Database , Tables , Constraints

1. Create new database & employee table (based on give sample data)

create employee table with primary key (EmployeeID)

Sql>create database assignment1;

use assignment1;

create table employee(

emp_id int(3) primary key,

first_name varchar(20),

last_name varchar(20),

salary decimal,

joining_date date,

department varchar(20),

gender varchar(20),

job_title varchar(20)

);

Sql> select * from employee;

3. Write a query to create a clone of an existing table using Create Command.

Sql> create table employee_clone as select * from employee;

4. Write a query to get all employee detail from "employee" table

Sql> select * from employee;

5. Select only top 1 record from employee table

Sql> select * from employee limit 1;

6. Select only bottom 1 record from employee table

select * from employee order by emp_id desc limit 1;

7. How to select a random record from a table?

Sql> Select * from employee ORDER BY RAND() LIMIT 1;

8. Write a query to get

“first_name” in upper case as "first_name_upper"

Sql> Select upper(first_name) as first_name_upper from employee;

‘first_name’ in lower case as ‘first_name_lower”

Sql> Select lower(first_name) as first_name_lower from employee;

Create a new column “full_name” by combining “first_name” &

“last_name” with space as a separator.

```
Sql> Select   first_name,
                last_name,
                concat(first_name, ' ', last_name) as full_name
               from employee;
```

Add 'Hello ' to first_name and display result

Sql> select concat('hello ',first_name) from employee;

9.Select the employee details of

Whose “first_name” is ‘Malli’

```
Sql> select * from employee
 where first_name='malli';
```

Whose “first_name” present in ("Malli","Meena", "Anjali")

```
Sql> select * from employee
where first_name in ('malli','meena','anjali');
```

Whose “first_name” not present in ("Malli","Meena", "Anjali")

```
Sql> select * from employee
 where first_name  not  in ('malli','meena','anjali');
```

Whose “first_name” starts with “v”

```
Sql> select * from employee
where first_name like 'v%';
```

Whose “first_name” ends with “i”

Sql> select * from employee where first_name like '%i';

Whose “first_name” contains “o”

Sql> select * from employee where first_name like '%o%';

Whose "first_name" start with any single character between 'm-v'

Whose "first_name" not start with any single character between 'm-v'

```
Sql> select  *from  employee
where first_name NOT LIKE 'm%'
```

AND first_name NOT LIKE 'n%'

AND first_name NOT LIKE 'o%'

AND first_name NOT LIKE 'p%'

AND first_name NOT LIKE 'q%'

AND first_name NOT LIKE 'r%'

AND first_name NOT LIKE 's%'

AND first_name NOT LIKE 't%'

AND first_name NOT LIKE 'u%'

AND first_name NOT LIKE 'v%';

Whose "first_name" start with 'M' and contain 5 letters

Sql>select * from employee where first_name like 'm____';

10. Write a query to get all unique values of"department" from the employee table.

Sql>select distinct(department) from employee;

11. Query to check the total records present in a table.

select count(*) from employee;

12. Write down the query to print first letter of a Name in Upper Case and all other letter in Lower Case.(EmployDetail table)

Sql>select concat(substring(first_name,1,1),substring(first_name,2)) from employee;

13. Write down the query to display all employee name in one cell separated by ',' ex:- "Vikas, nikita, Ashish, Nikhil , anish"(EmployDetail table).

Sql>select group_concat(first_name , ', ') from employee;

14. Query to get the below values of "salary" from employee table

Highest salary

Average salary

Highest salary - Lowest salary as diff_salary

% of difference between Highest salary and lowest salary. (sample output format: 10.5%)

Sql> select min(salary) as min_salary,

max(salary) as max_salary ,

round(avg(salary),2) as avg_salary,

```
 abs(min(salary)-max(salary)) as sallary_diff,
concat(round(((max(salary)-min(salary))/min(salary))*100,1),'%')
 from employee;
```

15. Select “first_name” from the employee table after removing white spaces from

Right side spaces

Left side spaces

Both right & left side spaces

```
Sql> select rtrim(first_name) as right_trim,
 ltrim(first_name) as left_trim ,
 trim(first_name) as full_trim
from employee;
```

16. Query to check no.of records present in a table where employees having 50k salary.

```
select count(*) from employee  where salary>=50000;
```

17. Find the most recently hired employee in each department.

```
Sql> select * from employee
 where joining_date in
 (select max(joining_date) from employee group by department) ;
```

Case When Then End Statement Queries

1.Display first_name and gender as M/F.(if male then M, if Female then F)

```
Sql> select *, case
when gender ='Male' then 'M'
else 'f'
end as gender  from employee;
```

2.Display first_name, salary, and a salary category. (If salary is below 50,000, categorize as 'Low'; between 50,000 and 60,000 as 'Medium';

above 60,000 as 'High')

```
Sql> select first_name, salary ,
case
when salary<50000 then 'low'
when salary between 500000 and 60000 then 'medium'
```

else 'high'

end as salary_category

from employee;

3.Display first_name, department, and a department classification. (If department is 'IT', display 'Technical'; if 'HR', display 'Human Resources'; if 'Finance', display 'Accounting'; otherwise, display 'Other')

Sql>select first_name, department,

case

when department='it' then 'Technical'

when department="hr" then 'human resuorses'

when department='finance' then 'accounting'

else 'others'

end as department_classification

from employee;

4.Display first_name, salary, and eligibility for a salary raise. (If salary is less than 50,000, mark as 'Eligible for Raise'; otherwise, 'Not Eligible')

Sql>select first_name, salary , case

when salary<50000 then 'eligible for raise'

else 'not eligible'

end as eligibility_for_salary_raise

from employee;

5.Display first_name, joining_date, and employment status. (If joining date is before '2022-01-01', mark as 'Experienced'; otherwise, 'New Hire')

Sql>select first_name, joining_date,

case

when joining_date<'2022-01-01' then 'experinced'

else 'new hire'

end as status

from employee;

6.Display first_name, salary, and bonus amount. (If salary is above 60,000, add10% bonus; if between 50,000 and 60,000, add 7%; otherwise, 5%)

```
Sql> select first_name, salary,
case
when salary>60000 then salary+((10/100)*salary)
when salary between 50000 and 60000 then salary+((7/100)*salary)
else salary+((5/100)*salary)
end as bonus_amount
 from employee;
```

7.Display first_name, salary, and seniority level. (If salary is greater than 60,000, classify as 'Senior'; between 50,000 and 60,000 as 'Mid-Level'; below 50,000 as 'Junior')

```
Sql> select first_name, salary ,
case
when salary>60000 then 'senior'
when salary between 50000 and 60000 then 'mid-level'
else 'junior'
end as seniority_level
from employee;
```

8.Display first_name, department, and job level for IT employees. (If department is 'IT' and salary is greater than 55,000, mark as 'Senior IT Employee'; otherwise, 'Other').

```
Sql>select first_name, department,
case
when department='it' and salary>=55000 then 'senior-it employee'
when department='it' and salary<55000 then 'others'
end as level_it_employees
from employee;
```

9.Display first_name, joining_date, and recent joiner status. (If an employee joined

after '2024-01-01', label as 'Recent Joiner'; otherwise, 'Long-Term Employee')

```
Sql>select first_name, joining_date,
case
when joining_date>='2024-01-01' then 'recent joiners'
else 'long-term employee'
end as joining_status
 from employee;
```

10.Display first_name, joining_date, and leave entitlement. (If joined before '2021-01-01', assign '10 Days Leave'; between '2021-01-01' and '2023-01-01', assign '20 Days Leave'; otherwise, '25 Days Leave')

```
Sql> select first_name, joining_date,
 case
when joining_date<='2021-01-01' then 10
when joining_date between '2021-01-01' and '2023-01-01' then 20
else 25
end as leave_entitlement
 from employee;
```

11.. Display first_name, salary, department, and promotion eligibility. (If salary is above 60,000 and department is 'IT', mark as 'Promotion Eligible'; otherwise, 'Not Eligible')

```
Sql>select first_name,salary, department,
case
when department='it' and salary>=600000 then 'promotion eligible'
else 'not eligible'
end as promotion_eligibility
 from employee;
```

12.Display first_name, salary, and overtime pay eligibility. (If salary is below 50,000, mark as 'Eligible for Overtime Pay'; otherwise, 'Not Eligible')

```
select first_name, salary ,
case
```

when salary<50000 then 'eligible for over time pay'

else 'not eligible'

end as over_time_pay

from employee;

13. Display first_name, department, salary, and job title. (If department is 'HR' and salary

is above 60,000, mark as 'HR Executive'; if department is 'Finance' and salary is above

55,000, mark as 'Finance Manager'; otherwise, 'Regular Employee')

Sql>select first_name, department,salary,

case

when department='hr' and salary>=60000 then 'hr exicutive'

when department='finance' and salary>=55000 then 'finance manager'

else 'regular employee'

end as job_title

from employee;

14.Display first_name, salary, and salary comparison to the company average. (If salary is

above the company's average salary, mark as 'Above Average'; otherwise, 'Below

Average')

Sql>select first_name, salary,

case

when salary>(select avg(salary) from employee) then 'above average'

else 'below average'

end as salary_comparision

from employee;

Group by

1.Write the query to get the department and department wise total(sum) salary,

display it in ascending and descending order according to salary.

Sql>select department,sum(salary) as department_total_sum

```
from employee
group by department
order by sum(salary);
select department,sum(salary) as department_total_sum
 from employee
group by department
order by sum(salary) desc;
```

2.Write down the query to fetch Project name assign to more than one Employee

```
select emp_id_no ,
project_name from project_details
group by project_name
 having COUNT(emp_id_no) > 1;
```

3.Write the query to get the department, total no. of departments, total(sum) salary with respect to department from "employee table" table.*/

```
Sql>select department,
(select count(distinct department) from employee) as total_Departments, sum(salary)
 from employee
group by department;
```

4**.Get the department-wise salary details from the "employee table" table:**
What is the average salary? (Order by salary ascending)
What is the maximum salary? (Order by salary ascending)

```
Sql>select department,
round(avg(salary),2) as average_department_salary,
 max(salary) as max_salary_department
 from employee
group by department
 order by average_department_salary;
```

5. Display department-wise employee count and categorize based on size. (If a department

has more than 5 employees, label it as 'Large'; between 3 and 5 as 'Medium'; otherwise, 'Small')*/

```
Sql>select department,
count(emp_id) as total_employeed,
case
when count(emp_id)>=5 then 'large'
when count(emp_id) between 3 and 5 then 'medium'
else 'samll'
end as department_category from employee
 group by department;
```

6.Display department-wise average salary and classify pay levels. (If the average salary in a

department is above 60,000, label it as 'High Pay'; between 50,000 and 60,000

as 'Medium Pay'; otherwise, 'Low Pay').

```
Sql>select department,round(avg(salary),2) as average_department_salary,
 case
when avg(salary)>=60000 then 'high pay'
when avg(salary) between 50000 and 60000 then 'medium pay'
else 'low pay'
end as pay_level
 from employee
 group by department;
```

7. Display department, gender, and count of employees in each category. (Group by

department and gender, showing total employees in each combination)

```
select department,
 gender,
```

count(emp_id)

from employee

group by department, gender

order by department, gender;

8. Display the number of employees who joined each year and categorize hiring trends. (If a

year had more than 5 hires, mark as 'High Hiring'; 3 to 5 as 'Moderate Hiring'; otherwise, 'Low Hiring')

Sql>select year(joining_date) as joined_year,

count(emp_id) as total_employees,

case

when count(emp_id)>=5 then 'high hiring'

when count(emp_id) between 3 and 5 then 'medium hiring'

else 'low hiring'

end as hiring_trends

from employee

group by joined_year;

9. Display department-wise highest salary and classify senior roles. (If the highest salary in a

department is above 70,000, label as 'Senior Leadership'; otherwise, 'Mid-Level')

Sql>select department,

case

when max(salary)>=70000 then 'senior leadership'

else 'mid leadership'

end as senior_clasification

from employee

group by department;

10. Display department-wise count of employees earning more than 60,000. (Group

employees by department and count those earning above 60,000, labeling departments with more than 2 such employees as 'High-Paying Team')

```
select department, count(emp_id) as high_salary,
 case
when count(emp_id)>=2 then 'high-paying team'
else 'low paying team'
end as department_labling
from employee
where salary>=60000
 group by department;
```

Date and Time

1.Query to extract the below things from joining_date column. (Year, Month, Day, Current Date)

```
Sql>select year(joining_date) as year_joining,
month(joining_date) as joining_month,
 day(joining_date) as joining_day,
 date(now()) as today
 from employee;
```

2.Create two new columns that calculate the difference between joining_date and the current date. One column should show the difference in months, and the other should show the difference in days

```
select
    emp_id,
    FIRST_NAME,
    joining_date,
    DATEDIFF(CURRENT_DATE, joining_date) AS diff_days,

    TIMESTAMPDIFF(MONTH, joining_date, CURRENT_DATE) AS diff_months
FROM employee;
```

#**3.Get all employee details from the employee table whose joining year is 2020.**

```
select * from employee
where year(joining_date)=2020;
```

#**4.Get all employee details from the employee table whose joining month is Feb.**

```
select * from employee
where month(joining_date)=2;
```

/***5.Get all employee details from employee table whose joining date between "2021-01-01" and "2021-12-01"***/

```
select * from employee
where joining_date between '2021-01-01' and  "2021-12-01";
```

#----------JOINS----------

1.Get the employee name and project name from the "employee table" and "ProjectDetail" for employees who have been assigned a project, sorted by first name.

```
Sql>select e.first_name,
e.last_name,
p.project_name from employee as e
inner join pro_details as p
on e.emp_id=p.emp_id_no
order by e.first_name;
```

2.Get the employee name and project name from the "employee table" and "ProjectDetail" for all employees, including those who have not been assigned a project, sorted by first name.

```
Sql>select a.first_name,
 p.project_name from employee as a
left join
pro_details as p
on a.emp_id=p.emp_id_no
order by a.first_name;
```

3.**Get the employee name and project name from the "employee table" and**

"ProjectDetail" for all employees. If an employee has no assigned project, display "-No Project Assigned," sorted by first name.*/

```
Sql>Select a.first_name,
   ifnull(p.project_name, 'not assigned') as project_name
from employee as a
left join pro_details as p
on a.emp_id = p.emp_id_no
order  by a.first_name;
```

4.Get all project names from the "ProjectDetail" table, even if they are not linked to any employee, sorted by first name from the "employee table" and "ProjectDetail" table

```
Sql >Select  a.first_name,
     p.project_name
from employee as a
right join pro_details as p on a.emp_id = p.emp_id_no
order by a.first_name;
```

5.Find the project names from the "ProjectDetail" table that have not been assigned to any employee using the "employee table" and "ProjectDetail" table.

```
Sql> select p.project_name
From Pro_Details as p
Left join employee as e
on p.emp_id_no = e.emp_id
where  e.emp_id is null;
```

6.Get the employee name and project name for employees who are assigned to more than one project.

```
Sql> Select e.first_name,
   p.project_name
from employee e
join pro_details p
 on e.emp_id = p.emp_id_no
```

```
where
   e.emp_id in (
select emp_id_no
      from pro_details
      group by emp_id_no
      having count(emp_id_no) > 1
   )
Order by e.first_name;
```

7.Get the project name and the employee names of employees working on projects that have more than one employee assigned.

```
Sql> Select e.first_name,
   p.project_name
from
employee e
join pro_details p ON e.emp_id = p.emp_id_no
where
   e.emp_id in (
      select emp_id_no
      from pro_details
      group by emp_id_no
      having count(emp_id_no) > 1
      and status= 'ongoing'
   )
Order by
   e.first_name;
```

8.Get records from the "ProjectDetail" table where the corresponding employee ID does not exist in the "employee table.

```
select *
from Pro_Details
where emp_id_no not in(
   select  emp_id
```

```
    from employee
);
```