## MySQL Basic **Select** **Statement**:

1.

*-- Selecting the first\_name column and aliasing it as "First Name"*

**SELECT** first\_name "First Name",

*-- Selecting the last\_name column and aliasing it as "Last Name"*

last\_name "Last Name"

*-- Selecting data from the employees table*

**FROM** employees;

2.

*-- Selecting distinct values from the department\_id column*

**SELECT** **DISTINCT** department\_id

*-- Selecting data from the employees table*

**FROM** employees;

3.

*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Ordering the result set by the first\_name column in descending order*

**ORDER** **BY** first\_name **DESC**;

4.

*-- Selecting the first\_name, last\_name, and salary columns*

**SELECT** first\_name, last\_name, salary,

*-- Calculating 15% of the salary and aliasing it as "PF" (Provident Fund)*

salary \* 0.15 **AS** PF

*-- Selecting data from the employees table*

**FROM** employees;

5.

*-- Selecting specific columns: employee\_id, first\_name, last\_name, and salary*

**SELECT** employee\_id, first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Ordering the result set by the salary column in ascending order*

**ORDER** **BY** salary;

6.

*-- Calculating the sum of salaries for all employees*

**SELECT** **SUM**(salary)

*-- Selecting data from the employees table*

**FROM** employees;

7.

*-- Selecting the maximum and minimum salary values from the employees table*

**SELECT** **MAX**(salary), **MIN**(salary)

*-- Selecting data from the employees table*

**FROM** employees;

8.

*-- Calculating the average salary and counting the total number of employees*

**SELECT** **AVG**(salary), **COUNT**(\*)

*-- Selecting data from the employees table*

**FROM** employees;

9.

*-- Counting the total number of records (rows) in the employees table*

**SELECT** **COUNT**(\*)

*-- Selecting data from the employees table*

**FROM** employees;

10.

*-- Counting the total number of distinct job IDs in the employees table*

**SELECT** **COUNT**(**DISTINCT** job\_id)

*-- Selecting data from the employees table*

**FROM** employees;

11.

*-- Converting the first\_name column values to uppercase*

**SELECT** **UPPER**(first\_name)

*-- Selecting data from the employees table*

**FROM** employees;

12.

*-- Extracting the substring of the first three characters from the first\_name column*

**SELECT** **SUBSTRING**(first\_name, 1, 3)

*-- Selecting data from the employees table*

**FROM** employees;

13.

*-- Performing arithmetic operations: multiplication and addition*

**SELECT** 171 \* 214 + 625 **Result**;

14.

*-- Concatenating the first\_name and last\_name columns with a space in between*

**SELECT** CONCAT(first\_name, ' ', last\_name) 'Employee Name'

*-- Selecting data from the employees table*

**FROM** employees;

15.

*-- Removing leading and trailing whitespace characters from the first\_name column*

**SELECT** **TRIM**(first\_name)

*-- Selecting data from the employees table*

**FROM** employees;

16.

*-- Selecting the first\_name and last\_name columns*

**SELECT** first\_name, last\_name,

*-- Calculating the sum of the lengths of first\_name and last\_name columns and aliasing it as 'Length of Names'*

**LENGTH**(first\_name) + **LENGTH**(last\_name) 'Length of Names'

*-- Selecting data from the employees table*

**FROM** employees;

17.

*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the first\_name column contains a digit*

**WHERE** first\_name REGEXP '[0-9]';

18.

*-- Selecting the employee\_id and first\_name columns*

**SELECT** employee\_id, first\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Limiting the result set to only include the first 10 rows*

**LIMIT** 10;

19.

*-- Selecting the first\_name, last\_name, and calculating the monthly salary*

**SELECT** first\_name, last\_name,

*-- Dividing the salary by 12 to calculate the monthly salary and rounding it to 2 decimal places*

ROUND(salary / 12, 2) **AS** 'Monthly Salary'

*-- Selecting data from the employees table*

**FROM** employees;

MySQL Restricting **and** Sorting **data**:

1.

*-- Selecting the first\_name, last\_name, and salary columns*

**SELECT** first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the salary is not between 10000 and 15000*

**WHERE** salary **NOT** **BETWEEN** 10000 **AND** 15000;

2.

*-- Selecting the first\_name, last\_name, and department\_id columns*

**SELECT** first\_name, last\_name, department\_id

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the department\_id is either 30 or 100*

**WHERE** department\_id **IN** (30, 100)

*-- Ordering the result set by the department\_id column in ascending order*

**ORDER** **BY** department\_id **ASC**;

3.

*-- Selecting the first\_name, last\_name, salary, and department\_id columns*

**SELECT** first\_name, last\_name, salary, department\_id

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the salary is not between 10000 and 15000*

*-- and the department\_id is either 30 or 100*

**WHERE** salary **NOT** **BETWEEN** 10000 **AND** 15000

**AND** department\_id **IN** (30, 100);

4.

*-- Selecting the first\_name, last\_name, and hire\_date columns*

**SELECT** first\_name, last\_name, hire\_date

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the year part of the hire\_date is like '1987%'*

**WHERE** **YEAR**(hire\_date) **LIKE** '1987%';

5.

*-- Selecting the first\_name column*

**SELECT** first\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the first\_name column contains both 'b' and 'c'*

**WHERE** first\_name **LIKE** '%b%'

**AND** first\_name **LIKE** '%c%';

6.

*-- Selecting the last\_name, job\_id, and salary columns*

**SELECT** last\_name, job\_id, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the job\_id is either 'IT\_PROG' or 'SH\_CLERK'*

*-- and the salary is not 4500, 10000, or 15000*

**WHERE** job\_id **IN** ('IT\_PROG', 'SH\_CLERK')

**AND** salary **NOT** **IN** (4500, 10000, 15000);

7.

*-- Selecting the last\_name column*

**SELECT** last\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the last\_name column consists of exactly six characters*

**WHERE** last\_name **LIKE** '\_\_\_\_\_\_';

8.

*-- Selecting the last\_name column*

**SELECT** last\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the last\_name column starts with two characters followed by 'e' and any other characters*

**WHERE** last\_name **LIKE** '\_\_e%';

9.

*-- Selecting distinct values from the job\_id column*

**SELECT** **DISTINCT** job\_id

*-- Selecting data from the employees table*

**FROM** employees;

10.

*-- Selecting the first\_name, last\_name, salary columns, and calculating 15% of the salary as PF (Provident Fund)*

**SELECT** first\_name, last\_name, salary, salary \* 0.15 **AS** PF

*-- Selecting data from the employees table*

**FROM** employees;

11.

*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the last\_name column matches one of the specified values*

**WHERE** last\_name **IN** ('JONES', 'BLAKE', 'SCOTT', 'KING', 'FORD');

**Aggregate** Functions **and** **Group** **by** :

1.

*-- Counting the number of distinct job IDs in the employees table*

**SELECT** **COUNT**(**DISTINCT** job\_id)

*-- Selecting data from the employees table*

**FROM** employees;

2.

*-- Calculating the total sum of salaries for all employees*

**SELECT** **SUM**(salary)

*-- Selecting data from the employees table*

**FROM** employees;

3.

*-- Retrieving the minimum salary from the employees table*

**SELECT** **MIN**(salary)

*-- Selecting data from the employees table*

**FROM** employees;

4.

*-- Retrieving the maximum salary among employees with the job\_id 'IT\_PROG'*

**SELECT** **MAX**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees with the job\_id 'IT\_PROG'*

**WHERE** job\_id = 'IT\_PROG';

5.

*-- Calculating the average salary and counting the number of employees in department 90*

**SELECT** **AVG**(salary), **COUNT**(\*)

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees in department 90*

**WHERE** department\_id = 90;

6.

*-- Calculating various statistics (maximum, minimum, sum, average) for the salary column*

**SELECT**

*-- Rounding the maximum salary to 0 decimal places and aliasing it as 'Maximum'*

ROUND(**MAX**(salary), 0) 'Maximum',

*-- Rounding the minimum salary to 0 decimal places and aliasing it as 'Minimum'*

ROUND(**MIN**(salary), 0) 'Minimum',

*-- Rounding the sum of salaries to 0 decimal places and aliasing it as 'Sum'*

ROUND(**SUM**(salary), 0) 'Sum',

*-- Rounding the average salary to 0 decimal places and aliasing it as 'Average'*

ROUND(**AVG**(salary), 0) 'Average'

*-- Selecting data from the employees table*

**FROM** employees;

7.

*-- Counting the number of employees for each job ID*

**SELECT** job\_id, **COUNT**(\*)

*-- Selecting data from the employees table*

**FROM** employees

*-- Grouping the result set by the job\_id column*

**GROUP** **BY** job\_id;

8.

*-- Calculating the difference between the maximum and minimum salaries*

**SELECT** **MAX**(salary) - **MIN**(salary) DIFFERENCE

*-- Selecting data from the employees table*

**FROM** employees;

9.

*-- Selecting the manager\_id and the minimum salary for each manager*

**SELECT** manager\_id, **MIN**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where manager\_id is not NULL*

**WHERE** manager\_id **IS** **NOT** **NULL**

*-- Grouping the result set by manager\_id*

**GROUP** **BY** manager\_id

*-- Sorting the result set by the minimum salary in descending order*

**ORDER** **BY** **MIN**(salary) **DESC**;

10.

*-- Calculating the total salary for each department*

**SELECT** department\_id, **SUM**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Grouping the result set by department\_id*

**GROUP** **BY** department\_id;

11.

*-- Calculating the average salary for each job, excluding 'IT\_PROG'*

**SELECT** job\_id, **AVG**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the job\_id is not 'IT\_PROG'*

**WHERE** job\_id <> 'IT\_PROG'

*-- Grouping the result set by job\_id*

**GROUP** **BY** job\_id;

12.

*-- Calculating various statistics for salaries of employees in department 90, grouped by job\_id*

**SELECT** job\_id, **SUM**(salary), **AVG**(salary), **MAX**(salary), **MIN**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only rows where the department\_id is '90'*

**WHERE** department\_id = '90'

*-- Grouping the result set by job\_id*

**GROUP** **BY** job\_id;

13.

*-- Retrieving the maximum salary for each job title, filtering out job titles where the maximum salary is less than 4000*

**SELECT** job\_id, **MAX**(salary)

*-- Selecting data from the employees table*

**FROM** employees

*-- Grouping the result set by job\_id*

**GROUP** **BY** job\_id

*-- Filtering the result set to include only groups where the maximum salary is greater than or equal to 4000*

**HAVING** **MAX**(salary) >= 4000;

14.

*-- Calculating the average salary and counting the number of employees for each department, filtering out departments with fewer than 10 employees*

**SELECT** department\_id, **AVG**(salary), **COUNT**(\*)

*-- Selecting data from the employees table*

**FROM** employees

*-- Grouping the result set by department\_id*

**GROUP** **BY** department\_id

*-- Filtering the result set to include only groups where the count of employees is greater than 10*

**HAVING** **COUNT**(\*) > 10;

MySQL Subquery :

1.

*-- Selecting the first name, last name, and salary of employees whose salary is higher than that of the employee with the last name 'Bull'*

**SELECT** FIRST\_NAME, LAST\_NAME, SALARY

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is higher than the salary of the employee with the last name 'Bull'*

**WHERE** SALARY >

*-- Subquery to fetch the salary of the employee with the last name 'Bull'*

(**SELECT** salary **FROM** employees **WHERE** last\_name = 'Bull');

2.

*-- Selecting the first name and last name of employees*

**SELECT** first\_name, last\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose department\_id is in the set of department\_ids where the department\_name is 'IT'*

**WHERE** department\_id

**IN** (**SELECT** department\_id **FROM** departments **WHERE** department\_name='IT');

3.

*-- Selecting the first name and last name of employees*

**SELECT** first\_name, last\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose manager\_id is in the set of employee\_ids*

*-- where the department\_id is in the set of department\_ids associated with locations in the US*

**WHERE** manager\_id **in**

(**SELECT** employee\_id

*-- Subquery to select employee\_ids from the employees table*

**FROM** employees

*-- Filtering the employee\_ids to include only those associated with departments*

*-- where the location\_id is in the set of location\_ids associated with countries having country\_id 'US'*

**WHERE** department\_id

**IN**

(**SELECT** department\_id

*-- Subquery to select department\_ids from the departments table*

**FROM** departments

*-- Filtering the department\_ids to include only those associated with locations*

*-- where the country\_id is 'US'*

**WHERE** location\_id

**IN**

(**SELECT** location\_id

*-- Subquery to select location\_ids from the locations table*

**FROM** locations

*-- Filtering the location\_ids to include only those associated with countries*

*-- having country\_id 'US'*

**WHERE** country\_id='US')

)

);

4.

*-- Selecting the first name and last name of employees*

**SELECT** first\_name, last\_name

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose employee\_id is in the set of manager\_ids*

**WHERE** (employee\_id **IN**

*-- Subquery to select manager\_ids from the employees table*

(**SELECT** manager\_id **FROM** employees)

);

5.

*-- Selecting the first name, last name, and salary of employees*

**SELECT** first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is greater than the average salary of all employees*

**WHERE** salary >

*-- Subquery to calculate the average salary from the employees table*

(**SELECT** **AVG**(salary) **FROM** employees);

6.

*-- Selecting the first name, last name, and salary of employees*

**SELECT** first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary matches the minimum salary defined for their job position*

**WHERE** employees.salary =

*-- Subquery to select the minimum salary for each job position from the jobs table*

(**SELECT** min\_salary

**FROM** jobs

*-- Matching the job\_id of each employee with the job\_id in the jobs table*

**WHERE** employees.job\_id = jobs.job\_id);

7.

Selecting the **first** name, **last** name, **and** salary **of** employees

**SELECT** first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees who belong to departments with names starting with 'IT'*

*-- and have a salary higher than the average salary of all employees*

**WHERE** department\_id **IN**

*-- Subquery to select department\_ids from the departments table where the department\_name starts with 'IT'*

(**SELECT** department\_id **FROM** departments **WHERE** department\_name **LIKE** 'IT%')

*-- Additional condition: Salary of employees should be higher than the average salary of all employees*

**AND** salary >

*-- Subquery to calculate the average salary from the employees table*

(**SELECT** **avg**(salary) **FROM** employees);

8.

S*-- Selecting the first name, last name, and salary of employees*

**SELECT** first\_name, last\_name, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is higher than that of the employee with the last name 'Bell'*

**WHERE** salary >

*-- Subquery to fetch the salary of the employee with the last name 'Bell'*

(**SELECT** salary **FROM** employees **WHERE** last\_name = 'Bell')

*-- Sorting the result set in ascending order based on the first name of employees*

**ORDER** **BY** first\_name;

9.

*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is equal to the minimum salary among all employees*

**WHERE** salary =

*-- Subquery to find the minimum salary from the employees table*

(**SELECT** **MIN**(salary) **FROM** employees);

10.

*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is greater than all average salaries within their respective departments*

**WHERE** salary >*-- Selecting all columns from the employees table*

**SELECT** \*

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is greater than all average salaries within their respective departments*

**WHERE** salary >

*-- Subquery to calculate the average salary for each department and compare with each employee's salary*

**ALL**(**SELECT** **avg**(salary) **FROM** employees **GROUP** **BY** department\_id);

11.

*-- Selecting the first name, last name, job ID, and salary of employees*

**SELECT** first\_name, last\_name, job\_id, salary

*-- Selecting data from the employees table*

**FROM** employees

*-- Filtering the result set to include only employees whose salary is greater than all salaries of employees with job\_id 'SH\_CLERK'*

**WHERE** salary >

*-- Subquery to select all salaries of employees with job\_id 'SH\_CLERK' and compare with each employee's salary*

**ALL** (**SELECT** salary **FROM** employees **WHERE** job\_id = 'SH\_CLERK')

*-- Sorting the result set in ascending order based on salary*

**ORDER** **BY** salary;

12.

*-- Selecting the first name and last name of employees who are not managers*

**SELECT** b.first\_name, b.last\_name

*-- Selecting data from the employees table, aliasing it as 'b'*

**FROM** employees b

*-- Filtering the result set to include only employees who are not managers*

**WHERE** **NOT** **EXISTS**

*-- Subquery to check if there is no employee with manager\_id equal to the employee\_id of each employee in the outer query*

(**SELECT** 'X' **FROM** employees a **WHERE** a.manager\_id = b.employee\_id);

13.

*-- Selecting the employee\_id, first name, last name, and department name of employees*

**SELECT** employee\_id, first\_name, last\_name,

*-- Subquery to fetch the department name for each employee's department\_id*

(**SELECT** department\_name **FROM** departments d

*-- Joining the employees table with the departments table based on the department\_id*

**WHERE** e.department\_id = d.department\_id) department

*-- Selecting data from the employees table, aliasing it as 'e'*

**FROM** employees e

*-- Sorting the result set based on the department name*

**ORDER** **BY** department;

14.

*-- Selecting the employee\_id and first name of employees*

**SELECT** employee\_id, first\_name

*-- Selecting data from the employees table, aliasing it as 'A'*

**FROM** employees **AS** A

*-- Filtering the result set to include only employees whose salary is greater than the average salary of their department*

**WHERE** salary >

*-- Subquery to calculate the average salary for each department and compare with each employee's salary*

(**SELECT** **AVG**(salary) **FROM** employees **WHERE** department\_id = A.department\_id);

15.

*-- Setting user-defined variable @i to 0*

**SET** @i = 0;

*-- Selecting the sequential number and employee\_id of every second row in the employees table*

**SELECT** i, employee\_id

*-- Subquery to generate a sequential number for each row in the employees table, starting from 1*

**FROM** (**SELECT** @i := @i + 1 **AS** i, employee\_id **FROM** employees) a

*-- Filtering the result set to include only rows with even sequential numbers*

**WHERE** **MOD**(a.i, 2) = 0;

16.

*-- Selecting distinct salaries from the employees table for which there are exactly 5 distinct salaries greater than or equal to it*

**SELECT** **DISTINCT** salary

*-- Selecting data from the employees table, aliasing it as 'e1'*

**FROM** employees e1

*-- Filtering the result set to include only those salaries for which there are exactly 5 distinct salaries greater than or equal to it*

**WHERE** 5 =

*-- Subquery to count the number of distinct salaries greater than or equal to each salary*

(**SELECT** **COUNT**(**DISTINCT** salary)

*-- Selecting data from the employees table, aliasing it as 'e2'*

**FROM** employees e2

*-- Filtering the result set to include only distinct salaries greater than or equal to the salary in the outer query (e1.salary)*

**WHERE** e2.salary >= e1.salary);

17.

*-- Selecting distinct salaries from the employees table for which there are exactly 4 distinct salaries less than or equal to it*

**SELECT** **DISTINCT** salary

*-- Selecting data from the employees table, aliasing it as 'e1'*

**FROM** employees e1

*-- Filtering the result set to include only those salaries for which there are exactly 4 distinct salaries less than or equal to it*

**WHERE** 4 =

*-- Subquery to count the number of distinct salaries less than or equal to each salary*

(**SELECT** **COUNT**(**DISTINCT** salary)

*-- Selecting data from the employees table, aliasing it as 'e2'*

**FROM** employees e2

*-- Filtering the result set to include only distinct salaries less than or equal to the salary in the outer query (e1.salary)*

**WHERE** e2.salary <= e1.salary);

18.

*-- Selecting all columns from a subset of the employees table, ordered by employee\_id in descending order, and limiting the result to the first 10 rows*

**SELECT** \* **FROM** (

**SELECT** \* **FROM** employees **ORDER** **BY** employee\_id **DESC** **LIMIT** 10

) sub

*-- Sorting the result set from the subset in ascending order based on employee\_id*

**ORDER** **BY** employee\_id **ASC**;

19.

*-- Selecting all columns from the departments table*

**SELECT** \*

*-- Selecting data from the departments table*

**FROM** departments

*-- Filtering the result set to include only departments whose department\_id is not present in the set of department\_ids associated with employees*

**WHERE** department\_id

**NOT** **IN**

*-- Subquery to select department\_ids from the employees table*

(**select** department\_id **FROM** employees);

20.

*-- Selecting distinct salary values from the employees table*

**SELECT** **DISTINCT** salary

*-- Selecting data from the employees table, aliasing it as 'a'*

**FROM** employees a

*-- Filtering the result set to include only salary values where there are at most 3 distinct salary values greater than or equal to it*

**WHERE** 3 >=

*-- Subquery to count the number of distinct salaries greater than or equal to each salary*

(**SELECT** **COUNT**(**DISTINCT** salary)

*-- Selecting data from the employees table, aliasing it as 'b'*

**FROM** employees b

*-- Filtering the result set to include only distinct salaries greater than or equal to the salary in the outer query (a.salary)*

**WHERE** b.salary >= a.salary)

*-- Sorting the result set in descending order based on salary*

**ORDER** **BY** a.salary **DESC**;

21.

*-- Selecting distinct salary values from the employees table*

**SELECT** **DISTINCT** salary

*-- Selecting data from the employees table, aliasing it as 'a'*

**FROM** employees a

*-- Filtering the result set to include only salary values where there are at most 3 distinct salary values less than or equal to it*

**WHERE** 3 >=

*-- Subquery to count the number of distinct salaries less than or equal to each salary*

(**SELECT** **COUNT**(**DISTINCT** salary)

*-- Selecting data from the employees table, aliasing it as 'b'*

**FROM** employees b

*-- Filtering the result set to include only distinct salaries less than or equal to the salary in the outer query (a.salary)*

**WHERE** b.salary <= a.salary)

*-- Sorting the result set in descending order based on salary*

**ORDER** **BY** a.salary **DESC**;

22.

*-- Selecting all columns from the employees table, aliasing it as 'emp1'*

**SELECT** \*

*-- Selecting data from the employees table, aliasing it as 'emp1'*

**FROM** employees emp1

*-- Filtering the result set to include only employees where the count of distinct salaries greater than the salary of the current employee is 1*

**WHERE** (1) = (

*-- Subquery to count the number of distinct salaries greater than the salary of each employee*

**SELECT** **COUNT**(**DISTINCT**(emp2.salary))

*-- Selecting data from the employees table, aliasing it as 'emp2'*

**FROM** employees emp2

*-- Filtering the result set to include only distinct salaries greater than the salary of the employee in the outer query (emp1.salary)*

**WHERE** emp2.salary > emp1.salary

);

MySQL Joins :

1.

*-- This SQL query selects specific columns from the 'locations' table after performing a natural join with the 'countries' table.*

**SELECT**

location\_id, *-- Selecting the 'location\_id' column from the result set.*

street\_address, *-- Selecting the 'street\_address' column from the result set.*

city, *-- Selecting the 'city' column from the result set.*

state\_province, *-- Selecting the 'state\_province' column from the result set.*

country\_name *-- Selecting the 'country\_name' column from the result set.*

**FROM**

locations *-- Specifying the 'locations' table.*

**NATURAL** **JOIN**

countries; *-- Performing a natural join with the 'countries' table based on any common columns.*

2.

*-- This SQL query selects specific columns from the 'employees' table after performing an inner join with the 'departments' table using the 'department\_id' column.*

**SELECT**

first\_name, *-- Selecting the 'first\_name' column from the result set.*

last\_name, *-- Selecting the 'last\_name' column from the result set.*

department\_id, *-- Selecting the 'department\_id' column from the result set.*

department\_name *-- Selecting the 'department\_name' column from the result set.*

**FROM**

employees *-- Specifying the 'employees' table.*

**JOIN**

departments *-- Specifying the 'departments' table.*

**USING**

(department\_id); *-- Performing an inner join using the 'department\_id' column, which is common in both tables.*

3.

*-- This SQL query selects specific columns from the 'employees' and 'departments' tables, as well as the 'locations' table, to retrieve information about employees in the London city.*

**SELECT**

e.first\_name, *-- Selecting the 'first\_name' column from the 'employees' table and aliasing it as 'e'.*

e.last\_name, *-- Selecting the 'last\_name' column from the 'employees' table and aliasing it as 'e'.*

e.job\_id, *-- Selecting the 'job\_id' column from the 'employees' table and aliasing it as 'e'.*

e.department\_id, *-- Selecting the 'department\_id' column from the 'employees' table and aliasing it as 'e'.*

d.department\_name *-- Selecting the 'department\_name' column from the 'departments' table and aliasing it as 'd'.*

**FROM**

employees e *-- Specifying the 'employees' table and aliasing it as 'e'.*

**JOIN**

departments d *-- Specifying the 'departments' table and aliasing it as 'd'.*

**ON**

(e.department\_id = d.department\_id) *-- Performing a join between 'employees' and 'departments' based on the 'department\_id' column.*

**JOIN**

locations l **ON** *-- Joining the 'locations' table and aliasing it as 'l'.*

(d.location\_id = l.location\_id) *-- Performing a join between 'departments' and 'locations' based on the 'location\_id' column.*

**WHERE**

**LOWER**(l.city) = 'London'; *-- Filtering the result to only include rows where the city in lowercase is 'London'.*

4.

*-- This SQL query selects specific columns from the 'employees' table, twice aliased as 'e' and 'm', to retrieve information about employees and their managers.*

**SELECT**

e.employee\_id 'Emp\_Id', *-- Selecting the 'employee\_id' column from the 'employees' table, aliased as 'e', and renaming it as 'Emp\_Id'.*

e.last\_name 'Employee', *-- Selecting the 'last\_name' column from the 'employees' table, aliased as 'e', and renaming it as 'Employee'.*

m.employee\_id 'Mgr\_Id', *-- Selecting the 'employee\_id' column from the 'employees' table, aliased as 'm', and renaming it as 'Mgr\_Id'.*

m.last\_name 'Manager' *-- Selecting the 'last\_name' column from the 'employees' table, aliased as 'm', and renaming it as 'Manager'.*

**FROM**

employees e *-- Specifying the 'employees' table and aliasing it as 'e'.*

**JOIN**

employees m *-- Joining the 'employees' table again and aliasing it as 'm'.*

**ON**

(e.manager\_id = m.employee\_id); *-- Performing a join between 'employees' and itself based on the 'manager\_id' column to associate employees with their managers.*

5.

*-- This SQL query selects specific columns from the 'employees' table to retrieve information about employees hired after an employee with the last name 'Jones'.*

**SELECT**

e.first\_name, *-- Selecting the 'first\_name' column from the 'employees' table.*

e.last\_name, *-- Selecting the 'last\_name' column from the 'employees' table.*

e.hire\_date *-- Selecting the 'hire\_date' column from the 'employees' table.*

**FROM**

employees e *-- Specifying the 'employees' table and aliasing it as 'e'.*

**JOIN**

employees davies *-- Joining the 'employees' table again and aliasing it as 'davies'.*

**ON**

(davies.last\_name = 'Jones') *-- Performing a join based on the condition where the last name in 'davies' is 'Jones'.*

**WHERE**

davies.hire\_date < e.hire\_date; *-- Filtering the result to include only employees hired after the employee with the last name 'Jones'.*

6.

*-- This SQL query retrieves the count of employees in each department, along with the department names, from the 'departments' and 'employees' tables.*

**SELECT**

department\_name **AS** 'Department Name', *-- Selecting the 'department\_name' column from the 'departments' table and aliasing it as 'Department Name'.*

**COUNT**(\*) **AS** 'No of Employees' *-- Counting the number of records (employees) in each department and aliasing it as 'No of Employees'.*

**FROM**

departments *-- Specifying the 'departments' table.*

**INNER** **JOIN**

employees *-- Performing an inner join with the 'employees' table.*

**ON**

employees.department\_id = departments.department\_id *-- Joining the 'employees' and 'departments' tables based on the 'department\_id' column.*

**GROUP** **BY**

departments.department\_id, department\_name *-- Grouping the result set by department ID and department name.*

**ORDER** **BY**

department\_name; *-- Ordering the result set by department name in ascending order.*

7.

*-- This SQL query retrieves specific columns from the 'job\_history' table and calculates the duration of each job in days for employees in the specified department.*

**SELECT**

employee\_id, *-- Selecting the 'employee\_id' column from the result set.*

job\_title, *-- Selecting the 'job\_title' column from the result set.*

end\_date - start\_date **AS** Days *-- Calculating the difference between 'end\_date' and 'start\_date' columns and aliasing it as 'Days'.*

**FROM**

job\_history *-- Specifying the 'job\_history' table.*

**NATURAL** **JOIN**

jobs *-- Performing a natural join with the 'jobs' table.*

**WHERE**

department\_id = 90; *-- Filtering the result to include only records where the department ID is 90.*

8.

*-- This SQL query retrieves specific columns from the 'departments' and 'employees' tables to get information about department managers.*

**SELECT**

d.department\_id, *-- Selecting the 'department\_id' column from the 'departments' table.*

d.department\_name, *-- Selecting the 'department\_name' column from the 'departments' table.*

d.manager\_id, *-- Selecting the 'manager\_id' column from the 'departments' table.*

e.first\_name *-- Selecting the 'first\_name' column from the 'employees' table and aliasing it as 'e'.*

**FROM**

departments d *-- Specifying the 'departments' table and aliasing it as 'd'.*

**INNER** **JOIN**

employees e *-- Performing an inner join with the 'employees' table and aliasing it as 'e'.*

**ON**

(d.manager\_id = e.employee\_id); *-- Joining the 'departments' and 'employees' tables based on the 'manager\_id' column to associate departments with their managers.*

9.

*-- This SQL query retrieves specific columns from the 'departments', 'employees', and 'locations' tables to get information about department managers and their corresponding locations.*

**SELECT**

d.department\_name, *-- Selecting the 'department\_name' column from the 'departments' table.*

e.first\_name, *-- Selecting the 'first\_name' column from the 'employees' table and aliasing it as 'e'.*

l.city *-- Selecting the 'city' column from the 'locations' table and aliasing it as 'l'.*

**FROM**

departments d *-- Specifying the 'departments' table and aliasing it as 'd'.*

**JOIN**

employees e *-- Performing a join with the 'employees' table and aliasing it as 'e'.*

**ON**

(d.manager\_id = e.employee\_id) *-- Joining the 'departments' and 'employees' tables based on the 'manager\_id' column to associate departments with their managers.*

**JOIN**

locations l **USING** (location\_id); *-- Performing a join with the 'locations' table based on the 'location\_id' column.*

10.

*-- This SQL query calculates the average salary for each job title by joining the 'employees' and 'jobs' tables.*

**SELECT**

job\_title, *-- Selecting the 'job\_title' column from the result set.*

**AVG**(salary) *-- Calculating the average salary and selecting it from the result set.*

**FROM**

employees *-- Specifying the 'employees' table.*

**NATURAL** **JOIN**

jobs *-- Performing a natural join with the 'jobs' table.*

**GROUP** **BY**

job\_title; *-- Grouping the result set by job title to calculate the average salary for each job.*

11.

*-- This SQL query selects specific columns from the 'employees' and 'jobs' tables and calculates the difference between each employee's salary and the minimum salary for their job title.*

**SELECT**

job\_title, *-- Selecting the 'job\_title' column from the result set.*

first\_name, *-- Selecting the 'first\_name' column from the result set.*

salary - min\_salary **AS** 'Salary - Min\_Salary' *-- Calculating the difference between the salary and the minimum salary for each job title and aliasing it as 'Salary - Min\_Salary'.*

**FROM**

employees *-- Specifying the 'employees' table.*

**NATURAL** **JOIN**

jobs; *-- Performing a natural join with the 'jobs' table.*

12.

*-- This SQL query retrieves all columns from the 'job\_history' table for employees whose salary is greater than 10000.*

**SELECT**

jh.\* *-- Selecting all columns from the 'job\_history' table.*

**FROM**

job\_history jh *-- Specifying the 'job\_history' table and aliasing it as 'jh'.*

**JOIN**

employees e *-- Performing a join with the 'employees' table and aliasing it as 'e'.*

**ON**

(jh.employee\_id = e.employee\_id) *-- Joining the 'job\_history' and 'employees' tables based on the 'employee\_id' column to associate job history with employees.*

**WHERE**

salary > 10000; *-- Filtering the result to include only records where the salary is greater than 10000.*

13.

*-- This SQL query selects specific columns from the 'departments' and 'employees' tables to retrieve information about department managers with more than 15 years of experience.*

**SELECT**

first\_name, *-- Selecting the 'first\_name' column from the result set.*

last\_name, *-- Selecting the 'last\_name' column from the result set.*

hire\_date, *-- Selecting the 'hire\_date' column from the result set.*

salary, *-- Selecting the 'salary' column from the result set.*

(DATEDIFF(now(), hire\_date))/365 Experience *-- Calculating the experience in years and aliasing it as 'Experience'.*

**FROM**

departments d *-- Specifying the 'departments' table and aliasing it as 'd'.*

**JOIN**

employees e *-- Joining the 'employees' table and aliasing it as 'e'.*

**ON**

(d.manager\_id = e.employee\_id) *-- Joining the 'departments' and 'employees' tables based on the 'manager\_id' column to associate department managers with their departments.*

**WHERE**

(DATEDIFF(now(), hire\_date))/365 > 15; *-- Filtering the result to include only records where the experience in years is greater than 15.*

Date **and** **Time** functions :

1.

*-- This SQL query calculates a date that is three months prior to the current date.*

**SELECT**

date(((PERIOD\_ADD *-- Calculates a period by adding a specified number of months to a given period.*

(**EXTRACT**(YEAR\_MONTH *-- Extracts the year and month from the current date (CURDATE()).*

**FROM** CURDATE()) *-- Specifies the current date.*

,-3)\*100)+1)); *-- Subtracts three months from the current date, then multiplies by 100 to convert it to a period, and finally adds 1 to convert it back to a date.*

2.

*-- This SQL query calculates the last day of the current month.*

**SELECT**

(SUBDATE(ADDDATE *-- Calculates the date obtained by adding a specified interval (1 month) to the current date.*

(CURDATE(),INTERVAL 1 **MONTH**), *-- Adds 1 month to the current date (CURDATE()).*

INTERVAL DAYOFMONTH(CURDATE()) **DAY**)) *-- Subtracts the number of days from the current date to get the last day of the current month.*

**AS** LastDayOfTheMonth; *-- Alias for the calculated last day of the month.*

3.

*-- This SQL query retrieves distinct dates representing the first day of the week for each hire date in the 'employees' table.*

**SELECT** **DISTINCT**( *-- Selects distinct values of the result set.*

STR\_TO\_DATE( *-- Converts a string into a date value.*

CONCAT( *-- Concatenates multiple strings into one string.*

YEARWEEK(hire\_date), *-- Gets the year and week number for each hire date.*

'1' *-- Appends '1' to the end of the concatenated string.*

),

'%x%v%w' *-- Specifies the format of the input string.*

)

)

**FROM**

employees; *-- Specifies the 'employees' table.*

4.

*-- This SQL query creates a date using the year extracted from the current date and the day '1'.*

**SELECT**

MAKEDATE( *-- Creates a date using the specified year and day.*

**EXTRACT**(**YEAR** **FROM** CURDATE()), *-- Extracts the year from the current date (CURDATE()).*

1 *-- Specifies the day as '1'.*

);

5.

*-- This SQL query creates a date using the year extracted from the current date, the month '12', and the day '31'.*

**SELECT**

STR\_TO\_DATE( *-- Converts a string into a date value.*

CONCAT( *-- Concatenates multiple strings into one string.*

12, *-- Specifies the month as '12'.*

31, *-- Specifies the day as '31'.*

**EXTRACT**(**YEAR** **FROM** CURDATE()) *-- Extracts the year from the current date (CURDATE()).*

),

'%m%d%Y' *-- Specifies the format of the input string.*

);

6.

*-- This SQL query calculates the age based on the difference between the current year and the year of birth.*

**SELECT**

**YEAR**(**CURRENT\_TIMESTAMP**) - *-- Calculates the current year.*

**YEAR**("1967-06-08") - *-- Calculates the year from the given birthdate.*

(**RIGHT**(**CURRENT\_TIMESTAMP**, 5) < *-- Checks if the month and day of the current date are before the month and day of the birthdate.*

**RIGHT**("1967-06-08", 5)) *-- Extracts the month and day from the birthdate.*

**as** age; *-- Alias for the calculated age.*

7.

*-- This SQL query formats the current date in a specified format and aliases it as 'Current\_date'.*

**SELECT**

DATE\_FORMAT( *-- Formats a date value according to the specified format.*

CURDATE(), *-- Retrieves the current date using the CURDATE() function.*

'%M %e, %Y' *-- Specifies the desired format for the date: '%M' for full month name, '%e' for day of the month without leading zeros, and '%Y' for four-digit year.*

)

**AS** 'Current\_date'; *-- Alias for the formatted current date.*

8.

*-- This SQL query formats the current date and time in a specified format.*

**SELECT**

DATE\_FORMAT( *-- Formats a date and time value according to the specified format.*

NOW(), *-- Retrieves the current date and time using the NOW() function.*

''%W %M %Y' -- Specifies the desired format for the date and time: '%W' for the full weekday name, '%M' for the full month name, and '%Y' for the four-digit year.

);

9.

-- This SQL query extracts the year component from the current date and time.

SELECT

EXTRACT(YEAR FROM NOW());

10.

-- This SQL query converts a number representing the number of days since year 0 to a date value.

SELECT

FROM\_DAYS(730677);

11.

-- This SQL query retrieves the first name and hire date of employees hired within a specific date range.

SELECT

FIRST\_NAME, -- Selecting the 'FIRST\_NAME' column from the 'employees' table.

HIRE\_DATE -- Selecting the 'HIRE\_DATE' column from the 'employees' table.

FROM

employees -- Specifying the 'employees' table.

WHERE

HIRE\_DATE -- Filtering the rows based on the hire date being within a specific date range.

BETWEEN '1987-06-01 00:00:00' -- Specifies the start of the date range.

AND '1987-07-30 23:59:59'; -- Specifies the end of the date range.

12.

-- This SQL query formats the current date and time in a specific format.

SELECT

date\_format( -- Formats a date and time value according to the specified format.

CURDATE(), -- Retrieves the current date using the CURDATE() function.

'%W %D %M %Y %T' -- Specifies the desired format for the date and time.

);

13.

-- This SQL query formats the current date in a specific date format.

SELECT

date\_format( -- Formats a date value according to the specified format.

CURDATE(), -- Retrieves the current date using the CURDATE() function.

'%d/%m/%Y' -- Specifies the desired format for the date.

);

14.

-- This SQL query formats the current date and time in a specific format.

SELECT

date\_format( -- Formats a date and time value according to the specified format.

CURDATE(), -- Retrieves the current date using the CURDATE() function.

'%l:%i %p %b %e, %Y' -- Specifies the desired format for the date and time.

);

15.

-- This SQL query retrieves the first name and last name of employees hired in June.

SELECT

first\_name, -- Selecting the 'first\_name' column from the 'employees' table.

last\_name -- Selecting the 'last\_name' column from the 'employees' table.

FROM

employees -- Specifying the 'employees' table.

WHERE

MONTH(HIRE\_DATE) = 6; -- Filtering the rows to include only those where the hire date month is June.

16.

-- This SQL query retrieves the year part of the hire date for employees and groups them by year, filtering only those years where the count of employees hired is greater than 10.

SELECT

DATE\_FORMAT(HIRE\_DATE,'%Y') -- Formats the hire date to extract the year part and returns it as 'YYYY'.

FROM

employees -- Specifies the 'employees' table.

GROUP BY

DATE\_FORMAT(HIRE\_DATE,'%Y') -- Groups the result set by the year part of the hire date.

HAVING

COUNT(EMPLOYEE\_ID) > 10; -- Filters the grouped results to include only those years where the count of employees is greater than 10.

17.

-- This SQL query retrieves the first name and hire date of employees hired in the year 1987.

SELECT

FIRST\_NAME, -- Selecting the 'FIRST\_NAME' column from the 'employees' table.

HIRE\_DATE -- Selecting the 'HIRE\_DATE' column from the 'employees' table.

FROM

employees -- Specifying the 'employees' table.

WHERE

YEAR(HIRE\_DATE)=1987; -- Filtering the rows to include only those where the hire date year is 1987.

18.

-- This SQL query retrieves the department name, first name, and salary of employees who have been working for more than 5 years and are also managers of their respective departments.

SELECT

DEPARTMENT\_NAME, -- Selecting the 'DEPARTMENT\_NAME' column from the 'departments' table.

FIRST\_NAME, -- Selecting the 'FIRST\_NAME' column from the 'employees' table.

SALARY -- Selecting the 'SALARY' column from the 'employees' table.

FROM

departments D -- Specifying the 'departments' table and aliasing it as 'D'.

JOIN

employees E -- Joining the 'employees' table and aliasing it as 'E'.

ON

(D.MANAGER\_ID=E.MANAGER\_ID) -- Joining the 'departments' and 'employees' tables based on the manager ID to associate managers with their departments.

WHERE

(SYSDATE()-HIRE\_DATE) / 365 > 5; -- Filtering the result to include only records where the duration of employment is more than 5 years.

19.

-- This SQL query retrieves the employee ID, last name, hire date, and the last day of the month for each hire date.

SELECT

employee\_id, -- Selecting the 'employee\_id' column from the 'employees' table.

last\_name, -- Selecting the 'last\_name' column from the 'employees' table.

hire\_date, -- Selecting the 'hire\_date' column from the 'employees' table.

LAST\_DAY(hire\_date) -- Using the LAST\_DAY() function to calculate the last day of the month for each hire date.

FROM

employees; -- Specifying the 'employees' table.

20.

-- This SQL query retrieves the first name of employees, the current date, their hire date, and calculates their years of employment.

SELECT

FIRST\_NAME, -- Selecting the 'FIRST\_NAME' column from the 'employees' table.

SYSDATE(), -- Retrieving the current date using the SYSDATE() function.

HIRE\_DATE, -- Selecting the 'HIRE\_DATE' column from the 'employees' table.

DATEDIFF(SYSDATE(), hire\_date)/365 -- Calculating the difference in days between the current date and the hire date, then dividing by 365 to get years of employment.

FROM

employees; -- Specifying the 'employees' table.

21.

-- This SQL query calculates the count of employees hired in each department for each year, ordered by department ID.

SELECT

DEPARTMENT\_ID, -- Selecting the 'DEPARTMENT\_ID' column from the 'employees' table.

DATE\_FORMAT(HIRE\_DATE,'%Y'), -- Formatting the hire date to extract the year and selecting it as '**YEAR**'.

COUNT(EMPLOYEE\_ID) -- Counting the number of employees in each department for each year.

FROM

employees -- Specifying the 'employees' table.

GROUP BY

DEPARTMENT\_ID, DATE\_FORMAT(HIRE\_DATE, '%Y') -- Grouping the result set by department ID and year of hire date.

ORDER BY

DEPARTMENT\_ID; -- Ordering the result set by department ID.

MySQL String :

1.

-- This SQL query selects the job\_id and concatenates the employee\_id values separated by a space for each group of job\_id.

SELECT

job\_id,

GROUP\_CONCAT(employee\_id, ' ') AS 'Employees ID' -- Concatenates employee\_ids with a space separator and renames the resulting column as 'Employees ID'

FROM

employees

GROUP BY

job\_id; -- Groups the result by job\_id, so each row represents a unique job\_id with concatenated employee\_ids.

2.

-- This SQL statement updates the 'phone\_number' column in the 'employees' table.

-- It replaces any occurrence of the substring '124' in the 'phone\_number' column with '999'.

-- The WHERE clause ensures that only rows with phone numbers containing '124' are updated.

UPDATE employees

SET phone\_number = REPLACE(phone\_number, '124', '999')

WHERE phone\_number LIKE '%124%';

3.

-- This SQL query selects all columns from the 'employees' table where the length of the first name is greater than or equal to 8.

SELECT

\* -- Selecting all columns from the 'employees' table.

FROM

employees -- Specifying the 'employees' table.

WHERE

LENGTH(first\_name) >= 8; -- Filtering the rows to include only those where the length of the first name is greater than or equal to 8.

4.

-- This SQL query selects the job\_id, maximum salary, and minimum salary from the jobs table.

-- The LPAD function is used to left-pad the salary values with zeros to ensure they have a total width of 7 characters.

SELECT

job\_id,

LPAD(max\_salary, 7, '0') AS ' **Max** Salary', -- Left-pads the max\_salary column with zeros to ensure a width of 7 characters, and renames the resulting column as '**Max** Salary'.

LPAD(min\_salary, 7, '0') AS ' **Min** Salary' -- Left-pads the min\_salary column with zeros to ensure a width of 7 characters, and renames the resulting column as '**Min** Salary'.

FROM

jobs;

5.

-- Updating the email addresses of employees

UPDATE employees

-- Setting the email column to a concatenation of the existing email value and '@example.com'

SET email = CONCAT(email, '@example.com');

6.

-- Selecting the employee\_id, first\_name, and hire month from the employees table

SELECT employee\_id, first\_name,

-- Extracting the month portion of the hire\_date using the MID function, starting from the 6th character and taking 2 characters

MID(hire\_date, 6, 2) as hire\_month

-- Selecting data from the employees table

FROM employees;

7.

-- This SQL query selects the employee ID and extracts a portion of the reversed email address to create a new column called Email\_ID.

SELECT

employee\_id, -- Selecting the employee ID from the employees table.

-- Reverses the email address, extracts a substring starting from the fourth character, and then reverses it back.

REVERSE(SUBSTR(REVERSE(email), 4)) as Email\_ID

FROM

employees; -- Specifies the table from which data is being retrieved, in this case, it's the 'employees' **table**.

8.

*-- This SQL query selects all columns from the employees table where the first name is in uppercase.*

**SELECT**

\* *-- Selecting all columns from the employees table.*

**FROM**

employees *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

**WHERE**

first\_name = BINARY **UPPER**(first\_name);

9.

*-- This SQL query selects the last four digits of the phone numbers stored in the 'phone\_number' column and aliases the result as 'Ph.No.'.*

**SELECT**

**RIGHT**(phone\_number, 4) **as** 'Ph.No.' *-- Selecting the last four digits of the phone numbers and aliasing the result as 'Ph.No.'.*

**FROM**

employees; *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

10.

*-- This SQL query selects the location ID, street address, and extracts the last word from the street address.*

**SELECT**

location\_id, *-- Selecting the location ID from the locations table.*

street\_address, *-- Selecting the street address from the locations table.*

*-- Extracting the last word from the street address by replacing punctuation marks with spaces,*

*-- splitting the address into words, and then selecting the last word.*

SUBSTRING\_INDEX(

**REPLACE**(

**REPLACE**(

**REPLACE**(street\_address,',',' '),

')',' '),

'(',' '),

' ',-1) **AS** 'Last--word-of-street\_address'

**FROM**

locations; *-- Specifies the table from which data is being retrieved, in this case, it's the 'locations' table.*

11.

*-- This SQL query selects all columns from the locations table where the length of the street address is less than or equal to the minimum length of all street addresses in the locations table.*

**SELECT**

\* *-- Selecting all columns from the locations table.*

**FROM**

locations *-- Specifies the table from which data is being retrieved, in this case, it's the 'locations' table.*

**WHERE**

**LENGTH**(street\_address) <= ( *-- Filters the rows where the length of the street address is less than or equal to...*

**SELECT**

**MIN**(**LENGTH**(street\_address)) *-- ... the minimum length of all street addresses in the locations table.*

**FROM**

locations

);

12.

*-- This SQL query selects a portion of the job title from the jobs table.*

**SELECT**

job\_title, *-- Selecting the job title from the jobs table.*

*-- Extracting the substring from the job title starting from the first character up to the position of the first space.*

SUBSTR(job\_title, 1, INSTR(job\_title, ' ') - 1)

**FROM**

jobs; *-- Specifies the table from which data is being retrieved, in this case, it's the 'jobs' table.*

13.

*-- This SQL query selects the first name and last name of employees whose last name contains the letter 'C' after the second position.*

**SELECT**

first\_name, last\_name *-- Selecting the first name and last name from the employees table.*

**FROM**

employees *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

**WHERE**

INSTR(last\_name,'C') > 2; *-- Filters the rows where the letter 'C' is found in the last name after the second position.*

14.

*-- This SQL query selects the first name and its length from the employees table for names starting with 'J', 'M', or 'A', and sorts the result by first name.*

**SELECT**

first\_name "Name", *-- Selecting the first name and aliasing it as "Name".*

**LENGTH**(first\_name) "Length" *-- Calculating the length of the first name and aliasing it as "Length".*

**FROM**

employees *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

**WHERE**

first\_name **LIKE** 'J%' *-- Filters the rows where the first name starts with 'J'.*

**OR** first\_name **LIKE** 'M%' *-- Filters the rows where the first name starts with 'M'.*

**OR** first\_name **LIKE** 'A%' *-- Filters the rows where the first name starts with 'A'.*

**ORDER** **BY**

first\_name; *-- Orders the result set by the first name in ascending order.*

15.

*-- This SQL query selects the first name of employees and left-pads their salary values with '$' characters up to a total width of 10 characters.*

**SELECT**

first\_name, *-- Selecting the first name from the employees table.*

*-- Left-pads the salary values with '$' characters up to a total width of 10 characters.*

LPAD(salary, 10, '$') SALARY

**FROM**

employees; *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

16.

*-- This SQL query selects the first 8 characters of the first name, repeats '$' characters based on the salary divided by 1000, and retrieves the original salary from the employees table, ordering the result by salary in descending order.*

**SELECT**

**LEFT**(first\_name, 8), *-- Selecting the first 8 characters of the first name.*

*-- Repeats '$' characters a number of times based on the salary divided by 1000.*

REPEAT('$', FLOOR(salary/1000)) 'SALARY($)',

salary *-- Selecting the original salary from the employees table.*

**FROM**

employees *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

**ORDER** **BY**

salary **DESC**; *-- Orders the result set by salary in descending order.*

17.

*-- This SQL query selects the employee ID, first name, last name, and hire date from the employees table*

*-- where the hire date contains the month and day "07" in the specified format, '%d %m %Y'.*

**SELECT**

employee\_id, *-- Selecting the employee ID from the employees table.*

first\_name, *-- Selecting the first name from the employees table.*

last\_name, *-- Selecting the last name from the employees table.*

hire\_date *-- Selecting the hire date from the employees table.*

**FROM**

employees *-- Specifies the table from which data is being retrieved, in this case, it's the 'employees' table.*

**WHERE**

**POSITION**("07" **IN** DATE\_FORMAT(hire\_date, '%d %m %Y')) > 0;

*-- Checks if the position of "07" exists in the formatted hire date string,*

*-- indicating that the month and day of hiring are "07".*