Introduction to SQL

Overview:

SQL (Structured Query Language) is a programming language used to manage and manipulate relational databases.

It allows users to interact with databases like MySQL, PostgreSQL, and SQL Server. SQL is fundamental for database management

and is widely used in both small applications and large-scale systems.

Key Concepts:

- What is SQL?

SQL is the language used to query, update, and manage relational databases.

- Purpose of SQL:

Used for database creation, modification, and querying.

- Relational Database Systems:

Examples include MySQL, PostgreSQL, SQL Server.

- Tables:

The basic structure for storing data in a relational database.

- Queries:

SQL queries allow users to retrieve, update, and manipulate data.

Practical Applications:

- Creating Databases:

Using SQL to create databases and define their structure.

- Querying Data:

Retrieving data using SELECT statements.

- Managing Data:

Inserting, updating, or deleting data as needed.

Understanding SQL Queries

Overview:

SQL queries are the commands that interact with the data stored in relational databases. The most common query is the SELECT

statement, which retrieves data based on specified conditions.

Key Concepts:

- SELECT Statement:

Used to retrieve data from one or more tables.

- WHERE Clause:

Used to filter results based on a condition.

- ORDER BY Clause:

Used to sort results based on specific columns, in ascending or descending order.

Example Query:

SELECT * FROM employees WHERE department = 'Sales' ORDER BY last_name;

Practical Applications:

- Filter Data:

Use the WHERE clause to narrow results based on specific conditions.

- Sort Data:

Order data alphabetically, numerically, or by any column.

Working with Tables

Overview:

SQL provides commands for creating, modifying, and deleting tables, which are the core storage structures in a database.

Understanding data types and constraints is important for maintaining data integrity.

Key Concepts:

- CREATE TABLE Statement:

Used to create new tables in a database.

- ALTER TABLE Statement:

Modifies the structure of an existing table.

- DROP TABLE Statement:

Deletes a table and all its data permanently.

Example Syntax:

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    department VARCHAR(50)
);
```

Practical Applications:

- Table Creation:

Design and implement new tables based on data requirements.

- Modify Tables:

Alter tables to accommodate changes in the business model.

Data Manipulation

Overview:

Data manipulation commands allow you to insert, update, and delete records in tables. Proper use of these commands ensures

data consistency and integrity.

Key Concepts:

- INSERT INTO Statement:

Adds new data to a table.

- UPDATE Statement:

Modifies existing records in a table.

- DELETE Statement:

Removes data from a table.

Example Syntax:

```
INSERT INTO employees (employee_id, first_name, last_name, department)
VALUES (1, 'John', 'Doe', 'Sales');

UPDATE employees SET department = 'Marketing' WHERE employee_id = 1;
```

```
DELETE FROM employees WHERE employee_id = 1;
```

Practical Applications:

- Insert New Records:

Add new data when a new employee or product is added.

- Update Records:

Modify existing data to reflect business changes.

- Delete Records:

Remove obsolete or irrelevant data.

SQL Joins and Aggregations

Overview:

SQL Joins are used to combine data from multiple tables based on related columns. Aggregation functions summarize or analyze

data, such as calculating totals or averages.

Key Concepts:

- INNER JOIN:

Returns rows where there is a match in both tables.

- LEFT JOIN:

Returns all rows from the left table and matching rows from the right table.

- RIGHT JOIN:

Returns all rows from the right table and matching rows from the left table.

- Aggregation Functions:

Functions like COUNT, SUM, AVG, MIN, and MAX to perform operations on data.

Example Query:

```
SELECT employees.first_name, employees.last_name, departments.department_name
FROM employees
INNER JOIN departments ON employees.department_id = departments.department_id;
```

Practical Applications:

- Combining Data from Multiple Tables:

Join related tables like employees and departments to get comprehensive information.

- Summarize Data:

Use aggregation functions to calculate totals, averages, or counts for reporting.