**1.Employee Table**

# Slip 2:

Q 1.Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the average salary of each job title, sorted in descending order of average salary:

2)List the department and the maximum salary among employees who joined after 1995, sorted by maximum salary in descending order

🡪 CREATE DATABASE Employee;

-- use Employee;

-- create table Employee(

-> emp\_id Int Primary Key,

-> emp\_name Varchar(255),

-> job\_name Varchar(50),

-> manager\_id Int,

-> hire\_date Date,

-> salary Decimal(10,2)

-> );

-- INSERT INTO Employee (emp\_id, emp\_name, job\_name, manager\_id, hire\_date, salary) VALUES

-> (68319, 'KAYLING', 'PRESIDENT', NULL, '1991-11-18', 6000.00),

-> (66928, 'BLAZE', 'MANAGER', 68319, '1991-05-01', 2750.00),

-> (67832, 'CLARE', 'MANAGER', 68319, '1991-06-09', 2550.00),

-> (65646, 'JONAS', 'MANAGER', 68319, '1991-04-02', 2957.00),

-> (67858, 'SCARLET', 'ANALYST', 65646, '1997-04-19', 3100.00),

-> (69062, 'FRANK', 'ANALYST', 65646, '1991-12-03', 3100.00),

-> (63679, 'SANDRINE', 'CLERK', 69062, '1990-12-18', 900.00),

-> (64989, 'ADELYN', 'SALESMAN', 66928, '1991-02-20', 1700.00),

-> (65271, 'WADE', 'SALESMAN', 66928, '1991-02-22', 1350.00),

-> (66564, 'MADDEN', 'SALESMAN', 66928, '1991-09-28', 1350.00),

-> (68454, 'TUCKER', 'SALESMAN', 66928, '1991-09-08', 1600.00),

-> (68736, 'ADNRES', 'CLERK', 67858, '1997-05-23', 1200.00),

-> (69000, 'JULIUS', 'CLERK', 66928, '1991-12-03', 1050.00),

-> (69324, 'MARKER', 'CLERK', 67832, '1992-01-23', 1400.00);

List the average salary of each job title, sorted in descending order of average salary:

select job\_name , AVG(salary) as avg\_salary

-> from Employee

-> group by job\_name

-> order by avg\_salary;

List the department and the maximum salary among employees who joined after 1995, sorted by maximum salary in descending order

-- select job\_name ,MAX(salary) as max\_salary

-> from Employee

-> where hire\_date > '1995-12-31'

-> group by job\_name

-> order by max\_salary DESC;

# Slip 7:

Que 1.Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

1)Find the total number of employees in each department whose salary is above $2000, sorted by department name:

→select job\_name as department, count(\*) as num\_emp\_above\_2000

-> from Employee

-> where salary > 2000

-> group by job\_name

-> order by department;

2)List the manager\_id and the number of employees managed by each manager who manages more than one employee, sorted by manager\_id:

→select manager\_id , count(\*) as num\_employees\_managed

-> from Employee

-> where manager\_id is not null

-> group by manager\_id

-> having count(\*) > 1

-> order by manager\_id;

# Slip- 8

Que 1.Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

1)Find the department with the highest average salary among departments with at least 3 employees:

→SELECT job\_name AS department, AVG(salary) AS avg\_salary

-> FROM Employee

-> WHERE job\_name IN (

-> SELECT job\_name

-> FROM Employee

-> GROUP BY job\_name

-> HAVING COUNT(\*) >= 3

-> )

-> GROUP BY job\_name

-> ORDER BY avg\_salary DESC

-> LIMIT 1;

2)Find the top 3 departments with the highest total salary expenditure, sorted by total salary expenditure in descending order

→SELECT job\_name AS department, SUM(salary) AS total\_salary\_expenditure

-> FROM Employee

-> GROUP BY job\_name

-> ORDER BY total\_salary\_expenditure DESC

-> LIMIT 3;

# Slip 10:

Que1)Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

1)List the manager\_id and the average salary of employees managed by each manager who manages at least 2 employees, sorted by average salary in descending order:

→select manager\_id , avg(salary) as avg\_salary

-> from Employee

-> where manager\_id is not null

-> group by manager\_id

-> having count(\*) >= 2

-> order by avg\_salary desc

-> ;

2)Find the departments where the average salary of employees is greater than the average salary of all employees, sorted by department name

à SELECT job\_name AS department, AVG(salary) AS department\_avg\_salary

-> FROM Employee

-> GROUP BY job\_name

-> HAVING AVG(salary) > (

-> SELECT AVG(salary)

-> FROM Employee

-> )

-> ORDER BY department;

**2.Worker Table**

# Slip 1:

Que1)Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)

Write a query to insert 5 rows in it as per the query requirements.

1)List the department and the total salary for each department, sorted by total salary in descending order:

2)Find the average salary for each department and display only those departments where the average salary is above $100,000, sorted by average salary in descending order

à CREATE DATABASE worker;

--- USE worker;

--- CREATE TABLE Worker (

-> WORKER\_ID INT PRIMARY KEY,

-> FIRST\_NAME VARCHAR(100),

-> LAST\_NAME VARCHAR(100),

-> SALARY DECIMAL(10, 2),

-> JOINING\_DATE DATE,

-> DEPARTMENT VARCHAR(100)

-> );

##Write a query to insert 5 rows in it as per the query requirements.

INSERT INTO worker (WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT)VALUES

-> (68319, 'KAYLING', '', 6000.00, '1991-11-18', 'PRESIDENT'),

-> (66928, 'BLAZE', '', 2750.00, '1991-05-01', 'MANAGER'),

-> (67832, 'CLARE', '', 2550.00, '1991-06-09', 'MANAGER'),

-> (65646, 'JONAS', '', 2957.00, '1991-04-02', 'MANAGER'),

-> (67858, 'SCARLET', '', 3100.00, '1997-04-19', 'ANALYST');

## List the department and the total salary for each department, sorted by total salary in descending order

SELECT DEPARTMENT, SUM(SALARY) AS total\_salary

-> FROM Worker

-> GROUP BY DEPARTMENT

-> ORDER BY total\_salary DESC;

Find the average salary for each department and display only those departments where the average salary is above $100,000, sorted by average salary in descending order

-- SELECT DEPARTMENT, AVG(SALARY) AS average\_salary

-> FROM worker

-> GROUP BY DEPARTMENT

-> HAVING AVG(SALARY) > 100000

-> ORDER BY average\_salary DESC;

# Slip 3:

Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)

Write a query to insert 5 rows in it as per the query requirements.

1)List the number of workers in each department who joined after January 1, 2021, sorted by department name:

SELECT DEPARTMENT, COUNT(\*) AS num\_workers

-> FROM Worker

-> WHERE JOINING\_DATE > '2021-01-01'

-> GROUP BY DEPARTMENT

-> ORDER BY DEPARTMENT;

2)Find the department with the highest number of workers earning a salary greater than $90,000, sorted by department name:

SELECT DEPARTMENT, COUNT(\*) AS num\_workers

-> FROM Worker

-> WHERE SALARY > 90000

-> GROUP BY DEPARTMENT

-> ORDER BY num\_workers DESC

-> LIMIT 1;

# Slip:6

Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)

Write a query to insert 5 rows in it as per the query requirements.

1)List the department and the number of workers who joined in February 2021 and have a salary greater than $80,000, sorted by department name:

select DEPARTMENT , count(\*) as num\_workers

-> from worker

-> WHERE JOINING\_DATE >= '2021-02-01' AND JOINING\_DATE < '2021-03-01'

-> AND SALARY > 80000

-> group by DEPARTMENT

-> order by DEPARTMENT;

2)Find the department with the highest average salary among departments with at least 2 workers, sorted by average salary in descending order:

select DEPARTMENT , avg(SALARY) as avg\_salary

-> from worker

-> group by DEPARTMENT

-> having count(\*) >= 2

# Slip 9

Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the departments where the total salary expenditure is less than $300,000, sorted by total salary expenditure in ascending order

select DEPARTMENT, sum(SALARY) as total\_exp

-> from worker

-> group by DEPARTMENT

-> having sum(SALARY) < 300000

-> order by total\_exp

-> ;

2)Find the worker with the highest salary in each department, sorted by department name:

SELECT DEPARTMENT, WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY

-> FROM Worker

-> WHERE (DEPARTMENT, SALARY) IN (

-> SELECT DEPARTMENT, MAX(SALARY)

-> FROM Worker

-> GROUP BY DEPARTMENT

-> )

-> ORDER BY DEPARTMENT;

# Slip 12:

Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)

Write a query to insert 5 rows in it that as per the query requirements.

1)Find the departments where the number of workers is equal to the number of distinct first names, sorted by department name

SELECT DEPARTMENT

FROM Worker

GROUP BY DEPARTMENT

HAVING COUNT(\*) = COUNT(DISTINCT FIRST\_NAME)

ORDER BY DEPARTMENT;

2)List the number of workers in each department who joined after January 1, 2021, sorted by department name

àSELECT DEPARTMENT, COUNT(\*) AS num\_workers

FROM Worker

WHERE JOINING\_DATE > '2021-01-01'

GROUP BY DEPARTMENT

ORDER BY DEPARTMENT;

**Patient data**

# Slip 4

Consider the Patient table

Patient(PatientID,Name,DateOfBirth,Gender,admit\_date,ward\_no,City)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of patients admitted to each ward, sorted by ward number:

SELECT WARD\_NO, COUNT(\*) AS num\_patients

-> from Patient

-> group by WARD\_NO

-> ORDER BY WARD\_NO;

2)Find the average age of patients admitted to each ward, and display only those wards where the average age is below 40, sorted by average age in descending order

SELECT WARD\_NO , AVG(DATEDIFF(CURRENT\_DATE(), DOB)/365) AS AVG\_AGE

-> FROM Patient

-> group by WARD\_NO

-> HAVING AVG\_AGE < 40

-> ORDER BY AVG\_AGE DESC;

# Slip 5

Consider the Patient table

Patient(PatientID,Name,DateOfBirth,Gender,admit\_date,ward\_no,City)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of male and female patients admitted to each ward, sorted by ward number and gender:

select WARD\_NO , GENDER , COUNT(\*) AS NUM\_PATIENTS

-> FROM Patient

-> group by WARD\_NO, GENDER

-> ORDER BY WARD\_NO ,GENDER;

2)Find the ward with the highest number of patients admitted, and display the top 3 wards with the highest number of patients, sorted by the number of patients in descending order

SELECT WARD\_NO , COUNT(\*) AS NUM\_PATIENTS

-> FROM Patient

-> group by WARD\_NO

-> ORDER BY NUM\_PATIENTS DESC

-> LIMIT 3;

BOOK Dataset

# Slip 11

Consider the Book table

book (book\_id,title,author,publication\_year,language,available\_copies,total\_copies)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of books published in each year, sorted by publication year in ascending order

select PUBLICATION\_YEAR, COUNT(\*) AS NUM\_BOOK\_PUBLISHED

-> FROM BOOK

-> GROUP BY PUBLICATION\_YEAR

-> ORDER BY PUBLICATION\_YEAR

-> ;

2)Find the authors who have written more than one book, along with the total number of books they have written, sorted by the number of books in descending order

SELECT AUTHOR , COUNT(\*) AS NUM\_BOOKS

-> FROM BOOK

-> GROUP BY AUTHOR

-> HAVING COUNT(\*)>1

-> ORDER BY NUM\_BOOKS DESC;

# Slip 13

Consider the Book table

book (book\_id,title,author,publication\_year,language,available\_copies,total\_copies)

Write a query to insert 5 rows in it that as per the query requirements.

1)Find the authors who have written books in more than one language, along with the total number of languages they have written books in, sorted by author name:

SELECT AUTHOR , COUNT(DISTINCT LANGUAGE) AS NUM\_LANGUAGES

-> FROM BOOK

-> GROUP BY AUTHOR

-> HAVING COUNT(DISTINCT LANGUAGE) >1

-> ORDER BY AUTHOR;

2)List the titles of books along with the average number of available copies per book, and display only those books where the average number of available copies is less than 5, sorted by average available copies in descending order

SELECT TITLE , AVG(AVAILABLE\_COPIES) AS AVG\_AVAI\_COPIES

-> FROM BOOK

-> GROUP BY TITLE

-> HAVING AVG(AVAILABLE\_COPIES) < 5

-> ORDER BY AVG\_AVAI\_COPIES;

# Slip 14

Consider the Book table

book (book\_id,title,author,publication\_year,language,available\_copies,total\_copies)

Write a query to insert 5 rows in it that as per the query requirements.

1)Find the most common publication language among books, along with the total number of books published in each language, sorted by the number of books in descending order

SELECT LANGUAGE , COUNT(\*) AS NUM\_BOOKS

-> FROM BOOK

-> GROUP BY LANGUAGE

-> ORDER BY NUM\_BOOKS DESC;

2)Find the authors who have written books with the highest average number of total copies available, and display the top 3 authors with the highest average, sorted by average copies in descending order

SELECT AUTHOR , AVG(TOTAL\_COPIES) AS AVG\_TOTAL\_COPIES

-> FROM BOOK

-> GROUP BY AUTHOR

-> ORDER BY AVG\_TOTAL\_COPIES DESC

-> LIMIT 3;

# Slip 15

Consider the Book table

book (book\_id,title,author,publication\_year,language,available\_copies,total\_copies)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the titles of books along with the total number of available copies for each book, and display only those books where the total number of available copies is less than the total number of copies, sorted by total available copies in ascending order

SELECT TITLE, AVAILABLE\_COPIES, TOTAL\_COPIES

-> FROM BOOK

-> WHERE AVAILABLE\_COPIES < TOTAL\_COPIES

-> ORDER BY AVAILABLE\_COPIES ASC;

2)Find the authors who have written books with the highest difference between available copies and total copies, and display the top 3 authors with the highest difference, sorted by difference in descending order

SELECT AUTHOR, MAX(AVAILABLE\_COPIES - TOTAL\_COPIES) AS difference

-> FROM BOOK

-> GROUP BY AUTHOR

-> ORDER BY difference DESC

-> LIMIT 3;

Course Data

# Slip 16

Consider the Course table

Courses(CourseID,Title,Instructor,Category,Price,Duration,EnrollmentCount)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of courses in each category, sorted by category name:

SELECT Category, COUNT(\*) AS NumberOfCourses

-> FROM course

-> GROUP BY Category

-> ORDER BY Category;

2)Find the instructors who teach more than one course, along with the total number of courses they teach, sorted by the number of courses in descending order:

SELECT Instructor, COUNT(\*) AS TotalCoursesTaught

-> FROM course

-> GROUP BY Instructor

-> HAVING COUNT(\*) > 1

-> ORDER BY TotalCoursesTaught DESC;

# Slip 17

Consider the Course table

Courses(CourseID,Title,Instructor,Category,Price,Duration,EnrollmentCount)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the categories along with the total number of courses in each category, and-+ display only those categories with more than 5 courses, sorted by the number of courses in descending order:

SELECT Category, COUNT(\*) AS NumberOfCourses

-> FROM course

-> GROUP BY Category

-> HAVING COUNT(\*) > 5

-> ORDER BY NumberOfCourses DESC;

2)Find the categories where the average duration of courses is less than 8 weeks, sorted by category name:

SELECT Category

-> FROM course

-> GROUP BY Category

-> HAVING AVG(Duration) < 8

-> ORDER BY Category;

# Slip 18

Consider the Course table

Courses(CourseID,Title,Instructor,Category,Price,Duration,EnrollmentCount)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the instructors along with the total number of courses they teach, and display only those instructors who teach courses with more than 500 enrollments, sorted by the number of courses in descending order:

SELECT Instructor, COUNT(\*) AS TotalCoursesTaught

-> FROM course

-> WHERE EnrollmentCount > 500

-> GROUP BY Instructor

-> HAVING COUNT(\*) > 0

-> ORDER BY TotalCoursesTaught DESC;

2)Find the categories where the average enrollment count of courses is greater than 700, sorted by average enrollment count in descending order:

SELECT Category, AVG(EnrollmentCount) AS AvgEnrollmentCount

-> FROM course

-> GROUP BY Category

-> HAVING AVG(EnrollmentCount) > 700

-> ORDER BY AvgEnrollmentCount DESC;

# Slip 19

Consider the Course table

Courses(CourseID,Title,Instructor,Category,Price,Duration,EnrollmentCount)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the categories where the maximum price of courses is less than $50, sorted by category name:

SELECT Category

-> FROM course

-> GROUP BY Category

-> HAVING MAX(Price) < 50

-> ORDER BY Category;

2)Find the instructors who teach courses with the highest average enrollment count, and display the top 3 instructors with the highest average, sorted by average enrollment count in descending order:

SELECT Instructor, AVG(EnrollmentCount) AS AvgEnrollmentCount

-> FROM course

-> GROUP BY Instructor

-> ORDER BY AvgEnrollmentCount DESC

-> LIMIT 3;

# Slip 20

Consider the Course table

Courses(CourseID,Title,Instructor,Category,Price,Duration,EnrollmentCount)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the categories where the total enrollment count of courses is more than 3000, sorted by total enrollment count in descending order:

SELECT Category, SUM(EnrollmentCount) AS TotalEnrollmentCount

-> FROM course

-> GROUP BY Category

-> HAVING SUM(EnrollmentCount) > 3000

-> ORDER BY TotalEnrollmentCount DESC;

2)Find the categories where the total number of courses is equal to the number of distinct instructors, sorted by category name:

SELECT Category

-> FROM (

-> SELECT Category, COUNT(\*) AS TotalCourses, COUNT(DISTINCT Instructor) AS DistinctInstructors

-> FROM course

-> GROUP BY Category

-> ) AS subquery

-> WHERE TotalCourses = DistinctInstructors

-> ORDER BY Category;