# DATABASE TESTING

What is SQL?

SQL is the standard language for dealing with Relational Databases. SQL is used to insert, search, update, and delete database records.

**CREATE DATABASE Statement**

The `CREATE DATABASE` statement is used to create a new SQL database.

Syntax: `CREATE DATABASE databasename;`

DROP DATABASE Statement

The `DROP DATABASE` statement is used to drop an existing SQL database.

Syntax: `DROP DATABASE databasename;`

CREATE TABLE Statement

The `CREATE TABLE` statement is used to create a new table in a database.

Syntax:
```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype, ....);
```

Sample:
```
CREATE TABLE Persons (
     ID int NOT NULL,
     LastName varchar(255) NOT NULL,
     FirstName varchar(255),
     Age int,
     PRIMARY KEY (ID)
);
```

## AUTO_INCREMENT Keyword

By default, the starting value for `AUTO_INCREMENT` is 1, and it will increment by 1 for each new record.

```
CREATE TABLE Persons (Personid int NOT NULL
AUTO_INCREMENT,    LastName varchar(255) NOT NULL,
FirstName varchar(255),Age int,PRIMARY KEY (Personid));
```

## INSERT INTO Statement

The `INSERT INTO` statement is used to insert new records in a table.

Syntax:
```
INSERT INTO table_name (column1, column2, column3,
...) VALUES (value1, value2, value3, ...);
```

```
Sample: INSERT INTO Customers (CustomerName, ContactName,
Address, City, PostalCode, Country) VALUES ('Cardinal',
'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006',
'Norway');
```

## SELECT Statement

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

Syntax: `SELECT column1, column2, …  FROM table_name;`

Sample: `SELECT * FROM Customers;`

## WHERE Clause

The `WHERE` clause is used to filter records.

WHERE Syntax: `SELECT column1, column2, … FROM table_name WHERE condition;`

## MySQL AND and OR Operators

The `WHERE` clause can be combined with `AND`, `OR` operators.

The `AND` and `OR` operators are used to filter records based on more than one condition:

- The `AND` operator displays a record if all the conditions separated by `AND` are TRUE.
- The `OR` operator displays a record if any of the conditions separated by `OR` is TRUE.

AND Syntax: `SELECT column1, column2, …  FROM table_name WHERE condition1 AND condition2 AND condition3 ...;`

OR Syntax: `SELECT column1, column2, … FROM table_name WHERE condition1 OR condition2 OR condition3 ...;`

**ORDER BY Keyword**

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

ORDER BY Syntax: `SELECT column1, column2, … FROM table_name ORDER BY column1, column2, ... ASC|DESC;`

UPDATE Statement

The `UPDATE` statement is used to modify the existing records in a table.

UPDATE Syntax: `UPDATE table_name SET column1 = value1, column2 = value2, … WHERE condition;`

Sample: `UPDATE Customers SET ContactName = 'Alfred Schmidt', City = 'Frankfurt' WHERE CustomerID = 1;`

DELETE Statement

The `DELETE` statement is used to delete existing records in a table.

DELETE Syntax: `DELETE FROM table_name WHERE condition;`

Example: `DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';`

LIMIT Clause

The `LIMIT` clause is used to specify the number of records to return.

LIMIT Syntax: `SELECT column_name(s) FROM table_name WHERE condition LIMIT number;`

Example: `SELECT * FROM Customers LIMIT 3;`

LIKE Operator

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

Syntax: `SELECT column1, column2, … FROM table_name WHERE columnN LIKE pattern;`

| **LIKE Operator** | **Description** |
|---|---|
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |

|  |  |
|---|---|
| WHERE CustomerName LIKE '_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a__%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

BETWEEN Operator

The `BETWEEN` operator selects values within a given range. The values can be numbers, text, or dates.

Syntax: `SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;`

*Example:* `SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;`

MySQL Joins

Reference: https://www.w3schools.com/MySQL/mysql_join.asp

## PRIMARY KEY Constraint

The `PRIMARY KEY` constraint uniquely identifies each record in a table.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

## FOREIGN KEY Constraint

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(ID)
);
```

## For Testing Sample Table:

```
CREATE TABLE EMPLOYEE
(
   EmpCode       INT(4),
   EmpFName      VARCHAR(255),
   EmpLName      VARCHAR(255),
   Job           VARCHAR(255),
   Manager       CHAR(4),
   HireDate      DATE,
   Salary        INT(6),
   Commission    INT(6),
   DEPTCODE      INT(2)
);


INSERT INTO EMPLOYEE
VALUES (9369, 'TONY', 'STARK', 'SOFTWARE ENGINEER', 7902, '1980-12-17', 2800,0,20),
       (9499, 'TIM', 'ADOLF', 'SALESMAN', 7698, '1981-02-20', 1600, 300,30),
       (9566, 'KIM', 'JARVIS', 'MANAGER', 7839, '1981-04-02', 3570,0,20),
       (9654, 'SAM', 'MILES', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30),
       (9782, 'KEVIN', 'HILL', 'MANAGER', 7839, '1981-06-09', 2940,0,10),
       (9788, 'CONNIE', 'SMITH', 'ANALYST', 7566, '1982-12-09', 3000,0,20),
       (9839, 'ALFRED', 'KINSLEY', 'PRESIDENT', 7566, '1981-11-17', 5000,0, 10),
       (9844, 'PAUL', 'TIMOTHY', 'SALESMAN', 7698, '1981-09-08', 1500,0,30),
       (9876, 'JOHN', 'ASGHAR', 'SOFTWARE ENGINEER', 7788, '1983-01-12',3100,0,20),
       (9900, 'ROSE', 'SUMMERS', 'TECHNICAL LEAD', 7698, '1981-12-03', 2950,0, 20),
       (9902, 'ANDREW', 'FAULKNER', 'ANAYLYST', 7566, '1981-12-03', 3000,0, 10),
       (9934, 'KAREN', 'MATTHEWS', 'SOFTWARE ENGINEER', 7782, '1982-01-23', 3300,0,20),
       (9591, 'WENDY', 'SHAWN', 'SALESMAN', 7698, '1981-02-22', 500,0,30),
       (9698, 'BELLA', 'SWAN', 'MANAGER', 7839, '1981-05-01', 3420, 0,30),
       (9777, 'MADII', 'HIMBURY', 'ANALYST', 7839, '1981-05-01', 2000, 200, NULL),
       (9860, 'ATHENA', 'WILSON', 'ANALYST', 7839, '1992-06-21', 7000, 100, 50),
       (9861, 'JENNIFER', 'HUETTE', 'ANALYST', 7839, '1996-07-01', 5000, 100, 50);
```