COUNT(DISTINCT customer_id) AS avg_revenue_per_user
FROM orders;

-- Product-wise Sales
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name;
-- View Creation
```

```
CREATE VIEW customer_total_revenue AS
SELECT c.customer_name, SUM(o.total_amount) AS total_revenue
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name;

-- NULL Handling
SELECT customer_name, COALESCE(city, 'Not Provided') AS city
FROM customers;

-- Index Creation
CREATE INDEX idx_customer_id ON orders(customer_id);
```

4. Conclusion

This project demonstrates practical knowledge of SQL for real-world data analysis. It showcases the ability to write optimized queries, perform joins, calculate metrics, create views, and manage indexing for performance improvement.