**SQL Basics**

What is SQL?

Structured Query Language is a computer language that we use to interact with a relational database.SQL is a tool for organizing, managing, and retrieving archived data from a computer database.

The original name was given by IBM as Structured

English Query Language, abbreviated by the acronym SEQUEL. When data needs to be retrieved from a database, SQL is used to make the request. The DBMS processes the SQL query retrieves the requested data and returns it to us. Rather, SQL statements describe how a collection of data should be organized or what data should be extracted or added to the database.

In common usage, SQL encompasses DDL and DML commands for create, updates, modified or other operations on database structure.

**SQL uses:**

* **Data definition**: It is used to define the structure and organization of the stored data and relationships among the stored data items.
* **Data retrieval**: SQL can also be used for data retrieval.
* **Data manipulation**: If the user wants to add new data, remove data, or modifying in existing data then SQL provides this facility also.
* **Access control**: SQL can be used to restrict a user’s ability to retrieve, add, and

modify data, protecting stored data against unauthorized access.

* **Data sharing**: SQL is used to coordinate data sharing by concurrent users, ensuring that changes made by one user do not inadvertently wipe out changes made at

nearly the same time by another user.

**What is Database ?**

The Database is an essential part of our life. As we encounter several activities that

involve our interaction with databases, for example in the bank, in the railway station, in school, in a grocery store, etc. These are the instances where we need to store a large amount of data in one place and fetch these data easily.

**What is a Relational Database (RDBMS)?**

A relational database management system or RDBMS is a database system that stores and fetches data in the form of tables. Thus the data is stored in the form of rows and columns. DBMS such as My-SQL and ORACLE are based on the principle of relational DBMS.

The most commonly used relational databases are:

MySQL

PostgreSQL

DB2

**What is MySQL?**

MySQL is a database management system that is required to store or retrieve data from a database. In this article, we'll be using MySQL for managing our databases. Let's have a brief introduction about it.

MySQL is a cross-platform tool compatible with most operating systems. Thus it can run on multiple computing platforms, including Windows, Linux, NetWare, Novell, and other variations of UNIX.

It can be used with multiple languages like PHP, NodeJS, Python, and C#.

MySQL is open-source software that can be downloaded, used, and modified by anyone worldwide.

MySQL is a Relational Database Management System that stores and presents data in a tabular form, organized in rows and columns. We can retrieve data from multiple tables by writing a few commands.

MySQL has a well-earned and established reputation for reliability, making it a good choice for use.

MySQL stores data efficiently in the memory, ensuring that data is consistent and non-redundant. This makes the access and manipulation of data very fast.

MySQL is scalable, i.e., it can work with both small and large databases.

MySQL Server works in client/server architecture in which a central program acts as a server, and various clients can request the server for various access or retrieval of data from the database. The central program is the MySQL database server.

**How to Create Database in MySQL?**

CREATE DATABASE <database-name>;

Listing All the Databases

SHOW DATABASES;

**How to Create Table in MySQL?**

USE <database\_name>;

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype,

....

);

Eg:

CREATE TABLE college\_students (

student\_name varchar(100),

student\_id int,

student\_city varchar(100)

);

student\_name student\_id student\_city

CREATE TABLE teachers (

teacher\_name varchar(100),

teacher\_id int,

teacher\_city varchar(100)

);

The table can be displayed using

SELECT \* FROM <table\_name>;

**How to Delete Database from SQL Server?**

DROP DATABASE IF EXISTS <database\_name>;

**Add new column to table:**

ALTER TABLE students ADD Email varchar(255);

**Drop column:**

ALTER TABLE students DROP COLUMN student\_city;

**What are Data Types in SQL?**

A data type is a property that describes the sort of data that an object can store, such as integer data, character data, monetary data, date and time data, binary strings, and so on.

For any database, data types are primarily categorized into three categories:

String Datatypes.

Numeric Datatypes.

Date and time Data types.

**Creating tables:**

CREATE TABLE CUSTOMER\_ID(ID INT(10));

CREATE TABLE SALES(ID INT(10));

**Deleting the table, student**

DELETE FROM student;

Delete All Records as well as the Structure of the table.

The **DROP command** is used to delete all records as well as the structure of the table.

DROP TABLE table\_name;

This syntax is used to delete all the records and the table's structure. This is the same as the delete command, but this will not preserve the structure of the table.

**Truncate Table**

The truncate table command is used to delete all the records from the table without deleting the structure of the table, but we cannot use the where clause in the truncate table command.

TRUNCATE TABLE tableName

**Truncate Table vs Delete**

| **Truncate table** | **Delete command** |
| --- | --- |
| Truncate command is a DDL command. | Delete command is a DML command. |
| The where clause cannot use in the truncate command | The where clause can be used in the Delete command. |
| Truncate command cannot delete any particular records. | Delete command can delete specified records of the table. |
| Truncate command is faster than the delete command | Delete command is slower than the Truncate command. |

**SQL INSERT INTO Statement**

1. INSERT INTO by Specifying the Column Names

INSERT INTO table\_name (column\_1, column\_2, ..., column\_n)

VALUES (value\_1, value\_2, ..., value\_n);

2. INSERT INTO without Specifying the Column Names

INSERT INTO table\_name

VALUES (value\_1, value\_2, ..., value\_n);

mysql> select \* from students;

+--------------+------------+--------------+

| student\_name | student\_id | student\_city |

+--------------+------------+--------------+

| Rohit Shetty | 101 | Mumbai |

+--------------+------------+--------------+

1 row in set (0.00 sec)

**Insert Multiple Records**

1. By specifying the names of the columns of the table:

INSERT INTO country (Name, Code, Continent)

VALUES

(India, IND, Asia),

(France, FRN, Europe),

(Japan, JPN, Asia),

(Brazil, BRZ, Africa);

2. Without specifying the names of the columns of the table:

INSERT INTO country

VALUES

(India, IND, Asia),

(France, FRN, Europe),

(Japan, JPN, Asia),

(Brazil, BRZ, Africa);

SQL INSERT INTO SELECT Statement

1. Without specifying the names of the columns of the table:

INSERT INTO destination\_table

SELECT \* FROM source\_table

WHERE condition;

2. By specifying the names of the columns of the table:

INSERT INTO destination\_table (column\_1, column\_2, ..., column\_n)

SELECT (column\_1, column\_2, ..., column\_n) FROM source\_table

WHERE condition;

**Updates:**

Syntax for Update in SQL

UPDATE tableName

SET column1 = value1, column2 = value2, ...

WHERE condition;

Eg: UPDATE Student SET name="Amit Sharma" WHERE AdminNo=102;

UPDATE Student SET name="Sourav Sharma", marks=15 WHERE AdminNo=97;

**SQL order of operations**

SQL query execution order means optimizing our search results from the database or where the query clauses are evaluated according to the requirements.

Like how we plan something step by step and reach the final result, similarly, we use clauses in a particular order known as the SQL query execution order. Here is the table that shows the order in which the clauses are used.

| **Order** | **Clause** | **Function** |
| --- | --- | --- |
| 1 | FROM | Tables are joined to get the base data. |
| 2 | WHERE | The base data is filtered. |
| 3 | GROUP BY | The filtered base data is grouped. |
| 4 | HAVING | The grouped base data is filtered. |
| 5 | SELECT | The final data is returned. |
| 6 | ORDER BY | The final data is sorted. |
| 7 | LIMIT | The returned data is limited to row count. |

Create Customers,Order table

create table Customers

(customer\_id int, first\_name varchar(100))

create table orders

(order\_id int, amount int, customer\_id int);

create table Customers

(customer\_id int, first\_name varchar(100), order\_id int, amount int)

insert into Customers values(101,'Rohit',1234,10000),(201,'Sandeep',2345,12000),(301,'Ankush',3456,9000),(401,'Abhishek',4567,12000),(101,'Rohit',6789,4567),(201,'Sandeep',8947,6500),(301,'Vishal',9876,4900),(101,'Rohit',7679,9000),(401,'Abhishek',9021,19000);

Explore few things with these 2 tables

where, group by, having , order by , limit

**What is an Alias in SQL Statements?**

Alias in SQL is basically a temporary name that is given to a table or a column while writing a query. This is usually done when the column or the table names are long, so in order to render more readability, alias is given. The alias in SQL is a temporary change and only exists till the duration of that query.

Syntax for Column Alias

SELECT column\_name AS alias\_name

FROM table\_name;

Syntax for Table Alias

SELECT column\_1,column\_2...

FROM table\_name AS alias\_name;

**What is SQL Operator?**

To retrieve the data stored in the databases by performing specific logical or mathematical computations, we use SQL Operators. Let us learn about SQL operators in detail.

We can categorize the SQL operators into six categories:

SQL Arithmetic Operators.

SQL Comparison Operators.

SQL Logical Operators.

SQL Set Operators.

SQL Unary Operators.

SQL Bitwise Operators.

**What are JOINS in SQL?**

SQL Joins are mostly used when a user is trying to extricate data from multiple tables (which have one-to-many or many-to-many relationships with each other) at one time. The join keyword merges two or more tables and creates a temporary image of the merged table. Then according to the conditions provided, it extracts the required data from the image table, and once data is fetched, the temporary image of the merged tables is dumped.

Example of SQL JOINS

We have a company’s employee database, where Table 1 (Employee) contains information about the employee like: employee EmpNo, employee name, and DeptNo.

Table 2 (Department) includes information on Departments, i.e., their DeptNo and name and location.

CREATE TABLE Department(

DeptNo int PRIMARY KEY,

DName varchar(266),

Location varchar(266)

);

insert into Department values(1,'IT','Pune'),(2,'CSE','Pune'),(3,'EIE','Pune');

CREATE TABLE Employee(

EmpNo int PRIMARY KEY,

EmpName varchar(266),

Salary int,

DeptNo int,

FOREIGN KEY (DeptNo) REFERENCES Department(DeptNo)

);

insert into Employee values(101,'S',2,1),(102,'Sa',20,10);

**Types of JOINS:**

1. Cross Join:

The Cartesian Join, a.k.a. Cross Join, is the cartesian product of all the rows of the first table with all the rows of the second table. Let’s say we have m rows in the first table and n rows in the second table. Then the resulting cartesian join table will have m\*n rows. This usually happens when the matching column or WHERE condition is not specified.

SELECT column-name(s)

FROM table1 CROSS JOIN table2;

2. SELF JOIN in SQL

In SQL Self Join, a table is joined to itself. This means each row of the table is joined with itself and all other rows concerning stated conditions if any. In other words, we can say that it is a merge between two copies of the same table. This is extremely helpful when the foreign key references the primary key of the same table.

SELECT a.column1 , b.column2

FROM table\_name a, table\_name b

WHERE some\_condition;

Here we reference the same table with different names, i.e., a and b. This signifies a SELF JOIN.

3. INNER JOIN in SQL

SQL Inner Join where all rows from the intended tables are cached together if they meet the stated condition. Two or more tables are required for this join. Inner Join can be used with various SQL conditional statements like WHERE, GROUP BY, ORDER BY, etc.

SELECT column-name

FROM table-1 INNER JOIN table-2

WHERE table-1.column-name = table-2.column-name;

We can alternately use just the “JOIN” keyword instead of “INNER JOIN”.

A diagram of a table

Description automatically generated

Outer JOINS:

1. LEFT OUTER JOIN

In this join, a.k.a. SQL Left Join, all the rows of the left-hand table, regardless of following the stated conditions are added to the output table. At the same time, only matching rows of the right-hand table are added.

Rows belonging to the left-hand table and not having values from the right-hand table are presented as NULL values in the resulting table.

SELECT column-name(s)

FROM table1 LEFT OUTER JOIN table2

ON table1.column-name = table2.column-name;

A diagram of a table and table

Description automatically generated

2. RIGHT OUTER JOIN

Similar to the left outer join, in the case of the Right Outer Join, a.k.a. SQL Right Join, all the rows on the right-hand table, regardless of following the stated conditions, are added to the output table. At the same time, only matching rows of the left-hand table are added.

Rows belonging to the right-hand table and not having values from the left-hand table are presented as NULL values in the resulting table.

SELECT column-name(s)

FROM table1 RIGHT OUTER JOIN table2

ON table1.column-name = table2.column-name;

A diagram of a table and table

Description automatically generated

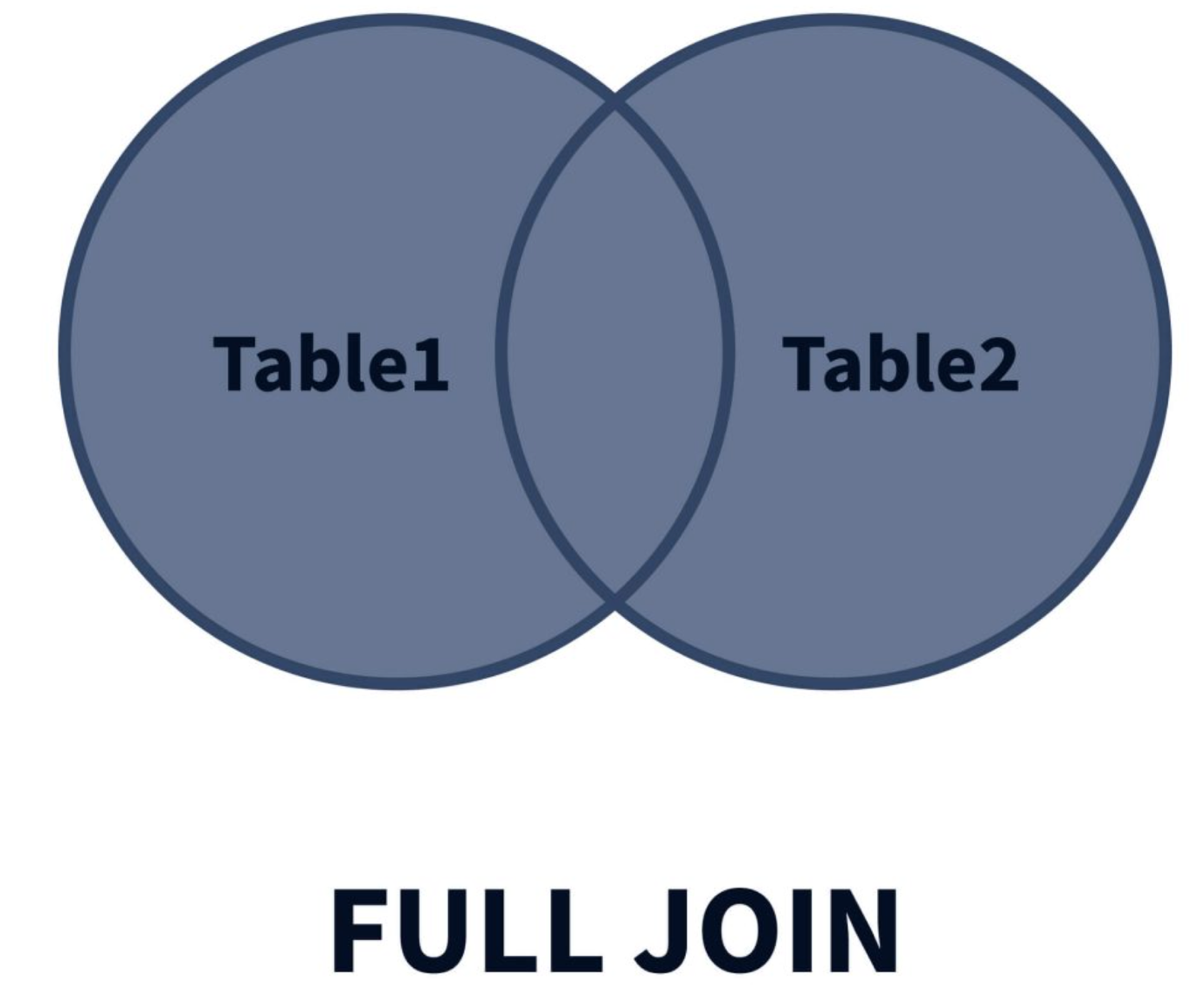
3. FULL OUTER JOIN

The full outer join (a.k.a. SQL Full Join) first adds all the rows matching the stated condition in the query and then adds the remaining unmatched rows from both tables. We need two or more tables for the join.

After the matched rows are added to the output table, the unmatched rows of the left-hand table are added with subsequent NULL values, and then unmatched rows of the right-hand table are added with subsequent NULL values.

SELECT column-name(s)

FROM table1 FULL OUTER JOIN table2

ON table1.column-name = table2.column-name;

Some databases like MYSQL do not directly support full outer join but we could use union of left join and right join

SELECT \*

FROM Students s

LEFT JOIN Marks m ON s.Id = m.StudentID

UNION

SELECT \*

FROM Students s

RIGHT JOIN Marks m ON s.Id = m.StudentID

WHERE s.Id IS NULL;

Aggregate Functions in SQL

An aggregate function in SQL performs a calculation on multiple values and returns a single value. SQL provides many aggregate functions that include avg, count, sum, min, max, etc. An aggregate function ignores NULL values when it performs the calculation, except for the count function.

A diagram of a function

Description automatically generated

Subqueries in SQL

A subquery is a query that is used within another SQL query embedded in the WHERE clause. Subqueries are majorly used in SELECT, INSERT, UPDATE, DELETE, FROM and WHERE clauses. The subqueries are used wherever an expression is allowed. The data returned by the subquery is used by the outer statement in the same way a literal value would be used. Subqueries in SQL provide an easy and efficient way to handle the queries which depend on the results from another query.

SELECT \*

FROM Students

WHERE marks = (

SELECT max(marks)

FROM Students

);

Subqueries with INSERT statement

INSERT INTO EmployeeNew

SELECT \* FROM Employee

WHERE emp\_id IN (

SELECT emp\_id

FROM Employee

);

Group By multiple columns:

We can group by multiple columns

SQL CASE Statement

What is CASE Statement in SQL?

The Case Statement in SQL returns a value when some condition is specified. This condition will be based on some other cell in the same table.

As in the if-else statement, several conditions are applied, and whichever first gives the result as true is executed. The case statement in SQL also works similarly. The conditions are specified using WHEN and THEN keywords. You can specify as many conditions using these keywords.

However, the CASE, WHEN, THEN, and END keywords are necessary while writing a case statement in SQL. The CASE keyword is used to specify the beginning of the case statement, whereas the END keyword specifies the end of the case statement and the end of the conditions. We can also write an ELSE keyword that would execute when all other conditions specified in the case statement return false.

The END keyword is followed by an alias\_name, which is the column name you are adding to the table based on the conditions.

SQL CASE Syntax

The syntax of the Case Statement in SQL is as follows:

CASE

WHEN condition1 THEN output1

WHEN condition2 THEN output2

...

...

WHEN conditionN THEN outputN

ELSE output

END AS alias\_name;

SELECT \*,

CASE

WHEN Marks>=85 THEN 'A'

WHEN Marks>=75 AND Marks<85 THEN 'B'

WHEN Marks>=50 AND Marks<74 THEN 'C'

ELSE 'F'

END AS Grade

FROM student;

SELECT \*,

CASE Grade

WHEN 'A' THEN "Excellent"

WHEN 'B' THEN "Good"

WHEN 'C' THEN "Improve more"

ELSE "Work Hard"

END AS Remarks

FROM student;