DATABASE MANAGEMENT SYSTEM LAB REPORT

Submitted for

Database Management System Laboratory

(5RISL1)

Submitted By

Name 1:ANUDEEP K S	**(1SI19IS003)**
Name 2:JEEVAN V S	**(1SI19IS017)**
Name 3:DHEERAJ K	**(1SI20IS018)**

Under the guidance of

Dr. Jagadamba G, Assistant Professor,
Mrs. Vidyashree, Assistant Professor,
Department of Information Science and Engineering
Siddaganga Institute of Technology, Tumakuru

Department of Information Science and Engineering

Siddaganga Institute of Technology

(An autonomous institution affiliated to VTU, Belagavi, Approved by AICTE, New Delhi, Accredited by NAAC with 'A' grade & ISO 9001:2015 Certified)

Tumakuru -572103

INDEX

PART A: SQL **PG NO:**

PART B: NO SQL

PART C: OPEN ENDED PROJECT

PART – A

SQL PROGRAMMING

1. LIBRARY DATABASE

A.1 Consider the following schema for a Library Database:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No-of_Copies)

BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**
2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2018 to Jan 2019**
3. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**
4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**
5. **Create a view of all books and its number of copies that are currently available in the Library.**

ER-DIAGRAM:

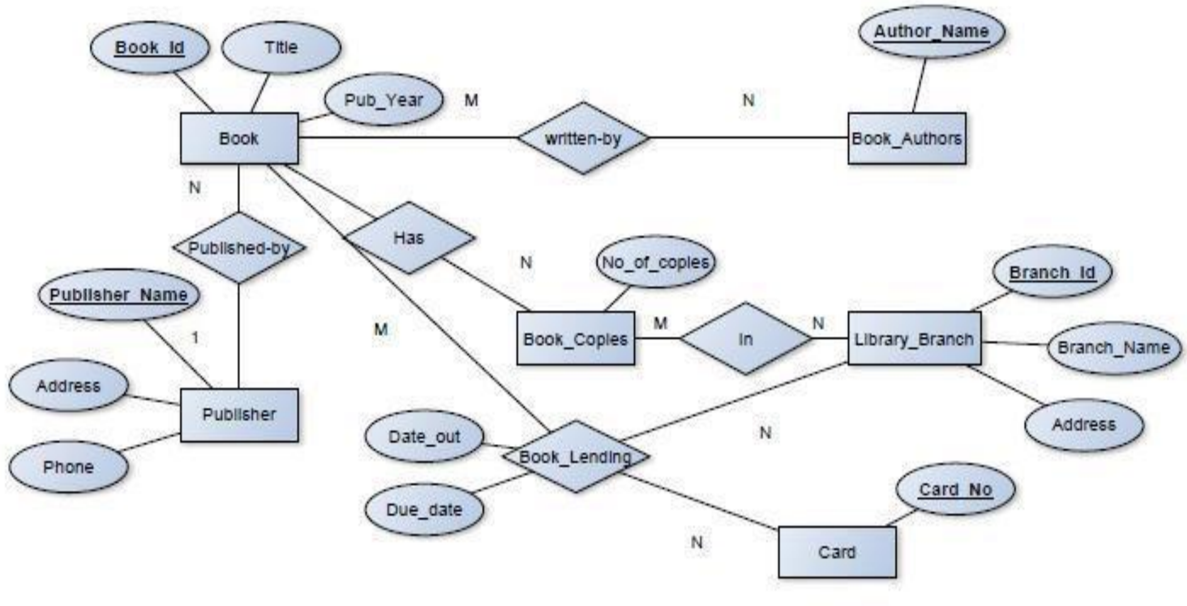


Table Creation:

PUBLISHER

```
SQL> CREATE TABLE PUBLISHER(
    NAME VARCHAR(18) PRIMARY KEY,
    ADDRESS VARCHAR(10),
    PHONE VARCHAR(10));
```

Table created.

BOOK

```
SQL> CREATE TABLE BOOK(
    BOOK_ID INTEGER PRIMARY KEY,
    TITLE VARCHAR(20),
    PUBLISHER_NAME VARCHAR(20)REFERENCES
PUBLISHER(NAME)ON DELETE
CASADE,   PUB_YEAR NUMBER(4));
```

Table created.

BOOK_AUTHORS

```
SQL> CREATE TABLE BOOK_AUTHORS(
    BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE
CASCADE,
    AUTHOR_NAME VARCHAR(20),
    PRIMARY KEY(BOOK_ID));


Table created.
```

LIBRARY_BRANCH

```
SQL> CREATE TABLE LIBRARY_BRANCH(
    BRANCH_ID INTEGER PRIMARY KEY,
    BRANCH_NAME VARCHAR(18),
    ADDRESS VARCHAR(15));


Table created.
```

BOOK_COPIES

```
SQL> CREATE TABLE BOOK_COPIES(
    BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE
CASCADE,
    BRANCH_ID INTEGER
REFERENCES
LIBRARY_BRANCH(BRANCH_
ID) ON DELETE CASCADE,
NO_OF_COPIES INTEGER,
PRIMARY
KEY(BOOK_ID,BRANCH_ID));


Table created.
```

BOOK_LENDING

```
SQL> CREATE TABLE BOOK_LENDING(
   BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE
CASCADE,
BRANCH_ID INTEGER REFERENCES
LIBRARY_BRANCH(BRANCH_ID) ON DELETE
CASCADE,
   CARD_NO INTEGER,
   DATE_OUT DATE,
   DUE_DATE DATE,
   PRIMARY KEY(BOOK_ID,BRANCH_ID,CARD_NO));

Table created.
```

Values for tables:

PUBLISHER

```
SQL>INSERT INTO PUBLISHER
VALUES('PEARSON','BANGALORE','9875462530');
SQL> INSERT INTO PUBLISHER
VALUES('MCGRAW','NEWDELHI','7845691234');

SQL> INSERT INTO PUBLISHER
VALUES('SAPNA','BANGALORE','7845963210');
```

BOOK

```
SQL> INSERT INTO BOOK
VALUES(1111,'SE','PEARSON',2005);
SQL> INSERT INTO BOOK
VALUES(2222,'DBMS','MCGRAW',2004);
```

```
SQL> INSERT INTO BOOK
VALUES(3333,'ANOTOMY','PEARSON',2010);
SQL> INSERT INTO BOOK
VALUES(4444,'ENCYCLOPEDIA','SAPNA',2010);
```

BOOK_AUTHORS

```
SQL> INSERT INTO BOOK_AUTHORS
VALUES(1111,'SOMMERVILLE');
SQL> INSERT INTO BOOK_AUTHORS
VALUES(2222,'NAVATHE');
SQL> INSERT INTO BOOK_AUTHORS
VALUES(3333,'HENRY GRAY');
SQL> INSERT INTO BOOK_AUTHORS VALUES(4444,'THOMAS');
```

LIBRARY_BRANCH

```
SQL> INSERT INTO LIBRARY_BRANCH VALUES(11,'CENTRAL
TECHNICAL','MG ROAD');
SQL> INSERT INTO LIBRARY_BRANCH
VALUES(22,'MEDICAL','BH ROAD');
SQL> INSERT INTO LIBRARY_BRANCH
VALUES(33,'CHILDREN','SS PURAM');
SQL> INSERT INTO LIBRARY_BRANCH
VALUES(44,'SECRETARIAT','SIRAGATE');
SQL> INSERT INTO LIBRARY_BRANCH
VALUES(55,'GENERAL','JAYANAGAR');
```

BOOK_COPIES

```
SQL> INSERT INTO BOOK_COPIES VALUES(1111,11,5);

SQL> INSERT INTO BOOK_COPIES
VALUES(3333,22,6);
```

```
SQL> INSERT INTO BOOK_COPIES VALUES(4444,33,10);

SQL> INSERT INTO BOOK_COPIES
VALUES(2222,11,12);
SQL> INSERT INTO BOOK_COPIES VALUES(4444,55,3);
```

BOOK_LENDING

```
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,1,'10-
JAN-2019','20-AUG-2019');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,2,'09-
JUL-2019','12-AUG-2019');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,55,1,'11-
APR-2018','09-AUG-2019');
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,5,'09-
AUG-2017','19-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,33,1,'10-
JUN-2017','15-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(1111,11,1,'12-
MAY-2017','10-JUN-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,1,'10-
JUL-2018','15-JUL-2019');
```

```
SQL> SELECT * FROM BOOK;

  BOOK_ID    TITLE           PUBLISHER_NAME   PUB_YEAR
---------- -----------------    ----------------     --------
 1111           SE             PEARSON              2005
 2222         DBMS            MCGRAW              2004
 3333     ANOTOMY          PEARSON              2010
4444     ENCYCLOPEDIA      SAPNA                2010

  4 rows selected.
```

```
SQL> SELECT * FROM BOOK_AUTHORS;

  BOOK_ID AUTHOR_NAME
 -------              ------------
1111        SOMMERVILLE
2222             NAVATHE
3333         HENRY GRAY
4444           THOMAS

4 rows selected.

SQL> SELECT * FROM PUBLISHER;

NAME          ADDRESS       PHONE
--------------- --------------- ----------
PEARSON        BANGALORE     9875462530
MCGRAW          NEWDELHI     7845691234
SAPNA         BANGALORE      7845963210
3 rows selected.

SQL> SELECT * FROM BOOK_COPIES;

BOOK_ID BRANCH_ID NO_OF_COPIES
  -------         ---------      ------------
 1111          11          5
 3333         22          6
 4444         33         10
2222          11       12
4444         55         3
5 rows selected.

SQL> SELECT * FROM BOOK_LENDING;

  BOOK_ID BRANCH_ID CARD_NO  DATE_OUT  DUE_DATE
------- --------- -------- --------- ---------
     2222       11       1 10-JAN-19 20-AUG-19
```

```
    3333       22       2 09-JUL-19 12-AUG-19
    4444       55       1 11-APR-18 09-AUG-18
    2222       11       5 09-AUG-17 19-AUG-17
    4444       33       1 10-JUL-17 15-AUG-17
    1111       11       1 12-MAY-17 10-JUN-17
    3333         22       1 10-JUL-18 15-JUL-19

7 rows selected.
```

```
SQL> SELECT * FROM LIBRARY_BRANCH;

            BRANCH_ID BRANCH_NAME
                                    ADDRESS
              --------- ------------------  ----------
 11      TECHNICAL                  MG ROAD
 22      MEDICAL                     BH ROAD
 33     CHILDREN                   SS PURAM
 44      SECRETARIAT               SIRAGATE
 55      GENERAL                  JAYANAGAR

5 rows selected.
```

Queries:

1) Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.

```
SELECT LB.BRANCH_NAME, B.BOOK_ID,TITLE,
PUBLISHER_NAME,AUTHOR_NAME,
        NO_OF_COPIES
  FROM BOOK B, BOOK_AUTHORS BA, BOOK_COPIES BC, LIBRARY_BRANCH LB
     WHERE B.BOOK_ID = BA.BOOK_ID AND
```

```
             BA.BOOK_ID = BC.BOOK_ID AND
             BC.BRANCH_ID = LB.BRANCH_ID
   GROUP BY LB.BRANCH_NAME, B.BOOK_ID, TITLE, PUBLISHER_NAME,
          AUTHOR_NAME,
NO_OF_COPIES;
```

```
 BRANCH_NAME     BOOK_ID TITLE       PUBLISHER_NAME   AUTHOR_NAME
NO_OF_COPIES
 ---------------- ------- ------------ ---------------- ------------- ------------
MEDICAL      3333 ANOTOMY     PEARSON        HENRY GRAY        6
CHILDREN     4444 ENCYCLOPEDIA SAPNA         THOMAS           10
TECHNICAL   1111   SE         PEARSON         SOMMERVILLE        5
TECHNICAL   2222 DBMS        MCGRAW         NAVATHE             12
ENCYCLOPEDIA 4444         SAPNATHOMAS    GENERAL              3
```

2) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2018 to Jan 2019.

```
    SELECT CARD_NO
   FROM BOOK_LENDING
   WHERE DATE_OUT BETWEEN '01-JAN-2018' AND '30-JAN-2019'
    GROUP BY CARD_NO
  HAVING COUNT(*) > 3;
```

```
     CARD_NO
        ------
         1
```

3) Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

```
    DELETE FROM BOOK
    WHERE BOOK_ID = '3333';
```

```
1 row deleted
```

```
SQL> SELECT * FROM BOOK;

  BOOK_ID TITLE              PUBLISHER_NAME      PUB_YEAR
--------- ------------------- ------------------- --------
1111 SE                         PEARSON             2005
 2222 DBMS                     MCGRAW              2004
4444 ENCYCLOPEDIA                 SAPNA              2010


SQL> SELECT * FROM BOOK_COPIES;

  BOOK_ID BRANCH_ID NO_OF_COPIES
------- --------- ------------
1111       11          5
 4444      33         10
2222        11        12
4444       55          3

SQL> SELECT * FROM BOOK_LENDING;

  BOOK_ID BRANCH_ID   CARD_NO DATE_OUT  DUE_DATE
  ------- --------- --------- --------- ---------
     2222       11        1 10-JAN-19 20-AUG-19
     4444       55        1 11-APR-18 09-AUG-18
     2222       11        5 09-AUG-17 19-AUG-17
     4444       33        1 10-JUN-17 15-AUG-17
1111            11         12-MAY-17 10-JUN-17
```

4) Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

```
          SELECT BOOK_ID, TITLE,
PUBLISHER_NAME, PUB_YEAR      FROM BOOK
```

```
    GROUP BY PUB_YEAR, BOOK_ID, TITLE,
PUBLISHER_NAME;
 BOOK_ID TITLE             PUBLISHER_NAME      PUB_YEAR
  ------- --------------------      --------------------       --------
 2222 DBMS                   MCGRAW             2004
  1111 SE                    PEARSON            2005
 3333 ANOTOMY             PEARSON               2010
4444 ENCYCLOPEDIA        SAPNA                 2010
```

5) Create a view of all books and its number of copies that are currently available in the Library.

```
        CREATE VIEW BOOKS_AVAILABLE AS
    SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
    FROM LIBRARY_BRANCH L, BOOK B, BOOK_COPIES C
    WHERE B.BOOK_ID = C.BOOK_ID   AND
         L.BRANCH_ID=C.BRANCH_ID;

        View created.

        SQL> SELECT * FROM BOOKS_AVAILABLE;

 BOOK_ID    TITLE              NO_OF_COPIES
------- --------------------      ------------
 1111 SE                          5
3333 ANOTOMY                  6
4444 ENCYCLOPEDIA            10
2222 DBMS                    12
4444 ENCYCLOPEDIA             3
```

2.ORDER DATABASE

A,2 Consider the following schema for Order Database:
SALESMAN (Salesman_id, Name, City, Commission)
CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
ORDERS(Ord_No,Purchase_Amt,Ord_Date,Customer_id,

Saleman_id)

Write SQL queries to

1. **Count the customers with grades above Bangalore’s average.**

2. **Find the name and numbers of all salesmen who had more than one customer.**
3. **List all salesmen and indicate those who have and don’t have customers in their cities**
   (Use UNION operation.)

4. **Create a view that finds the salesman who has the customer with the highest order of a**
   day.

5. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted**.

ER-Diagram:

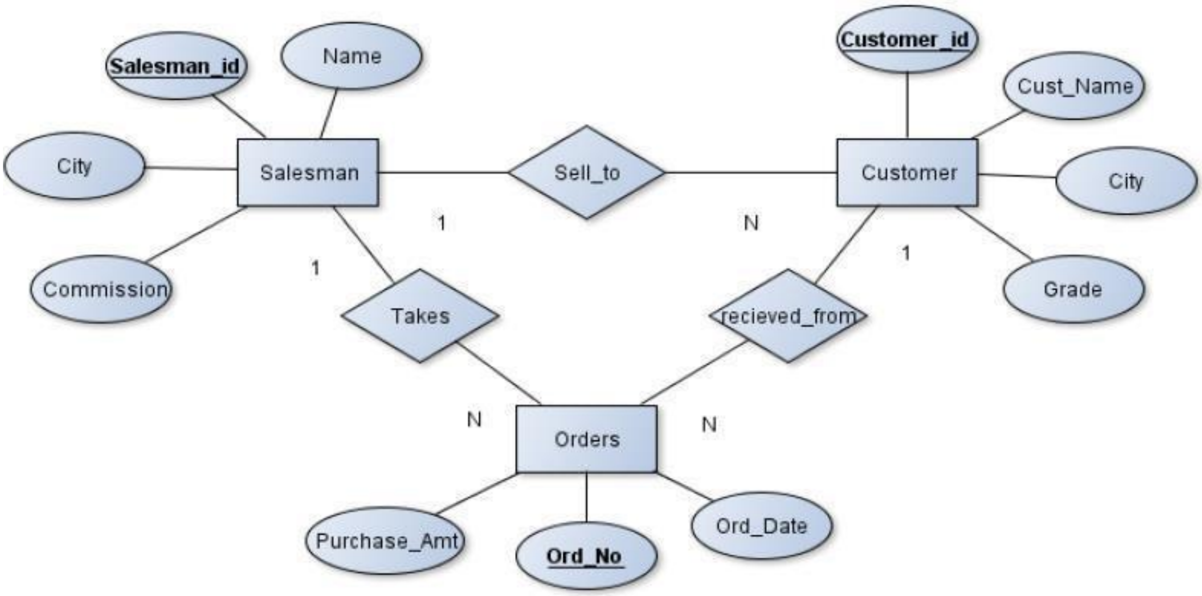


Table Creation:

SALESMAN

```
CREATE TABLE SALESMAN(
SALESMAN_ID NUMBER(5) CONSTRAINT SALESMAN_SALID
PRIMARY KEY,
NAME VARCHAR(10) CONSTRAINT SALESMAN_NAME_NN NOT
NULL,
CITY VARCHAR(15) CONSTRAINT SALESMAN_CITY_NN NOT NULL,
COMMISSION NUMBER(5));
```

```
Table created.
```

CUSTOMER

```
CREATE TABLE CUSTOMER(
CUSTOMER_ID NUMBER(5) CONSTRAINT CUSTOMER_CUSTID_PK
PRIMARY KEY,
CUST_NAME VARCHAR(10) CONSTRAINT
CUSTOMER_CUSTNAME_NN NOT NULL,
CITY VARCHAR(10) CONSTRAINT CUSTOMER_CITY_NN NOT NULL,
GRADE NUMBER(5) CONSTRAINT CUSTOMER_GRADE_NN NOT
NULL,
SALESMAN_ID NUMBER(5) CONSTRAINT CUSTOMER_SALEID_FKREFERENCES
SALESMAN(SALESMAN_ID) ON DELETE SET NULL);
```

```
Table created.
```

ORDERS

```
CREATE TABLE ORDERS(

ORD_NO NUMBER(5) CONSTRAINT ORDERS_ODNO_PK PRIMARY
KEY,

PURCHASE_AMT INTEGER CONSTRAINT ORDERS_PAMT_NN NOT
NULL,

ORD_DATE DATE CONSTRAINT ORDERS_ODATE_NN NOT NULL,

CUSTOMER_ID NUMBER(5) CONSTRAINT ORDERS_CUSTID_FK REFERENCES

CUSTOMER(CUSTOMER_ID),

SALESMAN_ID NUMBER(5) CONSTRAINT ORDERS_SALEID_FKREFERENCES
SALESMAN(SALESMAN_ID) ON DELETE CASCADE);
```

```
Table created.
```

Values for tables

```
SQL> INSERT INTO SALESMAN
VALUES(&SALESMAN_ID,'&NAME','&CITY',&COMMISSION);
SQL> INSERT INTO CUSTOMER

VALUES(&CUSTOMER_ID,'&CUST_NAME','&CITY','&GRADE',&SALE
SMAN_ID);

SQL> INSERT INTO ORDERS

VALUES(&ORD_NO,&PURCHASE_AMT,'&ORD_DATE',&CUSTOMER
_ID,&SALESMAN_ID); SELECT * FROM SALESMAN;

SALESMAN_ID NAME       CITY           COMMISSION
----------- ---------- -------------- ----------
 1000 RAJ       BENGALURU                  50
 2000 ASHWIN     TUMKUR                    30
  3000 BINDU      MUMBAI                   40
 4000LAVANYABENGALURU                      50
5000 ROHIT      MYSORE                     60
SELECT * FROM CUSTOMER;

CUSTOMER_ID CUST_NAME  CITY          GRADE SALESMAN_ID
----------- ---------- ---------- --------- -----------
   11 INFOSYS    BENGALURU     5       1000
  22 TCS       BENGALURU        4       2000
 33 WIPRO      MYSORE           7       1000
 44 TCS       MYSORE            6       2000
  55 ORACLE     TUMKUR          3       3000
SELECT * FROM ORDERS;

ORD_NO PURCHASE_AMT ORD_DATECUSTOMER_IDSALESMAN_ID
--------- ------------ --------- ----------- -----------
  1       200000 12-APR-16         11       1000
   2       300000 12-APR-16         11       2000
 3      400000 15-APR-17        22       1000
```

1. Count the customers with grades above Bangalore’s average.

```
SELECT COUNT(CUSTOMER_ID)
FROM CUSTOMER
WHERE GRADE>(SELECT AVG(GRADE)
          FROM CUSTOMER
          WHERE CITY LIKE '%BENGALURU');
```

```
COUNT(CUSTO
     MER_ID)
------------------
            3
```

2.Find the name and numbers of all salesmen who had more than one customer.

```
SELECT NAME, COUNT(CUSTOMER_ID)
FROM SALESMAN S, CUSTOMER C
WHERE S.SALESMAN_ID=C.SALESMAN_ID
GROUP BY NAME
HAVING COUNT(CUSTOMER_ID)>1;
```

```
NAME      COUNT(CUSTOMER_ID)
 ---------- ------------------
ASHWIN               2
RAJ                  2
```

2. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)

```
(SELECT NAME
 FROM SALESMAN S, CUSTOMER C
 WHERE S.SALESMAN_ID=C.SALESMAN_ID AND
      S.CITY=C.CITY)
UNION
(SELECT NAME
 FROM SALESMAN
 WHERE SALESMAN_ID NOT IN(SELECT S1.SALESMAN_ID
                        FROM SALESMAN S1, CUSTOMER C1
                        WHERE S1.SALESMAN_ID=C1.SALESMAN_ID AND
                              S1.CITY=C1.CITY));
```

```
 NAME
   ----------                                        ASHWIN
 BINDU                                               LAVANYA
 RAJ                                                 ROHIT
```

4. Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE VIEW SALES_HIGHERODER AS
SELECT SALESMAN_ID, PURCHASE_AMT
FROM ORDERS
WHERE PURCHASE_AMT=(SELECT MAX(O.PURCHASE_AMT)
                    FROM ORDERS O
                    WHERE O.ORD_DATE='12-APR-16');
```

```
View created.

SELECT * FROM SALES_HIGHERODER;

SALESMAN_ID PURCHASE_AMT
----------- ------------
 2000       300000
```

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

```
DELETE from salesman
WHERE salesman_id = 1000;

1 row deleted
```

```
SELECT * FROM SALESMAN;

SALESMAN_ID NAME        CITY              COMMISSION

----------- ---------- --------------- ----------
       2000 ASHWIN      TUMKUR                  30

       3000 BINDU       MUMBAI                  40

       4000 LAVANYA     BENGALURU               40

       5000 ROHIT       MYSORE                  60
```

```
SELECT * FROM CUSTOMER;

CUSTOMER_ID CUST_NAME CITY            GRADE SALESMAN_ID
----------- ---------- ---------- --------- -----------
         11 INFOSYS    BENGALURU          5        4000

         22 TCS        BENGALURU          4        2000

         33 WIPRO      MYSORE             7        5000

         44 TCS        MYSORE             6        2000

         55 ORACLE     TUMKUR             3        3000
```

```
SELECT * FROM ORDERS;

   ORD_NO PURCHASE_AMT ORD_DATE CUSTOMER_ID
   SALESMAN_ID
 --------- -------------------- ----------- -----------
```

```
        2          30000012-APR-16            11          2000
```

3.MOVIE DATABASE

A.3 Consider the schema for Movie Database:

ACTOR (*Act_id, Act_Name, Act_Gender*)

DIRECTOR (*Dir_id, Dir_Name, Dir_Phone*)

MOVIES (*Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id*)

MOVIE_CAST (*Act_id, Mov_id, Role*)

RATING (*Mov_id, Rev_Stars*)

Write SQL queries to

1. **List the titles of all movies directed by 'Mani Ratnam'.**
2. **Find the movie names where one or more actors acted in two or more movies.**
3. **List all actors who acted in a movie before 2010 and also in a movie after 2017 (use JOIN operation).**
4. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**

5.Update rating of all movies directed by 'Karan Johar' to 4.

ER-Diagram:

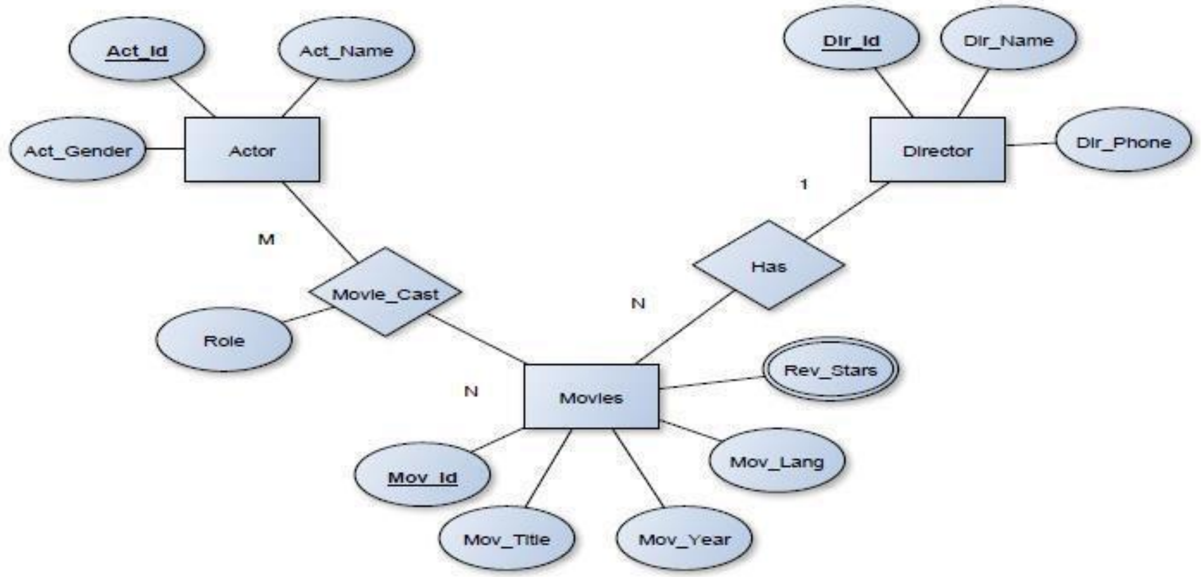


Table Creation:

ACTOR

```
CREATE TABLE ACTOR(
ACT_ID NUMBER(5) CONSTRAINT ACTOR_ACTID_PK PRIMARY KEY,
ACT_NAME VARCHAR(18) CONSTRAINT ACTOR_ACTNAME_NN
NOT NULL,
ACT_GENDER VARCHAR(2) CONSTRAINT ACTOR_ACTGENDER_NN
NOT NULL);
```

```
Table created.
```

DIRECTOR

```
CREATE TABLE DIRECTOR(
DIR_ID NUMBER(5) CONSTRAINT DIRECTOR_DIRID_PK PRIMARY
KEY,
DIR_NAME VARCHAR(18) CONSTRAINT DIRECTOR_DIRNAME_NN
NOT NULL,
DIR_PHONE VARCHAR(10) CONSTRAINT DIRECTOR_DIRPHONE_NN
NOT NULL);
```

```
Table created.
```

MOVIES

```
CREATE TABLE MOVIES(
MOV_ID NUMBER(5) CONSTRAINT MOVIES_MOVID_PK PRIMARY
KEY,
MOV_TITLE VARCHAR(10) CONSTRAINT MOVIES_MOVTITLE_NN
NOT NULL,
MOV_YEAR NUMBER(5) CONSTRAINT MOVIES_MOVYEAR_NN
NOT NULL,
MOV_LANG VARCHAR(10)  CONSTRAINT MOVIES_MOVLANG_NN
NOT NULL,
DIR_ID NUMBER(5) CONSTRAINT MOVIES_DIRID_FK REFERENCES
DIRECTOR(DIR_ID));
```

```
Table created.
```

MOVIE_CAST

```
CREATE TABLE MOVIE_CAST(
ACT_ID NUMBER(5) CONSTRAINT MOVIECAST_ACTID_FK
REFERENCES ACTOR(ACT_ID), MOV_ID NUMBER(5) CONSTRAINT
MOVIECAST_MOVID_FK REFERENCES MOVIES(MOV_ID),
ROLE VARCHAR(10),
CONSTRAINT MOVIECAST_ACTID_MOVID_PK PRIMARY
KEY(ACT_ID,MOV_ID));
```

```
Table created.
```

RATING

```
CREATE TABLE RATING(
MOV_ID NUMBER(5) CONSTRAINT RATING_MOVID_FK
REFERENCES MOVIES(MOV_ID),
REV_STARS NUMBER(1) CONSTRAINT RATING_REVSTARS_NN
NOT NULL,
CONSTRAINT RATING_MOVID_PK PRIMARY KEY(MOV_ID))
```

```
Table created
```

Values for tables:

```
SQL> INSERT INTO ACTOR
VALUES(&ACT_ID,'&ACT_NAME','&ACT_GENDER');   SQL>
INSERT INTO DIRECTOR
VALUES(&DIR_ID,'&DIR_NAME',&DIR_PHONE);

SQL> INSERT INTO MOVIES
VALUES(&MOV_ID,'&MOV_TITLE','&MOV_YEAR','&MOV_LANG',&DIR_ID);

SQL> INSERT INTO MOVIE_CAST VALUES(&ACT_ID,&MOV_ID,'&ROLE');

SQL> INSERT INTO RATING VALUES(&MOV_ID,&REV_STARS);
SQL> SELECT * FROM ACTOR;

   ACT_ID ACT_NAME        AC
  --------- --------------- --
 111 DEEPA SANNIDHI  F
 222 SUDEEP     M
 333 PUNEETH M
444DHIGANTH M
555RAMYA      F
SQL> SELECT * FROM DIRECTOR;

   DIR_ID DIR_NAME          DIR_PHONE
 --------- ------------------ ---------
101  Mani Ratnam        112267809
102  RAJ MOULI         152358709
103  YOGARAJ          272337808
104  Karan Johar       363445678
105   PAVAN KUMAR       385456809
SQL> SELECT * FROM MOVIES;

   MOV_ID MOV_TITLE   MOV_YEAR MOV_LANG      DIR_ID
--------- ----------- -------- ---------- ---------
1111 LASTWORLD       2009 ENGLISH         104
2222 EEGA           2010 TELUGU               102
4444 PARAMATHMA      2012 KANNADA   103
3333 MALE           2006 KANNADA            103
 5555 MANASARE       2010 KANNADA       103
 6666 REAR WINDOW     1954 ENGLISH      101
7777 NOTORIOUS       1946 ENGLISH         101

SQL> SELECT * FROM MOVIE_CAST;

   ACT_ID    MOV_ID ROLE
  --------- --------- ----------
 222     2222 VILLAIN
333     4444 HERO
111     4444 HEROINE
444     3333 GUEST
444     5555 HERO
 555     7777 MOTHER
SQL> SELECT * FROM RATING;
   MOV_ID REV_STARS
   --------- ---------
    1111        3
    2222        4
    3333        3
    5555        4
    4444        5
```

1. List the titles of all movies directed by ‘ManiRatnam’.

```
SELECT MOV_TITLE
FROM MOVIES M, DIRECTOR D
WHERE D.DIR_ID=M.DIR_ID AND
      DIR_NAME='ManiRatnam';

MOV_TITLE                                              -
---------
  NOTORIOUS
REAR WINDOW
```

2. Find the movie names where one or more actors acted in two or more movies.

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MC
WHERE M.MOV_ID=MC.MOV_ID AND
      MC.ACT_ID IN (SELECT ACT_ID
                    FROM MOVIE_CAST
                    GROUP BY ACT_ID
                    HAVING COUNT(MOV_ID)>=2);
MOV_TITLE
```

```
----------
 MALE
 MANASARE
```

3. List all actors who acted in a movie before 2010 and also in a movie after 2017 (use JOIN operation).

```
(SELECT ACT_NAME
FROM ACTOR A
JOIN
MOVIE_
CAST C
ON
A.ACT_ID
=C.ACT_ID 4.
JOIN   MOVIES
M  ON
C.MOV_ID=M.
MOV_ID
WHERE M.MOV_YEAR < 2000)
INTERSECT
(SELECT ACT_NAME
FROM ACTOR A JOIN
MOVIE_CAST C
ON A.ACT_ID=C.ACT_ID JOIN
MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE
M.MOV_YEAR
> 2015);
ACT_NAME
---------------
DHIGANTH
```

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest

number of stars that movie received. Sort the result by movie title.

```
SELECT MOV_TITLE,
REV_STARS FROM
MOVIES M, RATING R

WHERE      M.MOV_ID=R.MOV_ID      AND
REV_STARS>=1 ORDER BY MOV_TITLE
```

```
MOV_TITLE  REV_STARS
       ---------- ---------
EEGA              4
LASTWORLD         3
MALE              3
MANASARE          4
PARAMATHMA        5
```

5. Update rating of all movies directed by ‘Karan Johar’ to 4.

```
UPDATE  RATING
SET REV_STARS=4
WHERE MOV_ID IN (SELECT MOV_ID
FROM MOVIES M, DIRECTOR D
WHERE M.DIR_ID=D.DIR_ID  AND DIR_NAME='Karan Johar’);
```

```
1 row updated.
```

```
SELECT * FROM RATING
```

```
MOV_IDREV_STAR
------ ---------
    1111     4
    2222     4
    3333     3
    5555     4
    4444     4
```

4.COLLEGE DATABASE

A.4. Consider the schema for College Database:

STUDENT (USN, SName, Address, Phone, Gender)
SEMSEC (SSID, Sem, Sec)
CLASS (USN, SSID)
SUBJECT (Subcode, Title, Sem, Credits)
IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA) Write SQL queries to

1.List all the student details studying in fourth semester 'C' section.

2.Compute the total number of male and female students in each semester and in each section.

3.Create a view of Test1 marks of student USN '1S15CS101' in all subjects.

4.Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

5.Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = 'Outstanding'

If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA< 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students.

ER-Diagram:

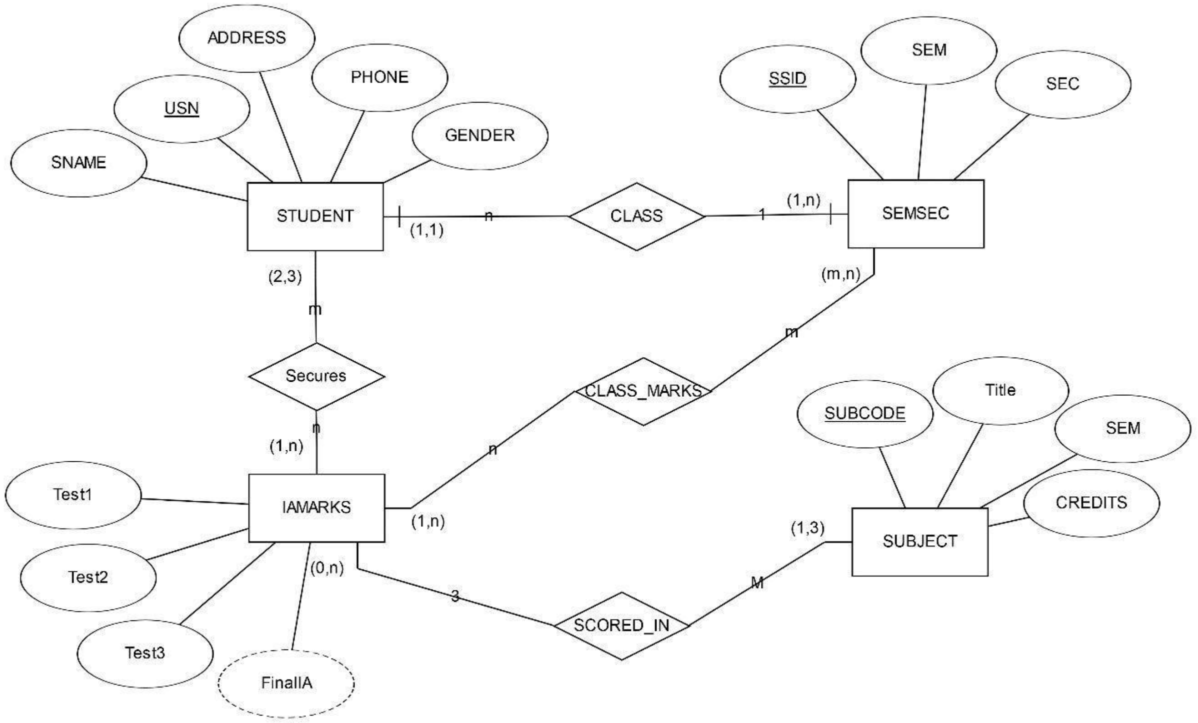


Table Creation:

STUDENT

CREATE TABLE STUDENT

(USN VARCHAR(10)

PRIMARY KEY, SNAME

VARCHAR(25), ADDRESS

VARCHAR(25), PHONE

VARCHAR(10), GENDER

CHAR(1));

```
Table created.
```

SEMSEC

```
CREATE TABLE SEMSEC        SSID VARCHAR(5) PRIMARY KEY, SEM
NUMBER(2), SEC CHAR(1));
```

```
Table created.
```

CLASS

```
CREATE TABLE CLASS ( USN VARCHAR(10), SSID VARCHAR(5),
PRIMARY  KEY(USN,SSID), FOREIGN KEY(USN) REFERENCES
STUDENT(USN), FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID));
```

```
Table created.
```

SUBJECT

```
CREATE TABLE SUBJECT
(SUBCODE VARCHAR(8)
PRIMARY KEY,  TITLE
VARCHAR(20), SEM
NUMBER(2), CREDITS
NUMBER(2));
```

```
Table created.
```

IAMARKS

```
CREATE TABLE IAMARKS (USN VARCHAR(10), SUBCODE
VARCHAR(8), SSID VARCHAR(5), TEST1 NUMBER(2), TEST2
NUMBER(2),
 TEST3 NUMBER(2),  FINALIA NUMBER(3),  PRIMARY
KEY(USN,SUBCODE,SSID), FOREIGN KEY(USN) REFERENCES
STUDENT(USN), FOREIGN KEY(SUBCODE) REFERENCES
SUBJECT(SUBCODE), FOREIGN KEY(SSID) REFERENCES
SEMSEC(SSID));

Table created.
```

Values for tables:

STUDENT:

```
INSERT INTO STUDENT VALUES
('&USN','&sname','&address',’&phone’,'&gender');

select * from student;
USN SNAME  ADDRESS PHONE GENDER
------ --------- ------------- ---------- ------------
```

```
1si15cs001 Abhi          tumkur          9875698410
                                         M
1si15cs002 amulya        gubbi           8896557412
                                         F
1si16me063 chethan       nittur          7894759522
                                         M
1si14ec055 raghavi       sspuram         9485675521
                                         F
1si15ee065 sanjay        bangalore       9538444404
                                         M
```

SEMSEC:

```
INSERT INTO SEMSEC VALUES ('&SSID', ‘&sem’,'&sec');

select * from semsec;
```

```
SSIDSEMS
------ ------
5A
3C
7B
4C
4B
```

```
2C
```

```
INSERT INTO CLASS VALUES (‘&USN’,’&SSID’);
```

```
select * from class;
```

```
USN SSID
---------- -----



1si16me063 3B
1si14ec055 7A
1si15ee065 3B
1si15ee065 4c
1si15cs002 4c
```

SUBJECT:

```
INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);
```

```
select * from subject;
```

```
SUBCODE TITLE                              SEM CREDITS

15cs53 dbms                                  5       4

15cs33 ds                                    3       4

15cs34 co                                    3       4

15csl58 dba                                 5        2
```

```
10cs71 oomd                          7          4
```

IAMARKS:

```
INSERT INTO IAMARKS VALUES
('&USN','&SUBCODE','&SSID','&TEST1','&TEST2','&TEST3')
;
```

```
select * from iamarks;
```

```
USN        SUBCODE SSID       TEST1     TEST2      TEST3 FINALIA
---------- -------- ----- --------- --------- --------- ---------

1si15cs001 15cs53    5A          18         19         15        19

1si15cs002 15cs53    5A          15         16         14        16

1si16me063 15cs33    3B          10         15         16        16

1si14ec055 10cs71    7A          18         20         21        21

1si15ee065 15cs33    3B          16         20         17        19

1si15ee065 15cs53    4c          19         20         18        20
```

Queries:

1. List all the student details studying in fourth semester ‘C’ section.

```
        select
        s.usn,sname,address,phone,gender from
        student s, class c, semsec ss where
sem=4 and
                 sec='c' and ss.ssid=c.ssid
                 and c.usn=s.usn;
USN        SNAME           ADDRESS          PHONE        G
---------- ------------- --------------- ------------ -----
1si15ee065 Sanjay          bangalore        9538444404 M
1si15cs002 Amulya          gubbi            8896557412 F
```

2. Compute the total number of male and female students in each semester and in each section.

```
        SELECT SEM,SEC,GENDER,COUNT(*)
        FROM STUDENT S, SEMSEC
        SS,CLASS C
     WHERE S.USN=C.USN AND
                 C.SSID=SS.SSID
        GROUP BY
        SEM,SEC,GENDER
        ORDER BY SEM;

        SEM S G  COUNT(*)
        ---- - - ---------
            3B M          2
```

```
4c F        1
4c M        1
5A F        1
5A M        1
7A F        1
```

3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.

```
 CREATE VIEW TEST1 AS
  SELECT SUBCODE,TEST1
FROM IAMARKS
  WHERE USN='1cg15ee065';

  View created.

  SQL> select * from test1;

 SUBCODE      TEST1
  -------- ---------
  15cs33      16 15cs53
  19
```

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

```
  CREATE OR REPLACE PROCEDURE
  AVG IS
```

```
        CURSOR C_IAMARKS IS
        SELECT GREATEST(TEST1,TEST2) AS A,GREATEST(TEST1,TEST3)
        AS B,
        GREATEST(TEST3,TEST2) AS C
FROM IAMARKS
        WHERE FINALIA IS NULL
        FOR UPDATE;
      C_A NUMBER;
      C_B NUMBER;
      C_C NUMBER;
        C_SM NUMBER;
      C_AV NUMBER;
        BEGIN
      OPEN C_IAMARKS;
        LOOP
      FETCH C_IAMARKS INTO C_A,C_B,C_C;
   EXIT WHEN C_IAMARKS%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(C_A||' '||C_B||' '||C_C);
         IF(C_A!=C_B) THEN
        C_SM:=C_A+C_B;
        ELSE
                C_SM:=C_A+C_C;
        END IF;
       C_AV:=C_SM/2;
        DBMS_OUTPUT.PUT_LINE('SUM='||C_SM);

        DBMS_OUTPUT.PUT_LINE('AVERAGE='||C_AV)
        ; UPDATE IAMARKS
       SET FINALIA=C_AV
```

```
 WHERE CURRENT OF C_IAMARKS;
END LOOP;
 CLOSE C_IAMARKS;
 END AVG;

 Procedure created.

SQL> BEGIN
   2   AVG;
   3   END;

 PL/SQL procedure successfully completed.

 SQL> SELECT * FROM IAMARKS;

USN          SUBCODE SSID TEST1      TEST2        TEST3 FINALIA
 ---------- -------- ----- --------- --------- --------- -----
 1si15cs001 15cs53    5A     18         19          15        19

 1si15cs002 15cs53    5A     15         16          14        16

 1si16me06315cs33     3B     10         15          16        16

 1si14ec055 10cs71    7A     18         20          21        21

 1si15ee065 15cs33    3B     16         20          17        19

 1si15ee065 15cs53    4c     19         20          18        20

6 rows selected.
```

5 . Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’ If FinalIA< 12 then CAT = ‘Weak’
Give these details only for 8th semester A, B, and C section students.

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, CASE
        WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE' ELSE
        'WEAK'
          END AS CAT
        FROM STUDENT S,SEMSEC SS,IAMARKS IA,SUBJECT SUB
     WHERE S.USN=IA.USN AND
              SS.SSID=IA.SSID AND
              SUB.SUBCODE=IA.SUBCODE AND
              SUB.SEM=7
```

```
USN        SNAME       ADDRESS     PHONE      G CAT
---------- ----------- ------------ ---------- - -----
1si14ec055 raghavi     sspuram     9485675521 F WEAK
```

5.COMPANY DATABASE

A.5 . Consider the schema for Company Database:

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,Hour
) 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.**
2. **Show the resulting salaries if every employee working on the 'IoT' project is**
   given a 10 percent raise.
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**
4. **Retrieve the name of each employee who works on all the projects controlled by department number 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**.

ER-Diagram:

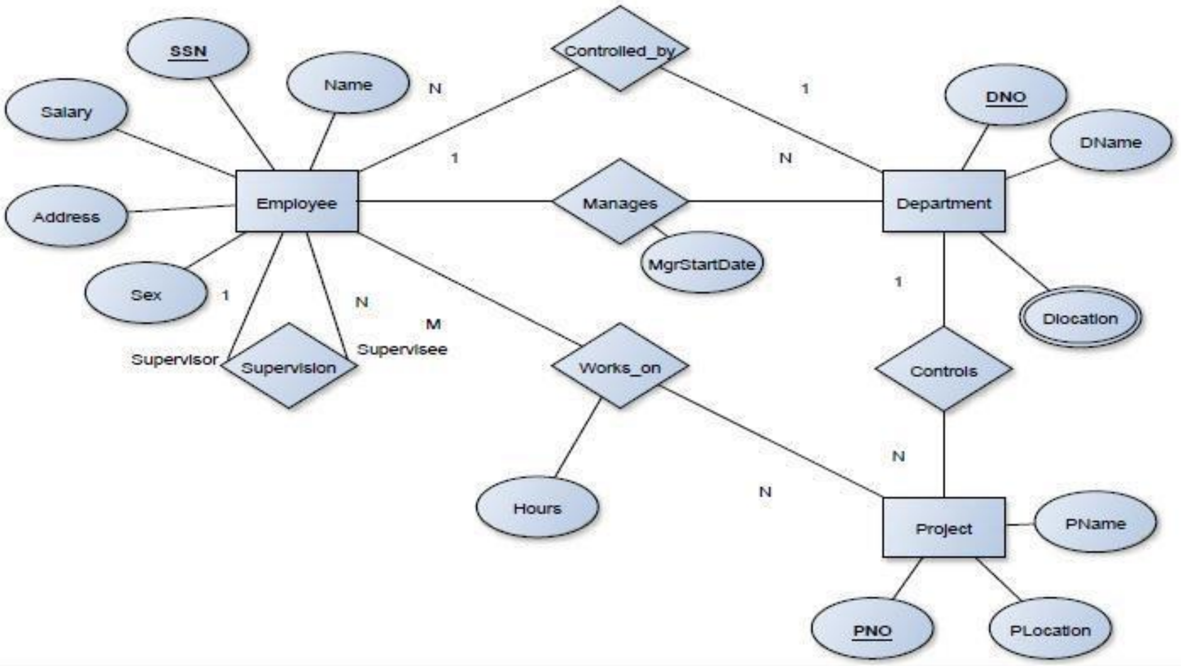


Table Creation:

DEPARTMENT

CREATE TABLE DEPARTMENT(DNO NUMBER(3) CONSTRAINT DEPT_DNO_PK PRIMARY KEY, DNAME VARCHAR(15) CONSTRAINT DEPT_DNAME_NN NOT NULL, MGRSSN CHAR(10), MGRSTARTDATE DATE);

EMPLOYEE

```
 CREATE TABLE EMPLOYEE( SSN CHAR(10) CONSTRAINT
EMP_SSN_PK PRIMARY KEY,          NAME VARCHAR(18)
CONSTRAINT EMP_NAME_NN NOT NULL, ADDRESS VARCHAR(18),
 SEX VARCHAR(3), SALARY REAL, SUPER_SSN  CHAR(10), DNO
NUMBER(3) CONSTRAINT EMP_DNO_FK REFERENCES
DEPARTMENT(DNO));
```

```
ALTER TABLE DEPARTMENT ADD CONSTRAINT DEPT_MGRSSN_FK
FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN);
```

```
Table altered.
```

DLOCATION

```
CREATE TABLE DLOCATION( DLOC VARCHAR2 (20), DNO
REFERENCES DEPARTMENT (DNO), PRIMARY KEY (DNO, DLOC));
```

PROJECT

```
CREATE TABLE PROJECT( PNO INTEGER PRIMARY KEY, PNAME
VARCHAR2 (20), PLOCATION VARCHAR2 (20), DNO REFERENCES
DEPARTMENT (DNO));
```

WORKS_ON____

```
CREATE TABLE WORKS_ON( HOURS NUMBER (2), SSN REFERENCES
EMPLOYEE (SSN), PNO REFERENCES PROJECT(PNO), PRIMARY KEY
(SSN, PNO));
```

Values for tables:

DEPARTMENT

```
INSERT INTO DEPARTMENT
VALUES(&DNO,'&DNAME',&MGRSSN,'&MGRSTARTDATE');
SELECT * FROM DEPARTMENT;
```

```
    DNO DNAME          MGRSSN MGRSTARTD
--------- --------------- ---------- ---------
        1 RESEARCH      111111    10AUG12

        2 ACCOUNTS      222222    10AUG10

        3 AI            333333    15-AP12

        4 NETWORKS      111111    18MAY14

        5 BIGDATA       666666    21-JAN10

5 rows selected.
```

EMPLOYEE

```
INSERT INTO EMPLOYEE
VALUES('&SSN','&NAME','&ADDRESS','&SEX',&SALARY,'&SUPERSSN',
&
DNO);

SELECT * FROM EMPLOYEE;

SSN      NAME               ADDRESS        SEX SALARY SUPERSSN    DNO

-------- ------------------------------ ---------------- -------      ---     ---     ---

111111 RAJ                BENGALURU   M    700000                   1

222222 RASHMI             MYSORE      F    400000 111111            2

333333 RAGAVI             TUMKUR      F    800000                   3

444444 RAJESH             TUMKUR      M    650000 333333            3

555555 RAVEESH            BENGALURU   M    500000 333333            3

666666 SCOTT              ENGLAND     M    700000 444444            5

777777 NIGANTH            GUBBI       M    200000 222222            2

888888 RAMYA              GUBBI       F    400000 222222            3

999999 VIDYA              TUMKUR      F    650000 333333            3

100000 GEETHA             TUMKUR      F    800000                   3

10 rows selected.
```

DLOCATION

```
INSERT INTO DLOCATION VALUES(&DNO,'&DLOC');

  SELECT * FROM DLOCATION;

              DNO DLOC --------- ---------------
              1 MYSORE  2 TUMKUR   3 BENGALURU  4GUBBI  5 DELHI
              6 BENGALURU

      6 rows selected.
```

PROJECT

```
INSERT INTO PROJECT
VALUES(&PNO,'&PNAME','&PLOCATION',’&DNO’);

  SELECT * FROM PROJECT;

             PNO PNAME     PLOCATION          DNO
                --------- ---------- ---------- ---------
                111 IOT       GUBBI                    3
```

```
222 TEXTSPEECH GUBBI              3

333 IPSECURITY DELHI              4

 444 TRAFICANAL BENGALURU         5

555 CLOUDSEC DELHI                1
```

```
     5 rows selected.
```

WORKS_ON

```
INSERT INTO WORKS_ON
VALUES('&SSN',&PNO,&HOURS);
```

```
 SELECT * FROM WORKS_ON;
```

```
     SSN        PNO    HOURS
      ---------- ---------    --------


     666666      111   2            111111    222   3
     555555      222   2            333333    111   4
     444444      111   6            222222    111   2
```

```
3 rows selected.
```

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 PNO

    FROM PROJECT P, DEPARTMENT D,
EMPLOYEE E WHERE P.DNO=D.DNO AND

         SSN=MGRSSN AND

          NAME='SCOTT')

  UNION
 (SELECT DISTINCT P.PNO

   FROM PROJECT P, WORKS_ON W,
EMPLOYEE E WHERE P.PNO=W.PNO AND

         W.SSN=E.SSN AND
          NAME='SCOTT');
```

```
     PNO

   ---------

     111

     333

     444
```

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

```
SELECT FNAME, LNAME, 1.1*SALARY AS INCR_SAL
FROM EMPLOYEE E, WORKS_ON W, PROJECT P
      WHERE E.SSN=W.SSN AND
       W.PNO=P.PNO AND
         P.PNAME=’IOT’;
```

```
      SSN      NAME            ADDRESS          SEX SALARY SUPERSSN DNO
       ---------- ----------------- ----------------- --- --------- ---------- --------
111111         RAJ           BENGALURU      M 700000                 1
222222       RASHMI         MYSORE           F 440000 111111        2
333333       RAGAVI         TUMKUR           F 880000                3
444444 RAJESH TUMKUR M 715000  333333                   3
555555 RAVEESH BENGALURU M 500000      333333             3
666666 SCOTT ENGLAND M 770000         444444              5
777777 NIGANTH GUBBI M 200000 222222                      2

888888 RAMYA GUBBI F 400000 222222                        3
999999 VIDYA TUMKUR F 650000 333333                       3
100000 GEETHA TUMKUR F 800000                            3

        10  rows selected.
```

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.

```
    SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY),
AVG(SALARY) FROM EMPLOYEE E, DEPARTMENT D

 WHERE DNAME='ACCOUNTS' AND

             D.DNO=E.DNO;
```

```
 SUM(SALARY) MAX(SALARY) MIN(SALARY) AVG(SALARY)
---------- ------------------------------   -       -       -
  640000   440000   200000       320000
```

4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

```
    SELECT NAME FROM
    EMPLOYEE E

    WHERE NOT EXISTS( (SELECT PNO
                           FROM ROJECT
                           WHERE
                           DNO=5)
                          MINUS
                          (SELECT PNO
                          FROM WORKS_ON W
                            WHERE E.SSN=W.SSN))
```

```
 NAME
----------------
 SCOTT
```

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(SSN)
    FROM
EMPLOYEE
    WHERE SALARY>600000 AND DNO
           IN(SELECT DNO
                        FROM EMPLOYEE
                      GROUP BY DNO
                      HAVING COUNT(SSN)>5)
    GROUP BY DNO ;
```

```
 DNO COUNT(SSN)
 --------- ----------
3              4
```

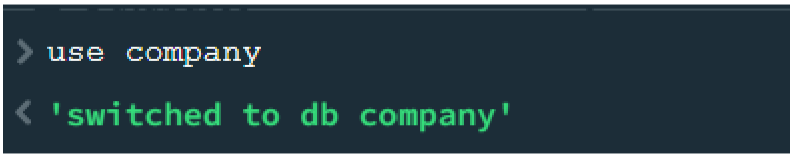
PART – B
NO SQL

B.1.Create the below tables, insert suitable tuples and perform the following operations using MongoDB EMPLOYEE (SSN, Name, DeptNo) ASSIGNED_TO (SSN , ProjectNo)

a. List all the employees of department “XYZ”.

b. Name the employees working on Project Number 6.

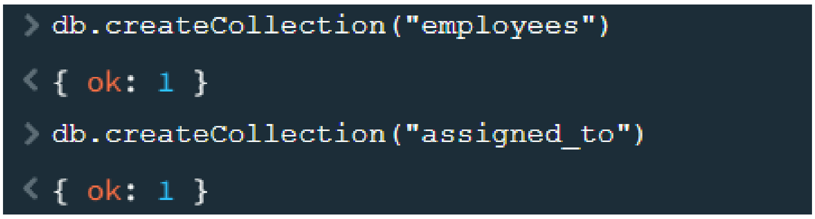
use company



```
> use company
< 'switched to db company'
```

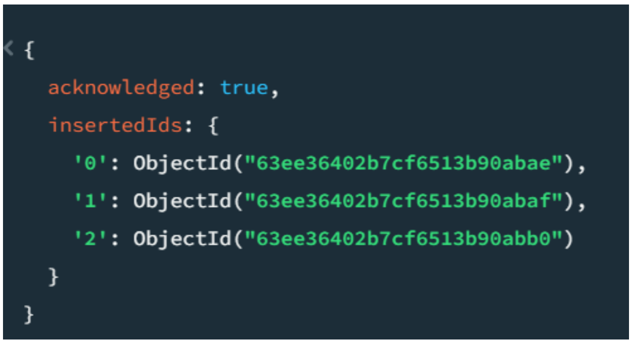
db.createCollection("employees")

db.createCollection("assigned_to")



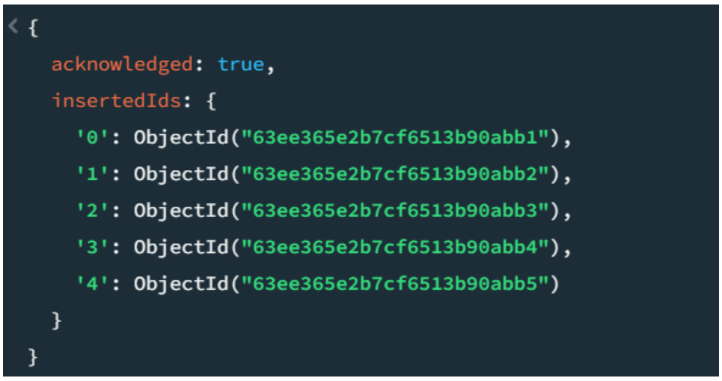
```
> db.createCollection("employees")
< { ok: 1 }
> db.createCollection("assigned_to")
< { ok: 1 }
```

```
db.employees.insertMany([
  {SSN: "111-11-1111", Name: "John Doe", DeptNo: "XYZ"},
  {SSN: "222-22-2222", Name: "Jane Smith", DeptNo: "ABC"},
  {SSN: "333-33-3333", Name: "Bob Johnson", DeptNo: "XYZ"}
])
```



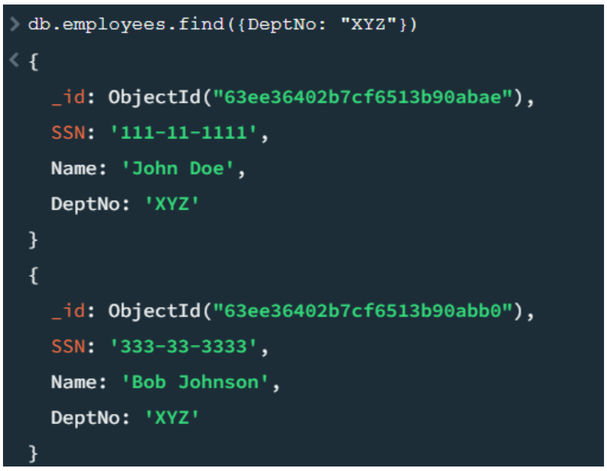
```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63ee36402b7cf6513b90abae"),
    '1': ObjectId("63ee36402b7cf6513b90abaf"),
    '2': ObjectId("63ee36402b7cf6513b90abb0")
  }
}
```

```
db.assigned_to.insertMany([
  {SSN: "111-11-1111", ProjectNo: 5},
  {SSN: "111-11-1111", ProjectNo: 6},
  {SSN: "222-22-2222", ProjectNo: 6},
  {SSN: "333-33-3333", ProjectNo: 5},
  {SSN: "333-33-3333", ProjectNo: 7}
])
```



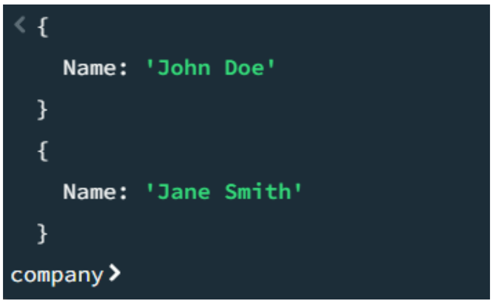
```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63ee365e2b7cf6513b90abb1"),
    '1': ObjectId("63ee365e2b7cf6513b90abb2"),
    '2': ObjectId("63ee365e2b7cf6513b90abb3"),
    '3': ObjectId("63ee365e2b7cf6513b90abb4"),
    '4': ObjectId("63ee365e2b7cf6513b90abb5")
  }
}
```

```
db.employees.find({DeptNo: "XYZ"})
```



```
> db.employees.find({DeptNo: "XYZ"})
< {
    _id: ObjectId("63ee36402b7cf6513b90abae"),
    SSN: '111-11-1111',
    Name: 'John Doe',
    DeptNo: 'XYZ'
  }
  {
    _id: ObjectId("63ee36402b7cf6513b90abb0"),
    SSN: '333-33-3333',
    Name: 'Bob Johnson',
    DeptNo: 'XYZ'
  }
```

```
db.employees.aggregate([
 {$lookup:
  {from: "assigned_to",
   localField: "SSN",
   foreignField: "SSN",
   as: "assignments"
  }
 },
 {$match:
  {"assignments.ProjectNo": 6}
 },
 {$project:
  {_id: 0, Name: 1}
 }
])
```



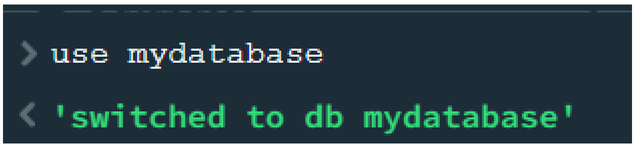
```
< {
    Name: 'John Doe'
  }
  {
    Name: 'Jane Smith'
  }
company >
```

B.2. Create the below tables, insert suitable tuples and perform the following operations using MongoDB PART (PNO, PNAME, COLOUR), SUPPLY (SNO, SNAME, PNO, ADDRESS)

a. Update the parts identifier

b. Display all suppliers who supply the part with part identifier: #PNO

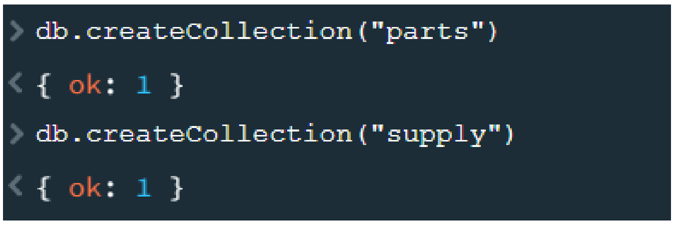
use mydatabase



```
> use mydatabase
< 'switched to db mydatabase'
```

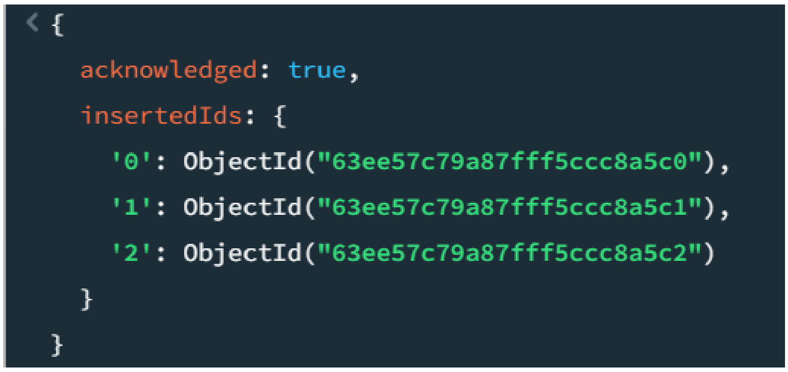
db.createCollection("parts")

db.createCollection("supply")



```
> db.createCollection("parts")
< { ok: 1 }
> db.createCollection("supply")
< { ok: 1 }
```

```
db.parts.insertMany([
 { PNO: "P1", PNAME: "Widget", COLOUR: "Blue" },
 { PNO: "P2", PNAME: "Gizmo", COLOUR: "Red" },
 { PNO: "P3", PNAME: "Thingamajig", COLOUR: "Green" }
])
```



```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee57c79a87fff5ccc8a5c0"),
      '1': ObjectId("63ee57c79a87fff5ccc8a5c1"),
      '2': ObjectId("63ee57c79a87fff5ccc8a5c2")
    }
  }
```

db.supply.insertMany([

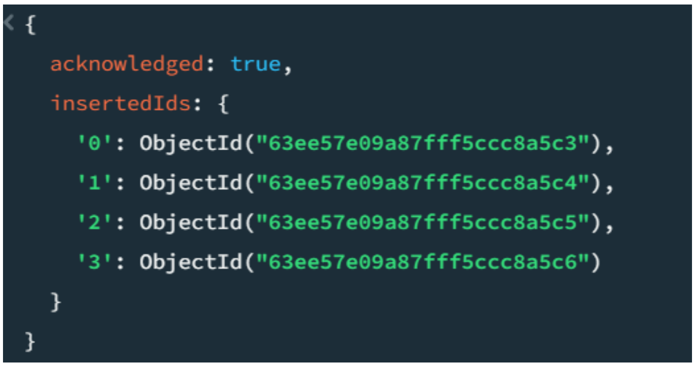
{ SNO: "S1", SNAME: "Supplier A", PNO: "P1", ADDRESS: "123 Main St." },

{ SNO: "S2", SNAME: "Supplier B", PNO: "P2", ADDRESS: "456 Elm St." },

{ SNO: "S3", SNAME: "Supplier C", PNO: "P1", ADDRESS: "789 Oak St." },

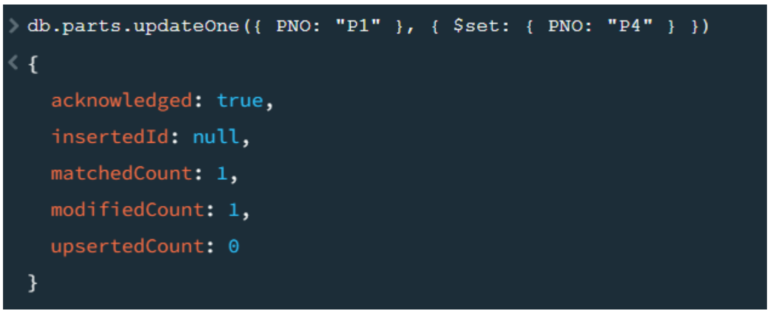
{ SNO: "S4", SNAME: "Supplier D", PNO: "P3", ADDRESS: "111 Pine St." }

])

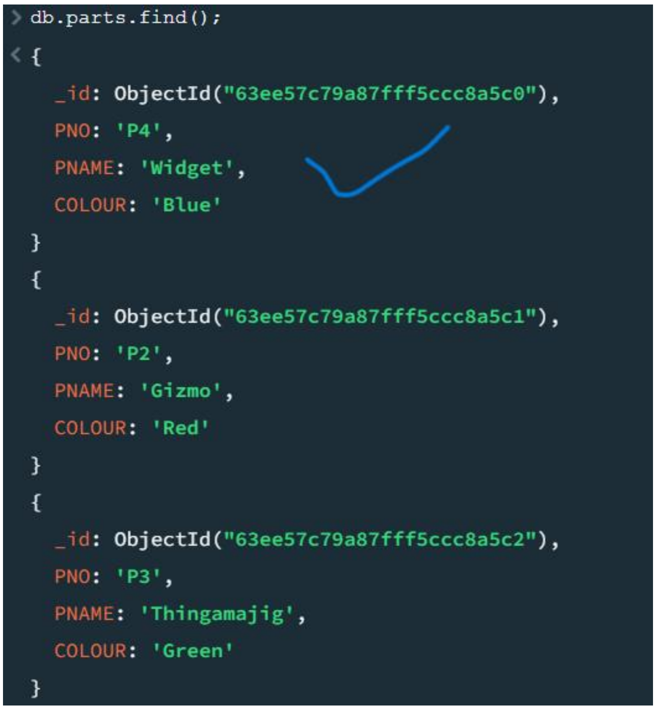


```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee57e09a87fff5ccc8a5c3"),
      '1': ObjectId("63ee57e09a87fff5ccc8a5c4"),
      '2': ObjectId("63ee57e09a87fff5ccc8a5c5"),
      '3': ObjectId("63ee57e09a87fff5ccc8a5c6")
    }
  }
```

db.parts.updateOne({ PNO: "P1" }, { $set: { PNO: "P4" } })

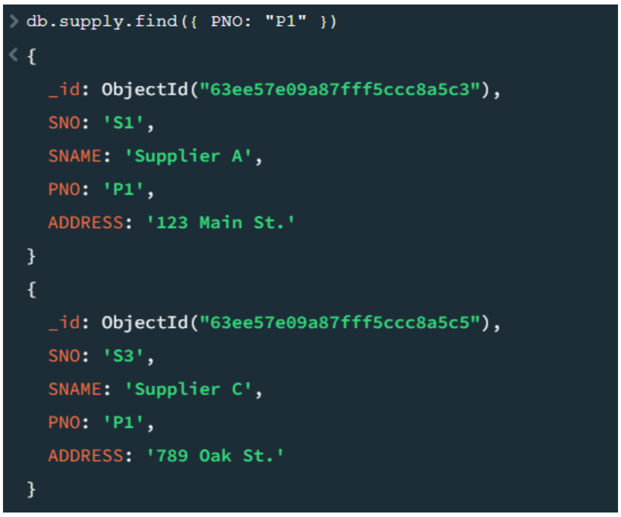


```
> db.parts.updateOne({ PNO: "P1" }, { $set: { PNO: "P4" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```



```
> db.parts.find();
< {
    _id: ObjectId("63ee57c79a87fff5ccc8a5c0"),
    PNO: 'P4',
    PNAME: 'Widget',
    COLOUR: 'Blue'
  }
  {
    _id: ObjectId("63ee57c79a87fff5ccc8a5c1"),
    PNO: 'P2',
    PNAME: 'Gizmo',
    COLOUR: 'Red'
  }
  {
    _id: ObjectId("63ee57c79a87fff5ccc8a5c2"),
    PNO: 'P3',
    PNAME: 'Thingamajig',
    COLOUR: 'Green'
  }
```

b.supply.find({ PNO: "P1" })



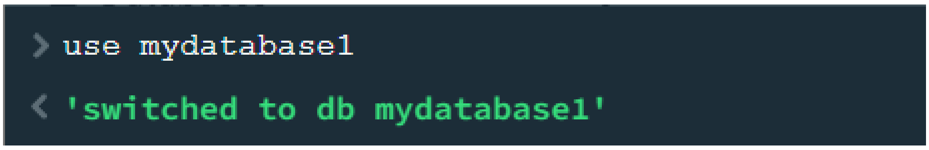
```
> db.supply.find({ PNO: "P1" })
< {
    _id: ObjectId("63ee57e09a87fff5ccc8a5c3"),
    SNO: 'S1',
    SNAME: 'Supplier A',
    PNO: 'P1',
    ADDRESS: '123 Main St.'
  }
  {
    _id: ObjectId("63ee57e09a87fff5ccc8a5c5"),
    SNO: 'S3',
    SNAME: 'Supplier C',
    PNO: 'P1',
    ADDRESS: '789 Oak St.'
  }
```

B.3. Create the below tables, insert suitable tuples and perform the following operations using MongoDB BOAT (BID, BNAME, COLOUR) RESERVES (BID, SNAME, SID, DAY)

a. Obtain the number of boats obtained by sailor :#sname

b. Retrieve boats of color :"White"

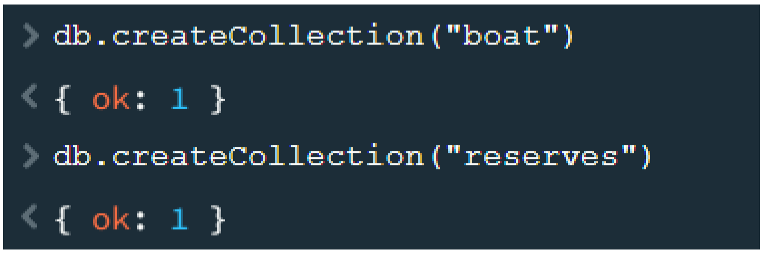
use mydatabase1 // create or use an existing database



```
> use mydatabase1
< 'switched to db mydatabase1'
```

db.createCollection("boat") // create a collection for the BOAT table

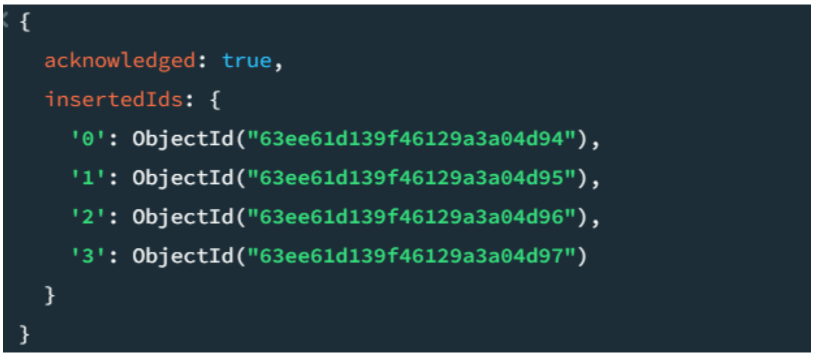
db.createCollection("reserves") // create a collection for the RESERVES table



```
> db.createCollection("boat")
< { ok: 1 }
> db.createCollection("reserves")
< { ok: 1 }
```

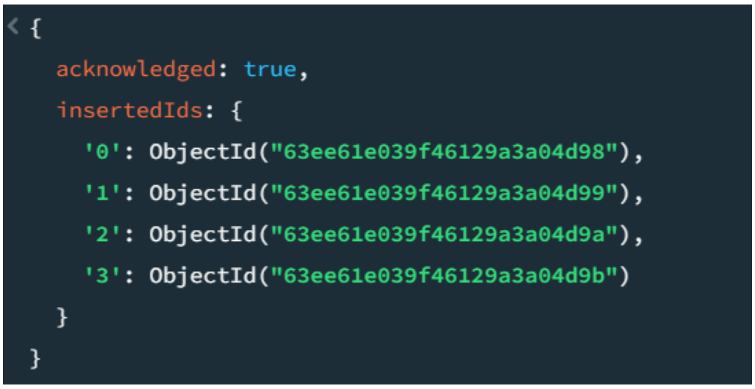
```
db.boat.insertMany([
 { BID: 1, BNAME: "Boat 1", COLOUR: "Blue" },
 { BID: 2, BNAME: "Boat 2", COLOUR: "Red" },
 { BID: 3, BNAME: "Boat 3", COLOUR: "White" },
 { BID: 4, BNAME: "Boat 4", COLOUR: "White" }
```

])



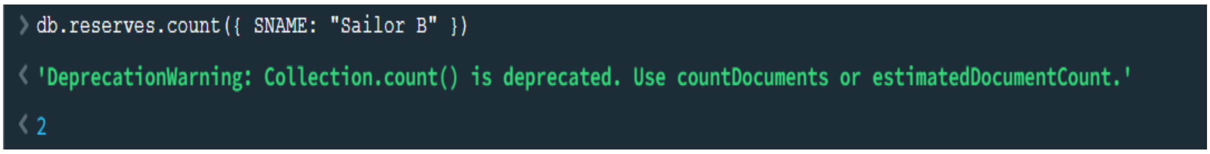
```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63ee61d139f46129a3a04d94"),
    '1': ObjectId("63ee61d139f46129a3a04d95"),
    '2': ObjectId("63ee61d139f46129a3a04d96"),
    '3': ObjectId("63ee61d139f46129a3a04d97")
  }
}
```

```
db.reserves.insertMany([
  { BID: 1, SNAME: "Sailor A", SID: 1, DAY: "2022-01-01" },
  { BID: 2, SNAME: "Sailor B", SID: 2, DAY: "2022-01-02" },
  { BID: 1, SNAME: "Sailor B", SID: 2, DAY: "2022-01-03" },
  { BID: 3, SNAME: "Sailor C", SID: 3, DAY: "2022-01-04" }
])
```



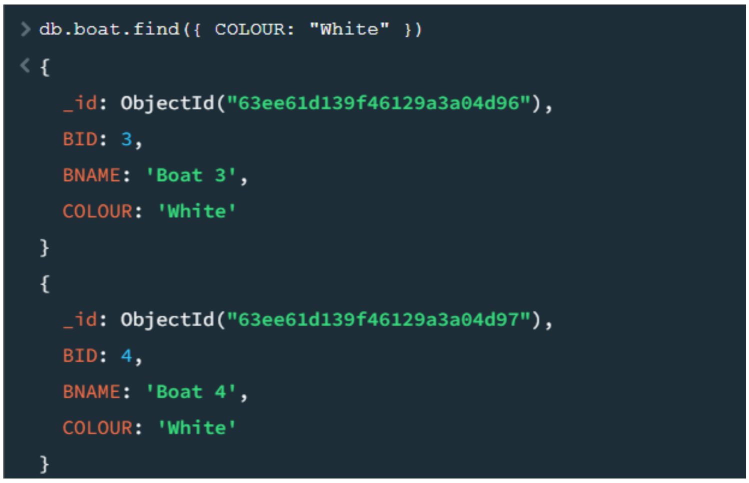
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee61e039f46129a3a04d98"),
      '1': ObjectId("63ee61e039f46129a3a04d99"),
      '2': ObjectId("63ee61e039f46129a3a04d9a"),
      '3': ObjectId("63ee61e039f46129a3a04d9b")
    }
  }
```

```
db.reserves.count({ SNAME: "Sailor B" })
```



```
> db.reserves.count({ SNAME: "Sailor B" })
< 'DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.'
< 2
```

db.boat.find({ COLOUR: "White" })



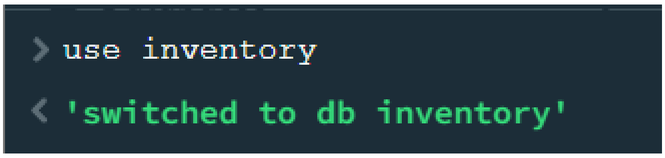
```
> db.boat.find({ COLOUR: "White" })
< {
    _id: ObjectId("63ee61d139f46129a3a04d96"),
    BID: 3,
    BNAME: 'Boat 3',
    COLOUR: 'White'
  }
  {
    _id: ObjectId("63ee61d139f46129a3a04d97"),
    BID: 4,
    BNAME: 'Boat 4',
    COLOUR: 'White'
  }
```

B.4. Create the below tables, insert suitable tuples and perform the following operations using MongoDB PART (PNO, PNAME, COLOUR) SHIPMENT (PNO, WNO, WNAME, QUANTITY, DATE)

a. Find the parts shipped from warehouse :”Wname”

b. List the total quantity supplied from each warehouse

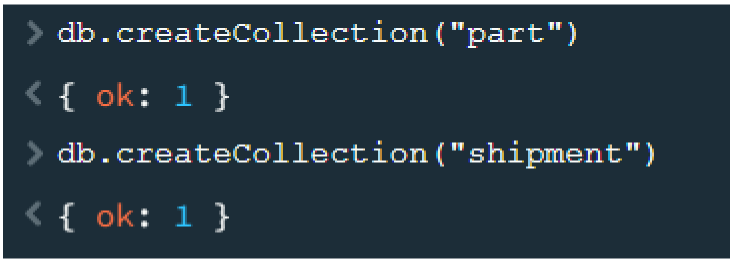
use inventory



```
> use inventory
< 'switched to db inventory'
```

db.createCollection("part")

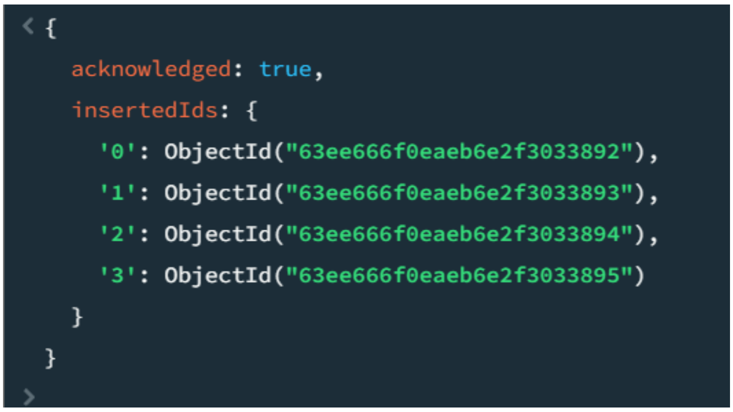
db.createCollection("shipment")



```
> db.createCollection("part")
< { ok: 1 }
> db.createCollection("shipment")
< { ok: 1 }
```

```
db.part.insertMany([
  { PNO: 1, PNAME: "Screw", COLOUR: "Silver" },
  { PNO: 2, PNAME: "Nut", COLOUR: "Gold" },
  { PNO: 3, PNAME: "Bolt", COLOUR: "Black" },
  { PNO: 4, PNAME: "Washer", COLOUR: "White" }
```

])



```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee666f0eaeb6e2f3033892"),
      '1': ObjectId("63ee666f0eaeb6e2f3033893"),
      '2': ObjectId("63ee666f0eaeb6e2f3033894"),
      '3': ObjectId("63ee666f0eaeb6e2f3033895")
    }
  }
>
```

db.shipment.insertMany([

{ PNO: 1, WNO: 1, WNAME: "Warehouse A", QUANTITY: 100, DATE: ISODate("2023-01-01") },

{ PNO: 1, WNO: 2, WNAME: "Warehouse B", QUANTITY: 200, DATE: ISODate("2023-01-02") },

{ PNO: 2, WNO: 1, WNAME: "Warehouse A", QUANTITY: 150, DATE: ISODate("2023-01-03") },

{ PNO: 2, WNO: 2, WNAME: "Warehouse B", QUANTITY: 50, DATE: ISODate("2023-01-04") },

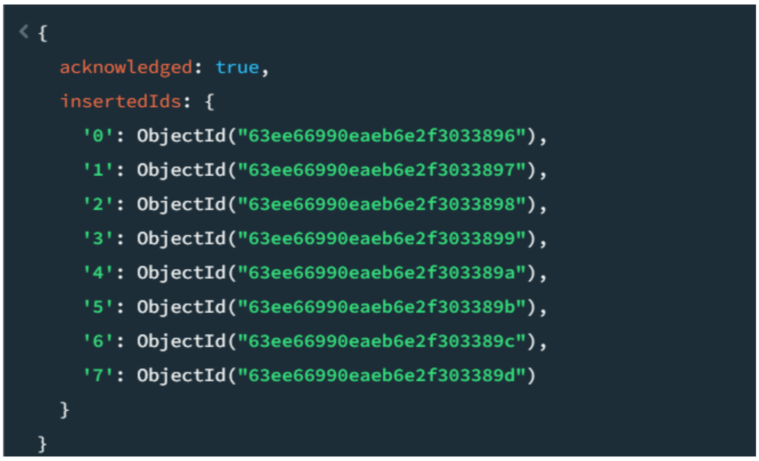
{ PNO: 3, WNO: 1, WNAME: "Warehouse A", QUANTITY: 300, DATE: ISODate("2023-01-05") },

{ PNO: 3, WNO: 3, WNAME: "Warehouse C", QUANTITY: 100, DATE: ISODate("2023-01-06") },

{ PNO: 4, WNO: 2, WNAME: "Warehouse B", QUANTITY: 75, DATE: ISODate("2023-01-07") },

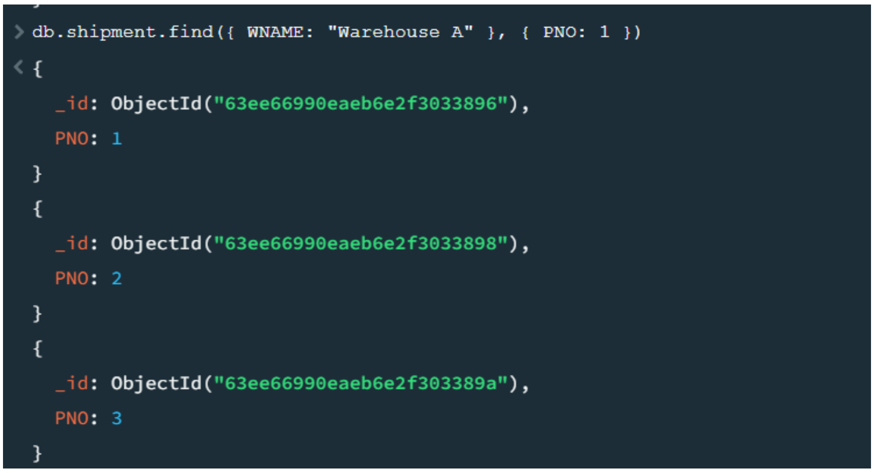
{ PNO: 4, WNO: 3, WNAME: "Warehouse C", QUANTITY: 125, DATE: ISODate("2023-01-08") }

])



```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee66990eaeb6e2f3033896"),
      '1': ObjectId("63ee66990eaeb6e2f3033897"),
      '2': ObjectId("63ee66990eaeb6e2f3033898"),
      '3': ObjectId("63ee66990eaeb6e2f3033899"),
      '4': ObjectId("63ee66990eaeb6e2f303389a"),
      '5': ObjectId("63ee66990eaeb6e2f303389b"),
      '6': ObjectId("63ee66990eaeb6e2f303389c"),
      '7': ObjectId("63ee66990eaeb6e2f303389d")
    }
  }
```

db.shipment.find({ WNAME: "Warehouse A" }, { PNO: 1 })

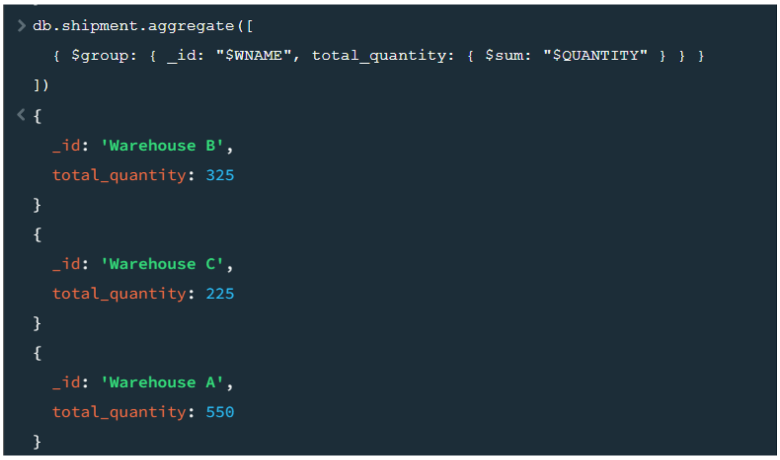


```
> db.shipment.find({ WNAME: "Warehouse A" }, { PNO: 1 })
< {
    _id: ObjectId("63ee66990eaeb6e2f3033896"),
    PNO: 1
  }
  {
    _id: ObjectId("63ee66990eaeb6e2f3033898"),
    PNO: 2
  }
  {
    _id: ObjectId("63ee66990eaeb6e2f303389a"),
    PNO: 3
  }
```

db.shipment.aggregate([

{ $group: { _id: "$WNAME", total_quantity: { $sum: "$QUANTITY" } } }

])



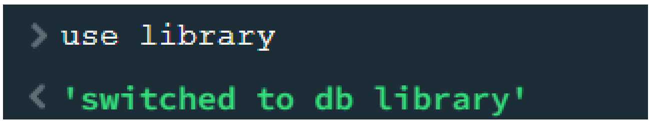
```
> db.shipment.aggregate([
    { $group: { _id: "$WNAME", total_quantity: { $sum: "$QUANTITY" } } }
  ])
< {
    _id: 'Warehouse B',
    total_quantity: 325
  }
  {
    _id: 'Warehouse C',
    total_quantity: 225
  }
  {
    _id: 'Warehouse A',
    total_quantity: 550
  }
```

B.5. Create the below tables, insert suitable tuples and perform the following operations using MongoDB BOOK (ISBN, TITLE, AUTHOR, PUBLISHER) BORROW (ISBN,USN,DATE)

a. Obtain the name of the student who has borrowed the book bearing ISBN "123‟.

b. Obtain the Names of students who have borrowed database books.

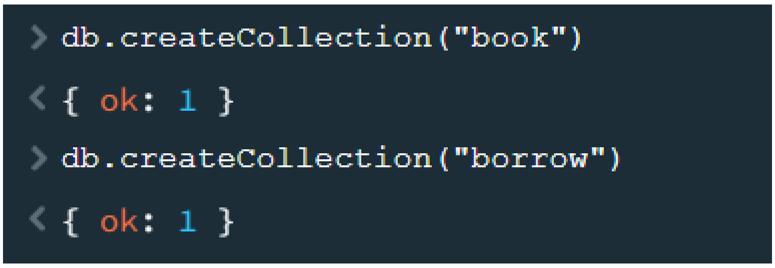
use library



```
> use library
< 'switched to db library'
```

db.createCollection("book")

db.createCollection("borrow")



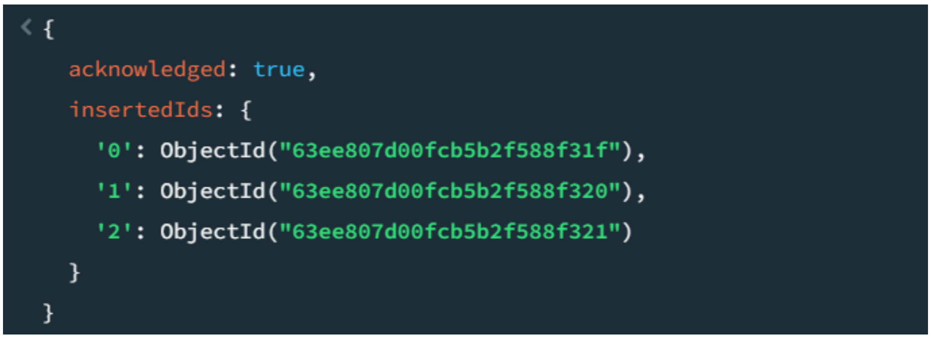
```
> db.createCollection("book")
< { ok: 1 }
> db.createCollection("borrow")
< { ok: 1 }
```

db.book.insertMany([

{ ISBN: "123", TITLE: "Introduction to Databases", AUTHOR: "John Smith", PUBLISHER: "Pearson" },

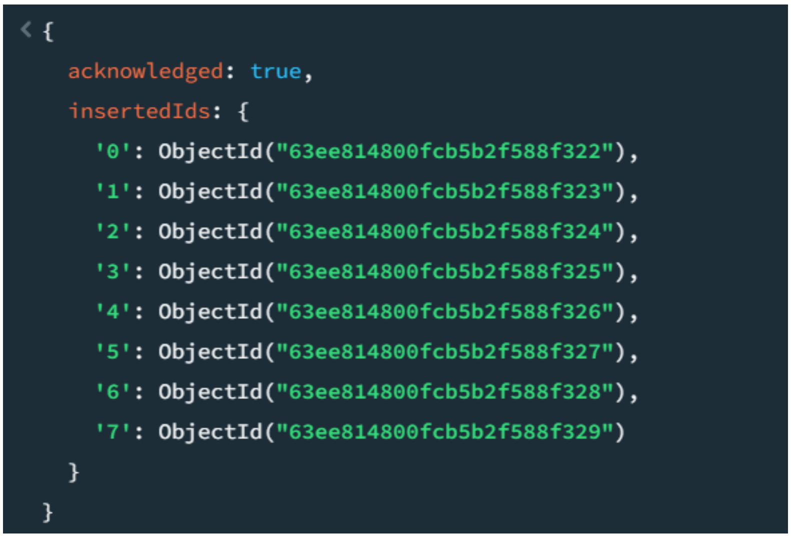
{ ISBN: "456", TITLE: "Data Structures and Algorithms", AUTHOR: "Jane Doe", PUBLISHER: "O'Reilly" },

```
  { ISBN: "789", TITLE: "Operating Systems", AUTHOR: "Bob Johnson",
PUBLISHER: "McGraw Hill" }
])
```



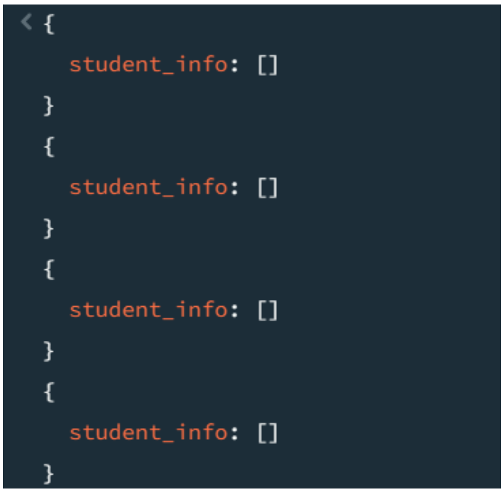
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee807d00fcb5b2f588f31f"),
      '1': ObjectId("63ee807d00fcb5b2f588f320"),
      '2': ObjectId("63ee807d00fcb5b2f588f321")
    }
  }
```

```
db.borrow.insertMany([
  { ISBN: "123", USN: "S001", DATE: ISODate("2023-01-01") },
  { ISBN: "456", USN: "S002", DATE: ISODate("2023-01-02") },
  { ISBN: "123", USN: "S003", DATE: ISODate("2023-01-03") },
  { ISBN: "789", USN: "S004", DATE: ISODate("2023-01-04") },
  { ISBN: "123", USN: "S005", DATE: ISODate("2023-01-05") },
  { ISBN: "456", USN: "S006", DATE: ISODate("2023-01-06") },
  { ISBN: "789", USN: "S007", DATE: ISODate("2023-01-07") },
  { ISBN: "123", USN: "S008", DATE: ISODate("2023-01-08") }
])
```



```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("63ee814800fcb5b2f588f322"),
      '1': ObjectId("63ee814800fcb5b2f588f323"),
      '2': ObjectId("63ee814800fcb5b2f588f324"),
      '3': ObjectId("63ee814800fcb5b2f588f325"),
      '4': ObjectId("63ee814800fcb5b2f588f326"),
      '5': ObjectId("63ee814800fcb5b2f588f327"),
      '6': ObjectId("63ee814800fcb5b2f588f328"),
      '7': ObjectId("63ee814800fcb5b2f588f329")
    }
  }
```

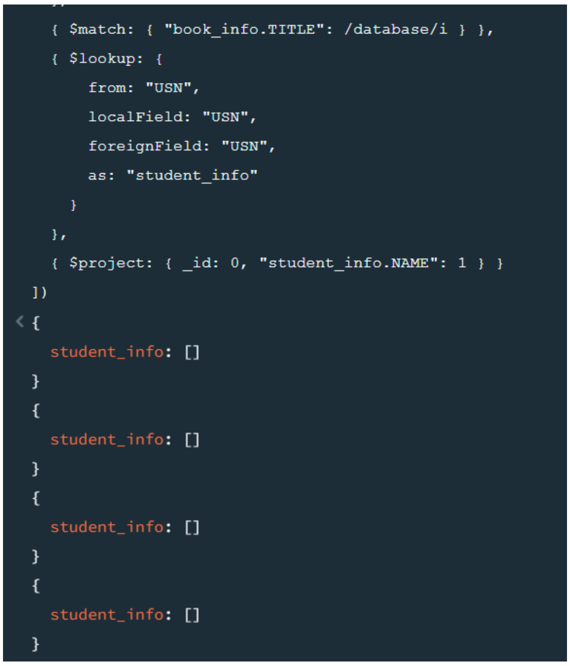
```
db.borrow.aggregate([
  { $match: { ISBN: "123" } },
  { $lookup: {
      from: "student",
      localField: "USN",
      foreignField: "USN",
      as: "student_info"
    }
  },
  { $project: { _id: 0, "student_info.NAME": 1 } }
])
```



```
< {
    student_info: []
  }
  {
    student_info: []
  }
  {
    student_info: []
  }
  {
    student_info: []
  }
```

```
db.borrow.aggregate([
 { $lookup: {
     from: "book",
     localField: "ISBN",
     foreignField: "ISBN",
     as: "book_info"
   }
 },
 { $match: { "book_info.TITLE": /database/i } },
 { $lookup: {
     from: "USN",
     localField: "USN",
     foreignField: "USN",
     as: "student_info"
   }
 },
 { $project: { _id: 0, "student_info.NAME": 1 } }
```

])



```
  { $match: { "book_info.TITLE": /database/i } },
  { $lookup: {
      from: "USN",
      localField: "USN",
      foreignField: "USN",
      as: "student_info"
    }
  },
  { $project: { _id: 0, "student_info.NAME": 1 } }
])
< {
  student_info: []
}
{
  student_info: []
}
{
  student_info: []
}
{
  student_info: []
}
```

PART – C
Open Ended Project

Paying Guest Management System

Paying Guest Management System in PHP is web based application.This project used to manage the student,employee , room details. Paying guest management system developed using PHP and MYSQL

Project Name	Hostel Management System Project in PHP
Language Used	PHP5.6, PHP7.x
Database	MySQL 5.x
User Interface Design	HTML,JAVASCRIPT
Web Browser	Mozilla, Google Chrome, IE8, OPERA
Software	XAMPP / Wamp / Mamp/ Lamp (anyone)

PHP is a popular server-side scripting language used to create dynamic websites and web applications. It is widely used for web development, content management systems, server-side scripting, and can be integrated with databases to create dynamic web applications. Its simplicity, flexibility, and support for a wide range of platforms and operating systems make it a preferred choice for our project.

MySQL is a popular open-source relational database management system used to store and manage data in web applications. It is widely used in web development, including for content management systems, e-commerce websites, and online reservation systems. MySQL can store various types of data, including user data, product information, and order details, and it allows for quick and easy retrieval of data, making applications faster and more efficient. Its reliability, scalability, and ease of use make it a preferred choice for many developers.

C.1 Description of Database

Table 1:-Admin

Name	type
id	Int(5)
username	Varchar(25)
email	Varchar(50)
password	Varchar(50)
Reg_date	timestamp
Updation_date	date

Table 2:-Adminlog

Name	type
id	Int(5)
adminid	Int(5)
ip	Varbinary(16)
logintime	timestamp

Table 3:-courses

Name	type
id	Int(5)
Course_code	Varchar(5)
Course_sn	char(7)
Course_fn	Varchar(50)
Posting_date	timestamp

Table 4:-room

Name	type
id	Int(5)
seater	Int(5)
Room_no	Int(5)
fees	Int(5)

Table 5:-states

Name	type
id	Int(11)
state	Varchar(75)

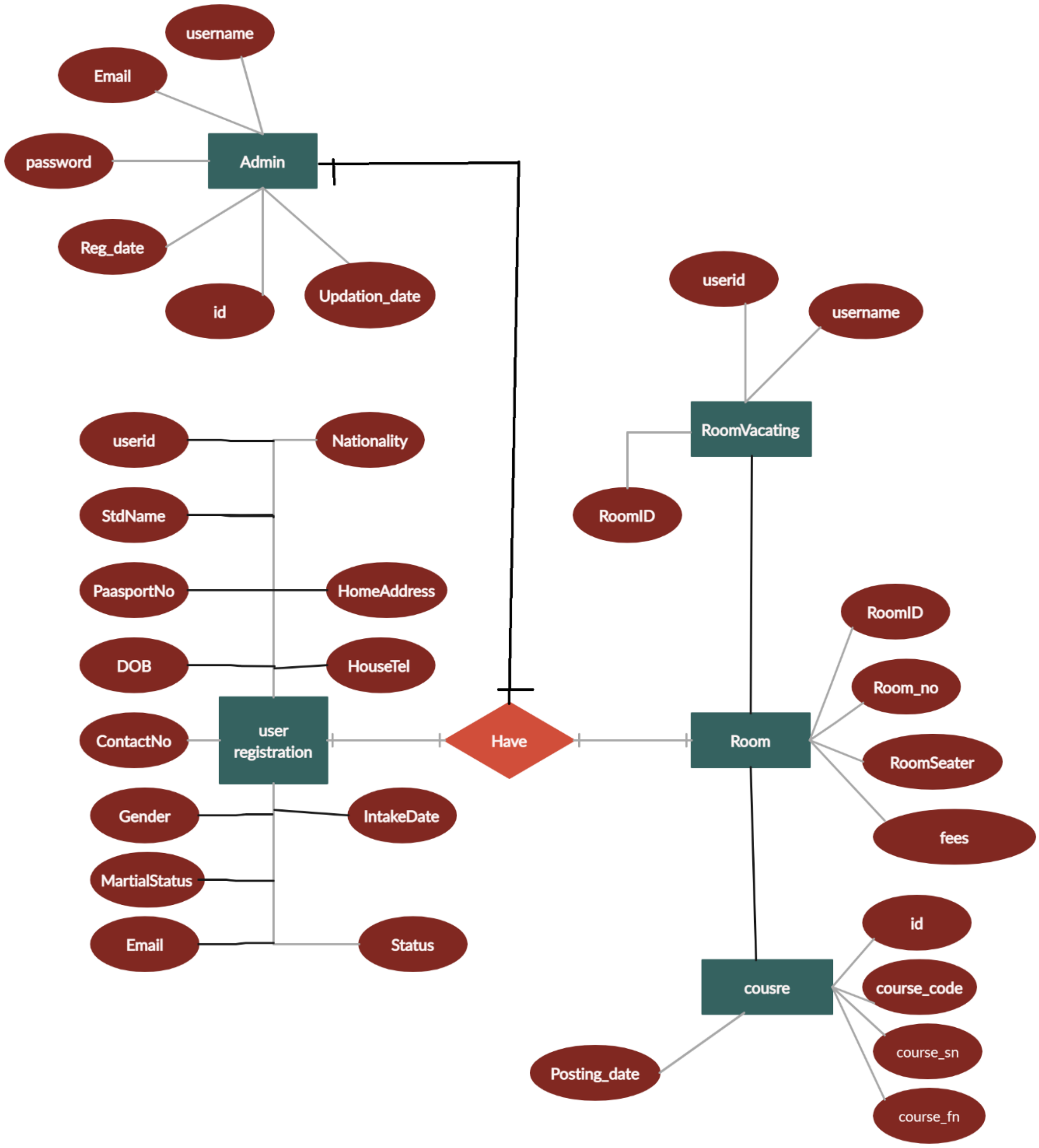
Table 6:-userlog

Name	type
id	Int(5)
usernameid	Int(5)
userEmail	Varchar(50)
userIp	Varbinary(16)
city	Chat(50)
country	Char(50)
LoginTime	timestamp

Table 7:-userregistration

Name	type
id	Int(5)
regNO	Varchar(5)
firstName	Char(25)
middleName	Char(25)
LastName	Char(25)
gender	char
contactno	Int(10)
email	Varchar(50)
password	Varchar(50)
regDate	timestamp

C.2 ER Diagram



C.3 Output

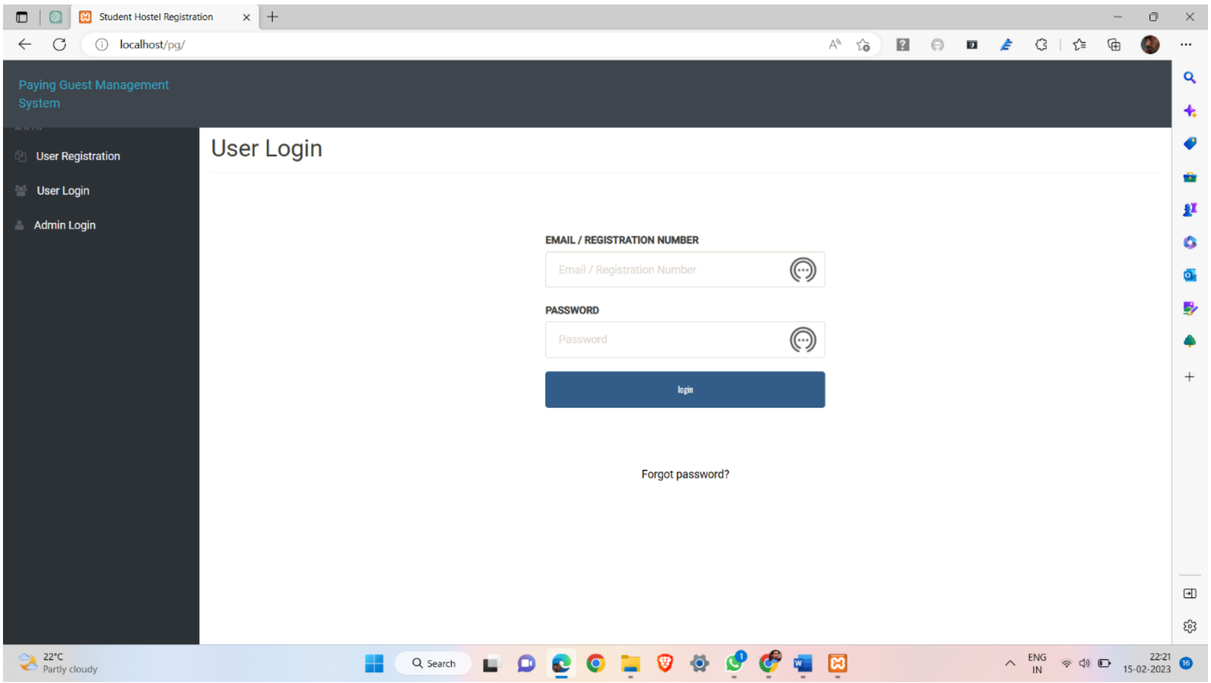


Fig c.3.1 home page

In home page user login, admin login and user registration are displayed.

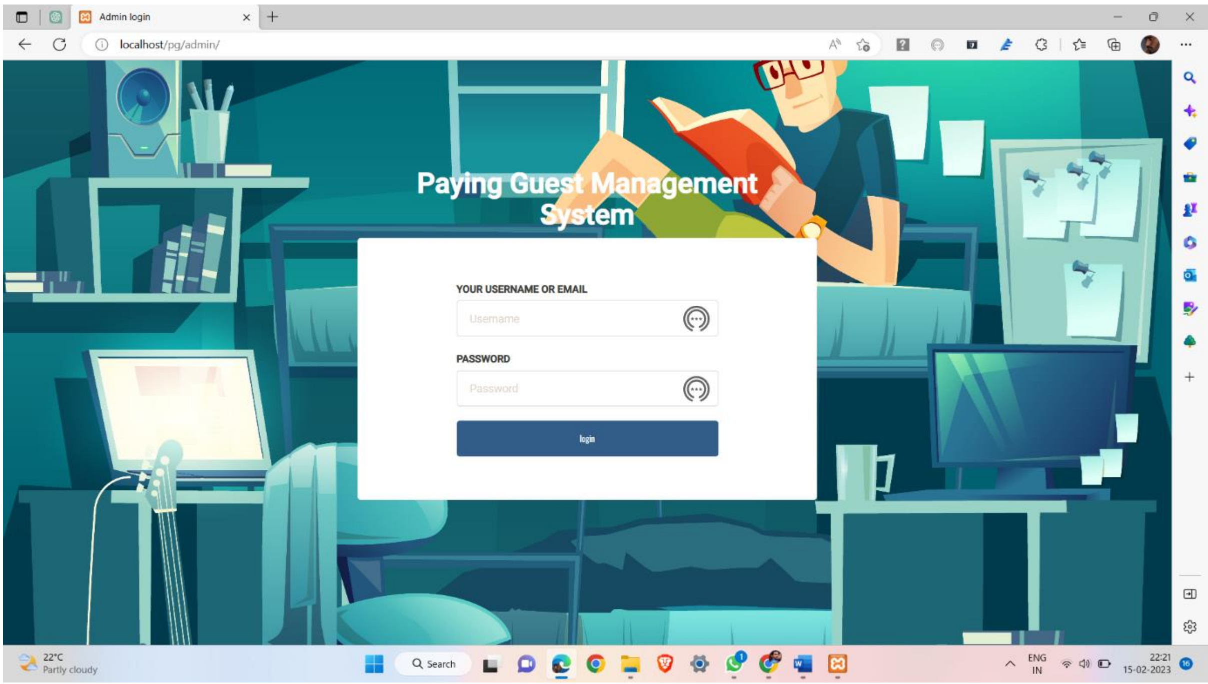


Fig c.3.2 admin login page

In admin login page, admin has to enter his user name or email and password.

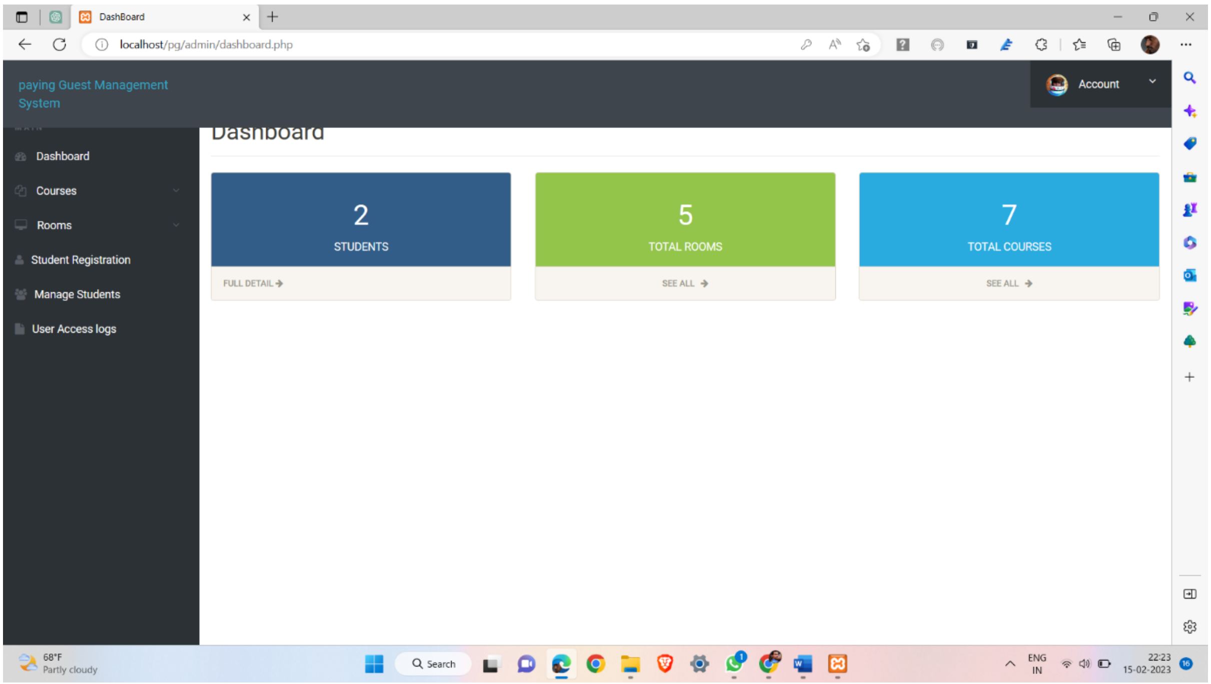
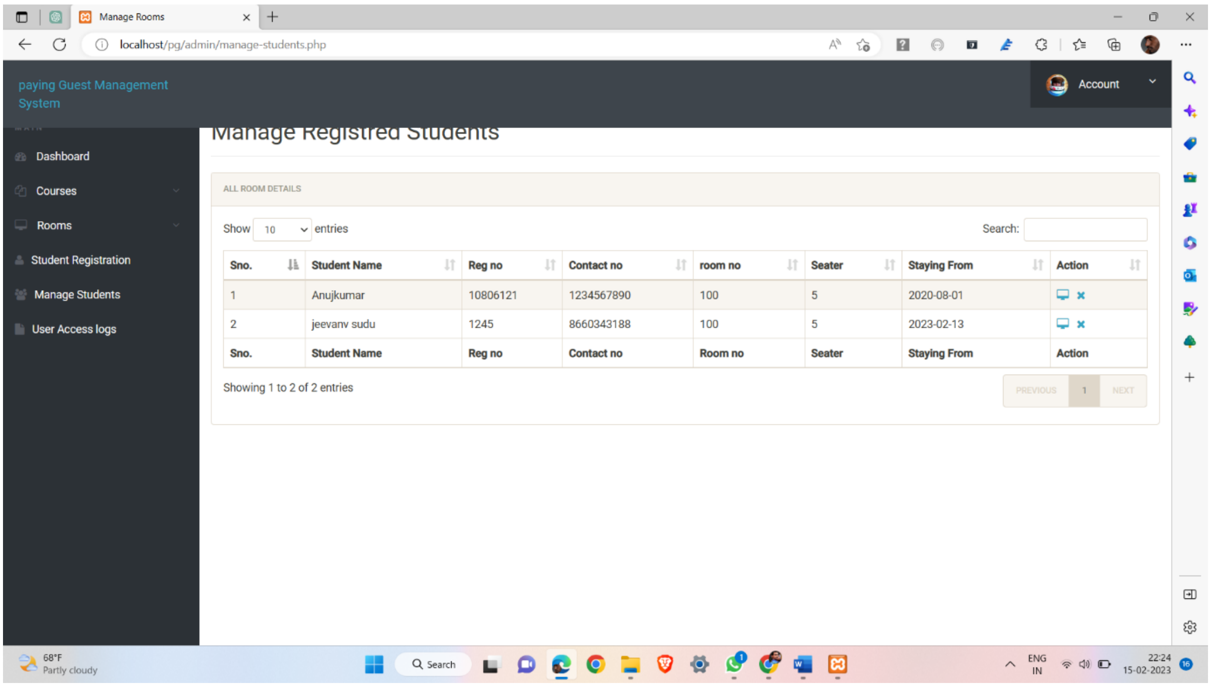


Fig c.3.3 admin dashboard

In admin dashboard , we get course details, number of rooms and number of student and student registration.

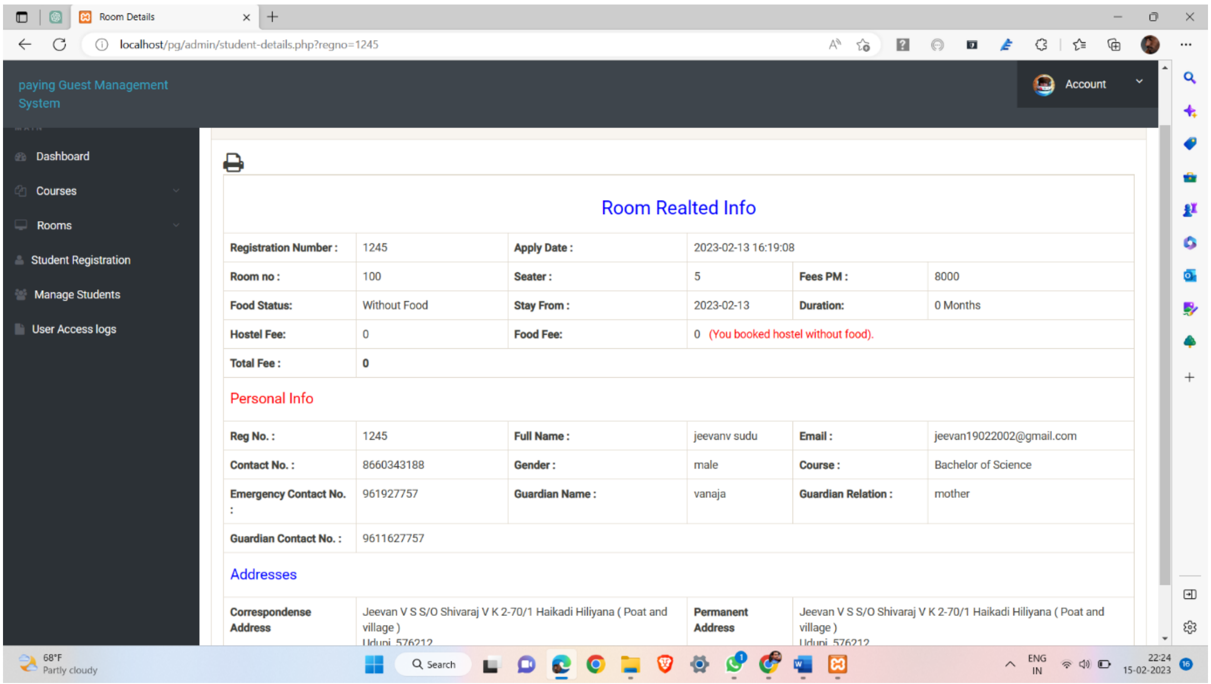


Sno.	Student Name	Reg no	Contact no	room no	Seater	Staying From	Action
1	Anujkumar	10806121	1234567890	100	5	2020-08-01	
2	jeevanv sudu	1245	8660343188	100	5	2023-02-13	
Sno.	Student Name	Reg no	Contact no	Room no	Seater	Staying From	Action

Showing 1 to 2 of 2 entries

Fig c.3.4 admin manage Registed Student

In admin manage registed student , we get the all the student list and their details.



Room Realted Info

Registration Number :	1245	Apply Date :	2023-02-13 16:19:08		
Room no :	100	Seater :	5	Fees PM :	8000
Food Status:	Without Food	Stay From :	2023-02-13	Duration:	0 Months
Hostel Fee:	0	Food Fee:	0 (You booked hostel without food).		
Total Fee :	**0**				

Personal Info

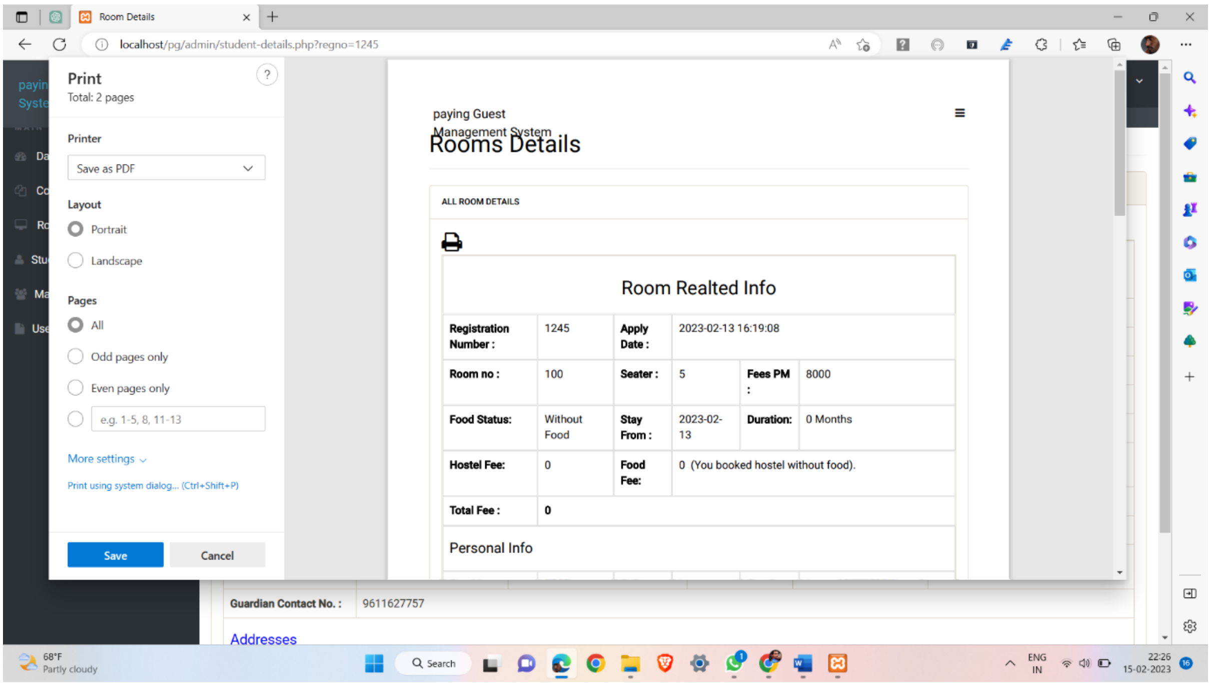
Reg No. :	1245	Full Name :	jeevanv sudu	Email :	jeevan19022002@gmail.com
Contact No. :	8660343188	Gender :	male	Course :	Bachelor of Science
Emergency Contact No. :	961927757	Guardian Name :	vanaja	Guardian Relation :	mother
Guardian Contact No. :	9611627757				

Addresses

Correspondense Address	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hiliyana (Poat and village) Udupi 576212	Permanent Address	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hiliyana (Poat and village) Udupi 576212

Fig c.3.5 Room Realted info

By click the student tab , we the details of particular student details.


Room Details
localhost/pg/admin/student-details.php?regno=1245
?
More settings
Print using system dialog... (Ctrl+Shift+P)
Save
Cancel
68°F
Partly cloudy
Search
ENG
IN
22:26
15-02-2023

Print
Total: 2 pages

Printer: Save as PDF

Layout
● Portrait
○ Landscape

Pages
● All
○ Odd pages only
○ Even pages only
○ e.g. 1-5, 8, 11-13

paying Guest Management System

Rooms Details

ALL ROOM DETAILS

Room Realted Info

Registration Number :	1245	Apply Date :	2023-02-13 16:19:08		
Room no :	100	Seater :	5	Fees PM :	8000
Food Status:	Without Food	Stay From :	2023-02-13	Duration:	0 Months
Hostel Fee:	0	Food Fee:	0 (You booked hostel without food).		
Total Fee :	**0**				

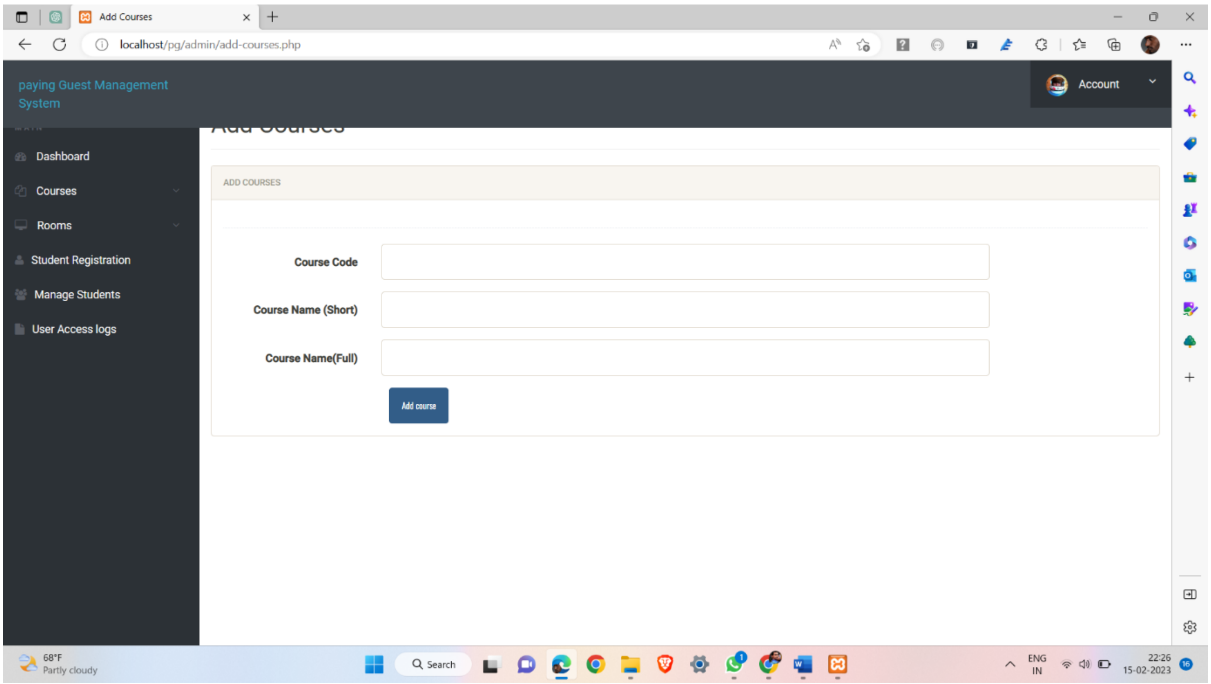
Personal Info

Guardian Contact No. :	9611627757

Addresses

Fig c.3.6 Room Details

By clicking the print button , we can get the details of students in pdf form.



Add Courses

ADD COURSES

Course Code

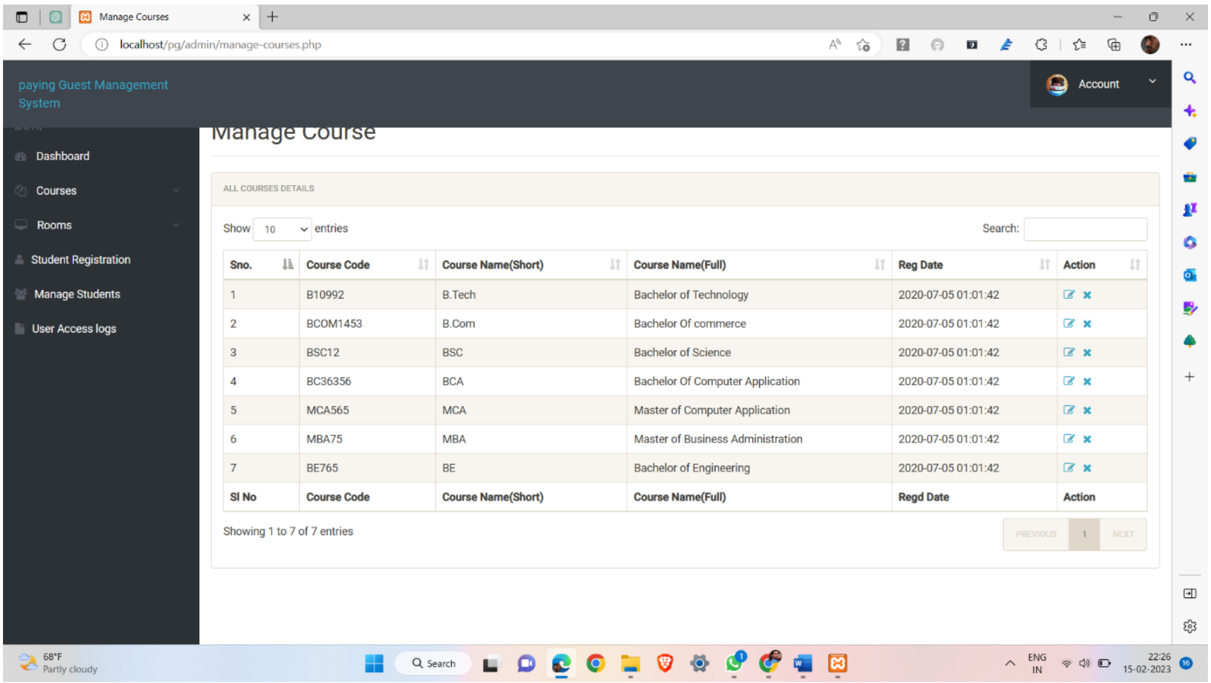
Course Name (Short)

Course Name(Full)

Add course

Fig c.3.7 add course

We can add the course details in add course column.



Manage Course

ALL COURSES DETAILS

Show 10 entries Search:

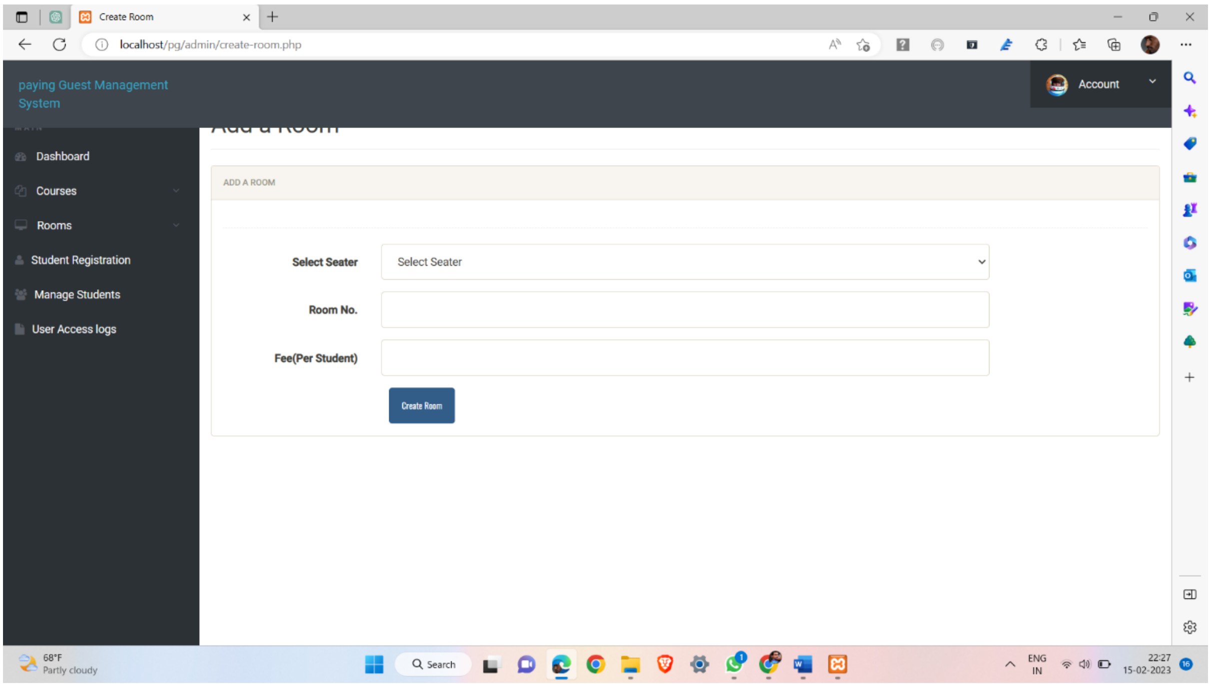
Sno.	Course Code	Course Name(Short)	Course Name(Full)	Reg Date	Action
1	B10992	B.Tech	Bachelor of Technology	2020-07-05 01:01:42	✎ ✖
2	BCOM1453	B.Com	Bachelor Of commerce	2020-07-05 01:01:42	✎ ✖
3	BSC12	BSC	Bachelor of Science	2020-07-05 01:01:42	✎ ✖
4	BC36356	BCA	Bachelor Of Computer Application	2020-07-05 01:01:42	✎ ✖
5	MCA565	MCA	Master of Computer Application	2020-07-05 01:01:42	✎ ✖
6	MBA75	MBA	Master of Business Administration	2020-07-05 01:01:42	✎ ✖
7	BE765	BE	Bachelor of Engineering	2020-07-05 01:01:42	✎ ✖
Sl No	Course Code	Course Name(Short)	Course Name(Full)	Regd Date	Action

Showing 1 to 7 of 7 entries

PREVIOUS 1 NEXT

Fig c.3.8 view course

We can see the total course in the manage course.



Add a Room

ADD A ROOM

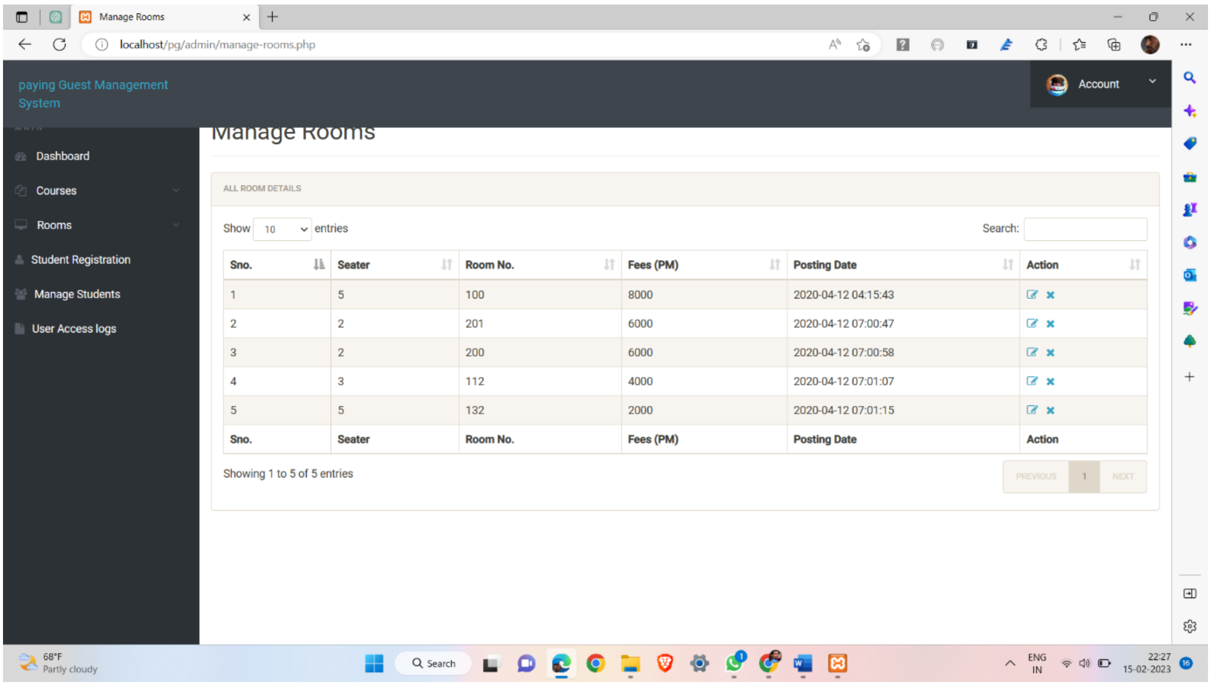
Select Seater: Select Seater

Room No.:

Fee(Per Student):

Fig c.3.9 add room

We can add new room , and room number and fees for room depending on the room sharing.



Manage Rooms

ALL ROOM DETAILS

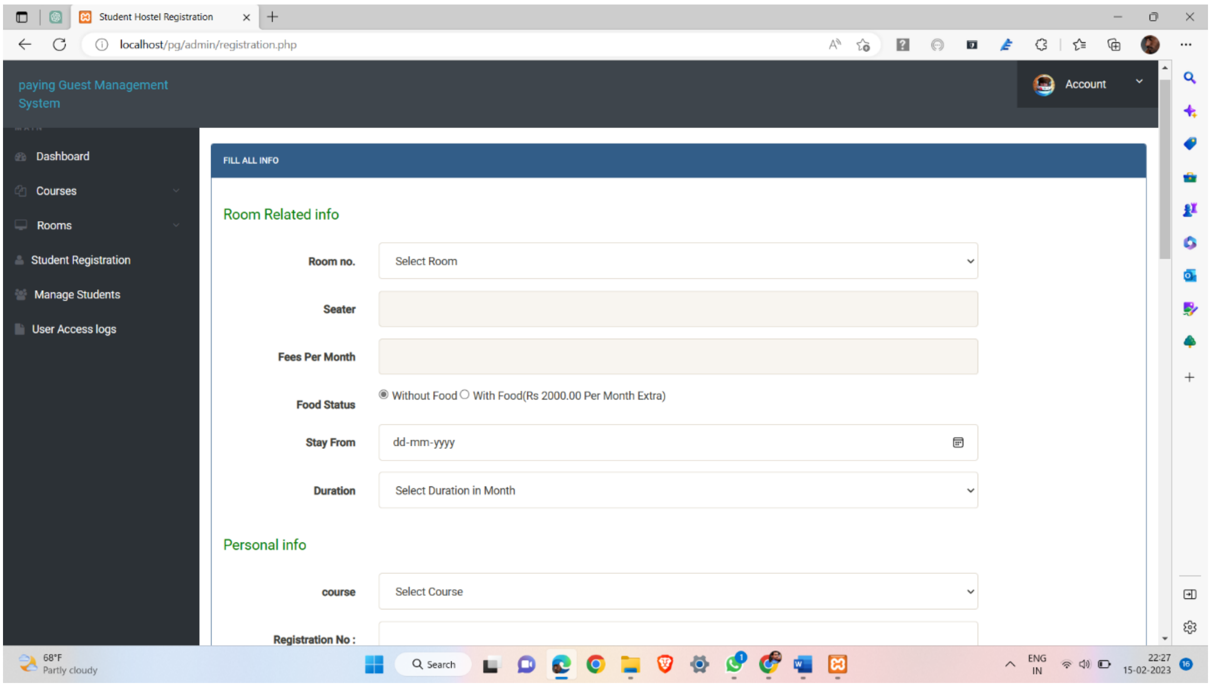
Show 10 entries

Sno.	Seater	Room No.	Fees (PM)	Posting Date	Action
1	5	100	8000	2020-04-12 04:15:43	✎ ✕
2	2	201	6000	2020-04-12 07:00:47	✎ ✕
3	2	200	6000	2020-04-12 07:00:58	✎ ✕
4	3	112	4000	2020-04-12 07:01:07	✎ ✕
5	5	132	2000	2020-04-12 07:01:15	✎ ✕
Sno.	Seater	Room No.	Fees (PM)	Posting Date	Action

Showing 1 to 5 of 5 entries

Fig c.3.10 view room

We can get list of room , number of seater , fees and we can also alter it.



FILL ALL INFO

Room Related info

Room no. Select Room

Seater

Fees Per Month

Food Status ● Without Food ○ With Food(Rs 2000.00 Per Month Extra)

Stay From dd-mm-yyyy

Duration Select Duration in Month

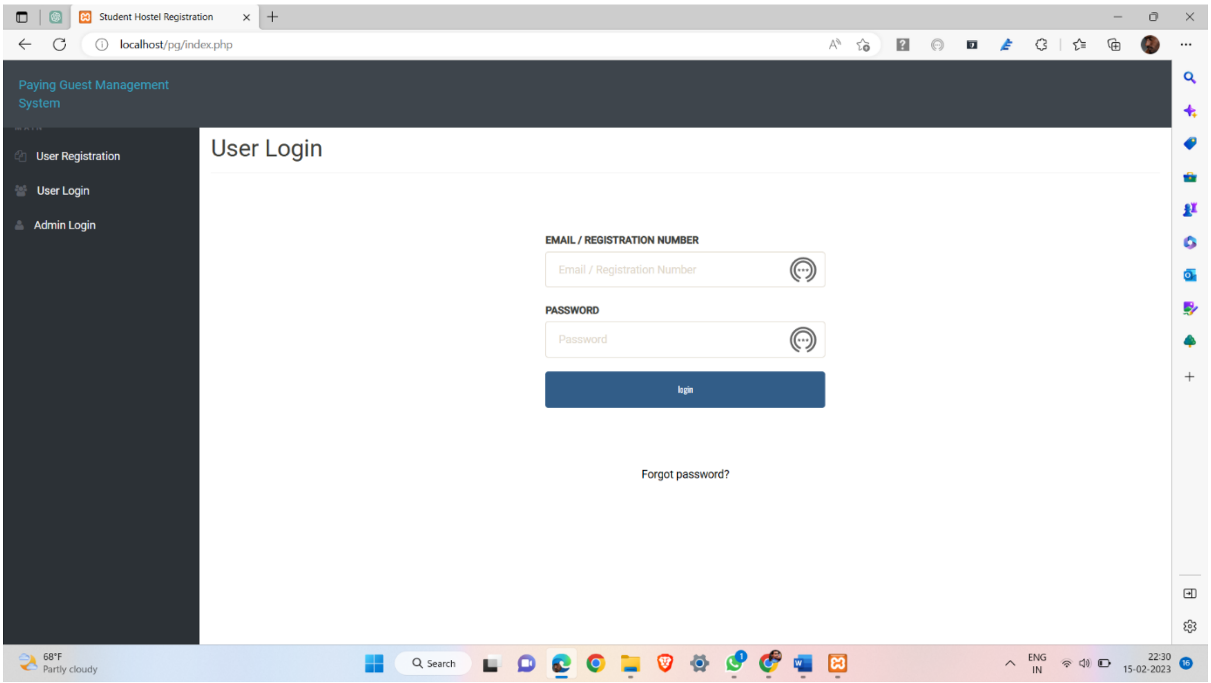
Personal info

course Select Course

Registration No :

Fig c.3.11 Student registration

We can do student registration like room no, date of join , period of living , and personal information like course , registration no, name , places , father and mother , email, phone number and so on



User Login

EMAIL / REGISTRATION NUMBER

Email / Registration Number

PASSWORD

Password

Fig c.3.12 user login

In user login , user has to enter the email address and phone number as the password for view their details.

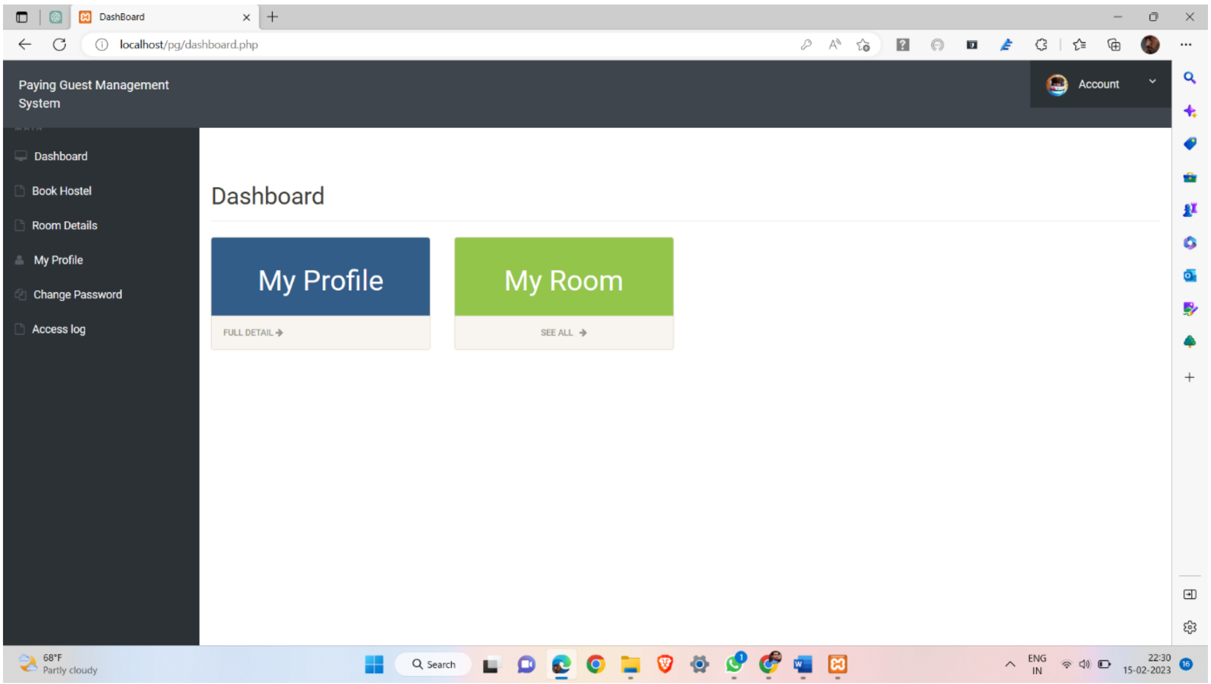
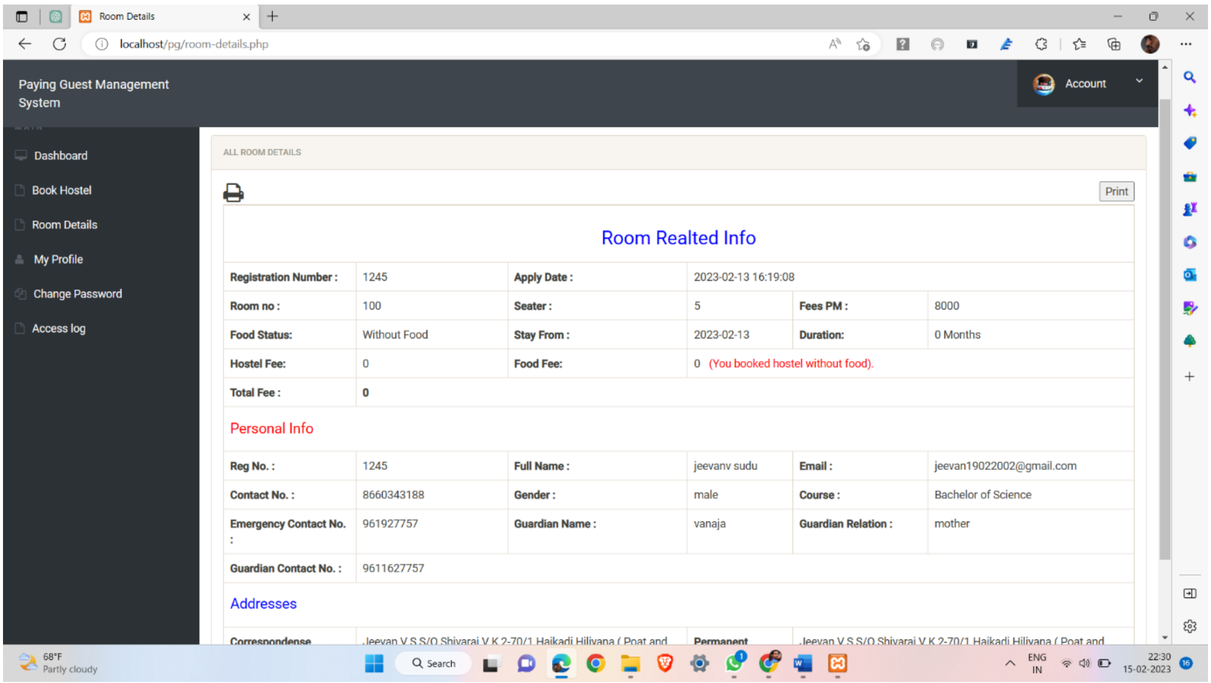


Fig c.3.13 user dashboard

After user login, we get user dashboard in that we get user profile and his room details.



Room Realted Info

Registration Number :	1245	**Apply Date :**	2023-02-13 16:19:08		
Room no :	100	**Seater :**	5	**Fees PM :**	8000
Food Status:	Without Food	**Stay From :**	2023-02-13	**Duration:**	0 Months
Hostel Fee:	0	**Food Fee:**	0 (You booked hostel without food).		
Total Fee :	**0**				

Personal Info

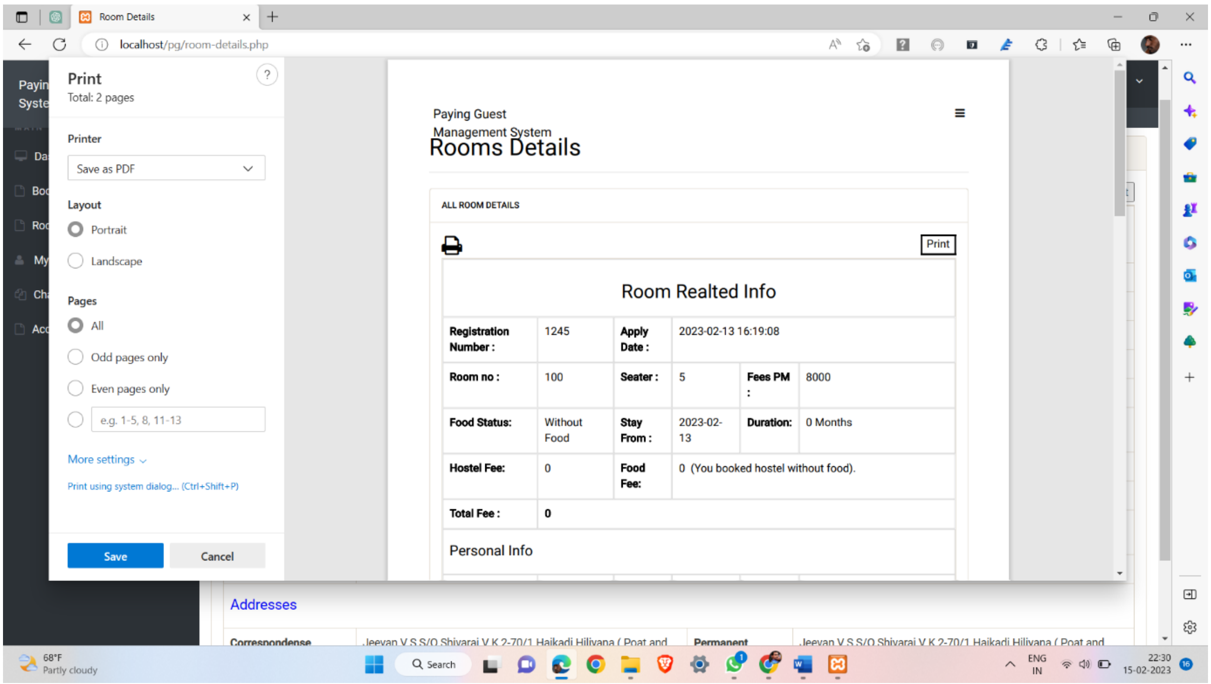
Reg No. :	1245	**Full Name :**	jeevanv sudu	**Email :**	jeevan19022002@gmail.com
Contact No. :	8660343188	**Gender :**	male	**Course :**	Bachelor of Science
Emergency Contact No. :	961927757	**Guardian Name :**	vanaja	**Guardian Relation :**	mother
Guardian Contact No. :	9611627757				

Addresses

Correspondense	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hilivana (Poat and	**Permanent**	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hilivana (Poat and

Fig c.3.14 student Details

By clicking on my profile , user can get his all details to view.



Paying Guest
Management System
Rooms Details

ALL ROOM DETAILS

Print

Room Realted Info

Registration Number :	1245	**Apply Date :**	2023-02-13 16:19:08		
Room no :	100	**Seater :**	5	**Fees PM :**	8000
Food Status:	Without Food	**Stay From :**	2023-02-13	**Duration:**	0 Months
Hostel Fee:	0	**Food Fee:**	0 (You booked hostel without food).		
Total Fee :	**0**				

Personal Info

Addresses

Correspondense	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hilivana (Post and	**Permanent**	Jeevan V S S/O Shivaraj V K 2-70/1 Haikadi Hilivana (Post and

Fig c.3.15 student print

In user view , we print user details in pdf format and can be shared to everyone.