Interview -02

Interview Questions

(Practice Project)

PW SKILLS

Easy

1.What is the difference between order and group by?

Ans: ORDER BY: The ORDER BY clause is used to sort the result set of a query in ascending (ASC) or descending (DESC) order. It's typically the last clause in a SELECT statement.

Key points:

- Can sort by multiple columns
- ASC is the default order if not specified
- Can use column positions instead of names (e.g., ORDER BY 1, 2)
- Can sort by expressions or computed values

The following is the syntax to use the ORDER BY clause in a SQL statement:

- SELECT expressions
- FROM tables
- [WHERE conditions]
- ORDER BY expression [ASC | DESC];

Example:

```
SELECT employee_name, department, salary
FROM employees
WHERE salary > 50000
ORDER BY department ASC, salary DESC;
```

GROUP BY: The GROUP BY clause is used to group rows that have the same values in specified columns. It's often used with aggregate functions to perform calculations on each group.

Key points:

- Typically used with aggregate functions (COUNT, SUM, AVG, MAX, MIN)
- All columns in the SELECT list that are not inside aggregate functions must be in the GROUP BY clause
- Comes after WHERE but before HAVING and ORDER BY
- Can group by multiple columns

The following is the syntax to use GROUP BY clause in a SQL statement:

- SELECT column_name,
- function(column_name)
- FROM table_name
- WHERE condition GROUP BY column_name;

Example:

```
SELECT department, AVG(salary) as avg_salary, COUNT(*) as employee_count
FROM employees
GROUP BY department
HAVING COUNT(*) > 5
ORDER BY avg_salary DESC;
```

2.What is ENUM?

Ans: An ENUM is a string object with a value chosen from a list of permitted values that are enumerated explicitly in the column specification at table creation time.

```
CREATE TABLE shirts (
    name VARCHAR(40),
    size ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);

INSERT INTO shirts (name, size) VALUES ('T-Shirt', 'medium');
```

Key points:

- Improves data integrity by restricting input to predefined values
- Can save storage space compared to VARCHAR
- The list of values is defined when creating the table
- Values are case-insensitive
- The first value in the list is the default value

3.What is CTE?

Ans: A CTE or common table expression is an expression that contains temporary result set which is defined in a SQL statement.

```
WITH high_salary_employees AS (
    SELECT * FROM employees WHERE salary > 100000
)
SELECT department, AVG(salary) as avg_high_salary
FROM high_salary_employees
GROUP BY department;
```

Key points:

- Improves readability and maintainability of complex queries
- Can be referenced multiple times in a single query
- Can be recursive (refer to itself)
- Exists only for the duration of the query

4.What is the difference between the RANK() and DENSE_RANK() function?

Ans: The only difference between the RANK() and DENSE_RANK() functions is in cases where there is a "tie"; i.e., in cases where multiple values in a set have the same ranking. In such cases, RANK() will assign non consecutive "ranks" to the values in the set (resulting in gaps between the integer ranking values when there is a tie), whereas DENSE_RANK() will assign consecutive ranks to the values in the set (so there will be no gaps between the integer ranking values in the case of a tie).

RANK():
- Assigns the same rank to tied values
- Skips the next rank(s) after ties

DENSE_RANK():
- Assigns the same rank to tied values
- Does not skip ranks after ties

For example, consider the set {25, 25, 50, 75, 75, 100}.
For such a set, RANK() will return {1, 1, 3, 4, 4, 6} (note that the values 2 and 5 are skipped), whereas DENSE_RANK() will return {1,1,2,3,3,4}.

Q5: What is a self-referencing foreign key? Give an example.

Ans: A self-referencing foreign key is a foreign key in a table that refers to the same table. For example, consider an `Employee` table where each employee may have a manager who is also an employee. In this case, the `manager_id` column in the `Employee` table is a self-referencing foreign key that refers to the `employee_id` column in the same table. Here is how you can define it:

Code:

```sql
CREATE TABLE Employee (
    name VARCHAR(25) NOT NULL,
    employee_id CHAR(9) NOT NULL,
    manager_id CHAR(9) NULL,
    salary DECIMAL(10,2) NULL,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (manager_id) REFERENCES Employee(employee_id) ON DELETE CASCADE
);
```

Q6: Both TIMESTAMP and DATETIME are used to store date and time. Explain the difference between them and when one should be used.

Ans:
- TIMESTAMP: Stores date and time in UTC and converts it to the server's time zone when retrieved. Useful when dealing with global applications where time zone consistency is important.
- DATETIME: Stores the date and time as provided without any time zone conversion. Use this when time zone awareness is not necessary, and you want to store the exact date and time as entered.

Q7: Explain the GRANT command in MySQL.

Ans: The `GRANT` command in MySQL is used to provide specific privileges to users on database objects. For example, to grant `SELECT` and `INSERT` permissions on a table named `customertable` to a user `username` at `localhost`, you would use:

```sql
GRANT SELECT, INSERT ON customertable TO 'username'@'localhost';
```

Q8: Explain the use of FEDERATED tables in MySQL.

Ans: FEDERATED tables allow access to tables located in remote MySQL servers as if they were local tables. The actual data resides on the remote server, but you can interact with it through the federated table. To create a federated table:

```sql
CREATE TABLE table_fed (
    ...
)
ENGINE=FEDERATED
CONNECTION='mysql://user@remote_hostname:port/federated_schema/table';
```

Q9: How can ENUM be used in MySQL? Give an example.

Ans: The `ENUM` type allows a column to store one value from a predefined list of values. For example:

```sql
CREATE TABLE Student (
    rollnumber INT NOT NULL,
    name VARCHAR(25) NOT NULL,
    country ENUM('USA', 'UK', 'Australia'),
    PRIMARY KEY(rollnumber)
);

INSERT INTO Student VALUES (6, 'John', 'USA');
```

Medium

Q10: What are different TEXT data types in MySQL? What is the difference between TEXT and VARCHAR?

Ans:
- TEXT Data Types:
- `TINYTEXT`: Up to 255 characters
- `TEXT`: Up to 65,535 characters
- `MEDIUMTEXT`: Up to 16,777,215 characters
- `LONGTEXT`: Up to 4,294,967,295 characters

Difference between TEXT and VARCHAR:
- `VARCHAR` stores variable-length strings and is stored inline in the table.
- `TEXT` stores large amounts of text and is stored outside the table with a pointer in the table. `TEXT` requires a prefix length for indexing and does not support default values.

Q11: What different stored objects are supported in MySQL?

Ans:
- VIEW: A virtual table based on the result set of a query.
- STORED PROCEDURE: A set of SQL statements stored in the database that can be executed with a `CALL` statement.
- STORED FUNCTION: Similar to a function that returns a single value and can be used in SQL expressions.
- TRIGGER: A program that automatically executes in response to specific events (INSERT, UPDATE, DELETE) on a table.
- EVENT: A scheduled task that runs at specified times or intervals.

Q12: What is a Stored Function in MySQL? How are they different from Stored Procedures?

Ans:
- Stored Function: A stored function contains complex logic and returns a single value. It can be used in SQL expressions and can perform operations like `SELECT`, `INSERT`, and `UPDATE`.
- Difference from Stored Procedures:
- Stored procedures do not return values and are invoked with the `CALL` statement to perform operations.
- Stored functions return a value and are invoked within SQL expressions. Functions have limitations compared to procedures, such as not being able to perform certain operations.

Q13: What is AUTO INCREMENT in MySQL? Explain with an example.

Ans: `AUTO_INCREMENT` is used to automatically generate a unique integer value for a column, usually for primary keys. Example:

```
CREATE TABLE Student (
    studentid INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL,
    PRIMARY KEY (studentid)
);
```

Inserting a row without specifying `studentid` will automatically assign the next unique value.

Hard

Question 14:
Practical Question:
1. How to find nth highest salary in SQL Server using a Sub-Query
2. How to find nth highest salary in SQL Server using a CTE
3. How to find the 2nd, 3rd or 15th highest salary

Let's use the following Employees table for this demo.

ID	FirstName	LastName	Gender	Salary
1	Amit	Sharma	Male	70000
2	Ravi	Patel	Male	60000
3	Suresh	Reddy	Male	45000
4	Amit	Sharma	Male	70000
5	Rajesh	Patel	Male	45000
6	Pooja	Iyer	Female	30000
7	Sita	Nair	Female	35000
8	Vikram	Kumar	Male	80000

Use the following script to create Employees table

```
-- Create the Employees table
CREATE TABLE Employees
(
    ID INT PRIMARY KEY IDENTITY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Gender NVARCHAR(50),
    Salary INT
);
GO

-- Insert data into Employees table with Indian names
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Amit', 'Sharma', 'Male', 70000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Ravi', 'Patel', 'Male', 60000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Suresh', 'Reddy', 'Male', 45000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Amit', 'Sharma', 'Male', 70000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Rajesh', 'Patel', 'Male', 45000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Pooja', 'Iyer', 'Female', 30000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Sita', 'Nair', 'Female', 35000);
INSERT INTO Employees (FirstName, LastName, Gender, Salary) VALUES ('Vikram', 'Kumar', 'Male', 80000);
GO
```

Q1: To find the highest salary:

Ans: Use the `MAX()` function to get the highest salary.

SELECT MAX(Salary) AS HighestSalary
FROM Employees;

Q2: To find the second highest salary:

Ans: Use a subquery with the `MAX()` function to get the second highest salary.

```
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

Q3: To find the nth highest salary using a subquery:

Ans:

```
SELECT TOP 1 Salary
FROM (
   SELECT DISTINCT TOP N Salary
   FROM Employees
   ORDER BY Salary DESC
) AS Result
ORDER BY Salary;
```

Replace `N` with the desired position to get the nth highest salary (e.g., 2 for the second highest).

Q4: To find the nth highest salary using a Common Table Expression (CTE):

Ans:

```
WITH Result AS (
   SELECT Salary,
        DENSE_RANK() OVER (ORDER BY Salary DESC) AS DenseRank
   FROM Employees
)
SELECT TOP 1 Salary
FROM Result
WHERE DenseRank = N;
```

Replace `N` with the desired position to get the nth highest salary (e.g., 2 for the second highest).

Note: If you need to find the nth highest salary where duplicates might be present, use this query with `ROW_NUMBER()`:

```
WITH Result AS (
   SELECT Salary,
        ROW_NUMBER() OVER (ORDER BY Salary DESC) AS RowNumber
   FROM Employees
)
SELECT Salary
FROM Result
WHERE RowNumber = 3;
```

Replace `3` with the desired rank position. This query assumes there are no duplicates for accurate results.

Q15: Find Employees with Salaries Higher than the Average Salary of Their Gender

Ans: Write a query to find employees whose salaries are higher than the average salary for their gender.

```
sql
SELECT FirstName, LastName, Salary, Gender
FROM Employees AS e
WHERE Salary > (
    SELECT AVG(Salary)
    FROM Employees
    WHERE Gender = e.Gender
);
```

Q16: Find the Third Highest Salary and the Employees Earning It

Ans: Write a query to find the third highest salary and list all employees who earn that salary.

```
sql
WITH SalaryRank AS (
    SELECT Salary,
           DENSE_RANK() OVER (ORDER BY Salary DESC) AS Rank
    FROM Employees
)
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary = (
    SELECT Salary
    FROM SalaryRank
    WHERE Rank = 3
);
```

Q17: Find the Employees with the Most Frequent Salary

Ans: Write a query to find the salary that occurs most frequently and list all employees earning that salary.

```
sql
WITH SalaryFrequency AS (
    SELECT Salary,
           COUNT(*) AS Frequency
    FROM Employees
    GROUP BY Salary
),
MaxFrequency AS (
    SELECT MAX(Frequency) AS MaxFreq
    FROM SalaryFrequency
)
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary IN (
    SELECT Salary
    FROM SalaryFrequency
    WHERE Frequency = (SELECT MaxFreq FROM MaxFrequency)
);
```

Q18: Find Employees Who Earn More Than All Employees in a Specific Department (Assuming a Department Column)

Ans: Assuming there is a `Department` column, write a query to find employees who earn more than every employee in the "Sales" department.

```
sql
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary > ALL (
    SELECT Salary
    FROM Employees
    WHERE Department = 'Sales'
);
```

Q19: Find Employees Whose Salary is Greater Than the Average Salary of Employees with the Same Last Name

Ans: Write a query to find employees whose salary is greater than the average salary of employees with the same last name.

```
sql
SELECT e.FirstName, e.LastName, e.Salary
FROM Employees AS e
WHERE e.Salary > (
    SELECT AVG(Salary)
    FROM Employees
    WHERE LastName = e.LastName
);
```