SQL CONCEPTS THAT WE HAVE LEARNT

1. Basic SELECT Query

- **SQL Concept**: **SELECT** is the most fundamental query used to retrieve data from one or more tables.

  Example:

  ```
  SELECT * FROM Sales;
  ```

- **Learning Focus**:
  - Retrieving all columns from the `Sales` table.
  - Understanding the basic structure of SQL queries (using `SELECT` and `FROM`).
  - Extracting raw data without any filters or conditions.

2. Filtering with WHERE Clause

- **SQL Concept**: The **WHERE** clause is used to filter records based on specific conditions.

  Example:

```
sql
```

```
SELECT * FROM Sales WHERE QuantitySold > 10;
```

- **Learning Focus**:
  - Applying conditions to retrieve only certain rows based on a given filter.
  - Using comparison operators (>, <, =, etc.) to narrow down results.

3. JOINs (Combining Multiple Tables)

- **SQL Concept**: **JOIN** statements combine rows from two or more tables based on related columns.

Example:

```
sql

        SELECT p.SKU, p.DesignNo, SUM(s.QuantitySold) AS TotalSold
        FROM Sales s
        JOIN Product p ON s.SKU = p.SKU
        GROUP BY p.SKU, p.DesignNo;
```

- **Learning Focus**:
  - **INNER JOIN**: Combines only the matching rows from the `Sales` and `Product` tables.
  - Understanding **table relationships** and how to pull data from multiple tables.
  - Learning how to match data using a common key (`SKU` in this case).

4. Aggregation Functions (SUM, COUNT, AVG)

- **SQL Concept**: Aggregation functions like **SUM**, **COUNT**, **AVG** are used to perform calculations on multiple rows.

  Example:

```
sql

        SELECT SUM(QuantitySold) AS TotalQuantitySold FROM Sales;
```

- **Learning Focus**:
  - Using **SUM()** to calculate the total of a numerical column.
  - Applying other aggregation functions like **COUNT()** for counting rows or **AVG()** for averages.
  - Learning to summarize data from many rows into a single result.

5. GROUP BY for Aggregation

- **SQL Concept**: **GROUP BY** groups rows that have the same values in specified columns, allowing aggregation on these groups.

  Example:

```
sql

        SELECT p.SKU, SUM(s.QuantitySold) AS TotalSold
        FROM Sales s
        JOIN Product p ON s.SKU = p.SKU
        GROUP BY p.SKU;
```

- **Learning Focus**:
  - Aggregating results per **SKU** or any other column.
  - Learning how to group data based on common attributes and then apply aggregate functions like **SUM** or **COUNT**.
  - Useful for creating summaries and reports.

6. ORDER BY (Sorting Results)

- **SQL Concept**: **ORDER BY** is used to sort the result set by one or more columns.

  Example:

  sql

```
SELECT p.SKU, SUM(s.QuantitySold) AS TotalSold
FROM Sales s
JOIN Product p ON s.SKU = p.SKU
GROUP BY p.SKU
ORDER BY TotalSold DESC;
```

- **Learning Focus**:
  - Sorting results in **ascending** or **descending** order using the **ORDER BY** clause.
  - Understanding how to present aggregated data in a meaningful way by sorting based on key metrics (like total sales).

7. Subqueries (Nested Queries)

- **SQL Concept**: **Subqueries** are queries embedded inside another query, often used to derive intermediate results.
- **Example**:

  sql

```
SELECT
    (SELECT SUM(QuantitySold * SalePrice) FROM Sales) AS TotalRevenue,
    (SELECT SUM(Amount) FROM Expense) AS TotalExpenses;
```

- **Learning Focus**:
  - Using subqueries to calculate multiple independent metrics in a single query (e.g., revenue and expenses).
  - Understanding how to nest one query inside another to simplify complex calculations.

8. Alias for Columns and Tables

- **SQL Concept**: **Aliases** give temporary names to columns or tables, making queries easier to read.

  Example:

  sql

  ```
  SELECT s.SKU, SUM(s.QuantitySold) AS TotalQuantity
  FROM Sales s;
  ```

- **Learning Focus**:
  - Using **AS** to assign aliases to columns and tables.
  - Understanding how to shorten long table names (e.g., `Sales` becomes `s`) and give meaningful names to calculated columns (e.g., `TotalQuantity`).

9. Mathematical Calculations

- **SQL Concept**: SQL allows for basic arithmetic calculations directly in the query.

  Example:

  sql

  ```
  SELECT QuantitySold * SalePrice AS Revenue
  FROM Sales;
  ```

- **Learning Focus**:
  - Performing calculations like **multiplication**, **addition**, **subtraction**, and **division** in the query.
  - Learning to derive new metrics such as **Revenue** or **Profit** from existing columns.

10. Conditional Logic (CASE Statements)

- **SQL Concept**: The **CASE** statement allows you to implement conditional logic directly in SQL queries.

Example:

```
sql


            SELECT
                SKU,
                CASE
                    WHEN QuantitySold > 100 THEN 'High Sales'
                    ELSE 'Low Sales'
                END AS SalesCategory
            FROM Sales;
```

- **Learning Focus**:
    - Using conditional statements to categorize or transform data in a query.
    - Learning to apply **if-else** type logic directly in SQL, which is useful for creating categorized reports or data segmentation.

Summary of Key SQL Learning Areas:

- **Data Retrieval (SELECT)**: Learnt how to extract data from tables and filter results using conditions.
- **Data Manipulation (JOINs, WHERE, GROUP BY)**: Understood how to combine, filter, and group data from multiple tables.
- **Aggregation and Summarization**: Learnt how to use SQL functions like **SUM()**, **COUNT()**, and **AVG()** to summarize data and generate useful metrics.
- **Sorting and Organizing Data (ORDER BY)**: Learnt how to sort data in a way that highlights the most relevant results.
- **Mathematical Operations and Subqueries**: Gained skills in performing calculations and using subqueries for more complex data extraction.