MCA 2nd Semester 2024 Jadavpur University

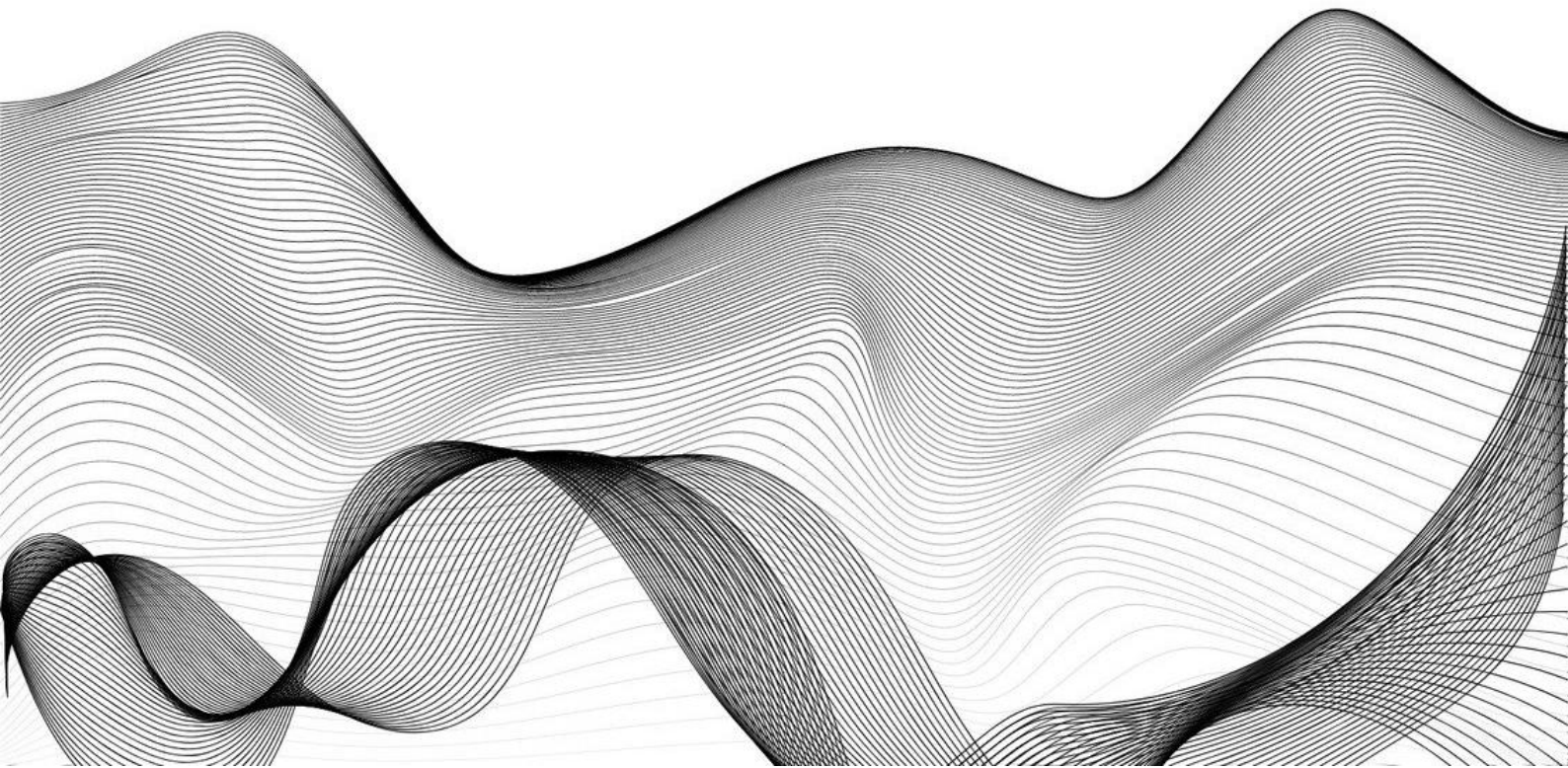
DBMS

Lab assignment

By

Jaidip Patra , roll 01

Joydipta Biswas , roll 02



Routine management system

General description

A Routine Management System (RMS) is a digital solution designed to assist individuals and organizations in planning, organizing, and managing their daily routines and tasks. The system aims to increase productivity by enabling users to schedule tasks, set reminders, track progress, and prioritize activities efficiently. RMS can be implemented as a web application, mobile app, or desktop software, catering to various user needs and preferences.

Problem Statement

Before the implementation of a computerized Routine Management System, several issues were prevalent due to the reliance on manual methods:

Task and Schedule Loss: Manual records are prone to being lost in a human-centric environment. Tasks, schedules, and important notes can easily be misplaced or forgotten, leading to inefficiencies and missed deadlines.

Task and Schedule Damage: Physical records are susceptible to damage from accidents such as spills or environmental factors like floods and fires. This results in the loss of crucial information and disrupts routine management.

Difficulty in Searching Records: Searching for specific tasks or schedules in a large volume of manual records is time-consuming and inefficient. Without a computerized system, finding particular information can be a daunting task.

Space Consumption: As the number of records increases, the physical storage space required also grows. Managing large volumes of physical records becomes increasingly challenging without a computerized system.

Cost Consumption: Maintaining manual records involves continuous expenses for paper, storage, and other materials. The absence of a computerized system increases the overall cost of managing routines and tasks.

System analysis and requirement

Functional Requirements

User Authentication: The system should provide secure user authentication, including registration, login, and password recovery.

Task Management: Users should be able to create, edit, delete, and view tasks.

Scheduling: The system should allow users to schedule tasks, set due dates, and prioritize tasks.

Reminders and Notifications: The system should send reminders and notifications for upcoming tasks and deadlines.

Progress Tracking: Users should be able to track the progress of their tasks and view completed tasks.

Calendar Integration: The system should integrate with popular calendar services (e.g., Google Calendar) to sync tasks and events.

Reporting: The system should generate reports on task completion, productivity, and time management.

User Interface: The system should have an intuitive and user-friendly interface accessible via web and mobile platforms.

Non-Functional Requirements

Scalability: The system should be able to handle a growing number of users and tasks without performance degradation.

Security: The system must ensure data privacy and security, including encryption and secure communication.

Performance: The system should provide quick response times for user actions and data retrieval.

Reliability: The system should be reliable and available with minimal downtime.

Usability: The system should be easy to use, with clear navigation and instructions.

Compatibility: The system should be compatible with various devices and operating systems.

Software and Hardware Analysis

Software Requirements

Programming Languages: JavaScript (for frontend), Python/Java (for backend)

Frameworks: React.js or Angular (frontend), Django or Spring Boot (backend)

Database: MySQL, PostgreSQL, or MongoDB

Cloud Services: AWS, Google Cloud, or Azure for hosting and storage

API Integration: RESTful APIs for external integrations (e.g., calendar services)

Development Tools: Git, Docker, Kubernetes for version control and containerization

Hardware Requirements

Server: High-performance server with at least 16GB RAM, multi-core processor, and SSD storage

User Devices: PCs, laptops, smartphones, or tablets with internet connectivity

Network: Reliable internet connection with adequate bandwidth

Data Collection

Data collection for the RMS involves gathering information from users during registration and task management activities. The key data points include:

User Information: Name, email, password (encrypted)

Task Details: Task name, description, due date, priority level, status (pending, in-progress, completed)

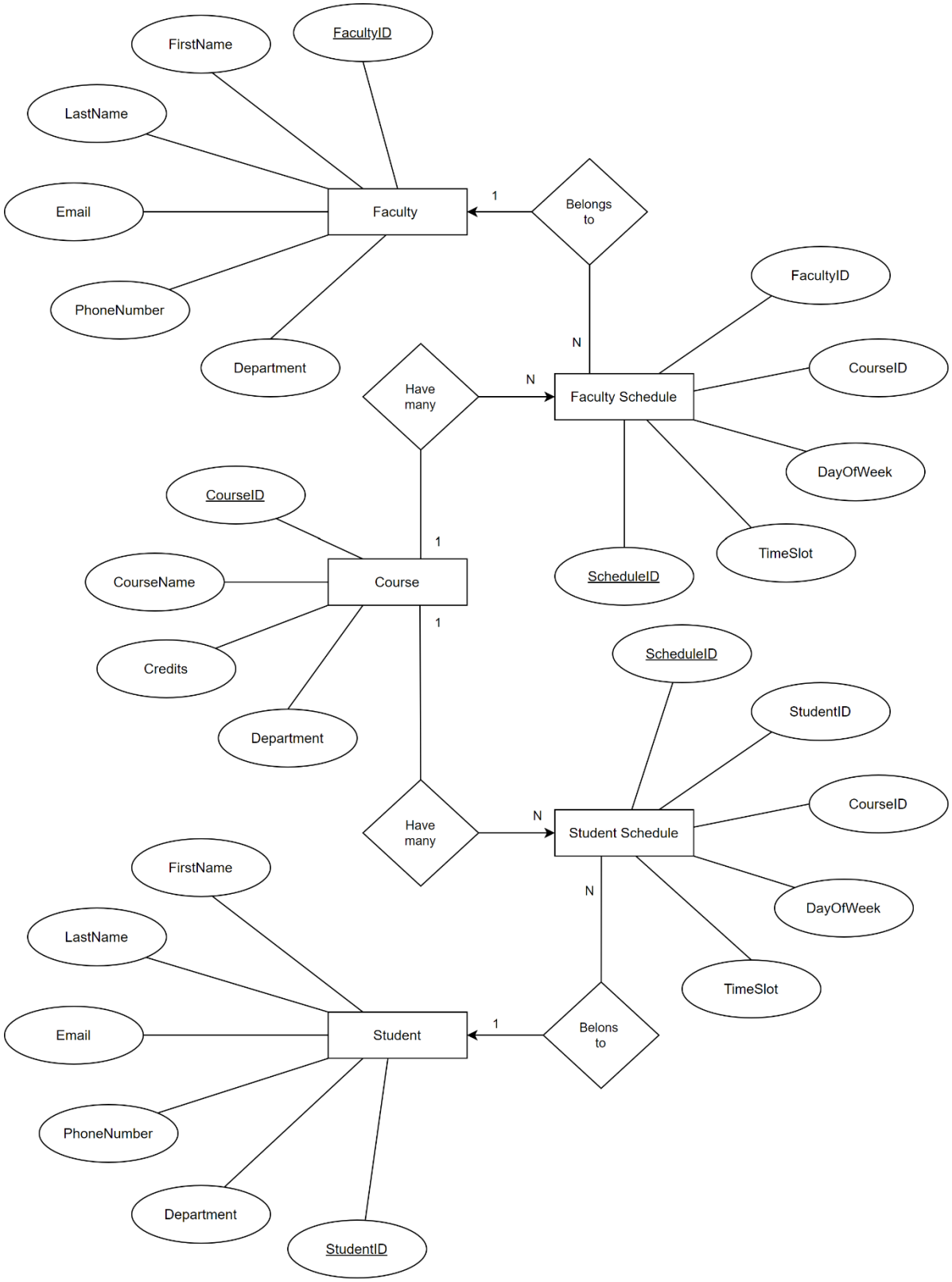
Reminder Settings: Reminder times, notification preferences

Progress Data: Task completion status, time spent on tasks

Calendar Events: Synced calendar events and schedules

ER diagram

Below is the Entity-Relationship (ER) diagram for the Routine Management System



Assignment 1

Q1) **Create the following tables:**

```
SQL:    CREATE TABLE EMPLOYEE (
        EMP_CODE char(16) primary key,      EMP_NAME char(20),    DEPT_CODE char(16),
DESIG_CODE char(16),
        SEX char(1),    ADDRESS char (25),     CITY char (20),     STATE char (20),     PIN
char (6),
        BASIC decimal(9,2),   JN_DT Date );

        CREATE TABLE DESIGNATION (
        DESIG_CODE char(16) primary key,      DESIG_DESC char(20)          );

        CREATE TABLE DEPARTMENT (
        DEPT_CODE char(16) primary key,     DEPT_NAME char(20)           );
```

Q2) **Display the structure of each table**

```
SQL:    desc EMPLOYEE;
        desc DESIGNATION;
        desc DEPARTMENT;
```

Column	Null?	Type
EMP_CODE	NOT NULL	CHAR(5)
EMP_NAME	-	CHAR(20)
DEPT_CODE	-	CHAR(5)
DESIG_CODE	-	CHAR(5)
SEX	-	CHAR(1)
ADDRESS	-	CHAR(25)
CITY	-	CHAR(20)
STATE	-	CHAR(20)
PIN	-	CHAR(6)
BASIC	-	NUMBER
JN_DT	-	DATE

Column	Null?	Type
DESIG_CODE	NOT NULL	CHAR(5)
DESIG_DESE	-	CHAR(20)

Column	Null?	Type
DEPT_CODE	NOT NULL	CHAR(5)
DEPT_NAME	-	CHAR(15)

Q3) **Insert few rows in each table.**

```
insert into DESIGNATION values ('D002','Executive')
insert into DESIGNATION values ('D001','Manager')
insert into DESIGNATION values ('D003','Officer')
insert into DESIGNATION values ('D004','Clerk')
insert into DESIGNATION values ('D005','Helper')
insert into DEPARTMENT values ('DP01','Personnel')
insert into DEPARTMENT values ('DP02','Production')
insert into DEPARTMENT values ('DP03','Purchase')
insert into DEPARTMENT values ('DP04','Finnance')
```

```
insert into DEPARTMENT values ('DP05','Research ')
insert into EMPLOYEE values ('E001','TAPAN','DP01', 'D001', 'M',
'BALARAMPUR','PURULIA','WEST BENGAL','723103',80000,
to_date('21:01:2024','dd:mm:yyyy'))
insert into EMPLOYEE values ('E002','ROHIT','DP02', 'D002', 'M',
'ABC','PURULIA','WEST BENGAL','723102',70000, to_date('21:01:2024','dd:mm:yyyy'))
insert into EMPLOYEE values ('E003','SUDIP',NULL, 'D002', 'M',
'ABC','PURULIA','WEST BENGAL','723102', NULL,  to_date('21:01:2024','dd:mm:yyyy'))
insert into EMPLOYEE values ('E004','AMIT','DP03', 'D003', 'M',
'ABC','PURULIA','WEST BENGAL','723102',0,
to_date('21:01:2024','dd:mm:yyyy'))
```

Q4) **In EMP table insert few rows without DEPT_CODE and BASIC.**

```
insert into EMPLOYEE values ('E003','SUDIP',NULL, 'D002', 'M',
'ABC','PURULIA','WEST BENGAL','723102', NULL,  to_date('21:01:2024','dd:mm:yyyy'))
```

Q5) **Find the rows with unassigned DEPT_CODE**

```
SELECT * FROM EMPLOYEE WHERE DEPT_CODE IS NULL
```

EMP_CODE	EMP_NAME	DEPT_CODE	DESIG_CODE	SEX	ADDRESS	CITY	STATE	PIN	BASIC	JN_DT
E003	SUDIP	-	D002	M	ABC	PURULIA	WEST BENGAL	723102	-	21-JAN-24

Q6) **Find the rows with BASIC unassigned**

```
SELECT * FROM EMPLOYEE WHERE BASIC IS NULL
```

EMP_CODE	EMP_NAME	DEPT_CODE	DESIG_CODE	SEX	ADDRESS	CITY	STATE	PIN	BASIC	JN_DT
E003	SUDIP	-	D002	M	ABC	PURULIA	WEST BENGAL	723102	-	21-JAN-24

Q7) **Find the rows with basic = 0**

```
SELECT * FROM EMPLOYEE WHERE BASIC =0
```

EMP_CODE	EMP_NAME	DEPT_CODE	DESIG_CODE	SEX	ADDRESS	CITY	STATE	PIN	BASIC	JN_DT
E004	AMIT	DP03	D003	M	ABC	PURULIA	WEST BENGAL	723102	0	21-JAN-24
E005	ABC	DP03	D004	M	ABC_D	PURULIA	WEST BENGAL	723102	0	01-JAN-24
E006	DEF	DP02	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	02-JAN-24
E007	GHI	DP01	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	22-JAN-24

Q8) **Find the average basic of the employees.**

```
SELECT AVG(BASIC) AS average_basic FROM EMPLOYEE
```

AVERAGE_BASIC
26250

Q9) **Replace the BASIC with 0 for the rows with unassigned Basic.**

```
UPDATE EMPLOYEE SET BASIC = 0 WHERE BASIC IS NULL
```

Q10) **Again, find the average Basic. (Note the difference of result obtained in Q.8**

```
SELECT AVG(BASIC) AS average_basic FROM EMPLOYEE
```

AVERAGE_BASIC
26250

Q11) **Delete the rows with unassigned DEPT_CODE.**

```
DELETE FROM EMPLOYEE WHERE DEPT_CODE IS NULL
```

Q12) **Say, Net pay of an employee= Basic+HRA+DA where HRA is 50% of the Basic & DA is 40% of Basic. Show the employee name & Net pay for all employees.**

```
SELECT EMP_NAME, 50/100 * BASIC AS HRA, 40/100 * BASIC AS DA, BASIC+(50/100 *
BASIC)+(40/100* BASIC) AS NET_PAY FROM EMPLOYEE
```

EMP_NAME	HRA	DA	NET_PAY
TAPAN	40000	32000	152000
ROHIT	35000	28000	133000
AMIT	0	0	0
ABC	0	0	0
DEF	0	0	0
GHI	0	0	0
ANITA	30000	24000	114000

Q13) **Show the EMP_NAME & BASIC in the ascending order of DEPT_CODE. The employee name must appear in uppercase.**

```
SELECT UPPER(EMP_NAME) AS EMPLOYEE_NAME , BASIC
FROM EMPLOYEE
ORDER BY DEPT_CODE ASC
```

EMPLOYEE_NAME	BASIC
GHI	0
TAPAN	80000
ANITA	60000
ROHIT	70000
DEF	0
AMIT	0
ABC	0

Q14) **Find the employees who have joined after 1 st January 2010.**

```
SELECT *
FROM EMPLOYEE
WHERE JN_DT  > TO_DATE('2010-01-01', 'YYYY-MM-DD')
```

EMP_CODE	EMP_NAME	DEPT_CODE	DESIG_CODE	SEX	ADDRESS	CITY	STATE	PIN	BASIC	JN_DT
E001	TAPAN	DP01	D001	M	BALARAMPUR	PURULIA	WEST BENGAL	723103	80000	21-JAN-24
E002	ROHIT	DP02	D002	M	ABC	PURULIA	WEST BENGAL	723102	70000	21-JAN-24
E004	AMIT	DP03	D003	M	ABC	PURULIA	WEST BENGAL	723102	0	21-JAN-24
E005	ABC	DP03	D004	M	ABC_D	PURULIA	WEST BENGAL	723102	0	01-JAN-24
E006	DEF	DP02	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	02-JAN-24

E007	GHI	DP01	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	22-JAN-24
E008	ANITA	DP01	D005	F	ABC	PURULIA	WEST BENGAL	723102	60000	23-JAN-24

Q15) **Find, how many employees have joined in the month of January?**

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE TO_CHAR(JN_DT, 'MM') = '01'
```

COUNT(*)
7

Q16) **Find the maximum & minimum Basic.**

```
SELECT MAX(BASIC) AS  MAXIMUM, MIN(BASIC) AS MINIMUM FROM EMPLOYEE
```

MAXIMUM	MINIMUM
80000	0

Q17) **Find how many Female employees are there?**

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE SEX = 'F'
```

COUNT(*)
1

Q18) **Replace CITY with existing value converted into uppercase for all rows.**

```
UPDATE EMPLOYEE SET CITY = UPPER(CITY)
```

Q19) **Find in how many different cities various employees are residing?**

```
SELECT COUNT(DISTINCT CITY) AS DISTINCT_CITY
FROM EMPLOYEE
```

DISTINCT_CITY
1

Q20) **Display the employee information in the ascending order of DEPT_CODE and with in a Department, it should be in the descending order of BASIC.**

```
SELECT *
FROM EMPLOYEE
ORDER BY DEPT_CODE ASC, BASIC DESC
```

EMP_CODE	EMP_NAME	DEPT_CODE	DESIG_CODE	SEX	ADDRESS	CITY	STATE	PIN	BASIC	JN_DT
E001	TAPAN	DP01	D001	M	BALARAMPUR	PURULIA	WEST BENGAL	723103	80000	21-JAN-24

E008	ANITA	DP01	D005	F	ABC	PURULIA	WEST BENGAL	723102	60000	23-JAN-24
E007	GHI	DP01	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	22-JAN-24
E002	ROHIT	DP02	D002	M	ABC	PURULIA	WEST BENGAL	723102	70000	21-JAN-24
E006	DEF	DP02	D005	M	ABC	PURULIA	WEST BENGAL	723102	0	02-JAN-24
E004	AMIT	DP03	D003	M	ABC	PURULIA	WEST BENGAL	723102	0	21-JAN-24
E005	ABC	DP03	D004	M	ABC_D	PURULIA	WEST BENGAL	723102	0	01-JAN-24

Assignment 2

Q1) **From the EMP table show the minimum, maximum and average basic for each department (show dept. Code).**

```
select DEPT_CODE, MIN(BASIC) as Minimum, MAX(BASIC) as Maximum, AVG(BASIC) as Average from employee group by DEPT_CODE
```

DEPT_CODE	MINIMUM	MAXIMUM	AVERAGE
DP01	0	80000	46666.666666666666666666666666666666666667
DP03	0	0	0
DP02	0	70000	35000

Q2) **Find the number of female employees in each department (show dept. Code).**

```
SELECT DEPT_CODE, COUNT(*) from employee where SEX='F' group by DEPT_CODE
```

DEPT_CODE	COUNT(*)
DP01	1

Q3) **Find the city wise no. of employees for each department (show dept. Code).**

```
SELECT count(*) as No_Of_Employee, CITY , DEPT_CODE from employee  GROUP BY CITY,DEPT_CODE
```

NO_OF_EMPLOYEE	CITY	DEPT_CODE
2	PURULIA	DP03
2	PURULIA	DP02
3	PURULIA	DP01

Q4) **Show the designation wise no of employees who have joined in the year 2000 in each department. The listing should appear in the ascending order of no. of employees.**

```
SELECT count(*) as No_Of_Employee, DESIG_CODE from employee where EXTRACT(YEAR FROM JN_DT)=2000 group by DESIG_CODE order by No_Of_Employee
```

Q5) **Find the department code wise total basic of male employees only for the departments for which such total is more than 50,000 and the listing should appear in the descending order of total basic.**

```
SELECT DEPT_CODE, SUM(BASIC) as Total_Basic from employee where SEX='M' group by DEPT_CODE having SUM(BASIC) >50000 order by Total_basic DESC
```

DEPT_CODE	TOTAL_BASIC
DP01	80000
DP02	70000

Q6) **Show the employee name, Designation description and basic for all employees.**

```
select EMP_NAME,DESIG_DESE,Basic from Employee,designation where Employee.DESIG_CODE=Designation.DESIG_CODE
```

EMP_NAME	DESIG_DESE	BASIC
TAPAN	Manager	80000
ROHIT	Executive	70000
AMIT	Officer	0
ABC	Clerk	0
DEF	Helper	0
GHI	Helper	0
ANITA	Helper	6000

Q7) **Show the employee name, Designation description, Department Name & Basic for all employees.**

```
select EMP_NAME,DESIG_DESE,Basic,DEPT_NAME from employee,designation,department where Employee.DESIG_CODE=Designation.DESIG_CODE AND Employee.DEPT_CODE = Department.DEPT_CODE
```

EMP_NAME	DESIG_DESE	BASIC	DEPT_NAME
TAPAN	Manager	80000	Personnel
GHI	Helper	0	Personnel
ANITA	Helper	60000	Personnel
ROHIT	Executive	70000	Production
DEF	Helper	0	Production
AMIT	Officer	0	Purchase
ABC	Clerk	0	Purchase

Q8) **Find the department Codes in which no employee works.**

```
SELECT Department.DEPT_CODE,
       count(Employee.DEPT_CODE)
FROM Department
LEFT OUTER JOIN Employee ON Employee.DEPT_CODE=Department.DEPT_CODE
GROUP BY Department.DEPT_CODE
HAVING count(Employee.EMP_CODE) = 0
```

DEPT_CODE	COUNT(EMPLOYEE.DEPT_CODE)
DP05	0
DP04	0

Q9) **Find the department names where at least one employee works.**

```
SELECT Department.DEPT_NAME
FROM Department
```

```
LEFT OUTER JOIN Employee ON Employee.DEPT_CODE=Department.DEPT_CODE
GROUP BY Department.DEPT_NAME
HAVING count(Employee.EMP_CODE)>=1
```

DEPT_NAME
Personnel
Production
Purchase

Q10) **Find the department names where at least 10 employees work.**

```
SELECT Department.DEPT_NAME
FROM Department
LEFT OUTER JOIN Employee ON Employee.DEPT_CODE=Department.DEPT_CODE
GROUP BY Department.DEPT_NAME
HAVING count(Employee.EMP_CODE)>=10
```

Q11) **Find the department code in which employee with highest Basic works.**

```
Select DEPT_CODE from EMPLOYEE where Basic=(Select MAX(BASIC) from Employee)
```

DEPT_CODE
DP01

Q12) **Find the Designation description of the employee with highest basic.**

```
Select DESIG_DESE from Designation where DESIG_CODE = (Select DESIG_CODE from
EMPLOYEE where Basic=(Select MAX(BASIC) from Employee))
```

DESIG_DESE
Manager

Q13) **Find the no. of managers in each department.**

```
SELECT COUNT(E.EMP_CODE) FROM EMPLOYEE E, DESIGNATION DES WHERE E.DESIG_CODE =
DES.DESIG_CODE AND DES.DESIG_DESC="MANAGER" GROUP BY E.DEPT_CODE ;
```

Q14) **Find the maximum basic from EMP table without using MAX().**

```
select DEPT_CODE, Sum(Basic) from employee  where ROWNUM=1 group by DEPT_CODE
```

DEPT_CODE	SUM(BASIC)
DP01	80000

Q15) **Find the minimum basic from EMP table without using MIN().**

```
SELECT BASIC MAX_BASIC FROM EMPLOYEE ORDER BY BASIC LIMIT 1;
```

Q16) **Find the name of the department with highest total basic. Do the same for highest average basic and maximum no. of employee.**

```
SQL: SELECT E.BASIC, DEP.DEPT_NAME FROM EMPLOYEE E, DEPARTMENT DEP WHERE E.DEPT_CODE =
DEP.DEPT_CODE ORDER BY E.BASIC DESC LIMIT 1;
+----------+------------+
| BASIC    | DEPT_NAME  |
+----------+------------+
| 36750.00 | PRODUCTION |
+----------+------------+
SQL: SELECT AVG(E.BASIC) AVG_BASIC, DEP.DEPT_NAME FROM EMPLOYEE E, DEPARTMENT DEP WHERE
E.DEPT_CODE = DEP.DEPT_CODE GROUP BY DEP.DEPT_CODE ORDER BY AVG_BASIC DESC  LIMIT 1;
+--------------+------------+
| AVG_BASIC    | DEPT_NAME  |
+--------------+------------+
```

```
| 36750.000000 | PRODUCTION |
+--------------+------------+
SQL: SELECT COUNT(E.EMP_CODE) COUNT_EMPLOYEE, DEP.DEPT_NAME FROM EMPLOYEE E, DEPARTMENT DEP
WHERE E.DEPT_CODE = DEP.DEPT_CODE GROUP BY DEP.DEPT_CODE ORDER BY COUNT_EMPLOYEE DESC LIMIT
1;
+----------------+-----------+
| COUNT_EMPLOYEE | DEPT_NAME |
+----------------+-----------+
|              1 | PURCHASE  |
+----------------+-----------+
```

Q17) **Insert same rows into EMP table with designation code not existing in DESIGNATION table.**

```
INSERT INTO EMPLOYEE VALUES('EMP011', 'Jaidip Sarkar', 'DEPT004', 'DESIG008', 'M',
       'Nowhere,  Bullygaunje', 'Kolkata', 'West Bengal', '325801', 116000.0, "2003-02-
       19");
       INSERT INTO EMPLOYEE VALUES('EMP012', 'Joydipto Biswas, 'DEPT001', 'DESIG010', 'F',
       'Anywhere, Garia', 'Kolkata', 'West Bengal', '700112', 80000.0, "1999-10-17");
```

Q18) **Delete the rows from EMP table with invalid DESIG_CODE.**

```
SQL: DELETE FROM EMPLOYEE WHERE DESIG_CODE NOT IN (SELECT DESIG_CODE FROM DESIGNATION);
```

Q19) **Find the name of the female employees with basic greater than the average basic of their respective department.**

```
SQL:   SELECT e.emp_name FROM EMPLOYEE e
       JOIN DEPARTMENT d ON e.dept_code = d.dept_code
       JOIN DESIGNATION des ON e.desig_code = des.desig_code
       WHERE e.sex = 'F' AND e.basic > (
         SELECT AVG(basic) FROM EMPLOYEE
         WHERE dept_code = e.dept_code
       );
```

Q20) **Find the number of female managers.**

```
SQL:   SELECT COUNT(*) AS female_managers
       FROM EMPLOYEE e
       JOIN DESIGNATION des ON e.desig_code = des.desig_code
       WHERE e.sex = 'F' AND des.desig_desc = 'Manager' ;
+-----------------+
| female_managers |
+-----------------+
|               1 |
+-----------------+
```

Assignment 3

Q1) **In an organization, number of departments exists. Each department has a name & unique code. Number of employees work in each department. Each employee has unique employee code. Detailed information**
like name, address, city, basic, date of join are also stored. In a leave register for each employee leave records are kept showing leave type (CL/EL/ML etc.), from-date and to-date. When an employee retires or resigns then all the leave information pertaining to him are also deleted. Basic salary must be within Rs.5000 to Rs.9000. A department can not be deleted if any employee record refers to it. Valid grades are A/B/C. Employee name must be in uppercase only. Default value for joining date is system date.
Design & implement the tables with necessary constraints to support the scenario depicted above.

```
-- Table for Department
```

```
CREATE TABLE DEPARTMENT (
  dcode varchar(10) primary key,
  dept_name varchar(30)
);
CREATE TABLE EMPLOYEE (
  ecode varchar(10) primary key,
  dcode varchar(10) not null,
  name varchar(30) check (name = UPPER(name)),
  address varchar(40),
  city varchar(20),
  basic decimal(9,2) check (basic >= 5000 AND basic <=9000),
  grades char(1) check(grades in ('A', 'B', 'C')),
  doj datetime default CURRENT_TIMESTAMP,
  foreign key(dcode) references DEPARTMENT(dcode)
);
CREATE TABLE LEAVE_REG (
  ecode varchar(10) not null,
  leave_type char(4) check (leave_type in ('CL','El','ML')),
  from_date date,
  to_date date,
  foreign key(ecode) references employee(ecode) on delete cascade
);

INSERT INTO DEPARTMENT VALUES (1, 'Sales');
INSERT INTO DEPARTMENT VALUES (2, 'Marketing');
INSERT INTO DEPARTMENT VALUES (3, 'Human Resources');

INSERT INTO employee (ecode, name, dcode, address, city, basic, grades)
VALUES (2, 'El', '2023-06-01', '2023-06-05');
INSERT INTO leave_reg (ecode, leave_type, from_date, to_date)
VALUES (3, 'ML', '2023-06-10', '2023-06-20');

INSERT INTO employee (ecode, name, dcode, address, city, basic, doj, grades)
VALUES (2, 'Jane Smith', 2, '456 Elm Avenue', 'Los Angeles', 8000.00, '2023-05-28
10:30:00', 'B');
INSERT INTO employee (ecode, name, dcode, address, city, basic, grades)
VALUES (3, 'Sarah Johnson', 1, '789 Oak Lane', 'Chicago', 5000.00, 'C');
INSERT INTO employee (ecode, name, dcode, address, city, basic, doj, grades)
VALUES (4, 'Michael Brown', 2, '987 Pine Street', 'San Francisco', 7500.00, '2022-12-15
09:00:00', 'A');
INSERT INTO employee (ecode, name, dcode, address, city, basic, grades)
VALUES (5, 'Emily Wilson', 3, '654 Maple Avenue', 'Seattle', 8500.00, 'B');
INSERT INTO employee (ecode, name, dcode, address, city, basic, grades)
VALUES (6, 'Suman Ghosh', 1, '123 Park Street', 'Kolkata', 6000.00, 'A');

INSERT INTO leave_reg (ecode, leave_type, from_date, to_date)
VALUES (1, 'CL', '2023-05-28', '2023-05-29');
INSERT INTO leave_reg (ecode, leave_type, from_date, to_date)
```

Q3) **a) create a view showing employee code, name, dcode & Basic For a particular department.**
b)Try to ensure a row into the view with valid department & also with invalid ones.
c)Find the newly inserted row in the table From which view was created .
d) Try to increment basic by Rs.100/-
e) Check it in the original table.
f) Delete the view.

```
create view emp_view as select ecode,name,dcode,basic from employee where dcode=1;
update emp_view set basic=basic+100;
select * from emp_view;
```

```
+-------+---------------+-------+---------+
| ecode | name          | dcode | basic   |
+-------+---------------+-------+---------+
| 1     | John Doe      | 1     | 7100.00 |
| 3     | Sarah Johnson | 1     | 5100.00 |
| 6     | Suman Ghosh   | 1     | 6100.00 |
+-------+---------------+-------+---------+
drop view emp_view;
```

Q4) **a) create a view Showing empcode, name, deptname, basic, leave type, From date & to date.**
b)Try to insert a row in the view. Check what happens?
c) Try to increment basic by Rs.100.
d) Delete the view.

```
create view emp_view as select e.ecode,dept_name,basic,leave_type,from_date,to_date from
employee e
join department d on e.dcode=d.dcode join leave_reg lr on lr.ecode=e.ecode;
select * from emp_view;
+-------+-----------+---------+------------+------------+------------+
| ecode | dept_name | basic   | leave_type | from_date  | to_date    |
+-------+-----------+---------+------------+------------+------------+
| 1     | Sales     | 7100.00 | CL         | 2023-05-28 | 2023-05-29 |
| 3     | Sales     | 5100.00 | ML         | 2023-06-10 | 2023-06-20 |
| 2     | Marketing | 8000.00 | El         | 2023-06-01 | 2023-06-05 |
+-------+-----------+---------+------------+------------+------------+
drop view emp_view;
```

Q5) **a) Create a table having empcode , Name, deptname, & basic From the existing tables along with the**
records of the employee who are in a particular department (say, d1) and with a basic Rs. 7000/-
b) From the existing table, add the employees with the basic salary greater than or equal to 7000/-
c) Alter the table to add a net pay column.
d) Replace net pay with 1.5* Basic.
e) Try to remove the net pay column.

```
SQL: create table emp as(
select ecode,name,dept_name,basic from employee e
join department d on e.dcode=d.dcode
where e.dcode=1 and basic>=7000
);
alter table emp
add column net_pay float8 not null;

update emp set net_pay=basic*1.5;
```

Q6) **Drop all the tables that you have created.**

```
SQL:
Drop table department;
Drop table  emp;
Drop table employee;
Drop table leave_reg;
```

Assignment 4

Q1) **a) Create EMP table with ECODE(primary key), ENAME, DCODE, GRADE, BASIC & JN-DT as the columns.[except BASIC & JN-DT,all columns are of char type and site of Grade is 1.]**

```
CREATE TABLE EMP (
        ECODE INT PRIMARY KEY,
        ENAME VARCHAR(50),
        DCODE CHAR(1),
        GRADE CHAR(1),
        BASIC FLOAT,
        JN_DT DATE      );
```

b) **Insert number of rows.**

```
SQL: INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (1, 'John Doe', 'A',
'B', 5000.00, '2023-01-01');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (2, 'Jane Smith', 'B',
'C', 7000.00, '2023-02-15');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (3, 'Michael Johnson',
'A', 'A', 9000.00, '2023-03-10');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (4, 'Chompers Ike', 'B',
'A', 1900.00, '2022-12-05');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (5, 'Emily Johnson', 'A',
'B', 5500.00, '2023-04-20');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (6, 'Robert Williams',
'B', 'C', 7200.00, '2023-05-10');
INSERT INTO EMP (ECODE, ENAME, DCODE, GRADE, BASIC, JN_DT) VALUES (7, 'Sarah Davis', 'A',
'A', 8500.00, '2023-06-01');
```

Q2) **change the column heading as shown below, So that in subsequent SELECT statement newly set**
heading will be shown:

ECODE	**EMPLOYEE CODE**
ENAME	**NAME**
DCODE	**DEPT.CODE**
JN-DT	**JONING DATE**

```
SQL: ALTER TABLE EMP RENAME COLUMN ECODE TO EMPLOYEE_CODE;
ALTER TABLE EMP RENAME COLUMN ENAME TO NAME;
ALTER TABLE EMP RENAME COLUMN DCODE TO DEPARTMENT_CODE;
ALTER TABLE EMP RENAME COLUMN `JN_DT` TO JOINING_DATE;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| EMPLOYEE_CODE    | int          | NO   | PRI | NULL    |       |
| NAME             | varchar(50)  | YES  |     | NULL    |       |
| DEPARTMENT_CODE  | char(1)      | YES  |     | NULL    |       |
| GRADE            | char(1)      | YES  |     | NULL    |       |
| BASIC            | float        | YES  |     | NULL    |       |
| JOINING_DATE     | date         | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
```

Q3) **Set the format of columns as mentioned below, So that in subsequent SELECT statement ,values**
appear in the specified format:

- ***format of BASIC is such that a value of 7000 will be shown as7,000**
- ***Format of GRADE will be such that full column name appears in the display.**
- ***For JN-DT format is such that 01-JAN-00 will be shown as JANURY 01,2000**

```
SELECT
  EMPLOYEE_CODE AS 'EMPLOYEE CODE',  NAME AS 'NAME',
  DEPARTMENT_CODE AS 'DEPT.CODE',  CONCAT('Grade: ', GRADE) AS 'GRADE',
  FORMAT(BASIC, 0) AS 'BASIC',
  DATE_FORMAT(JOINING_DATE, '%M %d, %Y') AS 'JOINING DATE' FROM EMP;
```

EMPLOYEE CODE	NAME	DEPT.CODE	GRADE	BASIC	JOINING DATE
1	John Doe	A	Grade: B	5,000	January 01, 2023
2	Jane Smith	B	Grade: C	7,000	February 15, 2023
3	Michael Johnson	A	Grade: A	9,000	March 10, 2023
4	Chompers Ike	B	Grade: A	1,900	December 05, 2022
5	Emily Johnson	A	Grade: B	5,500	April 20, 2023
6	Robert Williams	B	Grade: C	7,200	May 10, 2023
7	Sarah Davis	A	Grade: A	8,500	June 01, 2023

Q4) **a) Show the display attributes of all the columns.**

SQL: SHOW CREATE TABLE EMP;

Table	Create Table
EMP	CREATE TABLE `emp` (`EMPLOYEE_CODE` int NOT NULL, `NAME` varchar(50) DEFAULT NULL, `DEPARTMENT_CODE` char(1) DEFAULT NULL, `GRADE` char(1) DEFAULT NULL, `BASIC` float DEFAULT NULL, `JOINING_DATE` date DEFAULT NULL, PRIMARY KEY (`EMPLOYEE_CODE`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

b) Show the display attributes of particular column.

SQL: DESCRIBE EMP NAME;

Field	Type	Null	Key	Default	Extra
NAME	varchar(50)	YES		NULL	

c) Suppress the newly set attributes of JN-DT .Try a select statement.

SQL: ALTER TABLE EMP MODIFY COLUMN JOINING_DATE DATE;

d) Reset the newly set attributes of JN-DT

SQL: ALTER TABLE EMP MODIFY COLUMN JOINING_DATE DATE FORMAT 'YYYY-MM-DD';

e) Reset the newly set attributes of all columns.

```
SQL:    ALTER TABLE EMP MODIFY COLUMN EMPLOYEE_CODE INT;
        ALTER TABLE EMP MODIFY COLUMN NAME VARCHAR(50);
        ALTER TABLE EMP MODIFY COLUMN DEPARTMENT_CODE CHAR(1);
        ALTER TABLE EMP MODIFY COLUMN GRADE CHAR(1);
        ALTER TABLE EMP MODIFY COLUMN BASIC FLOAT;
        ALTER TABLE EMP MODIFY COLUMN JOINING_DATE DATE;
```

f) Shown the display attributes of all columns.

SQL: DESCRIBE EMP;

Field	Type	Null	Key	Default	Extra
EMPLOYEE_CODE	int	NO	PRI	NULL	
NAME	varchar(50)	YES		NULL	
DEPARTMENT_CODE	char(1)	YES		NULL	
GRADE	char(1)	YES		NULL	

```
| BASIC           | float       | YES  |     | NULL    |       |
| JOINING_DATE    | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```