SQL Basics

1. SELECT: Retrieving Data

- The `SELECT` statement is used to retrieve data from a database.

Basic Syntax:
sql

```
SELECT column1, column2, ...
FROM table_name;
```

-

Example:
sql

```
SELECT first_name, last_name
FROM employees;
```

-

Selecting All Columns:
sql

```
SELECT * FROM employees;
```

-

Filtering Data with WHERE:
sql

```
SELECT first_name, last_name
FROM employees
WHERE department = 'Sales';
```

-

Sorting Results with ORDER BY:
sql

```
SELECT first_name, last_name
FROM employees
ORDER BY last_name ASC;
```

-
- **Using Aggregate Functions**:
    - `COUNT`, `SUM`, `AVG`, `MIN`, `MAX` are examples of aggregate functions.

sql

```
SELECT COUNT(*)
FROM employees;
```

-

2. INSERT: Adding Data

- The `INSERT` statement is used to add new rows to a table.

Basic Syntax:
sql

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

-

Example:
sql

```
INSERT INTO employees (first_name, last_name, department)
VALUES ('John', 'Doe', 'HR');
```

-

Inserting Multiple Rows:
sql

```
INSERT INTO employees (first_name, last_name, department)
VALUES ('Jane', 'Smith', 'Finance'),
       ('Bob', 'Johnson', 'Sales');
```

-

3. UPDATE: Modifying Data

- The `UPDATE` statement is used to modify existing rows in a table.

Basic Syntax:
sql

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

-

Example:
sql

```
UPDATE employees
SET department = 'Marketing'
WHERE last_name = 'Doe';
```

-

Updating Multiple Columns:
sql

```
UPDATE employees
SET first_name = 'John', department = 'Finance'
WHERE last_name = 'Doe';
```

-
- **Caution**: Without a `WHERE` clause, `UPDATE` modifies all rows.

4. DELETE: Removing Data

- The `DELETE` statement is used to remove rows from a table.

Basic Syntax:
sql

```
DELETE FROM table_name
WHERE condition;
```

-

Example:
sql

```
DELETE FROM employees
WHERE last_name = 'Doe';
```

-

Deleting All Rows:
sql

```
DELETE FROM employees;
```

-
- **Caution**: Without a `WHERE` clause, `DELETE` removes all rows from the table.

5. JOIN: Combining Data from Multiple Tables

- `JOIN` operations are used to combine rows from two or more tables based on a related column.
- **Types of Joins**:
  - **INNER JOIN**: Returns rows with matching values in both tables.
  - **LEFT JOIN**: Returns all rows from the left table and the matched rows from the right table.
  - **RIGHT JOIN**: Returns all rows from the right table and the matched rows from the left table.
  - **FULL JOIN**: Returns all rows when there is a match in either left or right table.

INNER JOIN Example:
sql

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

-

LEFT JOIN Example:
sql

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

-

Summary

- **SELECT**: Retrieve data from one or more tables.
- **INSERT**: Add new data to a table.
- **UPDATE**: Modify existing data in a table.
- **DELETE**: Remove data from a table.
- **JOIN**: Combine rows from multiple tables based on related columns.