**SQL Data Analysis Project – E-commerce Database**

**□ Objective**

**The objective of this project is to practice SQL for data analysis using the Northwind E-commerce database.**
**It covers query writing, joins, subqueries, aggregate functions, views, and query optimization techniques.**
**The project also helps in preparing for SQL-related interview questions.**

**□ Project Files**

**SQL_Data_Analysis.ipynb → Notebook with SQL queries and outputs**

**Ecommerce_Analysis.sql → SQL script containing all queries**

**Screenshots/ → Contains query execution outputs (to be added after running)**

**README.md → Project documentation and interview question answers**

**□ Tools & Dataset**

**Database: Ecommerce (from Kaggle)**

**Tools: SQLite / MySQL / PostgreSQL, Jupyter Notebook**

**□ Queries Implemented**

**1. Basic SELECT, WHERE, ORDER BY queries**

**2. Aggregations using SUM, AVG, COUNT, GROUP BY**

**3. Joins – INNER, LEFT, RIGHT**

**4. Subqueries for advanced analysis**

**5. Views for simplified reporting**

**6. Indexing for query optimization**

**□ Interview Questions & Answers**

**1. Difference between WHERE and HAVING?**

**Ans: WHERE → Filters rows before aggregation.**

**HAVING → Filters groups after aggregation.**

**Example:**

```
SELECT country, COUNT(*)
FROM Customers
WHERE country != 'USA'
GROUP BY country
HAVING COUNT(*) > 5;
```

**2. Types of Joins?**

**Ans: INNER JOIN – Returns only matching rows**

**LEFT JOIN – All rows from left table + matches from right**

**RIGHT JOIN – All rows from right table + matches from left**

**FULL OUTER JOIN – All rows where a match exists in either table**

**CROSS JOIN – Cartesian product of two tables**

**3. Calculating Average Revenue per User?**

**Ans: Formula:**
**ARPU = Total Revenue ÷ Number of Unique Users**

**Example:**

```
SELECT
   SUM(order_amount) / COUNT(DISTINCT customer_id) AS avg_revenue_per_user
FROM Orders;
```

**4. Subqueries?**

**Ans: A query inside another query.**
**Can be used in SELECT, FROM, WHERE.**
**Helps simplify complex problems.**

**Example: Customers who spent above average:**

```
SELECT customer_id, SUM(order_amount) AS total_spent
FROM Orders
GROUP BY customer_id
HAVING SUM(order_amount) > (
```

```
    SELECT AVG(total)
    FROM (
        SELECT SUM(order_amount) AS total
        FROM Orders
        GROUP BY customer_id
    ) AS sub
);
```

**5. Optimizing SQL Queries?**

**Ans: Create indexes on frequently searched columns**

**Avoid SELECT *, fetch only required fields**

**Use proper joins instead of multiple subqueries**

**Partition large datasets**

**Use EXPLAIN to analyze performance**

**6. Views in SQL?**

**Ans: A view is a saved SQL query that behaves like a virtual table.**

**Simplifies complex queries and improves readability.**

**Can also be used for security by exposing only selected columns.**

**Example:**

```
CREATE VIEW customer_sales AS
SELECT customer_id, SUM(order_amount) AS total_sales
FROM Orders
GROUP BY customer_id;
```

**7. Handling NULL values?**

**Ans: Check: IS NULL / IS NOT NULL**

**Replace: COALESCE(column, 'default') or IFNULL(column, 'default')**

**Conditional: CASE WHEN column IS NULL THEN 'default' ELSE column END**

**Example:**

```
SELECT customer_name, COALESCE(phone, 'Not Provided') AS phone_number
FROM Customers;
```