CDS DS 594

Chapter Two: Understanding SQL

1. Introduction
    a. Structured Query Language (SQL) is backbone of modern relational database systems by providing powerful and versatile means of managing and querying data
2. SELECT Clause
    a. Allows to retrieve data from one or more tables.
    b. Syntax:
    SELECT column1, column2, …
    FROM table_name;
    c. Another Syntax:
    SELECT * (all)
    FROM employees;
3. WHERE Clause
    a. Allows to filter the data based on specific conditions.
    b. Essential when narrowing down the results of a query
    c. Syntax:
    SELECT column1, column2, …
    FROM table_name
    WHERE condition;
    d. Example:
    SELECT first_name, last_name
    FROM employees
    WHERE department = “HR”;
4. GROUP BY Clause
    a. Used to group rows from a result set based on values of one or more columns
    b. Useful when need to perform aggregate functions (such as SUM, COUNT, AVG) on groups of data
    c. Syntax:
    SELECT column1, aggregate_function(column2)
    FROM table_name
    GROUP BY column1
    d. Example:
    SELECT category, SUM(sales)
    FROM products
    GROUP BY category;

e. The following would group the products by category and calculate total sales for each category

5. JOIN Clause
    a. Data is distributed across multiple tables
    b. JOIN clause allows to combine data from two or more tables based on related column between them
    c. Syntax:

```
SELECT column1, column2, …
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

    d. Several joins
        i. INNER JOIN
            1. Also known as equi-join
            2. Returns only rows that have matching values in both left (table 1) and right (table 2) based on the specified join condition
            3. If there are no matching rows in either table, those rows are not included in the result set
            4. Example:

```
SELECT customers.customer_id, customers.first_name,
orders.order
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id
```

            5. The following joins “customers” and “orders” tables, retrieving customer information and order dates where there’s match on the “customer_id” column
        ii. LEFT JOIN
            1. Returns all the rows from the left (table 1) and the matching rows from the right (table 2)
            2. If no matches in the right table, the result will include all rows from the left table
            3. Useful for situations where you want to see all records from one table and the matching records from another table
            4. Example:

```
SELECT customers.customer_id, orders.order_date
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id;
```

5. Returns all customers, along with their order dates if they have made any orders
6. Customers without orders will have NULL values in the "order_date" column

iii. RIGHT JOIN

1. Reverse of the LEFT JOIN
2. Returns all rows from the right (table 2) and the matching rows from the left (table 1)
3. If there are no matches in the left table, result will still include all rows from the right table
4. Example:
   SELECT customers.customer_id, orders.order_date
   FROM customers
   RIGHT JOIN orders
   ON customers.customer_id = orders.customer_id;
5. Return all orders along with the customer information if available. Orders without a matching customer will have NULL values in the "customer_id" and customer-related columns

iv. FULL JOIN

1. Returns all rows from both left and right tables, regardless of whether they have matching values.
2. If there is no match for a particular row in one of the tables, the result will include NULL values for columns from the table that doesn't have a match
3. Example:
   SELECT customers.customer_id, orders.order_date
   FROM customers
   FULL JOIN orders
   ON customers.customer_id = orders.customer_id;
4. Query will return all customers and all orders, linking them when there is a match. If no match, corresponding columns will contain NULL values

e. In summary, choice between INNER, LEFT, RIGHT, FULL join depends on specific requirements of your query.
f. INNER JOIN is used when you only want matching records
g. LEFT and RIGHT JOINs are used when you want all records from one table and matching records from the other
h. A FULL JOIN is used when you want all records from both tables, with NULL values for non-matching records.

6. Database Technologies
    a. Several major database technologies, each with its own variations of SQL syntax and features.
        i. MySQL
            1. MySQL is an open-source relational database management system (RDBMS).
            2. It uses a SQL dialect that is very similar to the ANSI SQL standard with some MySQLspecific extensions.
            3. MySQL supports features like user-defined variables and non-standard SQL functions.
        ii. PostgreSQL
            1. PostgreSQL is another open-source RDBMS known for its extensibility and support for advanced data types.
            2. It adheres closely to the ANSI SQL standard and provides a rich set of SQL features.
            3. PostgreSQL has advanced support for procedural languages and custom functions.
        iii. SQLite
            1. SQLite is a lightweight, serverless, and self-contained database engine.
            2. Its SQL syntax closely follows the SQL-92 standard.
            3. SQLite is known for its simplicity and portability.
        iv. Microsoft SQL Server
            1. Microsoft SQL Server is a commercial RDBMS developed by Microsoft.
            2. It has its own Transact-SQL (T-SQL) language, which extends ANSI SQL.
            3. T-SQL includes features like stored procedures, triggers, and functions that are specific to SQL Server.
        v. Oracle Database
            1. Oracle Database is a powerful commercial RDBMS developed by Oracle Corporation.
            2. It uses the Oracle Database SQL dialect, which is compliant with ANSI SQL with Oracle-specific extensions.
            3. Oracle Database is known for its scalability and support for high-end enterprise applications.
        vi. Snowflake
            1. Snowflake is a cloud-based data warehousing platform known for its elasticity and ease of use.

2. Snowflake uses a variant of SQL known as “Snowflake SQL” or “SnowSQL.”
3. Snowflake’s SQL syntax is similar to ANSI SQL with some unique features and optimizations specific to Snowflake.
4. Snowflake is designed for data warehousing and supports semi-structured data, making it suitable for data analytics.

vii. Google BigQuery

1. Google BigQuery is a serverless, highly scalable, and fully-managed data warehouse service in the Google Cloud Platform.
2. It uses a version of SQL known as “BigQuery SQL.”
3. BigQuery SQL follows ANSI SQL standards but also introduces some extensions, particularly for handling large datasets and distributed computing.
4. BigQuery is optimized for running complex, high-performance analytical queries on vast amounts of data

b. SQL syntax can vary significantly between different database technologies due to the specific features and requirements of each system
c. While database adhere to ANSI SQL standards to some extent, they often introduce their own extensions and optimizations

7. Conclusion
   a. SQL is a powerful language for managing and querying relational databases.
   b. The SELECT, WHERE, GROUP BY, and JOIN clauses are essential tools for retrieving and manipulating data.
   c. Understanding how to use these clauses effectively is crucial for working with databases and deriving valuable insights from your data.