

ASSIGNMENT 12

Hotel (Hotel_No, Name, Address)

Room (Room_No, Hotel_No, Type, Price)

Booking (Hotel_No, Guest_No, Date_From, Date_To, Room_No)

Guest (Guest_No, Name, Address)

```
CREATE TABLE HOTEL (  
    HOTEL_NO NUMBER PRIMARY KEY,  
    NAME VARCHAR2(20),  
    ADDRESS VARCHAR2(20)  
);
```

```
SQL> CREATE TABLE HOTEL (  
2     HOTEL_NO NUMBER PRIMARY KEY,  
3     NAME VARCHAR2(20),  
4     ADDRESS VARCHAR2(20)  
5 );
```

Table created.

```
SQL> DESC HOTEL;
```

| Name | Null? | Type |
|----------|----------|--------------|
| HOTEL_NO | NOT NULL | NUMBER |
| NAME | | VARCHAR2(20) |
| ADDRESS | | VARCHAR2(20) |

```
SQL> █
```

```
CREATE TABLE ROOM (  
    ROOM_NO NUMBER PRIMARY KEY,  
    HOTEL_NO NUMBER,  
    TYPE VARCHAR2(50),  
    PRICE NUMBER(10, 2),  
    CONSTRAINT RFK1 FOREIGN KEY (HOTEL_NO) REFERENCES  
HOTEL(HOTEL_NO) ON DELETE CASCADE  
);
```

```
SQL> CREATE TABLE ROOM (
2     ROOM_NO NUMBER PRIMARY KEY,
3     HOTEL_NO NUMBER,
4     TYPE VARCHAR2(50),
5     PRICE NUMBER(10, 2),
6     CONSTRAINT RFK1 FOREIGN KEY (HOTEL_NO) REFERENCES HOTEL(HOTEL_NO) ON DELETE CASCADE
7 );
```

Table created.

```
SQL> DESC ROOM;
```

| Name | Null? | Type |
|----------|----------|--------------|
| ROOM_NO | NOT NULL | NUMBER |
| HOTEL_NO | | NUMBER |
| TYPE | | VARCHAR2(50) |
| PRICE | | NUMBER(10,2) |

```
SQL> █
```

```
CREATE TABLE BOOKING (
    HOTEL_NO NUMBER,
    GUEST_NO NUMBER,
    DATE_FROM DATE,
    DATE_TO DATE,
    ROOM_NO NUMBER,
    CONSTRAINT BKFK1 FOREIGN KEY (HOTEL_NO) REFERENCES
HOTEL(HOTEL_NO) ON DELETE CASCADE,
    CONSTRAINT BKFK2 FOREIGN KEY (GUEST_NO) REFERENCES
GUEST(GUEST_NO) ON DELETE CASCADE,
    CONSTRAINT BKFK3 FOREIGN KEY (ROOM_NO) REFERENCES
ROOM(ROOM_NO) ON DELETE CASCADE
);
```

```
SQL> CREATE TABLE BOOKING (
2     HOTEL_NO NUMBER,
3     GUEST_NO NUMBER,
4     DATE_FROM DATE,
5     DATE_TO DATE,
6     ROOM_NO NUMBER,
7     CONSTRAINT BKFK1 FOREIGN KEY (HOTEL_NO) REFERENCES HOTEL(HOTEL_NO) ON DELETE CASCADE,
8     CONSTRAINT BKFK2 FOREIGN KEY (GUEST_NO) REFERENCES GUEST(GUEST_NO) ON DELETE CASCADE,
9     CONSTRAINT BKFK3 FOREIGN KEY (ROOM_NO) REFERENCES ROOM(ROOM_NO) ON DELETE CASCADE
10 );
```

Table created.

```
SQL> DESC BOOKING;
```

| Name | Null? | Type |
|-----------|-------|--------|
| HOTEL_NO | | NUMBER |
| GUEST_NO | | NUMBER |
| DATE_FROM | | DATE |
| DATE_TO | | DATE |
| ROOM_NO | | NUMBER |

```
SQL> █
```

```
CREATE TABLE GUEST (
    GUEST_NO NUMBER PRIMARY KEY,
    NAME VARCHAR2(20),
    ADDRESS VARCHAR2(20)
);
```

```
SQL> CREATE TABLE GUEST (
2     GUEST_NO NUMBER PRIMARY KEY,
3     NAME VARCHAR2(20),
4     ADDRESS VARCHAR2(20)
5 );
```

Table created.

```
SQL> DESC GUEST;
```

| Name | Null? | Type |
|----------|----------|--------------|
| GUEST_NO | NOT NULL | NUMBER |
| NAME | | VARCHAR2(20) |
| ADDRESS | | VARCHAR2(20) |

```
SQL> █
```

INSERT ALL

INTO HOTEL VALUES (1, 'Hotel A', 'Address A')

INTO HOTEL VALUES (2, 'Hotel B', 'Address B')

INTO HOTEL VALUES (3, 'Hotel C', 'Address C')

SELECT * FROM DUAL;

```
SQL> INSERT ALL
2     INTO HOTEL VALUES (1, 'Hotel A', 'Address A')
3     INTO HOTEL VALUES (2, 'Hotel B', 'Address B')
4     INTO HOTEL VALUES (3, 'Hotel C', 'Address C')
5     SELECT * FROM DUAL;
```

3 rows created.

```
SQL> SELECT * FROM HOTEL;
```

| HOTEL_NO | NAME | ADDRESS |
|----------|---------|-----------|
| 1 | Hotel A | Address A |
| 2 | Hotel B | Address B |
| 3 | Hotel C | Address C |

```
SQL> █
```

INSERT ALL

INTO ROOM VALUES (101, 1, 'Single', 100.00)

INTO ROOM VALUES (102, 1, 'Double', 150.00)

INTO ROOM VALUES (103, 2, 'Single', 120.00)

INTO ROOM VALUES (104, 2, 'Double', 180.00)

INTO ROOM VALUES (201, 3, 'Single', 110.00)

INTO ROOM VALUES (202, 3, 'Double', 160.00)

SELECT * FROM DUAL;

```
SQL> SELECT * FROM ROOM;
```

| ROOM_NO | HOTEL_NO | TYPE | PRICE |
|---------|----------|--------|-------|
| 101 | 1 | Single | 100 |
| 102 | 1 | Double | 150 |
| 103 | 2 | Single | 120 |
| 104 | 2 | Double | 180 |
| 201 | 3 | Single | 110 |
| 202 | 3 | Double | 160 |

6 rows selected.

```
SQL> █
```

INSERT ALL

 INTO GUEST VALUES (1, 'John', 'Address 1')

 INTO GUEST VALUES (2, 'Jane', 'Address 2')

 INTO GUEST VALUES (3, 'Alice', 'Address 3')

SELECT * FROM DUAL;

```
SQL> INSERT ALL
```

```
2      INTO GUEST VALUES (1, 'John', 'Address 1')
```

```
3      INTO GUEST VALUES (2, 'Jane', 'Address 2')
```

```
4      INTO GUEST VALUES (3, 'Alice', 'Address 3')
```

```
5  SELECT * FROM DUAL;
```

3 rows created.

```
SQL> SELECT * FROM GUEST;
```

| GUEST_NO | NAME | ADDRESS |
|----------|-------|-----------|
| 1 | John | Address 1 |
| 2 | Jane | Address 2 |
| 3 | Alice | Address 3 |

```
SQL> █
```

INSERT ALL

 INTO BOOKING VALUES (1, 1, TO_DATE('2024-03-27', 'YYYY-MM-DD'),
TO_DATE('30-03-2024', 'DD-MM-YYYY'), 101)

 INTO BOOKING VALUES (2, 2, TO_DATE('2024-04-01', 'YYYY-MM-DD'),
TO_DATE('05-04-2024', 'DD-MM-YYYY'), 104)

 INTO BOOKING VALUES (3, 3, TO_DATE('2024-04-10', 'YYYY-MM-DD'),
TO_DATE('15-04-2024', 'DD-MM-YYYY'), 201)

SELECT * FROM DUAL;

```

SQL> INSERT ALL
2 INTO BOOKING VALUES (1, 1, TO_DATE('2024-03-27', 'YYYY-MM-DD'), TO_DATE('30-03-2024', 'DD-MM-YYYY'), 101)
3 INTO BOOKING VALUES (2, 2, TO_DATE('2024-04-01', 'YYYY-MM-DD'), TO_DATE('05-04-2024', 'DD-MM-YYYY'), 104)
4 INTO BOOKING VALUES (3, 3, TO_DATE('2024-04-10', 'YYYY-MM-DD'), TO_DATE('15-04-2024', 'DD-MM-YYYY'), 201)
5 SELECT * FROM DUAL;

3 rows created.

SQL> SELECT * FROM BOOKING ;

```

| HOTEL_NO | GUEST_NO | DATE_FROM | DATE_TO | ROOM_NO |
|----------|----------|-----------|-----------|---------|
| 1 | 1 | 27-MAR-24 | 30-MAR-24 | 101 |
| 2 | 2 | 01-APR-24 | 05-APR-24 | 104 |
| 3 | 3 | 10-APR-24 | 15-APR-24 | 201 |

```

SQL> █

```

Populate the tables Answer the following query using SQL.

1. List the names and addresses of all guests in London, alphabetically ordered by name

SELECT NAME, ADDRESS FROM GUEST WHERE ADDRESS LIKE '%LONDON%'
ORDER BY NAME;

```
SQL> UPDATE GUEST SET ADDRESS = 'LONDON' WHERE GUEST_NO = 2;
```

1 row updated.

```

SQL> SELECT NAME, ADDRESS
2 FROM GUEST
3 WHERE ADDRESS LIKE '%LONDON%'
4 ORDER BY NAME;

```

| NAME | ADDRESS |
|------|---------|
| Jane | LONDON |

```
SQL> █
```

2. List all double or family rooms with a price below £40.00 per night, in ascending order of price.

SELECT * FROM ROOM WHERE TYPE IN ('Double', 'Family') AND PRICE < 40.00
ORDER BY PRICE ASC;

```

SQL> SELECT *
2 FROM ROOM
3 WHERE TYPE IN ('Double', 'Family') AND PRICE < 40.00
4 ORDER BY PRICE ASC;

```

no rows selected

```
SQL> █
```

3. List the bookings for which no date_to has been specified.

SELECT * FROM BOOKING WHERE DATE_TO IS NULL;

```
SQL> SELECT *  
  2   FROM BOOKING  
  3   WHERE DATE_TO IS NULL;  
  
no rows selected  
  
SQL> █
```

4. How many hotels are there?

SELECT COUNT(*) AS TOTAL_HOTELS FROM HOTEL;

```
SQL> SELECT COUNT(*) AS TOTAL_HOTELS  
  2   FROM HOTEL;  
  
TOTAL_HOTELS  
-----  
              3  
  
SQL> █
```

5. What is the average price of a room?

SELECT AVG(PRICE) AS AVERAGE_PRICE FROM ROOM;

```
SQL> SELECT AVG(PRICE) AS AVERAGE_PRICE  
  2   FROM ROOM;  
  
AVERAGE_PRICE  
-----  
    136.666667  
  
SQL> █
```

6. What is the total revenue per night from all double rooms?

SELECT SUM(PRICE) AS TOTAL_REVENUE FROM ROOM WHERE TYPE =
'Double';

```
SQL> SELECT SUM(PRICE) AS TOTAL_REVENUE
2 FROM ROOM
3 WHERE TYPE = 'Double';
```

```
TOTAL_REVENUE
-----
490
```

```
SQL> █
```

7.How many different guests have made bookings for August?

```
SELECT COUNT(DISTINCT GUEST_NO) AS DISTINCT_GUESTS FROM
BOOKING WHERE DATE_FROM >= TO_DATE('2024-08-01', 'YYYY-MM-DD') AND
DATE_FROM < TO_DATE('2024-09-01', 'YYYY-MM-DD');
```

```
SQL> SELECT COUNT(DISTINCT GUEST_NO) AS DISTINCT_GUESTS
2 FROM BOOKING
3 WHERE DATE_FROM >= TO_DATE('2024-08-01', 'YYYY-MM-DD') AND DATE_FROM < TO_DATE('2024-09-01', 'YYYY-MM-DD');
```

```
DISTINCT_GUESTS
-----
0
```

```
SQL> █
```

8.List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

```
SELECT R.*, G.NAME AS GUEST_NAME FROM ROOM R LEFT JOIN BOOKING B
ON R.ROOM_NO = B.ROOM_NO LEFT JOIN GUEST G ON B.GUEST_NO =
G.GUEST_NO WHERE R.HOTEL_NO = (SELECT HOTEL_NO FROM HOTEL
WHERE NAME = 'Grosvenor Hotel');
```

```
SQL> SELECT R.*, G.NAME AS GUEST_NAME
2 FROM ROOM R
3 LEFT JOIN BOOKING B ON R.ROOM_NO = B.ROOM_NO
4 LEFT JOIN GUEST G ON B.GUEST_NO = G.GUEST_NO
5 WHERE R.HOTEL_NO = (SELECT HOTEL_NO FROM HOTEL WHERE NAME = 'Grosvenor Hotel');
```

```
no rows selected
```

```
SQL> █
```

9.What is the total income from bookings for the Grosvenor Hotel today?

```
SELECT SUM(PRICE) AS TOTAL_INCOME FROM ROOM
WHERE HOTEL_NO = (SELECT HOTEL_NO FROM HOTEL WHERE NAME =
'Grosvenor Hotel') AND ROOM_NO IN (SELECT ROOM_NO FROM BOOKING
WHERE DATE_FROM <= SYSDATE AND DATE_TO >= SYSDATE);
```



```

SQL> SELECT SUM(PRICE) AS TOTAL_INCOME
2 FROM ROOM
3 WHERE HOTEL_NO = (SELECT HOTEL_NO FROM HOTEL WHERE NAME = 'Grosvenor Hotel')
4 AND ROOM_NO IN (SELECT ROOM_NO FROM BOOKING WHERE DATE_FROM <= SYSDATE AND DATE_TO >= SYSDATE);

TOTAL_INCOME
-----
SQL> █

```

10. List the rooms that are currently unoccupied at the Grosvenor Hotel.

```

SELECT ROOM_NO FROM ROOM WHERE HOTEL_NO = (SELECT HOTEL_NO
FROM HOTEL WHERE NAME = 'Grosvenor Hotel') AND ROOM_NO NOT IN
(SELECT ROOM_NO FROM BOOKING WHERE DATE_FROM <= SYSDATE AND
DATE_TO >= SYSDATE);

```

```

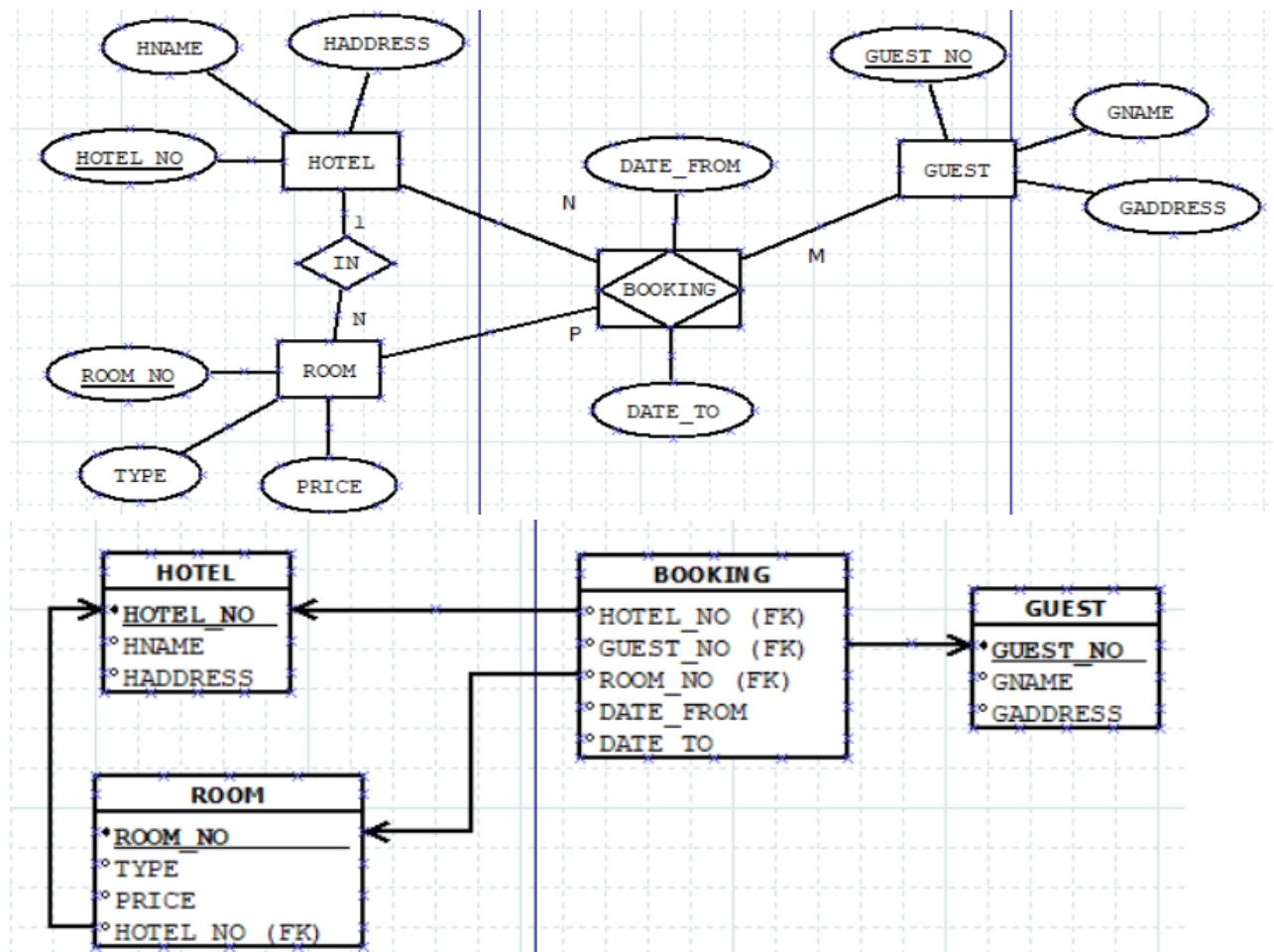
SQL> SELECT ROOM_NO
2 FROM ROOM
3 WHERE HOTEL_NO = (SELECT HOTEL_NO FROM HOTEL WHERE NAME = 'Grosvenor Hotel')
4 AND ROOM_NO NOT IN (SELECT ROOM_NO FROM BOOKING WHERE DATE_FROM <= SYSDATE AND DATE_TO >= SYSDATE);

no rows selected

SQL> █

```

Design an ER Model for an application where hotels are booked by guests wanting to go on a holiday in India or abroad. Your design should meet all requirements. Map into a relational model.



EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)
 DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)
 DLOCATION (DNo,DLoc)
 PROJECT (PNo, PName, PLocation, DNo)
 WORKS_ON (SSN, PNo, Hours)

CREATE TABLE EMPLOYEE (SSN VARCHAR2(10) PRIMARY KEY, NAME VARCHAR2(50), ADDRESS VARCHAR2(100), SEX CHAR(1), SALARY DECIMAL(10, 2), SUPERSSN VARCHAR2(10), DNO NUMBER, CONSTRAINT EMPLFK2 FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE);

```
SQL> CREATE TABLE EMPLOYEE (
2     SSN VARCHAR2(10) PRIMARY KEY,
3     NAME VARCHAR2(50),
4     ADDRESS VARCHAR2(100),
5     SEX CHAR(1),
6     SALARY DECIMAL(10, 2),
7     SUPERSSN VARCHAR2(10),
8     DNO NUMBER,
9     CONSTRAINT EMPLFK2 FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE
10 );
```

Table created.

```
SQL> DESC EMPLOYEE;
```

| Name | Null? | Type |
|----------|----------|---------------|
| SSN | NOT NULL | VARCHAR2(10) |
| NAME | | VARCHAR2(50) |
| ADDRESS | | VARCHAR2(100) |
| SEX | | CHAR(1) |
| SALARY | | NUMBER(10,2) |
| SUPERSSN | | VARCHAR2(10) |
| DNO | | NUMBER |

```
SQL> █
```

CREATE TABLE DEPARTMENT (DNO NUMBER PRIMARY KEY, DNAME VARCHAR2(50), MGRSSN VARCHAR2(10), MGRSTARTDATE DATE);

```
SQL> CREATE TABLE DEPARTMENT (
2     DNO NUMBER PRIMARY KEY,
3     DNAME VARCHAR2(50),
4     MGRSSN VARCHAR2(10),
5     MGRSTARTDATE DATE
6 );
```

Table created.

```
SQL> DESC DEPARTMENT;
```

| Name | Null? | Type |
|--------------|----------|--------------|
| DNO | NOT NULL | NUMBER |
| DNAME | | VARCHAR2(50) |
| MGRSSN | | VARCHAR2(10) |
| MGRSTARTDATE | | DATE |

```
SQL> █
```

```
CREATE TABLE DLOCATION (DNO NUMBER, DLOC VARCHAR2(100),
CONSTRAINT DLPK1 PRIMARY KEY (DNO, DLOC), CONSTRAINT DLFK1
FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE
CASCADE);
```

```
SQL> CREATE TABLE DLOCATION (
2     DNO NUMBER,
3     DLOC VARCHAR2(100),
4     CONSTRAINT DLPK1 PRIMARY KEY (DNO, DLOC),
5     CONSTRAINT DLFK1 FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE
6 );
```

Table created.

```
SQL> DESC DLOCATION;
```

| Name | Null? | Type |
|------|----------|---------------|
| DNO | NOT NULL | NUMBER |
| DLOC | NOT NULL | VARCHAR2(100) |

```
SQL> █
```

```
CREATE TABLE PROJECT ( PNO NUMBER PRIMARY KEY, PNAME
VARCHAR2(100), PLOCATION VARCHAR2(100), DNO NUMBER, CONSTRAINT
PRJFK1 FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE
CASCADE);
```

```
SQL> CREATE TABLE PROJECT (
2     PNO NUMBER PRIMARY KEY,
3     PNAME VARCHAR2(100),
4     PLOCATION VARCHAR2(100),
5     DNO NUMBER,
6     CONSTRAINT PRJFK1 FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE
7 );
```

Table created.

```
SQL> DESC PROJECT;
```

| Name | Null? | Type |
|-----------|----------|---------------|
| PNO | NOT NULL | NUMBER |
| PNAME | | VARCHAR2(100) |
| PLOCATION | | VARCHAR2(100) |
| DNO | | NUMBER |

```
SQL> █
```

```
CREATE TABLE WORKS_ON (SSN VARCHAR2(10), PNO NUMBER, HOURS
DECIMAL(5, 2), CONSTRAINT WOFK1 FOREIGN KEY (SSN) REFERENCES
EMPLOYEE(SSN) ON DELETE CASCADE, CONSTRAINT WOFK2 FOREIGN KEY
(PNO) REFERENCES PROJECT(PNO) ON DELETE CASCADE);
```

```
SQL> CREATE TABLE WORKS_ON (
2     SSN VARCHAR2(10),
3     PNO NUMBER,
4     HOURS DECIMAL(5, 2),
5     CONSTRAINT WOFK1 FOREIGN KEY (SSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE,
6     CONSTRAINT WOFK2 FOREIGN KEY (PNO) REFERENCES PROJECT(PNO) ON DELETE CASCADE
7 );
```

Table created.

```
SQL> DESC WORKS_ON;
```

| Name | Null? | Type |
|-------|-------|--------------|
| SSN | | VARCHAR2(10) |
| PNO | | NUMBER |
| HOURS | | NUMBER(5,2) |

```
SQL> █
```

INSERT ALL

```
    INTO EMPLOYEE VALUES ('1111111111', 'John Doe', '123 Main St', 'M', 50000,
NULL, 1)
```

```
    INTO EMPLOYEE VALUES ('2222222222', 'Alice Smith', '456 Elm St', 'F', 60000,
'1111111111', 2)
```

```
    INTO EMPLOYEE VALUES ('3333333333', 'Bob Johnson', '789 Oak St', 'M', 55000,
NULL, 1)
```

```
    INTO EMPLOYEE VALUES ('4444444444', 'Jane Doe', '101 Pine St', 'F', 65000,
'1111111111', 2)
```

```
    INTO EMPLOYEE VALUES ('5555555555', 'Chris Brown', '202 Maple St', 'M',
70000, NULL, 3)
```

```
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
2     INTO EMPLOYEE VALUES ('1111111111', 'John Doe', '123 Main St', 'M', 50000, NULL, 1)
3     INTO EMPLOYEE VALUES ('2222222222', 'Alice Smith', '456 Elm St', 'F', 60000, '1111111111', 2)
4     INTO EMPLOYEE VALUES ('3333333333', 'Bob Johnson', '789 Oak St', 'M', 55000, NULL, 1)
5     INTO EMPLOYEE VALUES ('4444444444', 'Jane Doe', '101 Pine St', 'F', 65000, '1111111111', 2)
6     INTO EMPLOYEE VALUES ('5555555555', 'Chris Brown', '202 Maple St', 'M', 70000, NULL, 3)
7     SELECT * FROM DUAL;
```

5 rows created.

```
SQL> SELECT * FROM EMPLOYEE;
```

| SSN | NAME | ADDRESS | S | SALARY | SUPERSSN | DNO |
|------------|-------------|-------------|---|--------|------------|-----|
| 1111111111 | John Doe | 123 Main St | M | 50000 | | 1 |
| 2222222222 | Alice Smith | 456 Elm St | F | 60000 | 1111111111 | 2 |
| 3333333333 | Bob Johnson | 789 Oak St | M | 55000 | | 1 |

| SSN | NAME | ADDRESS | S | SALARY | SUPERSSN | DNO |
|------------|-------------|--------------|---|--------|------------|-----|
| 4444444444 | Jane Doe | 101 Pine St | F | 65000 | 1111111111 | 2 |
| 5555555555 | Chris Brown | 202 Maple St | M | 70000 | | 3 |

INSERT ALL

```
    INTO DEPARTMENT VALUES (1, 'IT', '1111111111', TO_DATE('2022-01-01',
'YYYY-MM-DD'))
```

```
    INTO DEPARTMENT VALUES (2, 'HR', '2222222222', TO_DATE('2022-01-01',
'YYYY-MM-DD'))
```

```
    INTO DEPARTMENT VALUES (3, 'Finance', '5555555555', TO_DATE('2022-01-01', 'YYYY-MM-DD'))
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
2     INTO DEPARTMENT VALUES (1, 'IT', '1111111111', TO_DATE('2022-01-01', 'YYYY-MM-DD'))
3     INTO DEPARTMENT VALUES (2, 'HR', '2222222222', TO_DATE('2022-01-01', 'YYYY-MM-DD'))
4     INTO DEPARTMENT VALUES (3, 'Finance', '5555555555', TO_DATE('2022-01-01', 'YYYY-MM-DD'))
5  SELECT * FROM DUAL;
```

3 rows created.

```
SQL> SELECT * FROM DEPARTMENT;
```

| DNO | DNAME | MGRSSN | MGRSTARTD |
|-----|---------|------------|-----------|
| 1 | IT | 1111111111 | 01-JAN-22 |
| 2 | HR | 2222222222 | 01-JAN-22 |
| 3 | Finance | 5555555555 | 01-JAN-22 |

```
SQL> █
```

```
INSERT ALL
    INTO DLOCATION VALUES (1, 'New York')
    INTO DLOCATION VALUES (2, 'Los Angeles')
    INTO DLOCATION VALUES (3, 'Chicago')
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
2     INTO DLOCATION VALUES (1, 'New York')
3     INTO DLOCATION VALUES (2, 'Los Angeles')
4     INTO DLOCATION VALUES (3, 'Chicago')
5  SELECT * FROM DUAL;
```

3 rows created.

```
SQL> SELECT * FROM DLOCATION;
```

| DNO | DLOC |
|-----|-------------|
| 1 | New York |
| 2 | Los Angeles |
| 3 | Chicago |

```
SQL> █
```

```
INSERT ALL
    INTO PROJECT VALUES (101, 'Project X', 'New York', 1)
    INTO PROJECT VALUES (102, 'Project Y', 'Los Angeles', 2)
    INTO PROJECT VALUES (103, 'Project Z', 'Chicago', 3)
    INTO PROJECT VALUES (104, 'Project A', 'New York', 1)
```

```

    INTO PROJECT VALUES (105, 'Project B', 'Chicago', 3)
SELECT * FROM DUAL;

```

```

SQL> INSERT ALL
2   INTO PROJECT VALUES (101, 'Project X', 'New York', 1)
3   INTO PROJECT VALUES (102, 'Project Y', 'Los Angeles', 2)
4   INTO PROJECT VALUES (103, 'Project Z', 'Chicago', 3)
5   INTO PROJECT VALUES (104, 'Project A', 'New York', 1)
6   INTO PROJECT VALUES (105, 'Project B', 'Chicago', 3)
7   SELECT * FROM DUAL;

```

5 rows created.

```
SQL> SELECT * FROM PROJECT;
```

| PNO | PNAME | PLOCATION |
|-----|-----------|-------------|
| DNO | | |
| 101 | Project X | New York |
| 1 | | |
| 102 | Project Y | Los Angeles |
| 2 | | |
| 103 | Project Z | Chicago |
| 3 | | |
| 104 | Project A | New York |
| 1 | | |
| 105 | Project B | Chicago |
| 3 | | |

```
SQL> █
```

INSERT ALL

```

    INTO WORKS_ON VALUES ('1111111111', 101, 40)
    INTO WORKS_ON VALUES ('2222222222', 102, 35)
    INTO WORKS_ON VALUES ('3333333333', 103, 30)
    INTO WORKS_ON VALUES ('4444444444', 101, 45)
    INTO WORKS_ON VALUES ('5555555555', 104, 50)

```

```
SELECT * FROM DUAL;
```

```

SQL> INSERT ALL
2   INTO WORKS_ON VALUES ('1111111111', 101, 40)
3   INTO WORKS_ON VALUES ('2222222222', 102, 35)
4   INTO WORKS_ON VALUES ('3333333333', 103, 30)
5   INTO WORKS_ON VALUES ('4444444444', 101, 45)
6   INTO WORKS_ON VALUES ('5555555555', 104, 50)
7   SELECT * FROM DUAL;

```

5 rows created.

```
SQL> SELECT * FROM WORKS_ON;
```

| SSN | PNO | HOURS |
|------------|-------|-------|
| ----- | ----- | ----- |
| 1111111111 | 101 | 40 |
| 2222222222 | 102 | 35 |
| 3333333333 | 103 | 30 |
| 4444444444 | 101 | 45 |
| 5555555555 | 104 | 50 |

```
SQL> █
```

Write SQL queries to

1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.

```
SELECT DISTINCT P.PNO
FROM PROJECT P
JOIN WORKS_ON W ON P.PNO = W.PNO
JOIN EMPLOYEE E ON W.SSN = E.SSN
WHERE E.NAME LIKE '%Scott%'
UNION
SELECT DISTINCT P.PNO
FROM PROJECT P
JOIN DEPARTMENT D ON P.DNO = D.DNO
JOIN EMPLOYEE M ON D.MGRSSN = M.SSN
WHERE M.NAME LIKE '%Scott%';
```

```
SQL> SELECT DISTINCT P.PNO
  2  FROM PROJECT P
  3  JOIN WORKS_ON W ON P.PNO = W.PNO
  4  JOIN EMPLOYEE E ON W.SSN = E.SSN
  5  WHERE E.NAME LIKE '%Scott%'
  6  UNION
  7  SELECT DISTINCT P.PNO
  8  FROM PROJECT P
  9  JOIN DEPARTMENT D ON P.DNO = D.DNO
 10  JOIN EMPLOYEE M ON D.MGRSSN = M.SSN
 11  WHERE M.NAME LIKE '%Scott%';
```

no rows selected

SQL> █

2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

```
UPDATE PROJECT SET PNAME = 'IoT' WHERE PNO = 104;
```

```
UPDATE EMPLOYEE
SET SALARY = SALARY * 1.10
WHERE SSN IN (
    SELECT W.SSN
```



```

FROM WORKS_ON W
JOIN PROJECT P ON W.PNO = P.PNO
WHERE P.PNAME = 'IoT'
);

```

```

SQL> UPDATE EMPLOYEE
2  SET SALARY = SALARY * 1.10
3  WHERE SSN IN (
4      SELECT W.SSN
5      FROM WORKS_ON W
6      JOIN PROJECT P ON W.PNO = P.PNO
7      WHERE P.PNAME = 'IoT'
8  );

```

1 row updated.

```
SQL> SELECT * FROM EMPLOYEE;
```

| SSN S | NAME SALARY SUPERSSN | DNO | ADDRESS |
|------------|-------------------------|-----|--------------|
| 1111111111 | John Doe | | 123 Main St |
| M | 50000 | 1 | |
| 2222222222 | Alice Smith | | 456 Elm St |
| F | 60000 1111111111 | 2 | |
| 3333333333 | Bob Johnson | | 789 Oak St |
| M | 55000 | 1 | |
| 4444444444 | Jane Doe | | 101 Pine St |
| F | 65000 1111111111 | 2 | |
| 5555555555 | Chris Brown | | 202 Maple St |
| M | 77000 | 3 | |

```
SQL> █
```

3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department

```
UPDATE DEPARTMENT SET DNAME = 'Accounts' WHERE DNO = 3;
```

```

SELECT SUM(SALARY) AS Total_Salary,
       MAX(SALARY) AS Max_Salary,
       MIN(SALARY) AS Min_Salary,
       AVG(SALARY) AS Avg_Salary
FROM EMPLOYEE
WHERE DNO = (
    SELECT DNO
    FROM DEPARTMENT
    WHERE DNAME = 'Accounts'
);

```

```

SQL> SELECT SUM(SALARY) AS Total_Salary,
2         MAX(SALARY) AS Max_Salary,
3         MIN(SALARY) AS Min_Salary,
4         AVG(SALARY) AS Avg_Salary
5   FROM EMPLOYEE
6  WHERE DNO = (
7      SELECT DNO
8      FROM DEPARTMENT
9      WHERE DNAME = 'Accounts'
10 );

```

| TOTAL_SALARY | MAX_SALARY | MIN_SALARY | AVG_SALARY |
|--------------|------------|------------|------------|
| 77000 | 77000 | 77000 | 77000 |

```

SQL> █

```

4.Retrieve the name of each employee who works on all the projects controlled by department number

```

SELECT DISTINCT E.NAME
FROM EMPLOYEE E
WHERE NOT EXISTS (
    SELECT P.PNO
    FROM PROJECT P
    WHERE NOT EXISTS (
        SELECT *
        FROM WORKS_ON W
        WHERE W.PNO = P.PNO AND W.SSN = E.SSN
    )
    AND P.DNO = E.DNO
);

```

```

SQL> SELECT DISTINCT E.NAME
  2   FROM EMPLOYEE E
  3   WHERE NOT EXISTS (
  4       SELECT P.PNO
  5       FROM PROJECT P
  6       WHERE NOT EXISTS (
  7           SELECT *
  8           FROM WORKS_ON W
  9           WHERE W.PNO = P.PNO AND W.SSN = E.SSN
10       )
11       AND P.DNO = E.DNO
12   );

```

NAME

 Alice Smith

SQL> █

5 (use NOT EXISTS operator).

5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.

```

SELECT DNO, COUNT(*) AS Num_Employees_Above_6_Lacs
FROM EMPLOYEE
GROUP BY DNO
HAVING COUNT(*) > 5 AND SUM(CASE WHEN SALARY > 600000 THEN 1 ELSE 0
END) > 0;

```

```

SQL> SELECT DNO, COUNT(*) AS Num_Employees_Above_6_Lacs
  2   FROM EMPLOYEE
  3   GROUP BY DNO
  4   HAVING COUNT(*) > 5 AND SUM(CASE WHEN SALARY > 600000 THEN 1 ELSE 0 END) > 0;

```

no rows selected

SQL> █

B. Write a program in PL/SQL to create a procedure to displays the GCD of nos.

```

CREATE OR REPLACE PROCEDURE Calculate_GCD(x IN NUMBER, y IN
NUMBER) AS
  num1 NUMBER := x;
  num2 NUMBER := y;
  gcd NUMBER;
BEGIN

```

```

WHILE num2 != 0 LOOP
    gcd := num1;
    num1 := num2;
    num2 := MOD(gcd, num2);
END LOOP;
DBMS_OUTPUT.PUT_LINE('GCD of ' || x || ' and ' || y || ' is ' || num1);
END;
/

```

```

SQL> CREATE OR REPLACE PROCEDURE Calculate_GCD(x IN NUMBER, y IN NUMBER) AS
2     num1 NUMBER := x;
3     num2 NUMBER := y;
4     gcd NUMBER;
5 BEGIN
6     WHILE num2 != 0 LOOP
7         gcd := num1;
8         num1 := num2;
9         num2 := MOD(gcd, num2);
10    END LOOP;
11    DBMS_OUTPUT.PUT_LINE('GCD of ' || x || ' and ' || y || ' is ' || num1);
12 END;
13 /

```

Procedure created.

```
SQL> EXEC Calculate_GCD(4,6);
```

PL/SQL procedure successfully completed.

```
SQL> set serveroutput on;
```

```
SQL> EXEC Calculate_GCD(4,6);
```

GCD of 4 and 6 is 2

PL/SQL procedure successfully completed.

```
SQL> █
```

C. Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passed-in parameter value.

```

CREATE OR REPLACE PROCEDURE DISSAL(salary_limit IN NUMBER) AS
    CURSOR Employee_Cur IS
        SELECT NAME, SALARY
        FROM EMPLOYEE
        WHERE SALARY < salary_limit;
    emp_name EMPLOYEE.NAME%TYPE;
    emp_salary EMPLOYEE.SALARY%TYPE;
BEGIN
    OPEN Employee_Cur;
    LOOP
        FETCH Employee_Cur INTO emp_name, emp_salary;
        EXIT WHEN Employee_Cur%NOTFOUND;
    
```

```
        DBMS_OUTPUT.PUT_LINE('Name: ' || emp_name || ', Salary: ' || emp_salary);
    END LOOP;
    CLOSE Employee_Cur;
END;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE DISSAL(salary_limit IN NUMBER) AS
2     CURSOR Employee_Cur IS
3         SELECT NAME, SALARY
4         FROM EMPLOYEE
5         WHERE SALARY < salary_limit;
6     emp_name EMPLOYEE.NAME%TYPE;
7     emp_salary EMPLOYEE.SALARY%TYPE;
8 BEGIN
9     OPEN Employee_Cur;
10    LOOP
11        FETCH Employee_Cur INTO emp_name, emp_salary;
12        EXIT WHEN Employee_Cur%NOTFOUND;
13        DBMS_OUTPUT.PUT_LINE('Name: ' || emp_name || ', Salary: ' || emp_salary);
14    END LOOP;
15    CLOSE Employee_Cur;
16 END;
17 /
```

Procedure created.

```
SQL> EXEC DISSAL(60000);
Name: John Doe, Salary: 50000
Name: Bob Johnson, Salary: 55000
```

PL/SQL procedure successfully completed.

```
SQL> █
```