DATABASE & SQL

ONE SHOULD HAVE A STRONG FOUNDATION

"SQL & Database understanding is a must-have skill for testers. It is always preferable to do some database checks when doing functional/web testing."

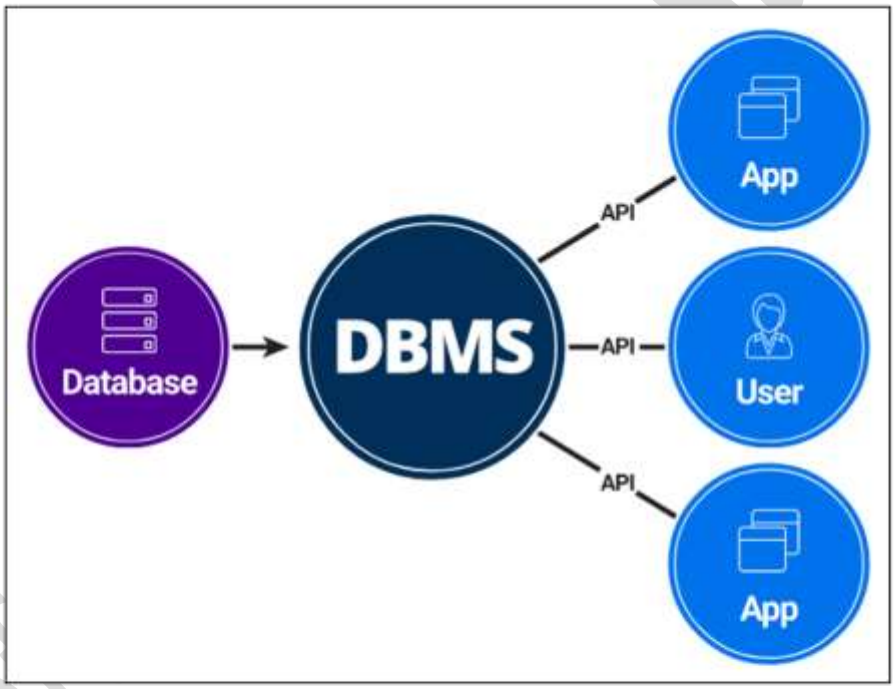
Introduction

Database

Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

Database Management System

The software which is used to manage database is called Database Management System (DBMS). For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

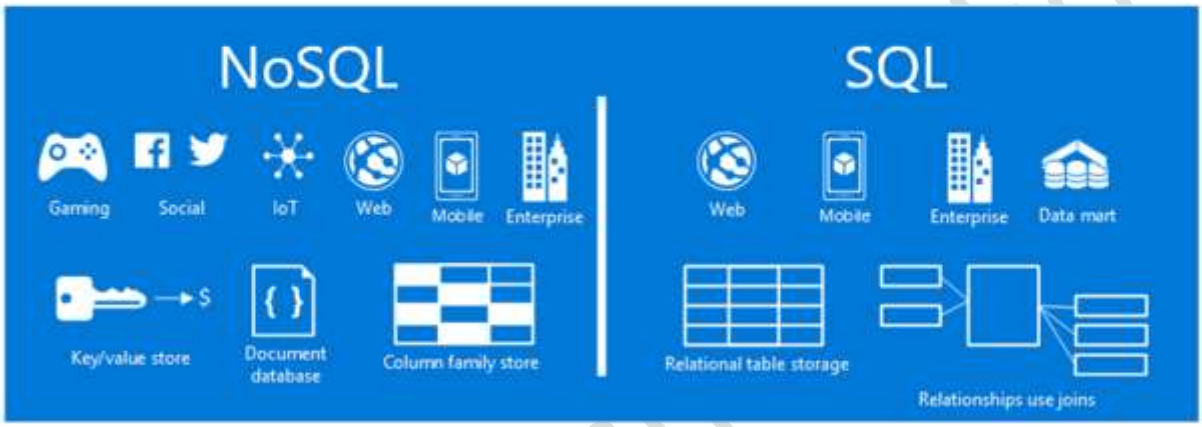


- **Data Definition**: It helps in creation, modification and removal of definitions that define the organization of data in database.
- **Data Updation**: It helps in insertion, modification and deletion of the actual data in the database.
- **Data Retrieval**: It helps in retrieval of data from the database which can be used by applications for various purposes.

Two main types of Databases

Knowledge of database is important, both from development as well as testing perspective. Recently (e.g., social media) there is a load of unstructured data which cannot be accommodated in traditional Relational-DBMS efficiently. Hence, we get two main types of databases,

- **SQL [relational]:** Table-based DB with predefined schema including tables-rows-columns. Use os structured query language (SQL) for defining and manipulating data. E.g., MySQL | Oracle | PostgreSQL | Microsoft SQL Server | etc.



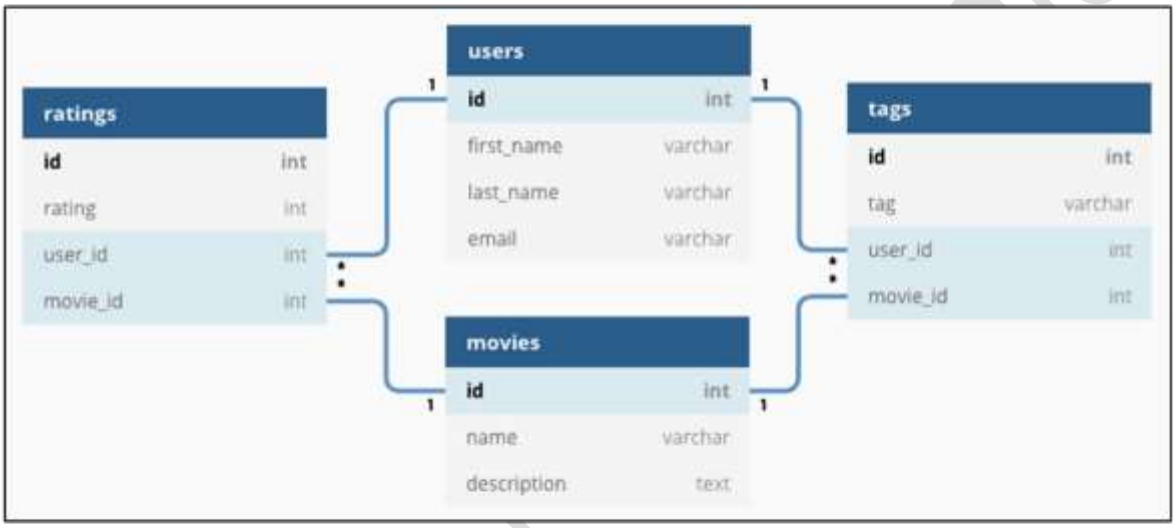
- **NoSQL [not-only-SQL, non-relational]:** dynamic schemas for unstructured data where data can be stored in many ways: column-oriented, document-oriented, graph-based or organized as a Key-Value store. E.g., MongoDB (document) | Cassandra (wide-column) | Redis (key-value) | Neptune (graph) | etc.

SQL

Structured Query Language or SQL is a standard Database language which is used to create, maintain and retrieve the data from relational databases like MySQL, Oracle, SQL Server, PostGreSQL, etc.

What is Relational Model?

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).



- SQL is **case insensitive**. But it is a recommended practice to use keywords (like SELECT, UPDATE, CREATE, etc) in capital letters and use user defined things (liked table name, column name, etc) in small letters.
- SQL is the programming language **for relational databases** (explained below) like MySQL, Oracle, Sybase, SQL Server, Postgre, etc. Other non-relational databases (also called NoSQL) databases like MongoDB, DynamoDB, Cassandra, etc do not use SQL
- Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax. So, we may encounter queries that work in SQL Server but do not work in MySQL.

DDL

Short name of **Data Definition Language**, which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE: to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER: alters the structure of the existing database
- DROP: delete objects from the database
- TRUNCATE: remove all records from a table, including all spaces allocated for the records are removed
- COMMENT: add comments to the data dictionary
- RENAME: rename an object

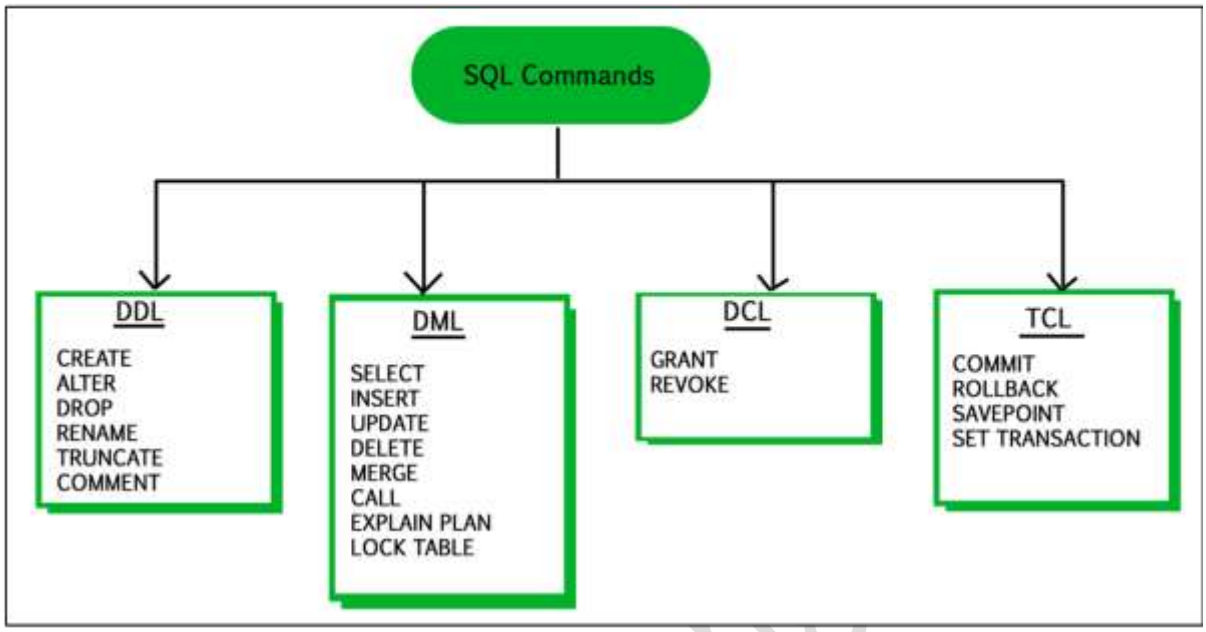
DML

Short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT: retrieve data from a database
- INSERT: insert data into a table
- UPDATE: updates existing data within a table
- DELETE: Delete all records from a database table
- MERGE: UPSERT operation (insert or update)
- CALL: call a PL/SQL or Java subprogram
- EXPLAIN PLAN: interpretation of the data access path
- LOCK TABLE: concurrency Control

Types of SQL commands | DDL, DML and DCL

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



Data Definition Language (DDL): changes the structure of the table like creating a table, deleting a table, altering a table, etc.
CREATE | ALTER | DROP | TRUNCATE

Data Manipulation Language (DML): used to modify the database. It is responsible for all form of changes in the database.
INSERT | UPDATE | DELETE

Data Control Language (DCL): used to grant and take back authority from any database user.
GRANT | REVOKE

Transaction Control Language (TCL): only used with DML commands like INSERT, DELETE and UPDATE only.
COMMIT | ROLLBACK | SAVEPOINT

Data Query Language (DQL): used to fetch the data from the database.
SELECT

SQL Commands

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

Select

The SQL SELECT statement is used to fetch the data from a database table which returns this data in the form of a result table.

SELECT column1, column2, columnN FROM table_name;

If you want to fetch all the fields, then use the following syntax,

*SELECT * FROM table_name;*

Order of SQL SELECT is: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY.

Only the SELECT and FROM clauses are mandatory.

Where

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records. The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc.

SELECT column1, column2, columnN
FROM table_name
WHERE [condition]

Specify a condition using the comparison or logical operators like >, <, =, LIKE, NOT, etc.

Update

The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

Delete

The SQL DELETE Query is used to delete the existing records from a table. You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

DELETE FROM table_name
WHERE [condition];

You can combine N number of conditions using AND or OR operators.

LIKE Clause

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

- The percent sign (%): represents zero, one or multiple characters
- The underscore (_): represents a single number or character

SELECT FROM table_name
WHERE column LIKE 'XXXX%'

or

SELECT FROM table_name
WHERE column LIKE '%XXXX%'

or

SELECT FROM table_name
WHERE column LIKE 'XXXX_'

or

```
SELECT FROM table_name
WHERE column LIKE '_XXXX'
```

or

```
SELECT FROM table_name
WHERE column LIKE '_XXXX_'
```

DISTINCT Keyword

Used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records. There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only those unique records instead of fetching duplicate records.

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

SQL Clauses

An SQL clause is defined to limit the queried results to certain specified conditions.

- **GROUP BY**: used in aggregation to arrange identical data into groups, the GROUP BY clause follows the WHERE clause in a SELECT statement and is followed by the ORDER BY clause

```
SELECT column1, column2
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

- **HAVING**: used to specify a search condition in a GROUP BY clause, HAVING can be used in the absence of a GROUP BY clause by using a WHERE clause
- **ORDER BY**: sorts the result set in ascending (default) or descending (using DESC keyword) order

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

- **WHERE**: used to define the condition of the records to be extracted

SQL Constraints

Constraints are statements used to establish the rules for all records in the table. If any action violates a constraint, that action will be aborted. Constraints are defined while creating the database itself with CREATE TABLE statement, or after the table is created, by using ALTER TABLE statement.

- **NOT NULL**: indicates that the column is required to have some value; it cannot be left null.
- **DEFAULT**: Provides a default value for a column when none is specified.
- **UNIQUE**: ensures that each row and column have unique value; no value is being repeated in any other row or column.
- **PRIMARY KEY**: used in association with NOT NULL and UNIQUE constraints to identify a particular unique record.
- **FOREIGN KEY**: used to ensure the referential integrity of data in the table and also matches the value in one table with another using PRIMARY KEY.
- **CHECK**: used to ensure the value in columns obeys specified conditions.
- **INDEX**: Used to create and retrieve data from the database very quickly.

Aggregate functions

SQL Aggregate functions determine and calculate values from multiple columns in a table and return a single value.

- **AVG():** Returns the average value from specified columns.
- **COUNT():** Returns number of table rows.
- **MAX():** Returns the largest value among the records.
- **MIN():** Returns smallest value among the records.
- **SUM():** Returns the sum of specified column values.
- **FIRST():** Returns the first value.
- **LAST():** Returns last value.

A vs. B

Database Triggers

Triggers in SQL is kind of stored procedures used to create a response to a specific action performed on the table such as INSERT, UPDATE or DELETE. You can invoke triggers explicitly on the table in the database.

Action and Event are two main components of SQL triggers. When certain actions are performed, the event occurs in response to that action.

Stored Procedures

A stored procedure is a collection of SQL statements that can be used as a function to access the database. We can create these stored procedures earlier before using it and can execute them wherever required by applying some conditional logic to it. Stored procedures are also used to reduce network traffic and improve performance.

Triggers	Stored Procedures
A special kind of stored procedure that is not called directly by a user. In fact, a trigger is created and is programmed to fire when a specific event occurs.	A group of SQL statements which can be reused again and again. These statements are created and stored in the database.
A trigger cannot be called or execute directly by a user. Only when the corresponding events are fired, triggers are created.	Can execute stored procedures by using the exec command, whenever we want.
You cannot schedule a trigger.	You can schedule a job to execute the stored procedure on a pre-defined time.
Cannot directly call another trigger within a trigger.	Call a stored procedure from another stored procedure.
Parameters cannot be passed as input	Parameters can be passed as input
Cannot return values.	Can return zero or n values.
Transactions are not allowed within a trigger.	You can use transactions within a stored procedure.

DELETE vs. DROP vs. TRUNCATE

- DELETE removes some or all rows from a table based on the condition. It can be rolled back.
- TRUNCATE removes ALL rows from a table by de-allocating the memory pages. The operation cannot be rolled back
- We can use the DELETE command with WHERE clause but cannot use the TRUNCATE command with it.
- DROP command removes a table from the database completely. Cannot be rolled-back.

UNION, MINUS, UNION ALL, INTERSECT

- **MINUS**: returns all distinct rows selected by the first query but not by the second.
- **UNION**: returns all distinct rows selected by either query
- **UNION ALL**: returns all rows selected by either query, including all duplicates.
- **INTERSECT**: returns all distinct rows selected by both queries.

NULL value, Zero, and Blank space

- Null value is a field with no value which is different from zero value and blank space.
- Zero is a number
- Blank space is the value we provide. The ASCII value of space is CHAR(32).

Having vs. Where clause

Where clause is used to fetch data from a database that specifies particular criteria.

Having clause is used along with 'GROUP BY' to fetch data that meets particular criteria specified by the Aggregate functions.

Where clause cannot be used with Aggregate functions, but the Having clause can.

Where-and-Having | Primary-and-Unique key

Where clause cannot be used with aggregates, but the Having clause can.

```
SELECT Student, SUM(score) AS total FROM Marks GROUP BY Student
HAVING total > 70
```

Generally, we use WHERE prior to GROUP BY and HAVING after GROUP BY. The Where clause acts as a pre-filter and Having clause as a post-filter.

- **Primary Key**: uniquely identifies each row in a table. It enforces integrity constraints to the table. Only one primary key is allowed for a table. It does not accept any duplicate OR NULL values.

- **Unique Key**: uniquely identifies each row in a table. But a table can have more than one unique key unlike primary key and Unique key constraints can accept only one NULL value for column. Generally used when we want to enforce unique constraint on a column (or a column group) which is not a primary key.

Union, Union All, Intersect and Minus operations

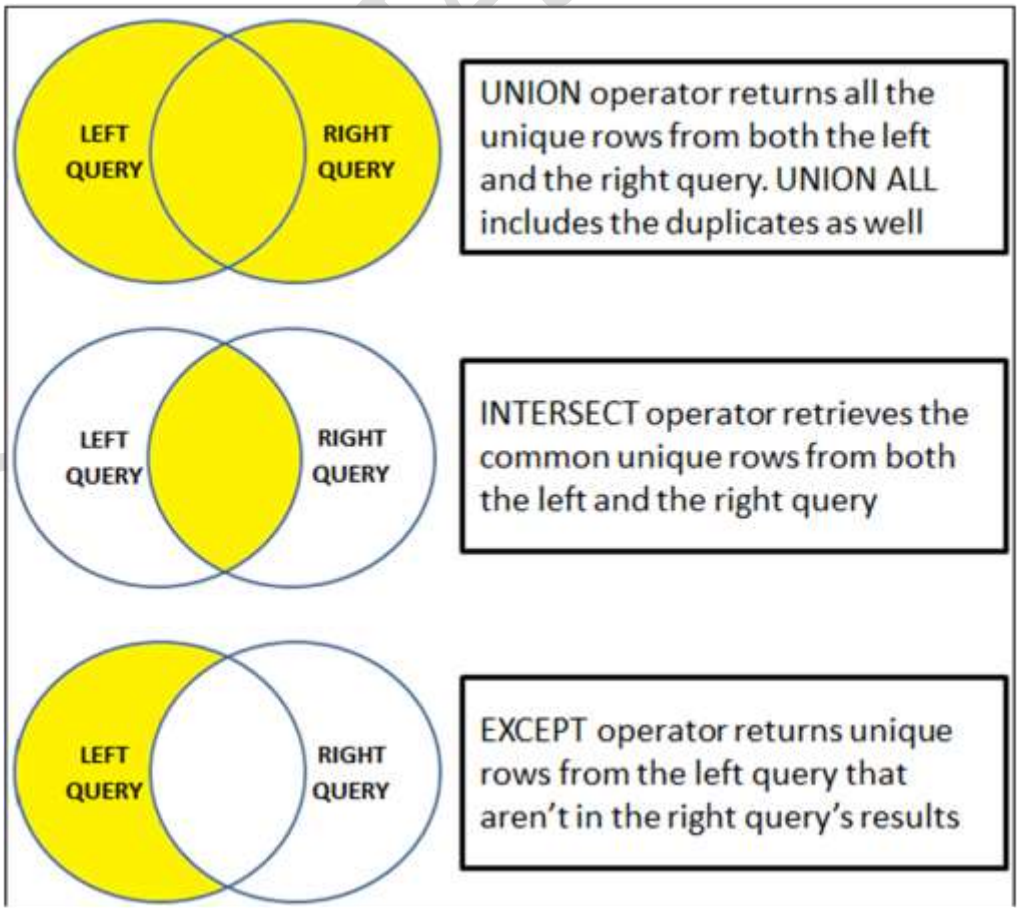
These are known as Set operations in SQL,

UNION: combine the results of two or more SELECT statements, eliminating duplicate rows. The number of columns and datatype must be same in both the tables.
*SELECT * FROM FirstTable UNION SELECT * FROM SecondTable;*

UNION ALL: similar to Union but it also shows the duplicate rows.
*SELECT * FROM FirstTable UNION ALL SELECT * FROM SecondTable;*

INTERSECT: combine two SELECT statements and return the records which are common from both SELECT statements. The number of columns and datatype must be same.
*SELECT * FROM FirstTable INTERSECT SELECT * FROM SecondTable;*

MINUS: combines results of two SELECT statements and return only those which belong to the first set.
*SELECT * FROM FirstTable MINUS SELECT * FROM SecondTable;*



Queries

1. Details of a student from Students table whose name start with K

*SELECT * FROM Student WHERE StudentName like 'K%';*

2. Select random rows from a table

Using a SAMPLE clause we can select random rows,

*SELECT * FROM TableName SAMPLE(10);*

3. Return each name only once from a table

Need to use the DISTINCT keyword,

SELECT DISTINCT Name FROM tableName;

4. To get the second largest value from a given column of a table

SELECT MAX(COLUMN_NAME) FROM TABLE_NAME
WHERE COLUMN_NAME < (SELECT MAX(COLUMN_NAME) FROM TABLE_NAME);

SELECT Max(Marks) from Students
WHERE Marks < (SELECT Max(Marks) from students);

5. Test for NULL Values

A field with a NULL value is a field with no value. NULL value cannot be compared with other NULL values. Hence, it is not possible to test for NULL values with comparison operators, such as =, <, or <>. For this, we have to use the IS NULL and IS NOT NULL operators.

- *SELECT column_names FROM table_name WHERE column_name IS NULL;*
- *SELECT column_names FROM table_name WHERE column_name IS NOT NULL;*

6. Fetch the 3rd highest Salary from the Employee table.

Query-1. *Select Name, Max (Salary) As Salary from Employee Where Salary < (Select Max (Salary) from Employee Where Salary < (Select Max (Salary) from Employee));*

Query-2. There should be just 2 Salaries greater than it.
*Select * from Employee E1 where 2 = (Select Count(Distinct(E2.Salary)) from Employee E2 where E2.Salary > E1.Salary)*

Query-3. (Oracle) Sort descending >> Then pick row number 3.
*Select * from (Select Name, Salary, ROW_NUMBER() OVER (Order by Salary Desc) As RowNum from Employee) as Temp where Temp.RowNum = 3;*

Query-4. (SQL Server). Top 3 by sorting in descending order >> Top 1 after sorting in ascending order, i.e. 3rd highest.
Select Top 1 Salary from (Select Distinct Top 3 Salary from Employee Order By Salary Desc) As Temp Order By Salary Asc

Query-5. (MySQL). Sort descending >> Skip top 2 & then fetch next record, i.e. 3rd highest salary.
Select Salary from Employee Order By Salary Desc LIMIT 2, 1;

1. Change a value of the field 'Salary' as 7500 for an Employee_Name 'John' in a table Employee_Details

UPDATE Employee_Details set Salary = 7500 where Employee_Name = 'John';

2. Select all the even number records from a table

*Select * from table where id % 2 = 0*

3. Fetch alternate records from a table

Records can be fetched for both Odd and Even row numbers using mod operation.

Select studentId from (Select rowno, studentId from student where mod(rowno,2)=0);

4. To display current date

There is a built in function in SQL called GetDate() which is used to return current timestamp.

Select getdate();

5. Case Function

Case facilitates conditional inquires just like an if-then-else statement. In SQL case works with either the select or update clauses.

```
SELECT Name,
CASE WHEN Sal > 0 AND Sal <= 100000 THEN 1
WHEN Sal > 100000 AND Sal < 250000 THEN 2
WHEN Sal > 250000 AND Sal < 5000000 THEN 3
ELSE 99
END AS Category
FROM Employee;
```

1. Get third-highest salary

Select TOP (1) salary from
(Select DISTINCT TOP (3) salary from Employee ORDER BY salary DESC) AS emp
ORDER BY salary ASC;

Subquery,
Select DISTINCT TOP (3) salary from Employee ORDER BY salary DESC
will select the top 3 salaried employees in the table listed in descending order.

Now picking the top 1 from that list will give you the highest salary not the 3rd highest salary. Therefore, the second query reorders the 3 records in ascending order and then selects the top record (which will now be the lowest of those 3 salaries).

Not all databases support the TOP keyword. For example, MySQL and PSQL use the LIMIT keyword, as follows,

Select Salary from
(Select DISTINCT Salary from Employee ORDER BY Salary DESC LIMIT 10) AS Emp
ORDER BY Salary LIMIT 1;

2. We have a table with employees' data, write a query to reverse the gender column value.

Approach-1: Use intermediate temporary value, i.e., update all M to T > update all F to M > update all T to F.

Approach-2: Single query,

Update EmployeeTable set "gender" = (case "gender" when 'male' then 'female' else 'male' end);

3. SQL query to add a column to table

ALTER TABLE command is used to alter any table layout,
ALTER TABLE table_name
ADD column_name datatype;

Example,
ALTER TABLE Customers
ADD Email varchar(255);

Conclusion

SQL & Database understanding is a must-have skill for testers. It is always preferable to do some database checks when doing functional/web testing.