MySQL Clauses

MySQL Clauses:

1. MySQL WHERE
2. MySQL DISTINCT
3. MySQL FROM
4. MySQL ORDER BY
5. MySQL GROUP BY
6. MySQL HAVING
7. MySQL LIMIT

---

1. WHERE Clause

The WHERE clause is used to filter records that meet specific conditions. It's applied after the FROM clause and before the ORDER BY clause in a SQL query.

I.M.P : Operators: =, !=, >, <, >=, <=, BETWEEN, LIKE, IN, IS NULL, IS NOT NULL, AND, OR, NOT.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

2. DISTINCT Keyword

The DISTINCT keyword is used to return unique values from a column or a set of columns. It eliminates duplicate rows from the result set.

Syntax:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

---

3. FROM Clause

The FROM clause specifies the table from which to retrieve the data. It is required in every SELECT statement.

Syntax:

```
SELECT column1, column2, ...
FROM table_name;
```

4. ORDER BY Clause

The ORDER BY clause is used to sort the result set in either ascending or descending order. By default, it sorts in ascending order.

I.M.P : Default: ASC (ascending). Use DESC for descending order.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC OR DESC], column2 [ASC OR DESC], ...;
```

Example:

```
SELECT * FROM Employees
ORDER BY Salary DESC;
```

5. GROUP BY Clause

The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows. Often used with aggregate functions like COUNT, SUM, AVG, MAX, and MIN.

Syntax:

```
SELECT column1, aggregate_function(column2)
FROM table_name
GROUP BY column1;
```

=> HAVING Clause:

The HAVING clause is used to filter groups based on a condition. It is applied after GROUP BY.

ex:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department
HAVING AVG(Salary) > 60000;
```

=> Order of Clauses:
GROUP BY comes after the WHERE clause and before the ORDER BY clause in a SQL query.

You can use the ORDER BY clause to sort the result set based on aggregated data.

ex.

```
SELECT Department, COUNT(*) AS NumberOfEmployees
FROM Employees
GROUP BY Department
ORDER BY NumberOfEmployees DESC;
```

---

6. HAVING Clause

The HAVING clause is used to filter groups created by the GROUP BY clause. It is similar to the WHERE clause but is used for filtering aggregated results.

HAVING vs. WHERE:

=> WHERE filters rows before grouping and is used to filter individual records.

=> HAVING filters groups after aggregation and is used to filter the results of a GROUP BY query.

Syntax:

```
SELECT column1, aggregate_function(column2)
FROM table_name
GROUP BY column1
HAVING condition;
```

I.M.P => Having mainly use with both Aggregate Functions and Non - Aggregate Functions

1. Using HAVING with Aggregate Functions:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department
HAVING AVG(Salary) > 60000;
```

In this example, HAVING filters departments where the average salary is greater than 60,000.

2. Using HAVING without Aggregate Functions:

```
SELECT Department, JobTitle
FROM Employees
GROUP BY Department, JobTitle
HAVING Department = 'Sales';
```

In this example, HAVING filters groups where the department is 'Sales'. Here, HAVING is used to filter based on a non-aggregated column.

---

7. LIMIT Clause:

The LIMIT clause in MySQL is used to specify the number of rows to return in a result set. It is particularly useful for paginating results or when you only want to see a specific number of rows from the query results.

Example:

1. Basic Usage of LIMIT

```
SELECT * FROM employees
LIMIT 5;
```

2. LIMIT with OFFSET

You can also use LIMIT with an OFFSET to skip a specific number of rows.

For example, to skip the first 10 rows and then return the next 5 rows:

```
SELECT * FROM employees
LIMIT 5 OFFSET 10;
```

⭐Practice Questions that focus on using MySQL clauses

1. Retrieve all records from the students table where the age is greater than 18.
2. Get the distinct list of job titles from the jobs table.
3. Select all columns from the employees table and order the results by hire_date in descending order.
4. Find the number of unique cities from the customers table.
5. Retrieve all records from the products table where the category is 'Electronics' and the price is greater than 100.
6. Get the list of all employees grouped by their

department from the employees table.
7. Select distinct departments from the employees table and order the results by department name.
8. Get the count of students in each grade level from the students table.
9. Retrieve all records from the sales table and order them by sales_date and amount in ascending order.
10. Get the average salary for each job title in the employees table where the average salary is greater than 50,000.
11. Get the top 3 highest-paid employees.

👉 ANS:

1. Retrieve all records from the students table where the age is greater than 18.
=> SELECT * FROM students WHERE age > 18;

2. Get the distinct list of job titles from the jobs table.
=> SELECT DISTINCT job_title FROM jobs;

3. Select all columns from the employees table and order the results by hire_date in descending order.
=> SELECT * FROM employees ORDER BY hire_date DESC;

4. Find the number of unique cities from the customers table.
=> SELECT DISTINCT cities FROM customers;

5. Retrieve all records from the products table where the category is 'Electronics' and the price is greater than 100.
=> SELECT * FROM products WHERE category = "Electronics" AND price > 100;

6. Get the list of all employees grouped by their department from the employees table.
=> SELECT employee_name, department FROM employees GROUP BY employee_name, department;

7. Select distinct departments from the employees table and order the results by department name.
=> SELECT DISTINCT department FROM employees ORDER BY department ASC;

8. Get the count of students in each grade level from the students table.
=> SELECT grade, COUNT(*) FROM students GROUP BY grade;

9. Retrieve all records from the sales table and order them by sales_date and amount in ascending order.
=> SELECT * FROM sales ORDER BY sales_date, amount;

10. Get the average salary for each job title in the employees table where the average salary is greater than 50,000.
=> SELECT AVG(salary), job_title FROM employees GROUP BY job_title HAVING AVG(salary) > 50000;

11. Get the top 3 highest-paid employees.
=> SELECT * from employees ORDER BY salary DESC LIMIT 3;