SQL Basics, CRUD, Joins, GROUP BY, and Window Functions

1. SQL Basics

What is SQL?

- **SQL (Structured Query Language)**: A programming language used to interact with relational databases.
- **Purpose**:
  - Store, retrieve, manipulate, and manage data in databases.
- **Components**:
  - **Data Definition Language (DDL)**: Define database schema (e.g., `CREATE`, `ALTER`, `DROP`).
  - **Data Manipulation Language (DML)**: Modify data (e.g., `INSERT`, `UPDATE`, `DELETE`).
  - **Data Query Language (DQL)**: Retrieve data (e.g., `SELECT`).
  - **Data Control Language (DCL)**: Control access (e.g., `GRANT`, `REVOKE`).

2. CRUD Operations

CRUD

CRUD stands for **Create**, **Read**, **Update**, and **Delete** – the four basic operations in relational databases.

Examples

- **Create**: Insert new records into a table.

```sql
INSERT INTO employees (id, name, age, position)
VALUES (1, 'Alice', 30, 'Engineer');
```

- **Read**: Retrieve data from a table.

```sql
SELECT * FROM employees;
```

- **Update**: Modify existing records.

```sql
UPDATE employees
SET age = 31
WHERE id = 1;
```

- **Delete**: Remove records from a table.

```sql
DELETE FROM employees
WHERE id = 1;
```

3. Joins

Definition

Joins combine rows from two or more tables based on related columns.

Types of Joins

- **Inner Join**: Returns rows with matching values in both tables.

```sql
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.id;
```

- **Left Join**: Returns all rows from the left table, with matching rows from the right table (if any).

```sql
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.id;
```

- **Right Join**: Returns all rows from the right table, with matching rows from the left table (if any).

```sql
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.id;
```

- **Full Outer Join**: Returns rows when there's a match in either table.

```sql
SELECT employees.name, departments.department_name
FROM employees
FULL OUTER JOIN departments
ON employees.department_id = departments.id;
```

4. GROUP BY Clause

Definition

Groups rows that have the same values in specified columns and allows aggregate functions like `COUNT`, `SUM`, `AVG`, `MAX`, and `MIN`.

Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name;
```

Example

Find the total salary for each department:

```
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id;
```

5. Window Functions

Definition

Perform calculations across a set of table rows related to the current row, without collapsing rows like `GROUP BY`.

Syntax

```
SELECT column_name,
       window_function() OVER (PARTITION BY column_name ORDER BY
column_name) AS alias
FROM table_name;
```

Common Window Functions

- **ROW_NUMBER()**: Assigns a unique number to each row.

  ```
  SELECT name, ROW_NUMBER() OVER (ORDER BY salary DESC) AS rank
  FROM employees;
  ```

- **RANK()**: Assigns a rank to rows, with ties receiving the same rank and skipping subsequent numbers.

  ```
  SELECT name, RANK() OVER (ORDER BY salary DESC) AS rank
  FROM employees;
  ```

- **DENSE_RANK()**: Similar to `RANK()` but without gaps in ranking.

  ```
  SELECT name, DENSE_RANK() OVER (ORDER BY salary DESC) AS rank
  FROM employees;
  ```

- **SUM()**: Computes running totals.

  ```
  SELECT department_id, salary,
         SUM(salary) OVER (PARTITION BY department_id ORDER BY id)
  AS running_total
  FROM employees;
  ```

Summary of Key Concepts

Concept	Definition	Example Use
CRUD	Basic operations to create, read, update, and delete data.	Manage employee records.
Joins	Combine rows from multiple tables based on relationships.	Link employees with their departments.
GROUP BY	Aggregate data grouped by one or more columns.	Total salary per department.
Window Functions	Perform calculations over a subset of data, maintaining row structure.	Ranking employees by salary.

Conclusion

- **SQL Basics** form the foundation for working with relational databases.
- **CRUD** operations handle fundamental data interactions.
- **Joins**, **GROUP BY**, and **Window Functions** offer advanced tools for querying and analyzing complex datasets efficiently.