# Introduction to SQL Queries

Introduction to SQL Queries

**Definition:**

- SQL (Structured Query Language) is a standard language for interacting with databases.
- SQL queries are used to perform tasks such as retrieving, updating, inserting, and deleting data.

**Basic SQL Commands:**

- `SELECT`: Retrieve data from a database.
- `INSERT`: Add new data into a database.
- `UPDATE`: Modify existing data.
- `DELETE`: Remove data from a database.

# SELECT Query Basics

Syntax:

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

Example:

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

**Key Points:**

- Use * to select all columns.
- Column names are case-insensitive, but best practice is to keep them consistent.

# Filtering Data with WHERE

Syntax:

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

Example:

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

**Key Points:**

- Operators: `=`, `>`, `<`, `>=`, `<=`, `<>` (not equal), `LIKE`, `IN`, `BETWEEN`
- Use `AND`, `OR` to combine multiple conditions.

# Using LIKE for Pattern Matching

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

Example:

```
SELECT first_name, last_name
FROM employees
WHERE first_name LIKE 'A%';
```

**Key Points:**

- `%` represents zero or more characters.
- `_` represents a single character.

# Sorting Data with ORDER BY

Syntax:

SELECT column1, column2, ...

FROM table_name

ORDER BY column1 [ASC|DESC];

Example:

SELECT first_name, last_name

FROM employees

ORDER BY last_name ASC;

**Key Points:**

- `ASC` is for ascending (default), `DESC` for descending.

# Using Aggregate Functions

**Functions:**

- `COUNT()`: Returns the number of rows.
- `SUM()`: Returns the total sum of a column.
- `AVG()`: Returns the average value.
- `MAX()`: Returns the maximum value.
- `MIN()`: Returns the minimum value.

Example:

```
SELECT COUNT(*), AVG(salary)

FROM employees;
```

# Grouping Data with GROUP BY

Syntax:

```
SELECT column1, COUNT(*)
FROM table_name
GROUP BY column1;
```

Example:

```
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

**Key Points:**

- Use `GROUP BY` to group rows that have the same values in specified columns.
- Often used with aggregate functions like `COUNT()`, `SUM()`, etc.

# Filtering Groups with HAVING

Syntax:

```
SELECT column1, COUNT(*)

FROM table_name

GROUP BY column1

HAVING COUNT(*) > value;
```

# Example:

```
SELECT department, COUNT(*)

FROM employees

GROUP BY department

HAVING COUNT(*) > 10;
```

**Key Points:**

- `HAVING` is used to filter records after grouping.
- Use `HAVING` instead of `WHERE` when filtering aggregates.

# Joining Tables (INNER JOIN)

Syntax:

```
SELECT columns

FROM table1

INNER JOIN table2

ON table1.column = table2.column;
```

# Example:

```
SELECT employees.first_name, departments.department_name

FROM employees

INNER JOIN departments

ON employees.department_id = departments.department_id;
```

**Key Points:**

- `INNER JOIN` returns rows when there is a match in both tables.

# Left Join (LEFT OUTER JOIN)

Syntax:

```
SELECT columns

FROM table1

LEFT JOIN table2

ON table1.column = table2.column;
```

# Example:

```
SELECT employees.first_name, departments.department_name

FROM employees

LEFT JOIN departments

ON employees.department_id = departments.department_id;
```

**Key Points:**

- `LEFT JOIN` returns all rows from the left table, even if there are no matches in the right table.

# Inserting Data (INSERT INTO)

Syntax:

INSERT INTO table_name (column1, column2, ...)

VALUES (value1, value2, ...);

Example:

INSERT INTO employees (first_name, last_name, department)

VALUES ('John', 'Doe', 'Sales');

**Key Points:**

- Columns and values must match in order and type.

# Updating Data (UPDATE)

Syntax:

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

Example:

```
UPDATE employees
SET department = 'Marketing'
WHERE employee_id = 101;
```

**Key Points:**

- Use `WHERE` to specify which rows to update; otherwise, all rows will be updated.

# Deleting Data (DELETE FROM)

Syntax:

DELETE FROM table_name

WHERE condition;

Example:

DELETE FROM employees

WHERE employee_id = 101;

**Key Points:**

- Use `WHERE` to specify which rows to delete; otherwise, all rows will be deleted.

# Creating a Table (CREATE TABLE)

Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
);
```

# Example:

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);
```

**Key Points:**

- Define column data types and constraints (e.g., `PRIMARY KEY`, `NOT NULL`, etc.).

# Altering a Table (ALTER TABLE)

ALTER TABLE table_name

ADD column_name datatype;

Example:

ALTER TABLE employees

ADD hire_date DATE;

**Key Points:**

- Use `ALTER TABLE` to add, modify, or delete columns.

# Dropping a Table (DROP TABLE)

DROP TABLE table_name;

Example:

DROP TABLE employees;

**Key Points:**

- Use `DROP TABLE` to permanently delete a table and its data.