TatvaSoft

sculpting thoughts...

TatvaSoft

-Ahmedabad

NAME : Makwana Sanjay l.

Technology : .PHP

1 . Brife Description on stored procedure.

- ➢ A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- ➢ So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- ➢ You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

❖ Stored Procedure Syntax

```
➢ CREATE PROCEDURE procedure_name
  AS
  sql_statement
➢
  GO;
```

A procedure (often called a stored procedure) is a **collection of pre-compiled SQL statements** stored inside the database. It is a subroutine or a subprogram in the regular computing language. **A procedure always contains a name, parameter lists, and SQL statements**. We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc. It was first introduced in MySQL **version 5**. Presently, it can be supported by almost all relational database systems.

If we consider the enterprise application, we always need to perform specific tasks such as database cleanup, processing payroll, and many more on the database regularly. Such tasks involve multiple SQL statements for executing each task. This process might easy if we group these tasks into a single task. We can fulfill this requirement in MySQL by creating a stored procedure in our database.

A procedure is called a **recursive stored procedure** when it calls itself. Most database systems support recursive stored procedures. But, it is not supported well in MySQL.

Stored Procedure Features

- Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.
- Stored procedure reduces the traffic between application and database server. Because the application has to send only the stored procedure's name and parameters instead of sending multiple SQL statements.
- Stored procedures are reusable and transparent to any applications.
- A procedure is always secure. The database administrator can grant permissions to applications that access stored procedures in the database without giving any permissions on the database tables.

How to call a stored procedure?

We can use the **CALL statement** to call a stored procedure. This statement returns the values to its caller through its parameters (IN, OUT, or INOUT). The following syntax is used to call the stored procedure in MySQL:

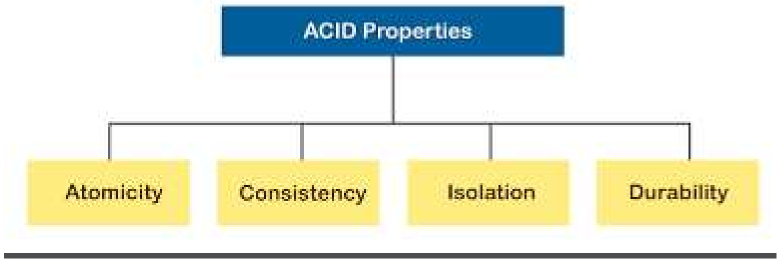
1. CALL procedure_name (parameter(s))

2. SELECT
   - customerName,
   - city,
   - state,
   - postalCode,
   - country

FROM Customers

ORDER BY customerName;

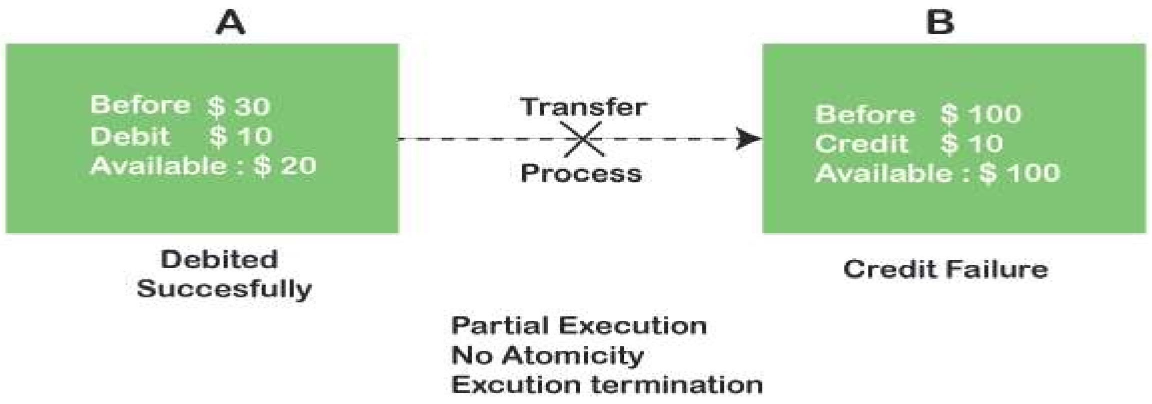
2. Explain ACID properties.



1) Atomicity

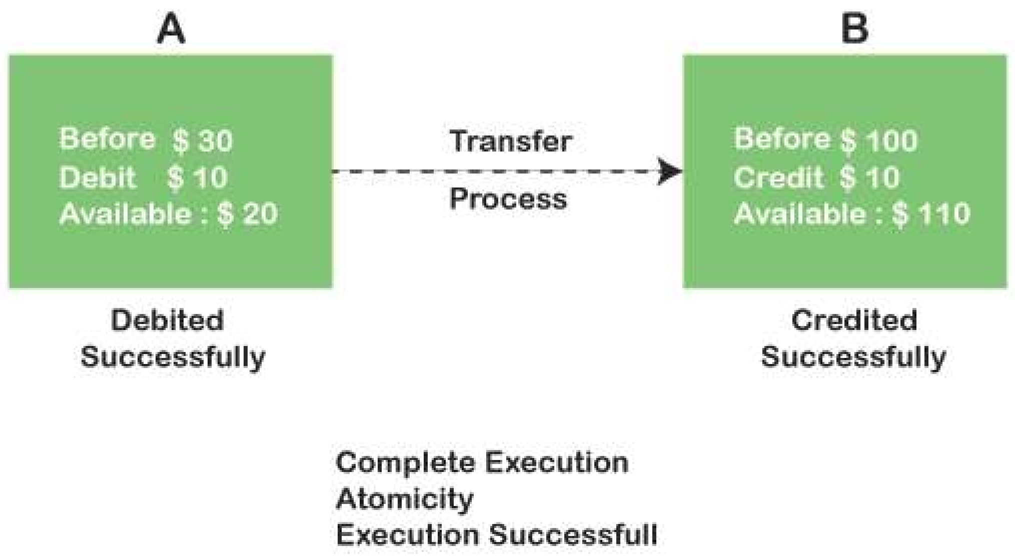
The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Example: If Remo has account A having $30 in his account from which he wishes to send $10 to Sheero's account, which is B. In account B, a sum of $ 100 is already present. When $10 will be transferred to account B, the sum will become $110. Now, there will be two operations that will take place. One is the amount of $10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account. Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails. Thus, in Remo's account A, the value becomes $20, and to that of Sheero's account, it remains $100 as it was previously present.



In the above diagram, it can be seen that after crediting $10, the amount is still $100 in account B. So, it is not an atomic transaction.

The below image shows that both debit and credit operations are done successfully. Thus the transaction is atomic.

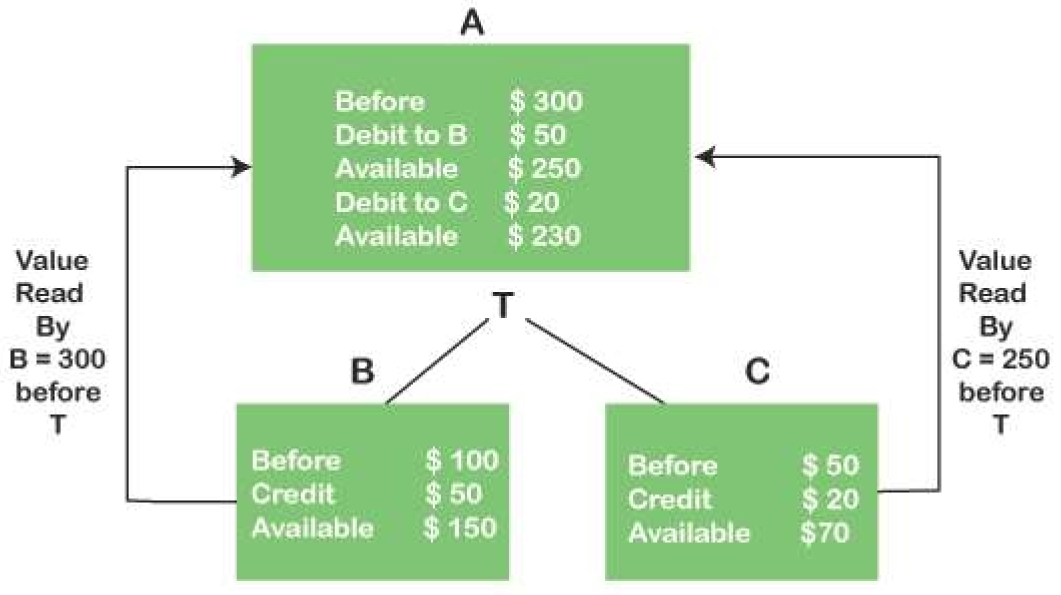


Thus, when the amount loses atomicity, then in the bank systems, this becomes a huge issue, and so the atomicity is the main focus in the bank systems.

2) Consistency

The word **consistency** means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

Example:



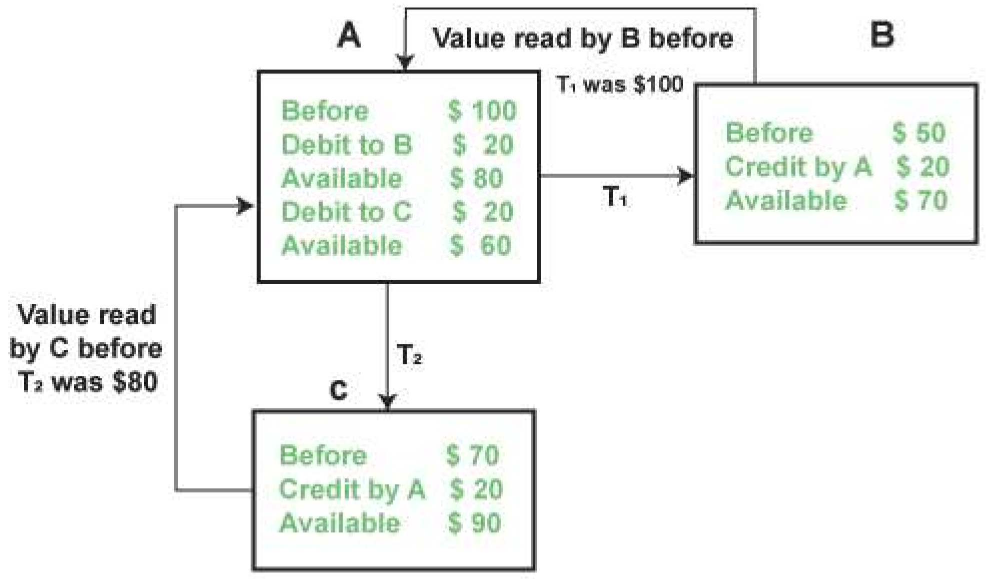
Data Consistent

In the above figure, there are three accounts, A, B, and C, where A is making a transaction T one by one to both B & C. There are two operations that take place, i.e., Debit and Credit. Account A firstly debits $50 to account B, and the amount in account A is read $300 by B before the transaction. After the successful transaction T, the available amount in B becomes $150. Now, A debits $20 to account C, and that time, the value read by C is $250 (that is correct as a debit of $50 has been successfully done to B). The debit and credit operation from account A to C has been done successfully. We can see that the transaction is done successfully, and the value is also read correctly. Thus, the data is consistent. In case the value read by B and C is $300, which means that data is inconsistent because when the debit operation executes, it will not be consistent.

3) Isolation

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Example: If two operations are concurrently running on two different accounts, then the value of both accounts should not get affected. The value should remain persistent. As you can see in the below diagram, account A is making T1 and T2 transactions to account B and C, but both are executing independently without affecting each other. It is known as Isolation.



Isolation - Independent execution of T_1 & T_2 by A

4) Durability

Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

Therefore, the ACID property of DBMS plays a vital role in maintaining the consistency and availability of data in the database.

Thus, it was a precise introduction of ACID properties in DBMS. We have discussed these properties in the transaction section also.

3. Explain offset in limit query with example.

Introduction to SQL LIMIT clause

To limit the number of rows returned by a select statement, you use the LIMIT and OFFSET clauses.

The following shows the syntax of LIMIT & OFFSET clauses:

```
SELECT
    column_list
FROM
    table1
ORDER BY column_list
LIMIT row_count OFFSET offset;
```

In this syntax:

- The LIMIT row_count determines the number of rows (row_count) returned by the query.
- The OFFSET offset clause skips the offset rows before beginning to return the rows.

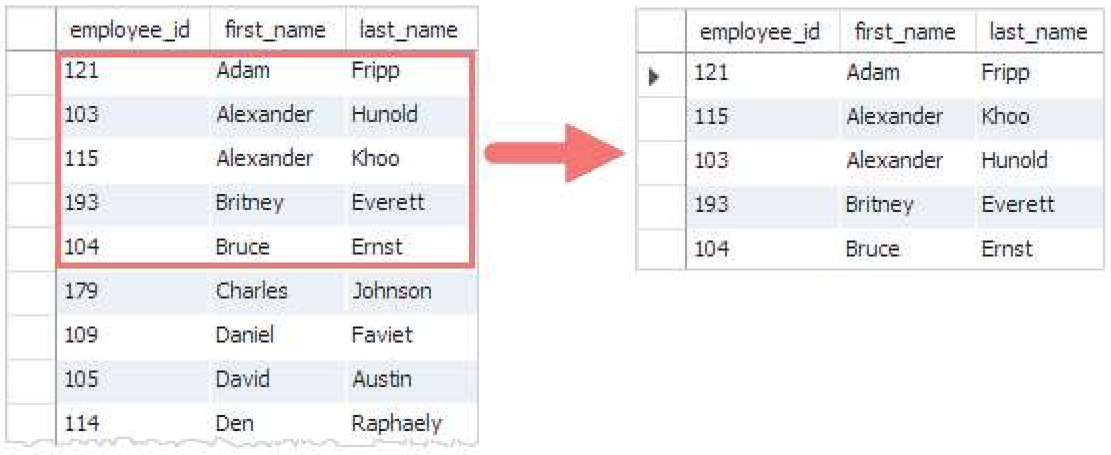
The OFFSET clause is optional. If you omit it, the query will return the row_count rows from the first row returned by the SELECT clause.

When you use the LIMIT clause, it is important to use an ORDER BY clause to ensure the order of rows in the result set.

Not all database systems support the LIMIT clause. Therefore, the LIMIT clause is available only in some database systems only such as MySQL, PostgreSQL, SQLite, Sybase SQL Anywhere, and HSQLDB. If you use SQL Server, you can use the SELECT TOP instead.

The following example uses the LIMIT clause to return the first 5 rows in the result set returned by the SELECT clause:

```
SELECT
    employee_id,
    first_name,
    last_name
FROM
    employees
ORDER BY
            first_name
LIMIT 5;
```

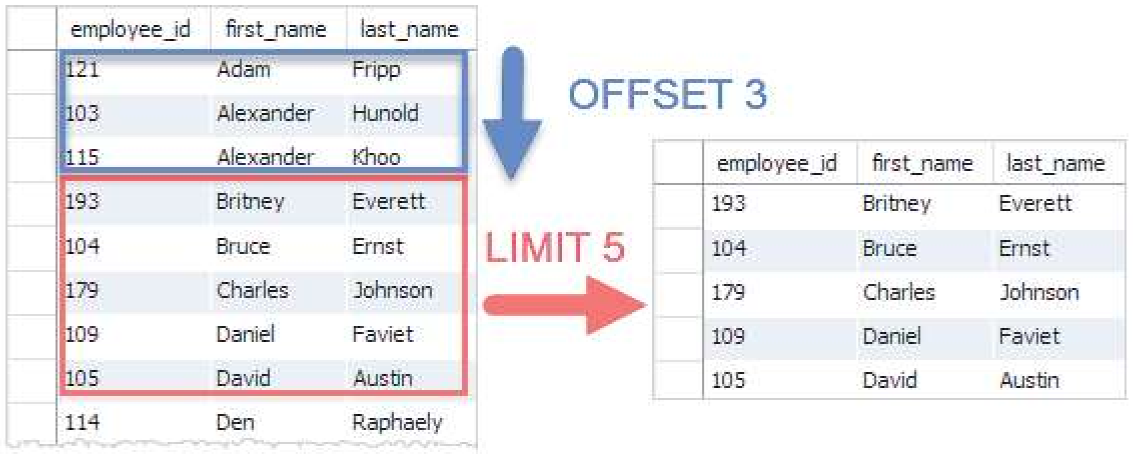


employee_id	first_name	last_name
121	Adam	Fripp
103	Alexander	Hunold
115	Alexander	Khoo
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
109	Daniel	Faviet
105	David	Austin
114	Den	Raphaely

employee_id	first_name	last_name
121	Adam	Fripp
115	Alexander	Khoo
103	Alexander	Hunold
193	Britney	Everett
104	Bruce	Ernst

The following example uses both LIMIT & OFFSET clauses to return five rows starting from the 4th row:

```
SELECT
    employee_id, first_name, last_name
FROM
    employees
ORDER BY first_name
LIMIT 5 OFFSET 3;
```



employee_id	first_name	last_name
121	Adam	Fripp
103	Alexander	Hunold
115	Alexander	Khoo
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
109	Daniel	Faviet
105	David	Austin
114	Den	Raphaely

employee_id	first_name	last_name
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
109	Daniel	Faviet
105	David	Austin

4.Left Join and Right Join with example.

MySQL has mainly two kinds of joins named LEFT JOIN and RIGHT JOIN. The main difference between these joins is the ***inclusion of non-matched rows***. The LEFT JOIN includes all records from the left side and matched rows from the right table, whereas RIGHT JOIN returns all rows from the right side and unmatched rows from the left table. In this section, we are going to know the popular differences between LEFT and RIGHT join. Before exploring the comparison, let us first understand JOIN, LEFT JOIN, and RIGHT JOIN clause in MySQL.

What is JOIN Clause?

A join is used to query data from multiple tables and returns the combined result from two or more tables through a condition. The condition in the join clause indicates how columns are matched between the specified tables.

What is the LEFT JOIN Clause?

The Left Join clause joins two or more tables and returns all rows from the left table and matched records from the right table or returns null if it does not find any matching record. It is also known as **Left Outer Join**. So, Outer is the optional keyword to use with Left Join.

To read more information about the LEFT join,

What is the RIGHT JOIN Clause?

The Right Join clause joins two or more tables and returns all rows from the right-hand table, and only those results from the other table that fulfilled the specified join condition. If it finds unmatched records from the left side table, it returns Null value. It is also known as **Right Outer Join**. So, Outer is the optional clause to use with Right Join.

We can understand it with the following **visual representation**.

To read more information about RIGHT JOIN,

Syntax of LEFT JOIN Clause

The following is the general syntax of LEFT JOIN:

```
SELECT column_list  FROM table_name1
LEFT JOIN table_name2
ON column_name1 = column_name2
WHERE join_condition
```

The following is the general syntax of LEFT OUTER JOIN:

```
SELECT column_list  FROM table_name1
LEFT OUTER JOIN table_name2
ON column_name1 = column_name2
WHERE join_condition
```

Syntax of RIGHT JOIN Clause

The following is the general syntax of RIGHT JOIN:

```
SELECT column_list  FROM table_name1
RIGHT JOIN table_name2
ON column_name1 = column_name2
WHERE join_condition
```

The following is the general syntax of RIGHT OUTER JOIN:

5. Explain Group by

SQL group by

In SQL, The **Group By** statement is used for organizing similar data into groups. The data is further organized with the help of equivalent function. It means, if different rows in a precise column have the same values, it will arrange those rows in a group.

- **The SELECT** statement is used with the **GROUP BY** clause in the SQL query.
- **WHERE** clause is placed before the **GROUP BY** clause in **SQL**.
- **ORDER BY** clause is placed after the **GROUP BY** clause in **SQL**.

Syntax:

1. **SELECT** column1, function_name(column2)
2. **FROM** table_name
3. **WHERE** condition
4. **GROUP BY** column1, column2
5. **ORDER BY** column1, column2;
6. function_name: **Table name**.
7. Condition: which we used.

Sample Table:

Employee

S.no	Name	AGE	Salary
1	**John**	**24**	**25000**
2	**Nick**	22	22000
3	**Amara**	25	15000
4	**Nick**	22	22000
5	**John**	24	25000

SUBJECT	YEAR	NAME
C language	**2**	**John**
C language	**2**	Ginny
C language	**2**	Jasmeen
C language	**3**	Nick

C language	**3**	Amara
Java	**1**	Sifa
Java	**1**	dolly

Student

Example:

Group By single column: Group By single column is used to place all the rows with the same value. These values are of that specified column in one group. It signifies that all rows will put an equal amount through a single column, which is of one appropriate column in one group.

Consider the below query:

```
1. SELECT NAME, SUM (SALARY) FROM Employee
2. GROUP BY NAME;
```

NAME	SALARY
John	50000
Nick	44000
Amara	15000

In the output, the rows which hold duplicate **NAME** are grouped under a similar NAME, and their corresponding SALARY is the sum of the SALARY of the duplicate rows.

SQL LOGICAL OPERATORS

The Logical Operator is nothing but which returns the result in one form, i.e., either it will display the query is true, or the query is false. The results displayed to combine or merge more than one true or false data.

The Logical Operators in SQL are as follows:

1. SQL AND OPERATOR
2. SQL OR OPERATOR
3. SQL NOT OPERATOR
4. SQL BETWEEN OPERATOR
5. SQL IN OPERATOR
6. SQL LIKE OPERATOR

Let's understand each and every operator one by one with the help of examples. All the queries in the examples will be written using the MySQL database.

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
2	Tejaswini Naik	75000	Delhi	System Engineer	2019-12-24
3	Anuja Sharma	40000	Jaipur	Manager	2021-08-15
4	Anushka Tripathi	90000	Mumbai	Software Tester	2021-06-13
5	Rucha Jagtap	45000	Bangalore	Project Manager	2020-08-09
6	Rutuja Deshmukh	60000	Bangalore	Manager	2019-07-17
7	Swara Baviskar	55000	Jaipur	System Engineer	2021-10-10

8	Sana Sheik	45000	Pune	Software Engineer	2020-09-10
9	Swati Kumari	50000	Pune	Software Tester	2021-01-01
10	Mayuri Patel	60000	Mumbai	Project Manager	2020-10-02
11	Simran Khanna	45500	Kolhapur	HR	2019-01-02
12	Shivani Wagh	50500	Delhi	Software Developer	2016-09-10
13	Kiran Maheshwari	50000	Nashik	HR	2013-12-12
14	Tejal Jain	40000	Delhi	Project Manager	2017-11-10
15	Mohini Shah	38000	Pune	Software Developer	2019-03-05

1. SQL AND Operator

The SQL AND operator is used with the where clause in the SQL Query. AND operator in SQL returns only those records which satisfy both the conditions in the SQL query.

Let's understand the below example, which explains how to execute AND operator in an SQL query.

Example:

Write a query to retrieve only those records of employees from the employees table where the designation is 'Project Manager' and the City to which the employee belongs to is Mumbai.

Query:

```
mysql> SELECT * FROM employees WHERE City = "Mumbai" AND Designation = "Project M
anager";
```

Here we have written a SELECT query with a WHERE clause on the City column and Designation column with 'AND' operator in between both the conditions. Any record in the employees table that meets both conditions, i.e., the city to which the employee belongs is Mumbai, and their designation is Project Manager, will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
10	Mayuri Patel	60000	Mumbai	Project Manager	2020-10-02

There are only two records in the employees table whose city name is equal to 'Mumbai' and designation name is equal to 'Project Manager'.

2. SQL BETWEEN Operator

This operator displays the records which fall between the given ranges in the SQL query. The results of the BETWEEN operator include begin and end values of the given range.

Let's understand the below example, which explains how to execute BETWEEN operator in an SQL query.

Example:

Write a query to retrieve only those records of an employee from the employees table where employee salary lies between 50000 to 90000.

Query:

```
1. mysql> SELECT * FROM employees WHERE Salary BETWEEN 50000 AND 90000;
```

Here we have written a SELECT query with a WHERE clause on the Salary column with the 'BETWEEN' operator. BETWEEN operator is followed by beginning and end values 50000 and 90000 respectively with 'AND' operator in between. Any record in the employees table that meets the condition, i.e., the employee's salary is between 50000 and 90000, will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
2	Tejaswini Naik	75000	Delhi	System Engineer	2019-12-24
4	Anushka Tripathi	90000	Mumbai	Software Tester	2021-06-13
6	Rutuja Deshmukh	60000	Bangalore	Manager	2019-07-17
7	Swara Baviskar	55000	Jaipur	System Engineer	2021-10-10
9	Swati Kumari	50000	Pune	Software Tester	2021-01-01
10	Mayuri Patel	60000	Mumbai	Project Manager	2020-10-02
12	Shivani Wagh	50500	Delhi	Software Developer	2016-09-10
13	Kiran Maheshwari	50000	Nashik	HR	2013-12-12

There are nine records in the employees table whose salary falls between 50000 to 90000.

3. SQL OR Operator

The SQL OR operator is used with the where clause in an SQL Query. AND operator in SQL returns only those records that satisfy any of the conditions in the SQL query.

Let's understand the below example, which explains how to execute OR operator an SQL query.

Example:

Write a query to retrieve only those records of employees from the employees table where the employee's designation is 'System Engineer' or the city to which the employee belongs is Mumbai.

Query:

```
1. mysql> SELECT * FROM employees WHERE Designation = "System Engineer" OR City = "Mumbai";
```

Here we have written a SELECT query with a WHERE clause on the City column and Designation column with the 'OR' operator in between both the conditions. Any record in the employees table that meets any of the conditions, i.e., the city to which the employee belongs is Mumbai, or their designation is System Engineer, will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
2	Tejaswini Naik	75000	Delhi	System Engineer	2019-12-24
4	Anushka Tripathi	90000	Mumbai	Software Tester	2021-06-13
7	Swara Baviskar	55000	Jaipur	System Engineer	2021-10-10
10	Mayuri Patel	60000	Mumbai	Project Manager	2020-10-02

There are only five records in the employees table whose city name is equal to 'Mumbai' or the employee's designation is equal to 'System Engineer'.

4. SQL IN Operator

When we want to check for one or more than one value in a single SQL query, we use IN operator with the WHERE clause in a SELECT query.

Let's understand the below example, which explains how to execute IN operator in an SQL query.

Example:

Write a query to retrieve only those records of employees from the employees table where the city to which the employee belongs to is either Mumbai, Bangalore, or Pune.

Query:

```
1. mysql> SELECT * FROM employees WHERE City IN ("Mumbai", "Bangalore", "Pune");
```

Here we have written a SELECT query with a WHERE clause on the City column followed by IN operator. Since we wanted only those records that belongs to Mumbai, Bangalore, or Pune, we have passed Mumbai, Bangalore, and Pune as parameters to the IN operator. So, if the City value of any record matches with the places passed to the IN operator will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
4	Anushka Tripathi	90000	Mumbai	Software Tester	2021-06-13
5	Rucha Jagtap	45000	Bangalore	Project Manager	2020-08-09
6	Rutuja Deshmukh	60000	Bangalore	Manager	2019-07-17
8	Sana Sheik	45000	Pune	Software Engineer	2020-09-10
9	Swati Kumari	50000	Pune	Software Tester	2021-01-01
10	Mayuri Patel	60000	Mumbai	Project Manager	2020-10-02
15	Mohini Shah	38000	Pune	Software Developer	2019-03-05

There are only eight records in the employees table where the city to which the employee belongs is either Mumbai, Bangalore, or Pune.

5. SQL NOT Operator

NOT operator in SQL shows those records from the table where the criteria is not met. NOT operator is used with where clause in a SELECT query.

Let's understand the below example, which explains how to execute NOT operator in SQL query.

Example:

Write a query to retrieve only those records of employees from the employees table where the employee's designation is not Project Manager.

Query:

```
1. mysql> SELECT * FROM employees WHERE NOT Designation = "Project Manager";
```

Here we have written a SELECT query with a WHERE clause on the Designation column followed by NOT operator. Since we wanted only those records whose designation is other than a project manager, we have given the designation value as Project Manager to the NOT operator. So, if the designation value of any record does not match with the value given to the NOT operator will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
2	Tejaswini Naik	75000	Delhi	System Engineer	2019-12-24
3	Anuja Sharma	40000	Jaipur	Manager	2021-08-15
4	Anushka Tripathi	90000	Mumbai	Software Tester	2021-06-13
6	Rutuja Deshmukh	60000	Bangalore	Manager	2019-07-17
7	Swara Baviskar	55000	Jaipur	System Engineer	2021-10-10
8	Sana Sheik	45000	Pune	Software Engineer	2020-09-10
9	Swati Kumari	50000	Pune	Software Tester	2021-01-01
11	Simran Khanna	45500	Kolhapur	HR	2019-01-02
12	Shivani Wagh	50500	Delhi	Software Developer	2016-09-10
13	Kiran Maheshwari	50000	Nashik	HR	2013-12-12

There are eleven records in the employees table whose designation is not a project manager.

6. SQL LIKE Operator

LIKE Operator in SQL displays only those data from the table which matches the pattern specified in the query. Percentage (%) and underscore (_) are the two wildcard operators used with LIKE Operator to perform pattern matching tasks.

Let's understand the below example, which explains how to execute the LIKE operator in an SQL query.

Example:

Write a query to retrieve only those records of employees from the employees table whose salary starts with the digit 5.

Query:

```
mysql> SELECT * FROM employees WHERE Salary LIKE "5%";
```

Here we have written a SELECT query with a WHERE clause on the Salary column followed by the LIKE operator. Since we wanted only those records whose salary starts with the digit 5, we have given the value to the LIKE operator as '5%'. So, if the salary value of any record starts with the digit 5, followed by any other digit will only be considered in output.

You will get the following output:

E_ID	Name	Salary	City	Designation	Date_of_Joining
1	Sakshi Kumari	50000	Mumbai	Project Manager	2021-06-20
7	Swara Baviskar	55000	Jaipur	System Engineer	2021-10-10
9	Swati Kumari	50000	Pune	Software Tester	2021-01-01
12	Shivani Wagh	50500	Delhi	Software Developer	2016-09-10
13	Kiran Maheshwari	50000	Nashik	HR	2013-12-12

7.Data types in mySQL

MySQL Data Types

A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc. It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored. In MySQL, each database table has many columns and contains specific data types for each column.

We can determine the data type in MySQL with the following characteristics:

- The type of values (fixed or variable) it represents.
- The storage space it takes is based on whether the values are a fixed-length or variable length.
- Its values can be indexed or not.
- How MySQL performs a comparison of values of a particular data type.

MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types that can be broken into the following categories: numeric, date and time, string types, spatial types, and JSON data types.

Numeric Data Type

MySQL has all essential SQL numeric data types. These data types can include the exact numeric data types (For example, integer, decimal, numeric, etc.), as well as the approximate numeric data types (For example, float, real, and double precision). It also supports BIT datatype to store bit values. In MySQL, numeric data types are categories into two types, either signed or unsigned except for bit data type.

Date and Time Data Type:

This data type is used to represent temporal values such as date, time, datetime, timestamp, and year. Each temporal type contains values, including zero. When we insert the invalid value, MySQL cannot represent it, and then zero value is used.

String Data Types:

The string data type is used to hold plain text and binary data, for example, files, images, etc. MySQL can perform searching and comparison of string value based on the pattern matching such as LIKE operator, Regular Expressions, etc.

Binary Large Object Data Types (BLOB):

BLOB in MySQL is a data type that can hold a variable amount of data. They are categories into four different types based on the maximum length of values can hold.

Spatial Data Types

It is a special kind of data type which is used to hold various geometrical and geographical values. It corresponds to OpenGIS classes. The following table shows all spatial types that support in MySQL:

8.Float and double, Where to use them.

- **FLOAT(M,D) −** A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

- **DOUBLE(M,D) −** A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

- **Floating-Point Types**
- The FLOAT and DOUBLE types represent approximate numeric data values. MySQL uses four bytes for single-precision values and eight bytes for double-precision values.

Types	**Description**
FLOAT	A precision from 0 to 23 results in a four-byte single-precision FLOAT column
DOUBLE	A precision from 24 to 53 results in an eight-byte double-precision DOUBLE column.

- MySQL allows a nonstandard syntax: FLOAT(M,D) or REAL(M,D) or DOUBLE PRECISION(M,D). Here values can be stored up to M digits in total where D represents the decimal point. For example, a column defined as FLOAT(8,5) will look like -999.99999. MySQL performs rounding when storing values, so if you insert 999.00009 into a FLOAT(7,4) column, the approximate result is 999.0001.

9.Functions in mySQL

MySQL has many built-in functions.

This reference contains string, numeric, date, and some advanced functions in MySQL.

MySQL String Functions

Function	Description
ASCII	Returns the ASCII value for the specific character
CHAR_LENGTH	Returns the length of a string (in characters)
CHARACTER_LENGTH	Returns the length of a string (in characters)
CONCAT	Adds two or more expressions together
CONCAT_WS	Adds two or more expressions together with a separator
FIELD	Returns the index position of a value in a list of values
FIND_IN_SET	Returns the position of a string within a list of strings
FORMAT	Formats a number to a format like "#,###,###.##", rounded to a spe
INSERT	Inserts a string within a string at the specified position and for a certain
INSTR	Returns the position of the first occurrence of a string in another string

LCASE	Converts a string to lower-case
LEFT	Extracts a number of characters from a string (starting from left)
LENGTH	Returns the length of a string (in bytes)
LOCATE	Returns the position of the first occurrence of a substring in a string
LOWER	Converts a string to lower-case
LPAD	Left-pads a string with another string, to a certain length
LTRIM	Removes leading spaces from a string
MID	Extracts a substring from a string (starting at any position)
POSITION	Returns the position of the first occurrence of a substring in a string
REPEAT	Repeats a string as many times as specified
REPLACE	Replaces all occurrences of a substring within a string, with a new subst
REVERSE	Reverses a string and returns the result

RIGHT	Extracts a number of characters from a string (starting from right)
RPAD	Right-pads a string with another string, to a certain length
RTRIM	Removes trailing spaces from a string
SPACE	Returns a string of the specified number of space characters
STRCMP	Compares two strings
SUBSTR	Extracts a substring from a string (starting at any position)
SUBSTRING	Extracts a substring from a string (starting at any position)
SUBSTRING_INDEX	Returns a substring of a string before a specified number of delimiter oc
TRIM	Removes leading and trailing spaces from a string
UCASE	Converts a string to upper-case
UPPER	Converts a string to upper-case

10. Having clause.

MySQL HAVING Clause is used with GROUP BY clause. It always returns the rows where condition is TRUE.

Syntax:

1. **SELECT** expression1, expression2, ... expression_n,

   aggregate_function (expression)
2. **FROM** tables
3. [**WHERE** conditions]
4. **GROUP BY** expression1, expression2, ... expression_n
5. **HAVING** condition;

Parameters

aggregate_function: It specifies any one of the aggregate function such as SUM, COUNT, MIN, MAX, or AVG.

expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

WHERE conditions: It is optional. It specifies the conditions for the records to be selected.

HAVING condition: It is used to restrict the groups of returned rows. It shows only those groups in result set whose conditions are TRUE.

- The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.
- The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

The following code block shows the position of the HAVING Clause in a query.

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used. The following code block has the syntax of the SELECT statement including the HAVING clause .