SQL Database

Baseer Ahmed Tahir

Table of Contents

Structured Query Language

City	Population (millions)	GDP (billion USD)
Karachi	14.91	164.0
Lahore	11.13	84.0
Islamabad	1.015	18.0
Quetta	1.001	4.2

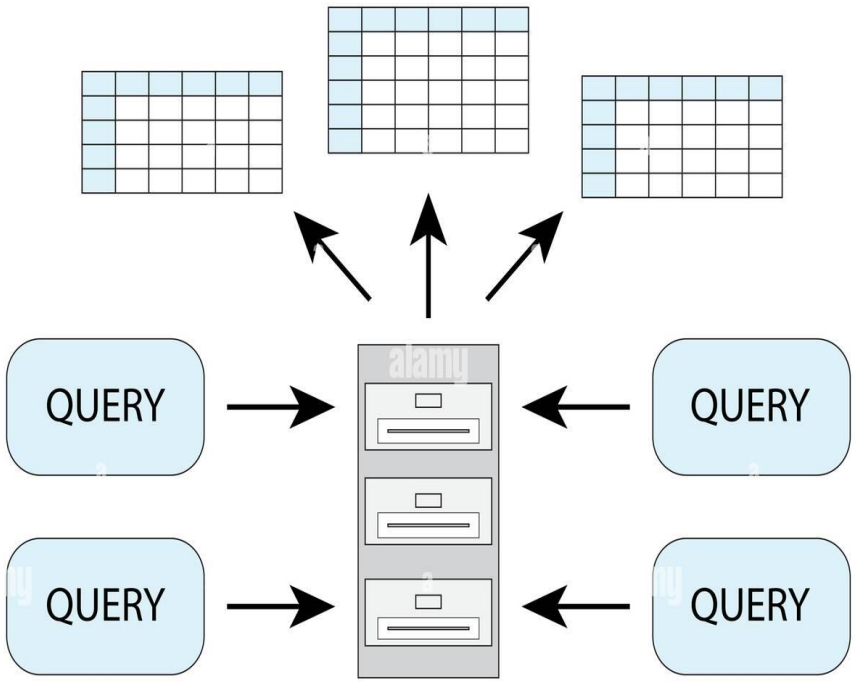
Introduction to SQL:

- SQL stands for Structured Query Language.
- It is a standard language for managing relational databases.

Key Components:

- SQL consists of commands for querying, updating, and managing databases.
- It includes data definition language (DDL), data manipulation language (DML), and data control language (DCL) commands.

Querying Single Table in SQL



Fetch all columns from single table

- **Syntax:** SELECT * FROM table_name;
- **Example:** SELECT * FROM employees;

Fetch specific columns from a single table:

- **Syntax:** SELECT column1, column2, ... FROM table_name WHERE condition;
- **Example:** SELECT name, age FROM customers;

Fetch specific columns and order the results:

- **Syntax:** SELECT column1, column2, ... FROM table_name ORDER BY column_name [ASC|DESC];
- **Example:** SELECT name FROM city ORDER BY rating ASC;

ALIASES

- **Syntax:** SELECT column_name AS alias_name FROM table_name;
- **Example:** SELECT first_name AS 'First', last_name AS 'Last' FROM employees;

Column Alias:

- SELECT first_name AS 'First', last_name AS 'Last' FROM employees;

Table Alias:

- SELECT e.first_name, d.department_name FROM employees AS e
- JOIN departments AS d ON e.department_id = d.department_id;

Alias for Aggregated Functions:

- SELECT AVG(salary) AS 'AverageSalary' FROM employees;

FILTERING THE OUTPUT

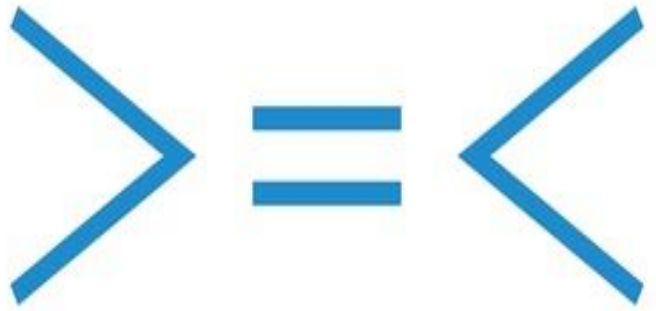
COMPARISON OPERATORS

Equal to Operator (=):

SELECT * FROM employees WHERE department = 'Sales'

Not Equal to Operator (<>, !=):

SELECT * FROM products WHERE price <> 100;



Greater Than Operator (>):

SELECT * FROM orders WHERE total_amount > 500;

Less Than Operator (<):

SELECT * FROM customers WHERE age < 30;

Greater Than or Equal to Operator (>=):

SELECT * FROM products WHERE stock_quantity >= 100;

Less Than or Equal to Operator (<=):

SELECT * FROM employees WHERE salary <= 50000;

JOINS

Types and syntax

INNER JOIN:

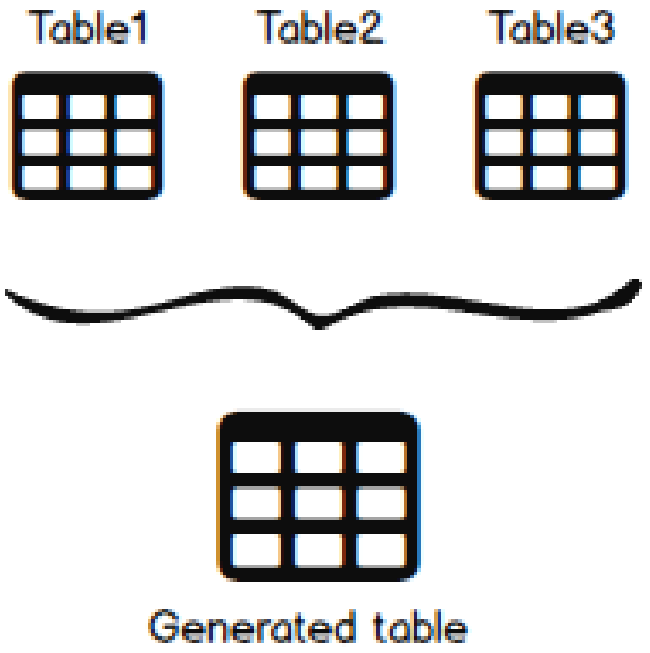
```
SELECT Students.Name, Courses.CourseName
FROM Students
INNER JOIN Courses ON Students.CourseID = Courses.ID;
```

LEFT JOIN:

```
SELECT Students.Name, Attendance.Status
FROM Students
LEFT JOIN Attendance ON Students.ID = Attendance.StudentID;
```

RIGHT JOIN:

```
SELECT Attendance.Status, Students.Name
FROM Attendance
RIGHT JOIN Students ON Attendance.StudentID = Students.ID;
```



JOINS Continue

Types and syntax

FULL JOIN:

```
SELECT Students.Name, Grades.Grade
FROM Students
FULL JOIN Grades ON Students.ID = Grades.StudentID;
```

CROSS JOIN:

```
SELECT Students.Name, Courses.CourseName
FROM Students
CROSS JOIN Courses;
```

NATURAL JOIN:

```
SELECT *
FROM Students
NATURAL JOIN Grades;
```

JOINS Continue

Types and syntax

Self Join:

```
SELECT e1.Name AS Employee, e2.Name AS Manager
FROM Employees e1
JOIN Employees e2 ON e1.ManagerID = e2.ID;
```

Non-Equijoin:

```
SELECT s.Name, c.CourseName
FROM Students s, Courses c
WHERE s.EnrollmentYear > c.StartYear;
```

Join with Multiple Conditions:

```
SELECT o.OrderID, c.CustomerName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID AND o.TotalAmount > 1000;
```

JOINS Continue

Multiple Joins

JOIN & JOIN:

```
SELECT o.OrderID, c.CustomerName, p.ProductName
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

Join & Left Join:

```
SELECT e.EmployeeID, e.EmployeeName, d.DepartmentName
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID
LEFT JOIN Managers m ON e.ManagerID = m.ManagerID;
```

Left Join & Left Join:

```
SELECT p.ProductName, s.SupplierName, c.CategoryName
FROM Products p
LEFT JOIN Suppliers s ON p.SupplierID = s.SupplierID
LEFT JOIN Categories c ON p.CategoryID = c.CategoryID;
```

Aggregation

AVG ():

SELECT AVG(Marks) AS AverageMarks
FROM Exams;

COUNT():

SELECT COUNT(*) AS TotalStudents
FROM Students;

MAX ():

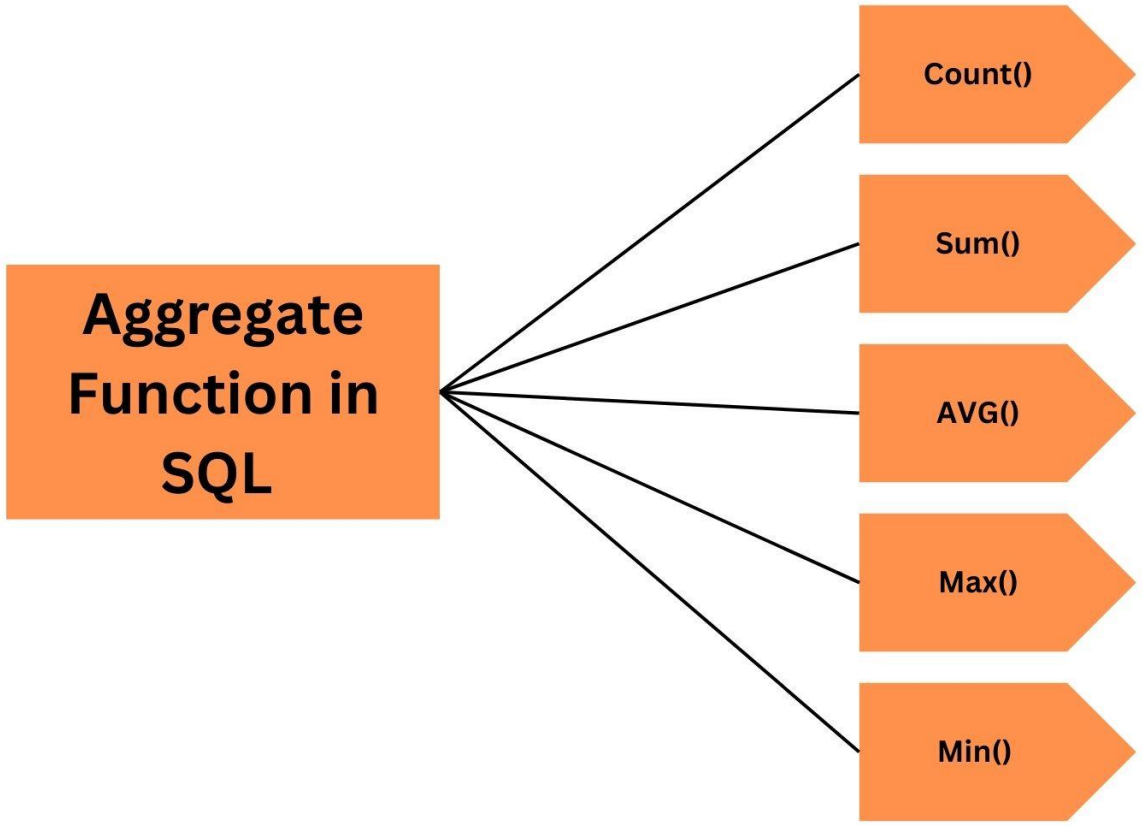
SELECT MAX(Salary) AS MaxSalary
FROM Employees;

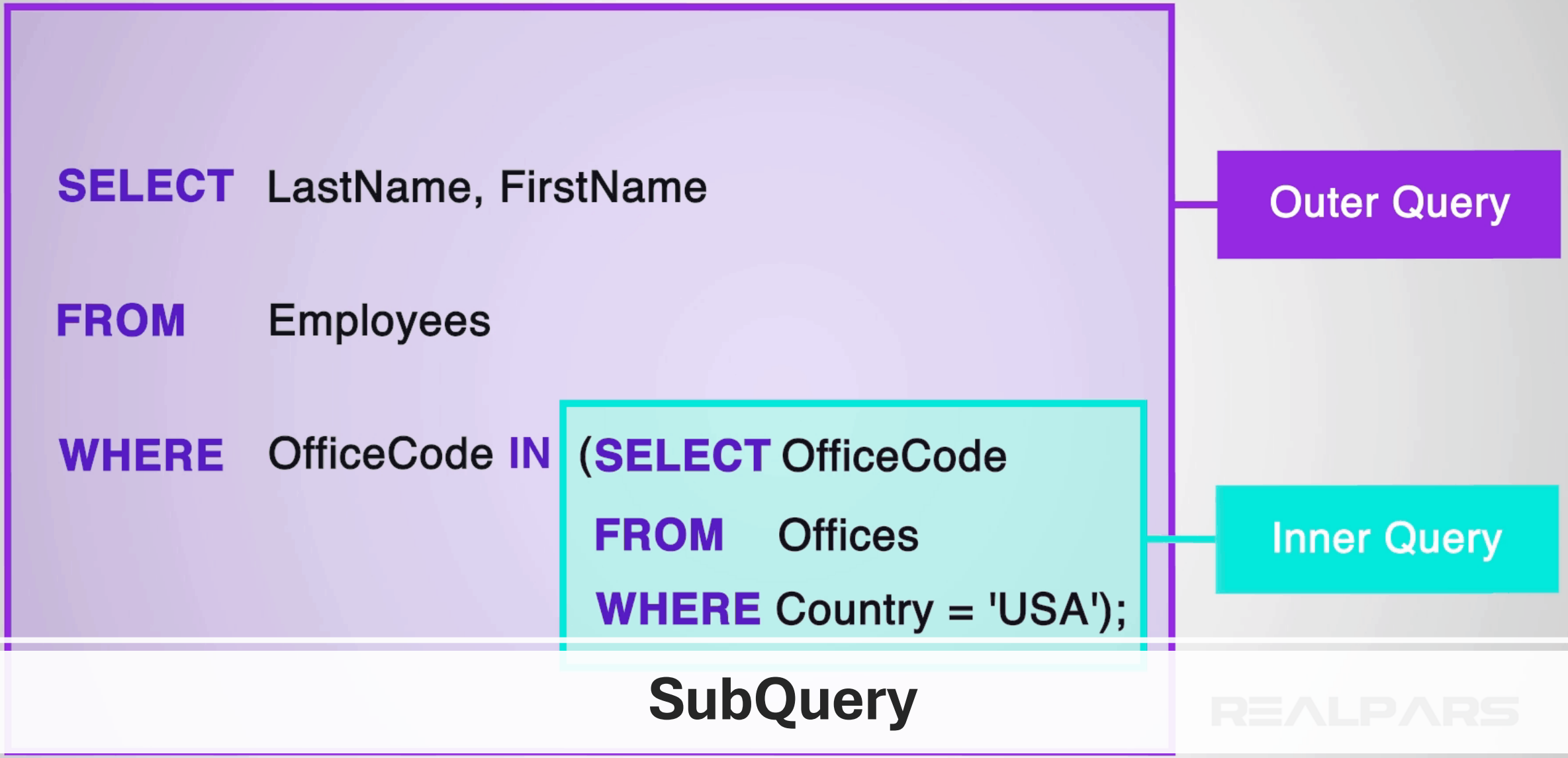
MIN():

SELECT MIN(Age) AS MinAge
FROM Customers;

SUM():

SELECT SUM(Price) AS TotalSales
FROM Orders;




SELECT LastName, FirstName
FROM Employees
WHERE OfficeCode IN (SELECT OfficeCode
FROM Offices
WHERE Country = 'USA');
Outer Query
Inner Query
SubQuery

SUBQUERIES

SINGLE VALUE :

```
SELECT Name, Age
FROM Students
WHERE Age = (SELECT MAX(Age) FROM Students);
```

Multiple Values:

```
SELECT Name
FROM Courses
WHERE CourseID IN (SELECT CourseID FROM Enrollments WHERE StudentID = 101);
```

Correlated :

```
SELECT Name
FROM Employees e
WHERE Salary > (SELECT AVG(Salary) FROM Employees WHERE Department =
e.Department);
```

SET OPERATIONS

UNION:

SELECT Name FROM Students

UNION

SELECT Name FROM Teachers;

INTERSECT:

SELECT Name FROM Students

INTERSECT

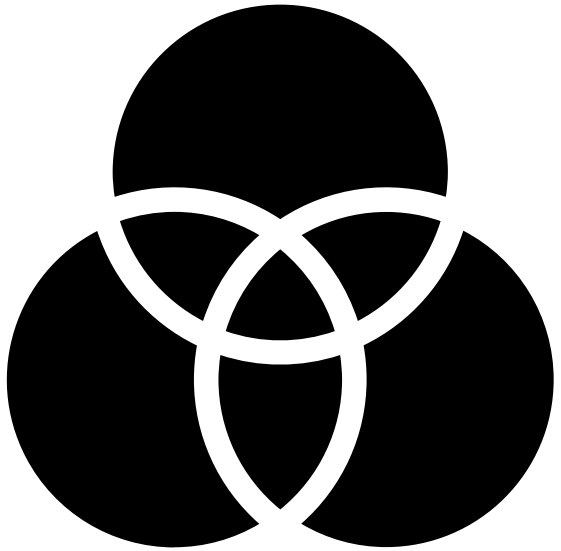
SELECT Name FROM TopPerformers;

EXCEPT:

SELECT Name FROM Students

EXCEPT

SELECT Name FROM Alumni;



Thank You