

Rithvik Ravilla
106121104

Lab 1 Queries:

-- 2)

Use the DDL commands performs the following operation:

(i)

Create a table called EMP with the following structure.

```
create table EMP(
```

```
    EMPNO INT (6),
```

```
    ENAME VARCHAR (20),
```

```
    JOB VARCHAR (10),
```

```
    DEPTNO INT (3),
```

```
    SAL DECIMAL (7,2)
```

```
);
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	YES		NULL	
ENAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPTNO	int	YES		NULL	
SAL	decimal(7,2)	YES		NULL	

-- 2) (ii)

Add a column experience to the EMP table. Experience numeric null allowed.

```
ALTER TABLE EMP ADD EXPERIENCE VARCHAR (5);
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	YES		NULL	
ENAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPTNO	int	YES		NULL	
SAL	decimal(7,2)	YES		NULL	
EXPERIENCE	varchar(5)	YES		NULL	

-- 2) (iii)

Modify the column width of the job field of EMP table.

```
ALTER TABLE EMP MODIFY EXPERIENCE VARCHAR (10);
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	YES		NULL	
ENAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPTNO	int	YES		NULL	
SAL	decimal(7,2)	YES		NULL	
EXPERIENCE	varchar(10)	YES		NULL	

Rithvik Ravilla
106121104

+-----+-----+-----+-----+-----+

-- 2) (iV)

Create dept table with the following structure.

```
CREATE TABLE DEPT (  
    DEPTNO INT (2),  
    LOC VARCHAR (10) ,  
    DNAME VARCHAR (10)  
);
```

Field	Type	Null	Key	Default	Extra
DEPTNO	int	YES		NULL	
LOC	varchar(10)	YES		NULL	
DNAME	varchar(10)	YES		NULL	

-- 2) (V)

drop a column experience from the EMP table.

```
ALTER TABLE EMP DROP EXPERIENCE;
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	YES		NULL	
ENAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPTNO	int	YES		NULL	
SAL	decimal(7,2)	YES		NULL	

-- 3) (I)

Insert a single record into dept table.

```
INSERT INTO DEPT(DEPTNO,LOC,DNAME) VALUES ( 1,"RITHVIK","RAVILLA");
```

-- 3) (II)

Insert more than a record into EMP table using a single insert command.

```
INSERT INTO EMP (EMPNO,ENAME,JOB,DEPTNO,SAL) VALUES  
( 1,"RITHVIK","TEST",1,1000),(2, "VIJAYA","TEST2",2,10000);
```

-- 3) (III)

Select employee name, job from the emp table

```
SELECT EMPNO,JOB FROM EMP;
```

EMPNO	JOB
1	TEST
2	TEST2

Rithvik Ravilla
106121104

-- 4) (I)

Truncate the EMP table and drop the dept table.

TRUNCATE TABLE EMP;

DROP TABLE DEPT;

-- 5) (I)

Use the DCL commands to perform the following operation

i. Create a new user 'dbuser' on the localhost

ii. Create a new database mysampldb and use that database for the following exercises.

iii. Grant all privileges for the dbuser on the mysampldb

CREATE USER 'rithvik'@'localhost';

CREATE DATABASE MYSAMPLEDB;

GRANT ALL PRIVILEGES ON MYSAMPLEDB TO 'rithvik'@'localhost';

-- 6) (I)

Use the DCL command to revoke privilege to the user.

i) Create a new user 'dbuser1' on the localhost

ii) Grant only select privileges for the dbuser1 on the EMP table

iii) Revoke the select privileges for the dbuser1 on the EMP table.

REVOKE ALL ON MYSAMPLEDB FROM 'rithvik'@'localhost';

CREATE USER 'dbuser1'@'localhost';

GRANT SELECT ON EMP TO 'dbuser1'@'localhost';

REVOKE SELECT ON EMP FROM 'dbuser1'@'localhost';

Lab 2 Queries:

1) (i)

Create the above tables by properly specifying the primary keys.

create table AUTHOR(AUTHOR_ID INT, NAME VARCHAR(30), CITY
VARCHAR(40),COUNTRY VARCHAR(50), PRIMARY KEY(AUTHOR_ID));

```
+-----+-----+-----+-----+-----+-----+
| Field  | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| AUTHOR_ID | int      | NO   | PRI | NULL    |      |
| NAME      | varchar(30) | YES  |     | NULL    |      |
| CITY      | varchar(40) | YES  |     | NULL    |      |
| COUNTRY   | varchar(50) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
```

create table PUBLISHER(PUBLISHER_ID INT, NAME VARCHAR(30), CITY
VARCHAR(40),COUNTRY VARCHAR(50), PRIMARY KEY(PUBLISHER_ID));

```
+-----+-----+-----+-----+-----+-----+
| Field  | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| PUBLISHER_ID | int      | NO   | PRI | NULL    |      |
| NAME      | varchar(30) | YES  |     | NULL    |      |
| CITY      | varchar(40) | YES  |     | NULL    |      |
| COUNTRY   | varchar(50) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
```

Rithvik Ravilla
106121104

```
create table CATALOG(BOOK_ID INT, TITLE VARCHAR(30), AUTHOR_ID INT,
PUBLISHER_ID INT,CATEGORY_ID INT, YEAR INT, PRICE INT, PRIMARY
KEY(BOOK_ID) );
```

Field	Type	Null	Key	Default	Extra
BOOK_ID	int	NO	PRI	NULL	
TITLE	varchar(30)	YES		NULL	
AUTHOR_ID	int	YES		NULL	
PUBLISHER_ID	int	YES		NULL	
CATEGORY_ID	int	YES		NULL	
YEAR	int	YES		NULL	
PRICE	int	YES		NULL	

```
create table CATEGORY(CATEGORY_ID INT, DESCRIPTION VARCHAR(30),PRIMARY
KEY(CATEGORY_ID) );
```

Field	Type	Null	Key	Default	Extra
CATEGORY_ID	int	NO	PRI	NULL	
DESCRIPTION	varchar(30)	YES		NULL	

```
create table ORDER_DETAILS(ORDER_NO INT, BOOK_ID INT, QUANTITY INT, PRIMARY
KEY(ORDER_NO) );
```

Field	Type	Null	Key	Default	Extra
ORDER_NO	int	NO	PRI	NULL	
BOOK_ID	int	YES		NULL	
QUANTITY	int	YES		NULL	

1) (ii)

Enter at least five tuples for each relation.

AUTHOR_ID	NAME	CITY	COUNTRY
1	RITHVIK	ABC	abc
2	RAVILLA	ABC	abc
3	RAMA	BENAGLORE	INDIA
4	VIJAYA	TRICHY	INDIA
5	ALLU	HYD	INDIA

PUBLISHER_ID	NAME	CITY	COUNTRY
--------------	------	------	---------

Rithvik Ravilla
106121104

1	RAM	BANG	INDIA
2	SITA	BANGAL	INDIA
3	LAX	BANGALA	INDIA
4	SAPNA	BANGALAR	INDIA
5	KRISHNA	BANGALORE	INDIA

CATEGORY_ID	DESCRIPTION
1	A
2	AA
3	AAA
4	AAAA
5	AAAAA

BOOK_ID	TITLE	AUTHOR_ID	PUBLISHER_ID	CATEGORY_ID	YEAR	PRICE
1	BOOK1	1	1	1	2023	1000
2	BOOK2	2	2	2	2023	2000
3	BOOK3	3	3	3	2023	3000
4	BOOK4	4	4	4	2023	4000
5	BOOK5	5	5	5	2023	5000

ORDER_NO	BOOK_ID	QUANTITY
1	1	150
2	2	250
3	3	350
4	4	450
5	5	550

1) (iii)
Find the total number of authors present in author relation.
SELECT COUNT(AUTHOR_ID) FROM AUTHOR;

COUNT(AUTHOR_ID)
5

1) (iV)
Find the book which has maximum sale.

Rithvik Ravilla
106121104

SELECT BOOK_ID, QUANTITY FROM ORDER_DETAILS WHERE QUANTITY IN (SELECT
MAX(QUANTITY) FROM ORDER_DETAILS);

```
+-----+-----+
| BOOK_ID | QUANTITY |
+-----+-----+
|    5    |    550   |
+-----+-----+
```

2)

```
+-----+-----+-----+
| ACC_NO | YEAR_PUB | TITLE      |
+-----+-----+-----+
| 237235 |    1995  | DBMS       |
| 376711 |    1992  | MACHINE_DESIGN |
| 543211 |    1991  | PROGRAMMING |
| 631523 |    1992  | COMPILER DESIGN |
| 734216 |    1982  | ALGO DESIGN |
+-----+-----+-----+
```

2)(i)

Select from the relation “Book” all the books whose year of publication is 1992.

SELECT * FROM BOOK WHERE YEAR_PUB = 1992;

```
+-----+-----+-----+
| ACC_NO | YEAR_PUB | TITLE      |
+-----+-----+-----+
| 376711 |    1992  | MACHINE_DESIGN |
| 631523 |    1992  | COMPILER DESIGN |
+-----+-----+-----+
```

2) (ii)

Select from the relation “Book” all the books whose Acc-no is greater than equal to 56782.

SELECT * FROM BOOK WHERE ACC_NO > 56782;

```
+-----+-----+-----+
| ACC_NO | YEAR_PUB | TITLE      |
+-----+-----+-----+
| 237235 |    1995  | DBMS       |
| 376711 |    1992  | MACHINE_DESIGN |
| 543211 |    1991  | PROGRAMMING |
| 631523 |    1992  | COMPILER DESIGN |
| 734216 |    1982  | ALGO DESIGN |
+-----+-----+-----+
```

2) (iii)

List all the Title and Acc-no of the “Book” relation.

SELECT ACC_NO, TITLE FROM BOOK;

```
+-----+-----+
| ACC_NO | TITLE      |
+-----+-----+
```

Rithvik Ravilla
106121104

237235 DBMS	
376711 MACHINE_DESIGN	
543211 PROGRAMMING	
631523 COMPILER DESIGN	
734216 ALGO DESIGN	
+-----+	+-----+

2) (IV)

Using 'Rename operator' to rename the 'Acc-no' and 'Yr_pub' into a 'SERIAL NO' and 'YEAR' in the 'Book' relation.

ALTER TABLE BOOK RENAME COLUMN ACC_NO TO SERIAL_NO;
ALTER TABLE BOOK RENAME COLUMN YEAR_PUB TO YEAR;

+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
Field	Type	Null	Key	Default	Extra
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
SERIAL_NO	int	NO	PRI	NULL	
YEAR	int	YES		NULL	
TITLE	varchar(40)	YES		NULL	
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

3) (i)

Create the above tables by properly specifying the primary keys.

create table BRANCH(BRANCH_NAME VARCHAR(40) NOT NULL, BRANCH_CITY VARCHAR(40), ASSETS INT, PRIMARY KEY(BRANCH_NAME));

+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
Field	Type	Null	Key	Default	Extra
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
BRANCH_NAME	varchar(40)	NO	PRI	NULL	
BRANCH_CITY	varchar(40)	YES		NULL	
ASSETS	int	YES		NULL	
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

create table ACCOUNT(ACCOUNT_NUMBER INT NOT NULL, BRANCH_NAME VARCHAR(40), BALANCE INT, PRIMARY KEY(ACCOUNT_NUMBER));

+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
Field	Type	Null	Key	Default	Extra
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
ACCOUNT_NUMBER	int	NO	PRI	NULL	
BRANCH_NAME	varchar(40)	YES		NULL	
BALANCE	int	YES		NULL	
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
Field	Type	Null	Key	Default	Extra
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
LOAN_NUMBER	int	NO	PRI	NULL	
BRANCH_NAME	varchar(40)	YES		NULL	
BALANCE	int	YES		NULL	
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

Rithvik Ravilla
106121104

create table DEPOSITOR(CUSTOMER_NAME VARCHAR(30) NOT NULL,
ACCOUNT_NUMBER INT, PRIMARY KEY(CUSTOMER_NAME));

Field	Type	Null	Key	Default	Extra
CUSTOMER_NAME	varchar(30)	NO	PRI	NULL	
ACCOUNT_NUMBER	int	YES		NULL	

create table BORROWER(CUSTOMER_NAME VARCHAR(30) NOT NULL, LOAN_NUMBER
INT, PRIMARY KEY(CUSTOMER_NAME));

Field	Type	Null	Key	Default	Extra
CUSTOMER_NAME	varchar(30)	NO	PRI	NULL	
LOAN_NUMBER	int	YES		NULL	

3) (ii)

Enter at least five tuples for each relation.

BRANCH_NAME	BRANCH_CITY	ASSETS
RAMA	BANG	3000
RAVI	BANG	2000
RAVILLA	BANG	4000
RITHVIK	BANG	1000
VIJAYA	BANG	4000

ACCOUNT_NUMBER	BRANCH_NAME	BALANCE
1	TEST1	1000
2	TEST2	2000
3	TEST3	3000
4	TEST4	4000
5	TEST5	5000

LOAN_NUMBER	BRANCH_NAME	BALANCE
1	ALPHA1	1000
2	ALPHA2	2000
3	ALPHA3	3000
4	ALPHA4	4000

Rithvik Ravilla
106121104

5	ALPHA5	5000
---	--------	------

CUSTOMER_NAME	ACCOUNT_NUMBER
RITHVIK	1
USER2	2
USER3	3
USER4	4
USER5	5

CUSTOMER_NAME	LOAN_NUMBER
USER1	1
USER2	2
USER3	3
USER4	4
USER5	5

3) (iii)

Find all loans of over 12000rs.

SELECT * FROM LOAN WHERE BALANCE > 1200;

LOAN_NUMBER	BRANCH_NAME	BALANCE
2	ALPHA2	2000
3	ALPHA3	3000
4	ALPHA4	4000
5	ALPHA5	5000

3) (iv)

display the branch names for a given city.

SELECT BRANCH_NAME FROM BRANCH WHERE BRANCH_CITY = "BANG";

BRANCH_NAME
RAMA
RAVI
RAVILLA
RITHVIK
VIJAYA

3) (V)

Rithvik Ravilla
106121104

display depositor name for a specific account number.

```
SELECT CUSTOMER_NAME FROM DEPOSITOR WHERE ACCOUNT_NUMBER = 1;
```

```
+-----+
| CUSTOMER_NAME |
+-----+
| RITHVIK      |
+-----+
```

3) (VI)

display customer names whose names starts with specified character.

```
SELECT CUSTOMER_NAME FROM DEPOSITOR WHERE CUSTOMER_NAME LIKE
"USER%";
```

```
+-----+
| CUSTOMER_NAME |
+-----+
| USER2         |
| USER3         |
| USER4         |
| USER5         |
|               |
|               |
+-----+
```

Lab 3 Queries:

1) (i)

Implement the above schema enforcing primary key and foreign key constraints and insert 5 records into the table.

```
create table DEPT ( DEPTNO INT, DNAME VARCHAR(10), LOC VARCHAR(10), LOCID INT,
PRIMARY KEY(DEPTNO) );
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| DEPTNO | int       | NO   | PRI | NULL    |      |
| DNAME  | varchar(10) | YES  |     | NULL    |      |
| LOC    | varchar(10) | YES  |     | NULL    |      |
| LOCID  | int       | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+
| DEPTNO | DNAME  | LOC   | LOCID |
+-----+-----+-----+-----+
| 1 | JOB   | BANG  | 1 |
| 2 | CLERK | HYD   | 2 |
| 3 | SEC   | CHE   | 3 |
| 4 | DRIVER | DELHI | 4 |
| 5 | TEACH | US    | 5 |
+-----+-----+-----+-----+
```

```
CREATE TABLE EMP (
    EMPNO INT,
    EFNAME VARCHAR(20),
```

Rithvik Ravilla
106121104

```
ELNAME VARCHAR(20),
JOB VARCHAR (10),
DEPTNAME VARCHAR(10),
DEPTNO INT REFERENCES DEPT(DEPTNO),
ECITY VARCHAR (10),
SAL DECIMAL (7,2),
WORKEXPERIENCE INT,
MANAGERNAME VARCHAR(10),
MANAGERNO INT,
PRIMARY KEY (EMPNO) );
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
EFNAME	varchar(20)	YES		NULL	
ELNAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPTNAME	varchar(10)	YES		NULL	
DEPTNO	int	YES	MUL	NULL	
ECITY	varchar(10)	YES		NULL	
SAL	decimal(7,2)	YES		NULL	
WORKEXPERIENCE	int	YES		NULL	
MANAGERNAME	varchar(10)	YES		NULL	
MANAGERNO	int	YES		NULL	

EMPNO	EFNAME	ELNAME	JOB	DEPTNAME	DEPTNO	ECITY	SAL	WORKEXPERIENCE	MANAGERNAME	MANAGERNO
1	RITHVIK	RAVILLA	TESTER	JOB	1	AUS	1.25	10	BLANK	
2	RAMA	RAVILLA	TESTER	CLERK	2	TORONTO	1234.50	20	NAME1	2
3	vijaya	allu	TESTER	SEC	3	HYD	123.50	15	NAME2	3
4	kumari	allu	doc	DRIVER	4	NYC	121.50	17	NAME3	4
5	KRISHNA	RAVI	GEN	TEACH	5	NJ	1121.45	13	NAME4	5

(ii)

Write a query to display the last name, department number, and department name for all employees.

Rithvik Ravilla
106121104

```
+-----+-----+-----+
| ELNAME | DEPTNAME | DEPTNO |
+-----+-----+-----+
| RAVILLA | JOB      | 1 |
| RAVILLA | CLERK    | 2 |
| allu    | SEC      | 3 |
| allu    | DRIVER   | 4 |
| RAVI    | TEACH    | 5 |
+-----+-----+-----+
```

(iii)

Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

```
SELECT DISTINCT JOB,LOC FROM EMP,DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO
AND DEPT.DEPTNO=5;
```

```
+-----+-----+
| JOB | LOC |
+-----+-----+
| GEN | US  |
+-----+-----+
```

(iv)

Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission.

```
SELECT ELNAME,DEPTNAME,ECITY,LOCID FROM EMP,DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND EMP.SAL > 1000;
```

```
+-----+-----+-----+-----+
| ELNAME | DEPTNAME | ECITY | LOCID |
+-----+-----+-----+-----+
| RAVILLA | CLERK    | TORONTO | 2 |
| RAVI    | TEACH    | NJ      | 5 |
+-----+-----+-----+-----+
```

(v):

Display the employee last name and department name for all employees who have an "a" (lowercase) in their last names.

```
SELECT ELNAME,DEPTNAME FROM EMP WHERE ELNAME LIKE "%a%";
```

```
+-----+-----+
| ELNAME | DEPTNAME |
+-----+-----+
| allu    | SEC      |
| allu    | DRIVER   |
+-----+-----+
```

(vi)

Display the employee last name and employee number along with their manager's name and manager number.

```
SELECT ELNAME,EMPNO,MANAGERNAME,MANAGERNO FROM EMP;
```

Rithvik Ravilla
106121104

```
+-----+-----+-----+-----+
| ELNAME | EMPNO | MANAGERNAME | MANAGERNO |
+-----+-----+-----+-----+
| RAVILLA | 1 | BLANK | 1 |
| RAVILLA | 2 | NAME1 | 2 |
| allu | 3 | NAME2 | 3 |
| allu | 4 | NAME3 | 4 |
| RAVI | 5 | NAME4 | 5 |
+-----+-----+-----+-----+
```

(vii)

Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

```
SELECT ELNAME,JOB,DEPT.DEPTNO,DEPTNAME FROM EMP,DEPT WHERE
EMP.DEPTNO = DEPT.DEPTNO AND DEPT.LOC = 'DELHI';
```

```
+-----+-----+-----+-----+
| ELNAME | JOB | DEPTNO | DEPTNAME |
+-----+-----+-----+-----+
| allu | doc | 4 | DRIVER |
+-----+-----+-----+-----+
```

(viii)

Modify the query 6 and display all employees including king, who has no manager and order the result by employee number.

```
SELECT EMPNO,ELNAME,MANAGERNAME,MANAGERNO FROM EMP ORDER BY
MANAGERNO;
```

```
+-----+-----+-----+-----+
| EMPNO | ELNAME | MANAGERNAME | MANAGERNO |
+-----+-----+-----+-----+
| 6 | RAV | NULL | NULL |
| 1 | RAVILLA | BLANK | 1 |
| 2 | RAVILLA | NAME1 | 2 |
| 3 | allu | NAME2 | 3 |
| 4 | allu | NAME3 | 4 |
| 5 | RAVI | NAME4 | 5 |
+-----+-----+-----+-----+
```

(ix)

Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label.

```
SELECT ELNAME,DEPTNO FROM EMP WHERE DEPTNAME = 'JOB';
```

```
+-----+-----+
| ELNAME | DEPTNO |
+-----+-----+
| RAVILLA | 1 |
| RAV | 1 |
+-----+-----+
```

Rithvik Ravilla
106121104

(x)

Find the sum and average of salary from the EMP table

```
SELECT AVG(SAL),SUM(SAL) FROM EMP;
```

```
+-----+-----+
| AVG(SAL) | SUM(SAL) |
+-----+-----+
| 434.575000 | 2607.45 |
+-----+-----+
```

(xi)

Find the employee who is having maximum year of experience.

```
SELECT * FROM EMP WHERE WORKEXPERIENCE=(SELECT MAX(WORKEXPERIENCE)
FROM EMP);
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| EMPNO | EFNAME | ELNAME | JOB   | DEPTNAME | DEPTNO | ECITY  | SAL   |
WORKEXPERIENCE | MANAGERNAME | MANAGERNO |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 2 | RAMA | RAVILLA | TESTER | CLERK   | 2 | TORONTO | 1234.50 | 20 |
NAME1 | 2 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
```

(xii)

Find the number of employees working.

```
SELECT COUNT(EMPNO) FROM EMP;
```

```
+-----+
| COUNT(EMPNO) |
+-----+
| 6 |
+-----+
```

(xiii)

Find the employee who is having very less work experience.

```
SELECT * FROM EMP WHERE WORKEXPERIENCE=(SELECT MIN(WORKEXPERIENCE)
FROM EMP);
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| EMPNO | EFNAME | ELNAME | JOB   | DEPTNAME | DEPTNO | ECITY | SAL   |
WORKEXPERIENCE | MANAGERNAME | MANAGERNO |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 1 | RITHVIK | RAVILLA | TESTER | JOB     | 1 | AUS   | 1.25 | 10 | BLANK |
1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
```

(xiv)

Rithvik Ravilla
106121104

Find the employee who is getting very high salary.

```
SELECT * FROM EMP WHERE SAL=(SELECT MAX(SAL) FROM EMP);
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| EMPNO | EFNAME | ELNAME | JOB   | DEPTNAME | DEPTNO | ECITY  | SAL   |
WORKEXPERIENCE | MANAGERNAME | MANAGERNO |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 2 | RAMA | RAVILLA | TESTER | CLERK   | 2 | TORONTO | 1234.50 | 20 |
NAME1 | 2 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
```

2) (i)

Implement the above schema enforcing primary key constraints and insert 5 records into the table.

```
CREATE TABLE DEPOSITOR ( CUSNAME VARCHAR(20), ACCNO VARCHAR (20) );
```

```
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| CUSNAME | varchar(20) | YES | | NULL | |
| ACCNO | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+
+-----+
| CUSNAME | ACCNO |
+-----+-----+
| RITHVIK | 123QWE |
| RAMA | 321SDC |
| VIJAYA | 456CSD |
| KRISHNA | 654FHN |
| KUMARI | 789ABC |
+-----+-----+
```

```
CREATE TABLE BORROWER ( CUSNAME VARCHAR(20), LOANNO VARCHAR (20) );
```

```
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| CUSNAME | varchar(20) | YES | | NULL | |
| LOANNO | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+
+-----+
| CUSNAME | LOANNO |
+-----+-----+
| RITHVIK | FHG464 |
| RAMA | DHS235 |
| TED | FDN452 |
| BOB | EKW745 |
```

Rithvik Ravilla
106121104

JEN	IKE957
+	+

(ii)

Find the names of all customers who have both loan and account in the bank

SELECT * FROM DEPOSITOR WHERE CUSNAME IN (SELECT CUSNAME FROM BORROWER);

CUSNAME	ACCNO
+	+
RITHVIK	123QWE
RAMA	321SDC
+	+

(iii)

Find the names of all customers who have only loan in the bank

SELECT * FROM BORROWER WHERE CUSNAME NOT IN (SELECT CUSNAME FROM DEPOSITOR);

SELECT CUSNAME FROM BORROWER EXCEPT SELECT CUSNAME FROM DEPOSITOR;

CUSNAME	LOANNO
+	+
TED	FDN452
BOB	EKW745
JEN	IKE957
+	+

(iv)

Find the names of all customers who have either loan or account in the bank

SELECT CUSNAME FROM BORROWER UNION SELECT CUSNAME FROM DEPOSITOR;

CUSNAME
+
RITHVIK
RAMA
TED
BOB
JEN
VIJAYA
KRISHNA
KUMARI

Lab 4 Queries:

```
CREATE TABLE LOCA(  
  -> LOCATION_ID INT,  
  -> STREET_ADDRESS VARCHAR(20),  
  -> POSTAL_CODE INT,  
  -> CITY VARCHAR(20),
```


Rithvik Ravilla
106121104

```
-> STATE_PROVINCE VARCHAR(20),
-> COUNTRY_ID INT,
-> PRIMARY KEY (LOCATION_ID) )
-> ;
```

Field	Type	Null	Key	Default	Extra
LOCATION_ID	int	NO	PRI	NULL	
STREET_ADDRESS	varchar(20)	YES		NULL	
POSTAL_CODE	int	YES		NULL	
CITY	varchar(20)	YES		NULL	
STATE_PROVINCE	varchar(20)	YES		NULL	
COUNTRY_ID	int	YES		NULL	
1	SARJAPURA	560035	BENGALURU	KARNATAKA	1
2	BELLANDUR	560036	BENGALURU	KARNATAKA	1
3	HSR	560037	BENGALURU	KARNATAKA	1
4	BTM	560038	BENGALURU	KARNATAKA	1
5	KORMANGLA	560039	BENGALURU	KARNATAKA	1

```
CREATE TABLE DEPT(
-> DEPARTMENT_ID INT,
-> DEPARTMENT_NAME VARCHAR(20),
-> MANAGER_ID INT,
-> LOCATION_ID INT,
-> PRIMARY KEY(DEPARTMENT_ID) );
```

```
ALTER TABLE DEPT ADD FOREIGN KEY(LOCATION_ID) REFERENCES
LOCA(LOCATION_ID);
```

Field	Type	Null	Key	Default	Extra
DEPARTMENT_ID	int	NO	PRI	NULL	
DEPARTMENT_NAME	varchar(20)	YES		NULL	
MANAGER_ID	int	YES		NULL	
LOCATION_ID	int	YES	MUL	NULL	
1	ADMINISTRATION	120		1	
2	MARKETING	136		2	
3	PURCHASING	342		3	
4	HR	45		4	
5	IT_SUPPORT	234		5	

Rithvik Ravilla
106121104

```
CREATE TABLE EMP(EMPLOYEE_ID INT, FIRST_NAME VARCHAR(20),LAST_NAME
VARCHAR(20),EMAIL VARCHAR(20), PHONE_NUMBER VARCHAR(20),HIRE_DATE
DATE,JOB_ID VARCHAR(20),COMMISSION_PCT DECIMAL(7,2),MANAGER_ID INT,
DEPARTMENT_ID INT, PRIMARY KEY(EMPLOYEE_ID));
ALTER TABLE EMP ADD FOREIGN KEY(DEPARTMENT_ID) REFERENCES
DEPT(DEPARTMENT_ID);
```

Field	Type	Null	Key	Default	Extra
EMPLOYEE_ID	int	NO	PRI	NULL	
FIRST_NAME	varchar(20)	YES		NULL	
LAST_NAME	varchar(20)	YES		NULL	
EMAIL	varchar(20)	YES		NULL	
PHONE_NUMBER	varchar(20)	YES		NULL	
HIRE_DATE	date	YES		NULL	
JOB_ID	varchar(20)	YES		NULL	
COMMISSION_PCT	decimal(7,2)	YES		NULL	
MANAGER_ID	int	YES		NULL	
DEPARTMENT_ID	int	YES	MUL	NULL	
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	
HIRE_DATE	JOB_ID	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	
1	RITHVIK	RAVILLA	RITHVIK@GMAIL.COM	43524354	2004-12-10
IT_PROG	1234.34	1	1		
2	RAMA	RAVILLA	RAMA@GMAIL.COM	6456223	1999-07-08
IT_PROG	5000.00	2	2		
3	VIJAYA	ALLU	VIJAYA@GMAIL.COM	235424	2002-11-05
HR	4003.00	234	2		
4	TRISHA	PATEL	TRISHA@GMAIL.COM	56346	2012-04-11
AD_PRES	2352.00	123	3		
5	JAYA	ALLU	JAYA@GMAIL.COM	84353	2022-09-23
VP_SALES	45622.00	12	4		

1)
Display all the information of an employee whose id is any of the number 134, 159 and 183. (use In)

```
SELECT * FROM EMP WHERE EMPLOYEE_ID IN (SELECT EMPLOYEE_ID FROM EMP
WHERE EMPLOYEE_ID = 1 OR EMPLOYEE_ID = 2 OR EMPLOYEE_ID = 5);
```

Rithvik Ravilla
106121104

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	RITHVIK	RAVILLA	RITHVIK@GMAIL.COM	43524354	2004-12-10	IT_PROG	1234.34	1	1
2	RAMA	RAVILLA	RAMA@GMAIL.COM	6456223	1999-07-08	IT_PROG	5000.00	2	2
5	JAYA	ALLU	JAYA@GMAIL.COM	84353	2022-09-23	VP_SALES	45622.00	12	4

2)

Write a query to display all the information of the employees who does not work in those departments where some employees work whose manager id within the range 100 and 200. (use Not in & Between)

```
SELECT * FROM EMP WHERE DEPARTMENT_ID NOT IN (SELECT DEPARTMENT_ID
FROM DEPT WHERE MANAGER_ID BETWEEN 100 AND 200);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
4	TRISHA	PATEL	TRISHA@GMAIL.COM	56346	2012-04-11	AD_PRES	2352.00	123	3
5	JAYA	ALLU	JAYA@GMAIL.COM	84353	2022-09-23	VP_SALES	45622.00	12	4

3)

Write a query to display all the information for those employees whose id is any id who earn the second highest salary. (use In & Max)

```
SELECT * FROM EMP WHERE COMMISSION_PCT IN (SELECT MAX(COMMISSION_PCT)
FROM EMP WHERE COMMISSION_PCT NOT IN (SELECT MAX(COMMISSION_PCT)
FROM EMP));
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
2	RAMA	RAVILLA	RAMA@GMAIL.COM	6456223	1999-07-08	IT_PROG	5000.00	2	2

Rithvik Ravilla
106121104

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
```

4)

Write a query in SQL to display all the information about those employees who earn second lowest salary of all the employees. (use distinct) Write a query to get the details of employees who are managers. (use exists)

```
SELECT * FROM EMP WHERE COMMISSION_PCT IN (SELECT MIN(COMMISSION_PCT)
FROM EMP WHERE COMMISSION_PCT NOT IN (SELECT MIN(COMMISSION_PCT)
FROM EMP));
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL          | PHONE_NUMBER |
| HIRE_DATE   | JOB_ID     | COMMISSION_PCT | MANAGER_ID | DEPARTMENT_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|          4 | TRISHA    | PATEL     | TRISHA@GMAIL.COM | 56346        | 2012-04-11 | AD_PRES |
| 2352.00    | 123       | 3         |              |              |              |              |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
```

```
SELECT * FROM EMP WHERE EXISTS (SELECT JOB_ID FROM EMP WHERE JOB_ID =
'VP_SALES');
```

5)

Write a subquery that returns a set of rows to find all departments that do actually have one or more employees assigned to them. (use distinct)

```
SELECT DISTINCT
DEPT.DEPARTMENT_ID,DEPARTMENT_NAME,DEPT.MANAGER_ID,LOCATION_ID
FROM EMP,DEPT WHERE DEPT.DEPARTMENT_ID=EMP.DEPARTMENT_ID;
```

```
+-----+-----+-----+-----+
| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
+-----+-----+-----+-----+
|          1 | ADMINISTRATION | 120        | 1           |
|          2 | MARKETING      | 136        | 2           |
|          3 | PURCHASING     | 342        | 3           |
|          4 | HR             | 45         | 4           |
+-----+-----+-----+-----+
```

6)

Write a query to display the employee name (first name and last name) and department for all employees for any existence of those employees whose salary is more than 3700. (use exists)

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_ID FROM EMP WHERE EXISTS
(SELECT COMMISSION_PCT FROM EMP WHERE COMMISSION_PCT > 3700);
```

```
+-----+-----+-----+
```

Rithvik Ravilla
106121104

FIRST_NAME	LAST_NAME	DEPARTMENT_ID
RITHVIK	RAVILLA	1
RAMA	RAVILLA	2
VIJAYA	ALLU	2
TRISHA	PATEL	3
JAYA	ALLU	4

8)

Write a query to display the employee number and name (first name and last name) for all employees who work in a department with any employee whose name contains a T. (use In)

```
SELECT FIRST_NAME, LAST_NAME FROM EMP WHERE DEPARTMENT_ID IN (SELECT DEPT.DEPARTMENT_ID FROM EMP, DEPT WHERE DEPT.DEPARTMENT_ID=EMP.DEPARTMENT_ID AND FIRST_NAME LIKE '%T%');
```

FIRST_NAME	LAST_NAME
RITHVIK	RAVILLA
TRISHA	PATEL

9)

Write a query to display the employee number, name (first name and last name), and salary for all employees who earn more than the average salary and who work in a department with any employee with a J in their name. (use avg & In)

```
SELECT FIRST_NAME, LAST_NAME FROM EMP WHERE DEPARTMENT_ID IN (SELECT DEPT.DEPARTMENT_ID FROM EMP, DEPT WHERE DEPT.DEPARTMENT_ID=EMP.DEPARTMENT_ID AND FIRST_NAME LIKE '%T%' AND EMP.COMMISSION_PCT > (SELECT AVG(COMMISSION_PCT) FROM EMP));
```

Empty set (0.00 sec)

10)

Write a query to display the employee number, name (first name and last name) and job title for all employees whose salary is smaller than any salary of those employees whose job title is IT_PROG . (use any)

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID FROM EMP WHERE COMMISSION_PCT < ANY(SELECT COMMISSION_PCT FROM EMP WHERE JOB_ID = 'IT_PROG');
```

FIRST_NAME	LAST_NAME	JOB_ID
RITHVIK	RAVILLA	IT_PROG
VIJAYA	ALLU	HR
TRISHA	PATEL	AD_PRES

Rithvik Ravilla
106121104

11)

Write a query to display the employee number, name (first name and last name) and job title for all employees whose salary is smaller than any salary of those employees whose job title is IT_PROG .Exclude Job title IT_PROG . (use any)

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID FROM EMP WHERE COMMISSION_PCT >
ANY(SELECT COMMISSION_PCT FROM EMP WHERE JOB_ID = 'IT_PROG') AND
JOB_ID != 'IT_PROG';
```

```
+-----+-----+-----+
| FIRST_NAME | LAST_NAME | JOB_ID |
+-----+-----+-----+
| VIJAYA    | ALLU     | HR     |
| TRISHA    | PATEL    | AD_PRES |
| JAYA      | ALLU     | VP_SALES |
+-----+-----+-----+
```

12)

Write a query to display the employee number, name (first name and last name) and job title for all employees whose salary is more than any salary of those employees whose job title is IT_PROG. Exclude job title IT_PROG. (use all)

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID FROM EMP WHERE COMMISSION_PCT >
ALL(SELECT COMMISSION_PCT FROM EMP WHERE JOB_ID = 'IT_PROG') AND JOB_ID !=
'IT_PROG';
```

```
+-----+-----+-----+
| FIRST_NAME | LAST_NAME | JOB_ID |
+-----+-----+-----+
| JAYA      | ALLU     | VP_SALES |
+-----+-----+-----+
```

14)

Write a query in SQL to display the first and last name, salary, and department ID for all those employees who earn more than the average salary and arrange the list in descending order on salary. (use order by)

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID, COMMISSION_PCT FROM EMP WHERE
COMMISSION_PCT < (SELECT AVG(COMMISSION_PCT) FROM EMP) ORDER BY
COMMISSION_PCT DESC;
```

```
+-----+-----+-----+-----+
| FIRST_NAME | LAST_NAME | JOB_ID | COMMISSION_PCT |
+-----+-----+-----+-----+
| RAMA      | RAVILLA  | IT_PROG | 5000.00 |
| VIJAYA    | ALLU     | HR      | 4003.00 |
| TRISHA    | PATEL    | AD_PRES | 2352.00 |
| RITHVIK   | RAVILLA  | IT_PROG | 1234.34 |
+-----+-----+-----+-----+
```

15)

Write a query to display all the information of the employees whose salary is within

Rithvik Ravilla
106121104

the range of smallest salary and 2500. (use Between & min)

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID FROM EMP WHERE COMMISSION_PCT  
BETWEEN (SELECT MIN(COMMISSION_PCT) FROM EMP) AND 2500;
```

+-----+-----+-----+		
FIRST_NAME	LAST_NAME	JOB_ID
+-----+-----+-----+		
RITHVIK	RAVILLA	IT_PROG
TRISHA	PATEL	AD_PRES

Lab 5 Queries:

1)

Create the Table and Perform Join Operations on them

```
CREATE TABLE STUDENT( ROLL_NO INT, NAME VARCHAR(20), ADDRESS  
VARCHAR(50), PHONE INT, AGE INT, PRIMARY KEY (ROLL_NO));
```

+-----+-----+-----+-----+-----+					
Field	Type	Null	Key	Default	Extra
+-----+-----+-----+-----+-----+					
ROLL_NO	int	NO	PRI	NULL	
NAME	varchar(20)	YES		NULL	
ADDRESS	varchar(50)	YES		NULL	
PHONE	int	YES		NULL	
AGE	int	YES		NULL	
+-----+-----+-----+-----+-----+					
+-----+-----+-----+-----+-----+					
ROLL_NO	NAME	ADDRESS	PHONE	AGE	
+-----+-----+-----+-----+-----+					
1	RITHVIK	SARJAPUR	12345	19	
2	RAMA	KASAVANAHALLI	52345	51	
3	VIJAYA	BELLANDUR	42345	45	
4	KRISHNA	KORMANGLA	42445	25	
5	KUMARI	MG ROAD	24735	38	
6	RAVI	HSR	63354	14	
+-----+-----+-----+-----+-----+					

```
CREATE TABLE STUDENTCOURSE( COURSE_ID INT,ROLL_NO INT);
```

+-----+-----+-----+-----+-----+					
Field	Type	Null	Key	Default	Extra
+-----+-----+-----+-----+-----+					
COURSE_ID	int	YES		NULL	
ROLL_NO	int	YES		NULL	
+-----+-----+-----+-----+-----+					
+-----+-----+					
COURSE_ID	ROLL_NO				

Rithvik Ravilla
106121104

1	1
1	2
1	3
2	1
2	4
3	3
3	4
4	7

I)

```
SELECT * FROM STUDENT INNER JOIN STUDENTCOURSE ON STUDENT.ROLL_NO = STUDENTCOURSE.ROLL_NO;
```

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID	ROLL_NO
1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2
3	VIJAYA	BELLANDUR	42345	45	1	3
1	RITHVIK	SARJAPUR	12345	19	2	1
4	KRISHNA	KORMANGLA	42445	25	2	4
3	VIJAYA	BELLANDUR	42345	45	3	3
4	KRISHNA	KORMANGLA	42445	25	3	4

II)

```
SELECT * FROM STUDENT LEFT JOIN STUDENTCOURSE ON STUDENT.ROLL_NO =
STUDENTCOURSE.ROLL_NO;
```

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID	ROLL_NO
1	RITHVIK	SARJAPUR	12345	19	2	1
1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2
3	VIJAYA	BELLANDUR	42345	45	3	3
3	VIJAYA	BELLANDUR	42345	45	1	3
4	KRISHNA	KORMANGLA	42445	25	3	4
4	KRISHNA	KORMANGLA	42445	25	2	4
5	KUMARI	MG ROAD	24735	38	NULL	NULL
6	RAVI	HSR	63354	14	NULL	NULL

III)

```
SELECT * FROM STUDENT RIGHT JOIN STUDENTCOURSE ON STUDENT.ROLL_NO =
STUDENTCOURSE.ROLL_NO;
```

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID
1	John	123 Main St	555-1234	20	CS101
2	Jane	456 Oak St	555-5678	22	CS101
3	Mike	789 Pine St	555-9012	21	CS101
4	Sarah	101 Elm St	555-3456	19	CS101
5	David	202 Maple St	555-7890	23	CS101
6	Emily	303 Birch St	555-2345	20	CS101
7	Chris	404 Cedar St	555-6789	22	CS101
8	Alex	505 Spruce St	555-0123	21	CS101
9	Mia	606 Willow St	555-4567	19	CS101
10	Noah	707 Ash St	555-8901	23	CS101

Rithvik Ravilla
106121104

1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2
3	VIJAYA	BELLANDUR	42345	45	1	3
1	RITHVIK	SARJAPUR	12345	19	2	1
4	KRISHNA	KORMANGLA	42445	25	2	4
3	VIJAYA	BELLANDUR	42345	45	3	3
4	KRISHNA	KORMANGLA	42445	25	3	4
NULL	NULL	NULL	NULL	NULL	4	7

IV)

(SELECT * FROM STUDENT LEFT JOIN STUDENTCOURSE ON STUDENT.ROLL_NO =
STUDENTCOURSE.ROLL_NO)

UNION ALL

(SELECT * FROM STUDENT RIGHT JOIN STUDENTCOURSE ON STUDENT.ROLL_NO =
STUDENTCOURSE.ROLL_NO);

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID	ROLL_NO
1	RITHVIK	SARJAPUR	12345	19	2	1
1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2
3	VIJAYA	BELLANDUR	42345	45	3	3
3	VIJAYA	BELLANDUR	42345	45	1	3
4	KRISHNA	KORMANGLA	42445	25	3	4
4	KRISHNA	KORMANGLA	42445	25	2	4
5	KUMARI	MG ROAD	24735	38	NULL	NULL
6	RAVI	HSR	63354	14	NULL	NULL
1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2
3	VIJAYA	BELLANDUR	42345	45	1	3
1	RITHVIK	SARJAPUR	12345	19	2	1
4	KRISHNA	KORMANGLA	42445	25	2	4
3	VIJAYA	BELLANDUR	42345	45	3	3
4	KRISHNA	KORMANGLA	42445	25	3	4
NULL	NULL	NULL	NULL	NULL	4	7

V) & VII)

EQUIJOIN == INNER JOIN == NATURAL JOIN

VI)

SELECT * FROM STUDENT INNER JOIN STUDENTCOURSE ON STUDENT.ROLL_NO <=
STUDENTCOURSE.ROLL_NO;

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID	ROLL_NO
1	RITHVIK	SARJAPUR	12345	19	1	1
2	RAMA	KASAVANAHALLI	52345	51	1	2

Rithvik Ravilla
106121104

1	RITHVIK	SARJAPUR	12345	19	1	2
3	VIJAYA	BELLANDUR	42345	45	1	3
2	RAMA	KASAVANAHALLI	52345	51	1	3
1	RITHVIK	SARJAPUR	12345	19	1	3
1	RITHVIK	SARJAPUR	12345	19	2	1
4	KRISHNA	KORMANGLA	42445	25	2	4
3	VIJAYA	BELLANDUR	42345	45	2	4
2	RAMA	KASAVANAHALLI	52345	51	2	4
1	RITHVIK	SARJAPUR	12345	19	2	4
3	VIJAYA	BELLANDUR	42345	45	3	3
2	RAMA	KASAVANAHALLI	52345	51	3	3
1	RITHVIK	SARJAPUR	12345	19	3	3
4	KRISHNA	KORMANGLA	42445	25	3	4
3	VIJAYA	BELLANDUR	42345	45	3	4
2	RAMA	KASAVANAHALLI	52345	51	3	4
1	RITHVIK	SARJAPUR	12345	19	3	4
6	RAVI	HSR	63354	14	4	7
5	KUMARI	MG ROAD	24735	38	4	7
4	KRISHNA	KORMANGLA	42445	25	4	7
3	VIJAYA	BELLANDUR	42345	45	4	7
2	RAMA	KASAVANAHALLI	52345	51	4	7
1	RITHVIK	SARJAPUR	12345	19	4	7

+-----+-----+-----+-----+-----+-----+

2)

I)

Create tables and insert 10 records into them

CREATE TABLE CUSTOMER (CUS_ID INT, CUS_NAME VARCHAR(20), PRIMARY KEY (CUS_ID));

Field	Type	Null	Key	Default	Extra
CUS_ID	int	NO	PRI	NULL	
CUS_NAME	varchar(20)	YES		NULL	

+-----+-----+

CUS_ID	CUS_NAME
1	NAME1
2	NAME2
3	NAME3
4	NAME4
5	NAME5
6	NAME6
7	NAME7
8	NAME8
9	NAME9
10	NAME10

Rithvik Ravilla
106121104

+-----+-----+

CREATE TABLE ITEM (ITEM_ID INT,ITEM_NAME VARCHAR(20),PRICE INT,PRIMARY
KEY (ITEM_ID));

+-----+-----+-----+-----+-----+-----+

Field	Type	Null	Key	Default	Extra
-------	------	------	-----	---------	-------

+-----+-----+-----+-----+-----+-----+

ITEM_ID	int	NO	PRI	NULL	
---------	-----	----	-----	------	--

ITEM_NAME	varchar(20)	YES		NULL	
-----------	-------------	-----	--	------	--

PRICE	int	YES		NULL	
-------	-----	-----	--	------	--

+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+

ITEM_ID	ITEM_NAME	PRICE
---------	-----------	-------

+-----+-----+-----+

1	ITEM1	100
---	-------	-----

2	ITEM2	200
---	-------	-----

3	ITEM3	300
---	-------	-----

4	ITEM4	400
---	-------	-----

5	ITEM5	500
---	-------	-----

6	ITEM6	600
---	-------	-----

7	ITEM7	700
---	-------	-----

8	ITEM8	800
---	-------	-----

9	ITEM9	900
---	-------	-----

10	ITEM10	1000
----	--------	------

+-----+-----+-----+

CREATE TABLE SALE(BILL_NO INT,BILL_DATE DATE,CUSTOMER_ID INT,ITEM_ID
INT,QTY_SOLD INT,AMOUNT INT,PRIMARY KEY (BILL_NO));

+-----+-----+-----+-----+-----+

Field	Type	Null	Key	Default	Extra
-------	------	------	-----	---------	-------

+-----+-----+-----+-----+-----+

BILL_NO	int	NO	PRI	NULL	
---------	-----	----	-----	------	--

BILL_DATE	date	YES		NULL	
-----------	------	-----	--	------	--

CUS_ID	int	YES	MUL	NULL	
--------	-----	-----	-----	------	--

ITEM_ID	int	YES	MUL	NULL	
---------	-----	-----	-----	------	--

QTY_SOLD	int	YES		NULL	
----------	-----	-----	--	------	--

AMOUNT	int	YES		NULL	
--------	-----	-----	--	------	--

+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+

BILL_NO	BILL_DATE	CUS_ID	ITEM_ID	QTY_SOLD
---------	-----------	--------	---------	----------

+-----+-----+-----+-----+-----+

1	2023-12-01	1	1	1
---	------------	---	---	---

2	2023-12-02	2	2	2
---	------------	---	---	---

3	2023-12-03	3	3	3
---	------------	---	---	---

4	2023-12-04	4	4	4
---	------------	---	---	---

5	2023-12-05	5	5	5
---	------------	---	---	---

6	2023-12-06	6	6	6
---	------------	---	---	---

7	2023-12-07	7	7	7
---	------------	---	---	---

8	2023-12-08	8	8	8
---	------------	---	---	---

Rithvik Ravilla
106121104

	9	2023-12-09	9		9		9	
	10	2023-12-10	10		10		10	
+-----+-----+-----+-----+-----+								

II)
Create a VIEW

```
CREATE VIEW SALE_VIEW AS SELECT  
BILL_NO,BILL_DATE,CUS_ID,SALE.ITEM_ID,PRICE,QTY_SOLD,PRICE*QTY_SOLD AS  
AMOUNT FROM SALE,ITEM WHERE SALE.ITEM_ID = ITEM.ITEM_ID;  
SELECT * FROM SALE_VIEW;
```

+-----+-----+-----+-----+-----+-----+-----+								
	BILL_NO		BILL_DATE		CUS_ID		ITEM_ID	
	PRICE		QTY_SOLD		AMOUNT			
+-----+-----+-----+-----+-----+-----+-----+								
	1	2023-12-01	1		1	100	1	100
	2	2023-12-02	2		2	200	2	400
	3	2023-12-03	3		3	300	3	900
	4	2023-12-04	4		4	400	4	1600
	5	2023-12-05	5		5	500	5	2500
	6	2023-12-06	6		6	600	6	3600
	7	2023-12-07	7		7	700	7	4900
	8	2023-12-08	8		8	800	8	6400
	9	2023-12-09	9		9	900	9	8100
	10	2023-12-10	10		10	1000	10	10000
+-----+-----+-----+-----+-----+-----+-----+								

III)
Create a view that lists daily sales

```
CREATE VIEW WEEK_VIEW AS SELECT * FROM SALE WHERE BILL_DATE BETWEEN  
'2023-12-03' AND '2023-12-10' ORDER BY BILL_DATE;
```

+-----+-----+-----+-----+-----+								
	BILL_NO		BILL_DATE		CUS_ID		ITEM_ID	
	QTY_SOLD							
+-----+-----+-----+-----+-----+								
	3	2023-12-03	3		3		3	
	4	2023-12-04	4		4		4	
	5	2023-12-05	5		5		5	
	6	2023-12-06	6		6		6	
	7	2023-12-07	7		7		7	
	8	2023-12-08	8		8		8	
	9	2023-12-09	9		9		9	
	10	2023-12-10	10		10		10	
+-----+-----+-----+-----+-----+								

IV)
Create a derived relation to get top 5 sales

Rithvik Ravilla
106121104

```
SELECT BILL_NO,BILL_DATE,CUS_ID,SALE.ITEM_ID,PRICE*QTY_SOLD AS AMOUNT
FROM SALE,ITEM WHERE SALE.ITEM_ID = ITEM.ITEM_ID ORDER BY AMOUNT DESC
LIMIT 5;
```

```
+-----+-----+-----+-----+-----+
| BILL_NO | BILL_DATE | CUS_ID | ITEM_ID | AMOUNT |
+-----+-----+-----+-----+-----+
| 10 | 2023-12-10 | 10 | 10 | 10000 |
| 9 | 2023-12-09 | 9 | 9 | 8100 |
| 8 | 2023-12-08 | 8 | 8 | 6400 |
| 7 | 2023-12-07 | 7 | 7 | 4900 |
| 6 | 2023-12-06 | 6 | 6 | 3600 |
+-----+-----+-----+-----+-----+
```

V)

Classify customers into 3 groups and count number in desired relation

```
SELECT COUNT(CUS_ID) AS SILVER FROM CUSTOMER WHERE CUS_ID IN (SELECT
CUS_ID FROM SALE,ITEM WHERE SALE.ITEM_ID = ITEM.ITEM_ID AND
PRICE*QTY_SOLD < 5000);
```

```
+-----+
| SILVER |
+-----+
| 7 |
+-----+
```

```
SELECT COUNT(CUS_ID) AS GOLD FROM CUSTOMER WHERE CUS_ID IN (SELECT
CUS_ID FROM SALE,ITEM WHERE SALE.ITEM_ID = ITEM.ITEM_ID
AND PRICE*QTY_SOLD > 5000);
```

```
+-----+
| GOLD |
+-----+
| 3 |
+-----+
```

VI)

Find top 5 customers based on their spending

```
CREATE VIEW TOP_CUSTOMER AS SELECT
CUSTOMER.CUS_ID,CUS_NAME,SALE.ITEM_ID,PRICE*QTY_SOLD AS AMOUNT FROM
SALE,ITEM,CUSTOMER
WHERE SALE.ITEM_ID = ITEM.ITEM_ID AND CUSTOMER.CUS_ID = SALE.CUS_ID
ORDER BY AMOUNT DESC LIMIT 5 WITH CHECK OPTION;
```

```
+-----+-----+-----+-----+
| CUS_ID | CUS_NAME | ITEM_ID | AMOUNT |
+-----+-----+-----+-----+
| 10 | NAME10 | 10 | 10000 |
| 9 | NAME9 | 9 | 8100 |
| 8 | NAME8 | 8 | 6400 |
```

Rithvik Ravilla
106121104

	7	NAME7		7	4900	
	6	NAME6		6	3600	
+-----+-----+-----+-----+						

NOTE: DERIVED RELATION

SELECT PRICE*QTY_SOLD AS AMOUNT FROM SALE,ITEM WHERE SALE.ITEM_ID =
ITEM.ITEM_ID AND AMOUNT > 5000;

Lab 6 Queries:

1)

Convert ER-diagram into relational database and to create the table for the relation by properly specifying the primary keys and foreign keys.

DESCRIBE COURSE;

+-----+-----+-----+-----+						
Field	Type	Null	Key	Default	Extra	
+-----+-----+-----+-----+						
COURSE_NUM	int	NO	PRI	NULL		
COURSE_NAME	varchar(20)	YES		NULL		
+-----+-----+-----+-----+						

CREATE TABLE SECTION(

TERM INT,

SECTION_NUM INT,

COURSE_NUM INT,

PROF_NUM INT,

FOREIGN KEY (COURSE_NUM) REFERENCES COURSE(COURSE_NUM),

FOREIGN KEY (PROF_NUM) REFERENCES PROFFESSOR(PROF_NUM),

PRIMARY KEY (COURSE_NUM,TERM,SECTION_NUM));

+-----+-----+-----+-----+						
Field	Type	Null	Key	Default	Extra	
+-----+-----+-----+-----+						
TERM	int	NO	PRI	NULL		
SECTION_NUM	int	NO	PRI	NULL		
COURSE_NUM	int	NO	PRI	NULL		
PROF_NUM	int	YES	MUL	NULL		
+-----+-----+-----+-----+						

CREATE TABLE OFF_SITE_SECTION(

TERM INT,

SECTION_NUM INT,

LOCATION VARCHAR(30),

COURSE_NUM INT,

PROF_NUM INT,

FOREIGN KEY (COURSE_NUM,TERM,SECTION_NUM) REFERENCES

SECTION(COURSE_NUM,TERM,SECTION_NUM),

FOREIGN KEY (PROF_NUM) REFERENCES SECTION(PROF_NUM),

PRIMARY KEY(COURSE_NUM,TERM,SECTION_NUM));

Rithvik Ravilla
106121104

Field	Type	Null	Key	Default	Extra
TERM	int	NO	PRI	NULL	
SECTION_NUM	int	NO	PRI	NULL	
LOCATION	varchar(30)	YES		NULL	
COURSE_NUM	int	NO	PRI	NULL	
PROF_NUM	int	YES	MUL	NULL	

```
CREATE TABLE PROFFESSOR ( PROF_NUM INT, PROF_NAME VARCHAR (20), PRIMARY KEY (PROF_NUM) );
```

Field	Type	Null	Key	Default	Extra
PROF_NUM	int	NO	PRI	NULL	
PROF_NAME	varchar(20)	YES		NULL	

```
CREATE TABLE STUDENT ( STUDENT_NUM INT, STUDENT_NAME VARCHAR (20), GPA INT, MARK INT, PRIMARY KEY (STUDENT_NUM) );
```

Field	Type	Null	Key	Default	Extra
STUDENT_NUM	int	NO	PRI	NULL	
STUDENT_NAME	varchar(20)	YES		NULL	
GPA	int	YES		NULL	
MARK	int	YES		NULL	

```
CREATE TABLE ENROLLED_IN (
    STUDENT_NUM INT,
    COURSE_NUM INT,
    TERM INT,
    SECTION_NUM INT,
    PRIMARY KEY (STUDENT_NUM,COURSE_NUM,TERM,SECTION_NUM)
);
```

Field	Type	Null	Key	Default	Extra
STUDENT_NUM	int	NO	PRI	NULL	
COURSE_NUM	int	NO	PRI	NULL	
TERM	int	NO	PRI	NULL	
SECTION_NUM	int	NO	PRI	NULL	

Rithvik Ravilla
106121104

2)

Create a trigger called updateAvailableQuantity that updates the quantity in stock in the Product table, for every product sold. The trigger should be executed after each insert operation on the SaleItem table: for the product with the given barcode (the one inserted into SaleItem), update the available quantity in Product table to be the old quantity minus the sold quantity.

```
CREATE TABLE PRODUCT (  
  -> BAR_CODE INT,  
  -> PNAME VARCHAR(20),  
  -> PRICE INT,  
  -> QUANTITY_IN_STOCK INT,  
  -> PRIMARY KEY (BAR_CODE) );
```

BAR_CODE	int	NO	PRI	NULL		
PNAME	varchar(20)	YES		NULL		
PRICE	int	YES		NULL		
QUANTITY_IN_STOCK	int	YES		NULL		

BAR_CODE	PNAME	PRICE	QUANTITY_IN_STOCK
1 A	100	20	
2 B	200	30	
3 C	300	40	

```
CREATE TABLE SALE ( SALE_ID INT, DELIVERY_ADDRESS VARCHAR(20),  
CREDIT_CARD INT, PRIMARY KEY (SALE_ID) );
```

Field	Type	Null	Key	Default	Extra
SALE_ID	int	NO	PRI	NULL	
DELIVERY_ADDRESS	varchar(20)	YES		NULL	
CREDIT_CARD	int	YES		NULL	

SALE_ID	DELIVERY_ADDRESS	CREDIT_CARD
1 ELAN	123	
2 PALM	425	

Rithvik Ravilla
106121104

```
CREATE TABLE SALE_ITEM (  
    SALE_ID INT,  
    BAR_CODE INT,  
    QUANTITY INT,  
    FOREIGN KEY (SALE_ID) REFERENCES SALE(SALE_ID),  
    FOREIGN KEY (BAR_CODE) REFERENCES PRODUCT(BAR_CODE));
```

Field	Type	Null	Key	Default	Extra
SALE_ID	int	YES	MUL	NULL	
BAR_CODE	int	YES	MUL	NULL	
QUANTITY	int	YES		NULL	

```
CREATE TRIGGER UPDATE_QUANTITY  
AFTER INSERT ON SALE_ITEM  
FOR EACH ROW UPDATE PRODUCT  
SET QUANTITY_IN_STOCK = QUANTITY_IN_STOCK - NEW.QUANTITY  
WHERE BAR_CODE = NEW.BAR_CODE;
```

```
INSERT INTO SALE_ITEM (SALE_ID,BAR_CODE,QUANTITY) VALUES (1,1,5);
```

```
SELECT * FROM SALE_ITEM;
```

SALE_ID	BAR_CODE	QUANTITY
1	1	5

```
SELECT * FROM PRODUCT;
```

BAR_CODE	PNAME	PRICE	QUANTITY_IN_STOCK
1	A	100	15
2	B	200	30
3	C	300	40

Employee(empNo, empName, jobPosition, managerId, salary)
Department(department number, department name)
Company(empNo, department number, joining date)

3)
create the following tables with given attributes by specifying appropriate primary key and foreign keys. Tables should be created with necessary constraints which enables to perform on delete, on update cascade functions and self-referential integrity constraints

Rithvik Ravilla
106121104

```
CREATE TABLE EMPLOYEE (  
  EMPNO INT,  
  EMPNAME VARCHAR(20),  
  JOB_POSITION VARCHAR(20),  
  MANAGER_ID INT(10),  
  SALARY NUMERIC(10, 2),  
  PRIMARY KEY (EMPNO),  
  FOREIGN KEY (MANAGER_ID) REFERENCES EMPLOYEE(EMPNO) ON DELETE  
  CASCADE  
);
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
EMPNAME	varchar(20)	YES		NULL	
JOB	varchar(20)	YES		NULL	
MANAGER_ID	int	YES		NULL	
SALARY	int	YES		NULL	

EMPNO	EMPNAME	JOB_POSITION	MANAGER_ID	SALARY
1	RITHVIK	A	1	100.00
2	RAMA	B	1	200.00
3	VIJAYA	C	1	300.00

```
CREATE TABLE DEPARTMENT (  
  DEPARTMENT_NO INT,  
  DEPARTMENT_NAME VARCHAR (20),  
  PRIMARY KEY (DEPARTMENT_NO)  
);
```

Field	Type	Null	Key	Default	Extra
DEPARTMENT_NO	int	NO	PRI	NULL	
DEPARTMENT_NAME	varchar(20)	YES		NULL	

DEPARTMENT_NO	DEPARTMENT_NAME
1	ABC
2	DEF
3	DFGS

Rithvik Ravilla
106121104

```
CREATE TABLE COMPANY(  
    EMPNO INT,  
    DEPARTMENT_NO INT,  
    JOINING_DATE DATE,  
    FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE(EMPNO) ON DELETE CASCADE  
ON UPDATE CASCADE,  
    FOREIGN KEY (DEPARTMENT_NO) REFERENCES DEPARTMENT(DEPARTMENT_NO)  
ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
SELECT * FROM COMPANY;
```

```
+-----+-----+-----+  
| EMPNO | DEPARTMENT_NO | JOINING_DATE |  
+-----+-----+-----+  
| 1 | 1 | 2021-09-08 |  
| 2 | 2 | 2021-09-04 |  
| 3 | 3 | 2021-09-05 |  
+-----+-----+-----+
```

--	--	--

Lab 7 Queries:

```
CREATE TABLE DEPT (DEPTNO INT,DNAME VARCHAR(10),LOC  
VARCHAR(10),PRIMARY KEY (DEPTNO) );
```

```
+-----+-----+-----+-----+-----+  
| Field | Type      | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| DEPTNO | int       | NO   | PRI | NULL    |      |  
| DNAME  | varchar(10) | YES  |     | NULL    |      |  
| LOC    | varchar(10) | YES  |     | NULL    |      |  
+-----+-----+-----+-----+-----+
```

```
+-----+-----+-----+  
| DEPTNO | DNAME | LOC |  
+-----+-----+-----+  
| 1 | A | BANG |  
| 2 | B | HYD |  
| 3 | C | CHE |  
| 4 | D | TIR |  
| 5 | E | VYJ |  
+-----+-----+-----+
```

```
CREATE TABLE EMPLOYEE (EMPNO NUMBER(6),ENAME VARCHAR(20),JOB VARCHAR  
(10), DEPTNO NUMBER(3),SAL NUMBER(7,2), PRIMARY KEY EMPNO)
```

```
+-----+-----+-----+-----+-----+  
| EMPNO | ENAME  | JOB  | DEPTNO | SAL |  
+-----+-----+-----+-----+-----+  
| 1 | RITHVIK | CODE | 1 | 1000 |
```

Rithvik Ravilla
106121104

	2		RAMA		ARCHI		2		2000	
	3		VIJAYA		OWNER		3		3000	
	4		KRISHNA		DRIVER		4		4000	
	5		KUMARI		COOK		5		5000	
+-----+-----+-----+-----+-----+										

1)

Create a procedure to display the details of an employee record from employee table for a given employee number.

DELIMITER //

```
CREATE PROCEDURE EMP_SEARCH (EMP INT) BEGIN SELECT * FROM EMPLOYEE  
WHERE EMPNO = EMP; END ;//
```

```
CALL EMP_SEARCH(1)//
```

+-----+-----+-----+-----+-----+										
	EMPNO		ENAME		JOB		DEPTNO		SAL	
+-----+-----+-----+-----+-----+										
	1		RITHVIK		CODE		1		1000	
+-----+-----+-----+-----+-----+										

2)

Create a procedure to add details of a new employee into employee table

```
CREATE PROCEDURE EMP_ADD (NO INT,NAME VARCHAR(20),JOB_INPUT  
VARCHAR(10),DNO INT, SAL_INPUT INT)  
BEGIN  
INSERT INTO EMPLOYEE (EMPNO,ENAME,JOB,DEPTNO,SAL) VALUES  
(NO,NAME,JOB_INPUT,DNO,SAL_INPUT);  
END //
```

```
CALL EMP_ADD(6,'RAVI','QA',1,6000)//
```

+-----+-----+-----+-----+-----+										
	EMPNO		ENAME		JOB		DEPTNO		SAL	
+-----+-----+-----+-----+-----+										
	1		RITHVIK		CODE		1		1000	
	2		RAMA		ARCHI		2		2000	
	3		VIJAYA		OWNER		3		3000	
	4		KRISHNA		DRIVER		4		4000	
	5		KUMARI		COOK		5		5000	
	6		RAVI		QA		1		6000	
+-----+-----+-----+-----+-----+										

3)

Write a procedure raise_sal which increases the salary of an employee. It accepts an employee number and salary increase amount. It uses the employee number to find the current salary from the EMPLOYEE table and update the salary.

```
CREATE PROCEDURE RAISE_SAL (EMPNO_INPUT INT,INCREASE INT)
```

Rithvik Ravilla
106121104

```
BEGIN UPDATE EMPLOYEE SET SAL = SAL + INCREASE WHERE EMPNO =  
EMPNO_INPUT END //  
CALL RAISE_SAL(1,1000);
```

```
+-----+-----+-----+-----+-----+  
| EMPNO | ENAME  | JOB   | DEPTNO | SAL |  
+-----+-----+-----+-----+-----+  
|  1 | RITHVIK | CODE  | 1 | 2000 |  
|  2 | RAMA    | ARCHI | 2 | 2000 |  
|  3 | VIJAYA  | OWNER | 3 | 3000 |  
|  4 | KRISHNA | DRIVER | 4 | 4000 |  
|  5 | KUMARI  | COOK  | 5 | 5000 |  
|  6 | RAVI    | QA    | 1 | 6000 |  
+-----+-----+-----+-----+-----+
```

4)

Create a procedure to delete a record from employee table for a given employee name.

```
//CHANGE NO TO NAME  
CREATE PROCEDURE DEL_EMPLOYEE (EMPNO_INPUT INT) BEGIN DELETE FROM  
EMPLOYEE WHERE EMPNO = EMPNO_INPUT; END //  
CALL DEL_EMPLOYEE(6);
```

```
+-----+-----+-----+-----+-----+  
| EMPNO | ENAME  | JOB   | DEPTNO | SAL |  
+-----+-----+-----+-----+-----+  
|  1 | RITHVIK | CODE  | 1 | 2000 |  
|  2 | RAMA    | ARCHI | 2 | 2000 |  
|  3 | VIJAYA  | OWNER | 3 | 3000 |  
|  4 | KRISHNA | DRIVER | 4 | 4000 |  
|  5 | KUMARI  | COOK  | 5 | 5000 |  
+-----+-----+-----+-----+-----+
```

5)

Write a function to display minimum salary of employees from the employee table.

```
CREATE FUNCTION MIN_SAL()  
RETURNS INT DETERMINISTIC  
BEGIN RETURN (SELECT MIN(SAL) FROM EMPLOYEE); END//  
SELECT MIN_SAL();
```

```
+-----+  
| MIN_SAL() |  
+-----+  
| 2000 |  
+-----+
```

6)

Write a function to display the number of employees working in the Organization.

```
CREATE FUNCTION COUNT_EMPLOYEE() RETURNS INT DETERMINISTIC BEGIN  
RETURN (SELECT COUNT(*) FROM EMPLOYEE); END//
```

Rithvik Ravilla
106121104

```
SELECT COUNT_EMPLOYEE();
```

```
+-----+  
| COUNT_EMPLOYEE() |  
+-----+  
|          5 |  
+-----+
```

7)

Write a function to display salary of an employee with the given employee number = 5.

```
CREATE FUNCTION SHOW_SAL(EMPNO_INPUT INT) RETURNS INT DETERMINISTIC  
BEGIN  
RETURN (SELECT SAL FROM EMPLOYEE WHERE EMPNO = EMPNO_INPUT);  
END//
```

```
SELECT SHOW_SAL(1)//
```

```
+-----+  
| SHOW_SAL(1) |  
+-----+  
|    2000 |  
+-----+
```

8)

Write a function average which takes DeptNo as input argument and returns the average salary received by the employee in the given department.

```
CREATE FUNCTION AVG_DEPT(DEPTNO_INPUT INT)  
RETURNS DECIMAL DETERMINISTIC  
BEGIN RETURN (SELECT AVG(SAL) FROM EMPLOYEE WHERE DEPTNO =  
DEPTNO_INPUT); END//  
SELECT AVG_DEPT(1);
```

```
+-----+  
| AVG_DEPT(1) |  
+-----+  
|    4000 |  
+-----+
```

9)

Write a procedure which takes the DeptNo =5 as input parameter and lists the names of all employees belonging to that department.

```
CREATE PROCEDURE SHOW_EMPLOYEE (DEPTNO_INPUT INT) BEGIN SELECT *  
FROM EMPLOYEE WHERE DEPTNO = DEPTNO_INPUT; END //  
CALL SHOW_EMPLOYEE(1);
```

```
+-----+-----+-----+-----+-----+  
| EMPNO | ENAME  | JOB  | DEPTNO | SAL |  
+-----+-----+-----+-----+-----+  
|    1 | RITHVIK | CODE |    1 | 2000 |  
|    6 | RAVI   | QA   |    1 | 6000 |
```

Rithvik Ravilla
106121104

+-----+-----+-----+-----+-----+

10)

Write procedure that lists the highest salary drawn by an employee in each of the departments. It should make use of a named procedure dept_highest which finds the highest salary drawn by an employee for the given department.

```
CREATE PROCEDURE MAX_EMPLOYEE (DEPTNO_INPUT INT) BEGIN
SELECT * FROM EMPLOYEE WHERE SAL = (SELECT MAX(SAL) FROM EMPLOYEE
WHERE DEPTNO = DEPTNO_INPUT); END //
```

```
CALL MAX_EMPLOYEE(1)//
```

+-----+-----+-----+-----+-----+

```
| EMPNO | ENAME | JOB | DEPTNO | SAL |
```

+-----+-----+-----+-----+-----+

```
| 6 | RAVI | QA | 1 | 6000 |
```

+-----+-----+-----+-----+-----+

11)

Write a function that will display the number of employees with salary more than 30000.

```
CREATE FUNCTION SAL_3000() RETURNS INT DETERMINISTIC
BEGIN RETURN (SELECT COUNT(*) FROM EMPLOYEE WHERE SAL >= 3000);
END//
```

```
SELECT SAL_3000()//
```

+-----+

```
| SAL_3000() |
```

+-----+

```
| 4 |
```

+-----+

12)

Write a function that will display the count of the number of employees working in Mumbai.

```
CREATE FUNCTION LOC_COUNT(LOC_INPUT VARCHAR(10))
RETURNS INT DETERMINISTIC
BEGIN RETURN (SELECT COUNT(*) FROM EMPLOYEE WHERE DEPTNO = (SELECT
DEPTNO FROM DEPT WHERE LOC = LOC_INPUT));
END//
```

```
SELECT LOC_COUNT('BANG')//
```

+-----+

```
| LOC_COUNT('BANG') |
```

+-----+

```
| 2 |
```

Lab 8 Code:

```
#include <bits/stdc++.h>
```

Rithvik Ravilla
106121104

```
#define int long long
using namespace std;
```

```
int length(int num) {
    int ans = 0;
    while (num) {
        ans++;
        num = num & (num - 1);
    }
    return ans;
}
```

```
bool BFS(int key, int x, vector<vector<int>> &adj) {
    queue<int> q;
    vector<int> vis(x, 0);
    int count = 0;
    for (int i = 0; i < x; i++) {
        if (key & (1LL << i)) {
            q.push(i);
        }
    }
    while (!q.empty()) {
        auto top = q.front();
        q.pop();
        if (vis[top])
            continue;
        vis[top] = 1;
        count++;
        for (int i : adj[top]) {
            if (!vis[i])
                q.push(i);
        }
    }
    return count == x;
}
```

```
int32_t main() {
    int n, m;
    cout << "No of attributes and functional dependencies: ";
    cin >> n >> m;
    vector<vector<int>> adj(n);
    for (int i = 0; i < m; i++) {
        cout << "Enter Input: ";
        int u;
        cin >> u;
        u--;
        cout << "Enter Output: ";
        string s;
        cin >> s;
```


Rithvik Ravilla
106121104

```
        for(auto i:s) {
            int a=i-'0';
            a--;
            adj[u].push_back(a);
        }
    }

    int limit = (1LL << n);
    map<int, vector<int>>> mp;

    cout << "Candidate keys are : \n";
    for (int i : mp.begin()->second) {
        for (int j = 0; j < n; j++) {
            if (i & (1 << j)) {
                cout << j + 1 << " ";
            }
        }
        cout << "\n";
    }
    cout << "Super keys are: \n";

    for (int i = 1; i < limit; i++) {
        if (BFS(i, n, adj)) {
            mp[length(i)].push_back(i);
            for (int j = 0; j < n; j++) {
                if (i & (1 << j)) {
                    cout << j + 1 << " ";
                }
            }
            cout << "\n";
        }
    }
    return 0;
}
```

Input and Output:

1)
No of attributes : 4
No of fds in set : 3
Enter LHS : 123
Enter RHS : 4
Enter LHS : 12
Enter RHS : 34
Enter LHS : 1
Enter RHS : 234
Super Keys :
1
1 2
1 3

Rithvik Ravilla
106121104

1 2 3
1 4
1 2 4
1 3 4
1 2 3 4

Candidate Keys are :

1

2)

No of attributes : 4

No of fds in set : 3

Enter LHS : 1

Enter RHS : 2

Enter LHS : 2

Enter RHS : 3

Enter LHS : 3

Enter RHS : 1

Super Keys :

1 4
2 4
1 2 4
3 4
1 3 4
2 3 4
1 2 3 4

Candidate Keys are :

1 4
2 4
3 4

Lab 9 Queries:

1)

Create a table called EMP with the following structure. Update any one attribute then show the result of following transaction operations.

```
CREATE TABLE EMP(EMPNO INT(6),ENAME VARCHAR(20),JOB VARCHAR(10),DEPT  
VARCHAR(10),DEPTNO INT(3),SAL DECIMAL(7,2), PR  
IMARY KEY (EMPNO));
```

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
ENAME	varchar(20)	YES		NULL	
JOB	varchar(10)	YES		NULL	
DEPT	varchar(10)	YES		NULL	
DEPTNO	int	YES		NULL	
SAL	decimal(7,2)	YES		NULL	

Rithvik Ravilla
106121104

+-----+-----+-----+-----+-----+-----+

SET autocommit = 0;

+-----+-----+-----+-----+-----+-----+

EMPNO	ENAME	JOB	DEPT	DEPTNO	SAL
-------	-------	-----	------	--------	-----

+-----+-----+-----+-----+-----+-----+

1	RITHVIK	CEO	IT	1	100.00
2	RAMA	CTO	IT	1	200.00
3	VIJAYA	CFO	FINANCE	2	300.00

+-----+-----+-----+-----+-----+-----+

COMMIT;

INSERT INTO EMP (EMPNO,ENAME,JOB,DEPT,DEPTNO,SAL) VALUES
(4,"KRISHNA","FOUNDER","TECH",3,400);

+-----+-----+-----+-----+-----+-----+

EMPNO	ENAME	JOB	DEPT	DEPTNO	SAL
-------	-------	-----	------	--------	-----

+-----+-----+-----+-----+-----+-----+

1	RITHVIK	CEO	IT	1	100.00
2	RAMA	CTO	IT	1	200.00
3	VIJAYA	CFO	FINANCE	2	300.00
4	KRISHNA	FOUNDER	TECH	3	400.00

+-----+-----+-----+-----+-----+-----+

ROLLBACK;

+-----+-----+-----+-----+-----+-----+

EMPNO	ENAME	JOB	DEPT	DEPTNO	SAL
-------	-------	-----	------	--------	-----

+-----+-----+-----+-----+-----+-----+

1	RITHVIK	CEO	IT	1	100.00
2	RAMA	CTO	IT	1	200.00
3	VIJAYA	CFO	FINANCE	2	300.00

+-----+-----+-----+-----+-----+-----+

2)

CREATE TABLE PRODUCT(BARCODE INT(6),PNAME VARCHAR(20),PRICE
INT(3),QUANTITY INT(2), PRIMARY KEY (BARCODE));

+-----+-----+-----+-----+-----+-----+

Field	Type	Null	Key	Default	Extra
-------	------	------	-----	---------	-------

+-----+-----+-----+-----+-----+-----+

BARCODE	int	NO	PRI	NULL	
PNAME	varchar(20)	YES		NULL	
PRICE	int	YES		NULL	
QUANTITY	int	YES		NULL	

+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+

BARCODE	PNAME	PRICE	QUANTITY_IN_STOCK
---------	-------	-------	-------------------

+-----+-----+-----+-----+-----+-----+

Rithvik Ravilla
106121104

1	BALL	100	10
2	BAT	200	20
3	WICKET	300	30

CREATE TABLE SALE(SALEID INT(6),DELIVERYADDRESS VARCHAR(20),CREDITCARD INT(7), PRIMARY KEY (SALEID));

Field	Type	Null	Key	Default	Extra
SALEID	int	NO	PRI	NULL	
DELIVERYADDRESS	varchar(20)	YES		NULL	
CREDITCARD	int	YES		NULL	

SALEID	DELIVERYADDRESS	CREDITCARD
1	ELAN	123
2	PALM	456
3	SENIORITA	789

CREATE TABLE SALEITEM(SALEID INT(6),BARCODE INT(6),QUANTITYBOUGHT INT(2), FOREIGN KEY (BARCODE) REFERENCES PRODUCT(BARCODE), FOREIGN KEY (SALEID) REFERENCES SALE(SALEID));

Field	Type	Null	Key	Default	Extra
SALEID	int	YES	MUL	NULL	
BARCODE	int	YES	MUL	NULL	
QUANTITY	int	YES		NULL	

(I)

Create a trigger called updateAvailableQuantity that updates the quantity in stock in the Product table, for every product sold. The trigger should be executed after each insert operation on the SaleItem table: for the product with the given barcode (the one inserted into SaleItem), update the available quantity in Product table to be the old quantity minus the sold quantity.

```
CREATE TRIGGER UPDATE_QUANTITY
AFTER INSERT ON SALEITEM
FOR EACH ROW UPDATE PRODUCT
SET QUANTITY_IN_STOCK = QUANTITY_IN_STOCK - NEW.QUANTITY
WHERE BARCODE = NEW.BARCODE;
```

INSERT INTO SALEITEM (SALEID,BARCODE,QUANTITY) VALUES (1,1,4);

--	--	--	--

Rithvik Ravilla
106121104

BARCODE	PNAME	PRICE	QUANTITY_IN_STOCK
1	BALL	100	6
2	BAT	200	20
3	WICKET	300	30

(II)

Create a stored procedure called spInsertProduct that inserts a new product into the database, under some conditions. The stored procedure has as input parameters the barcode, the product name, price, and quantityInStock. The stored procedure should insert a row in the Product table only if the price is greater than 0 and the quantity is greater or equal to 0

DELIMITER //

```
CREATE PROCEDURE INSERTPRODUCT (BARC INT,PN VARCHAR(20), PR INT, QNY INT)
BEGIN IF QNY>0 AND PR > 0 THEN
INSERT INTO PRODUCT(BARCODE,PNAME,PRICE,QUANTITY_IN_STOCK) VALUES
(BARC,PN,PR,QNY)
; END IF
; END;//
```

```
CALL INSERTPRODUCT(5,"SNACK",-2,50);
```

(III)

Create a function called spreturn that returns the total price of a product by passing the quantity and barcode.

```
CREATE FUNCTION SPRETURN(BARC INT, QTY INT)
RETURNS INT DETERMINISTIC
BEGIN RETURN QTY*(SELECT PRICE FROM PRODUCT WHERE BARCODE = BARC);
END//
```

```
SELECT SPRETURN(1,20)//
```

SPRETURN(1,20)
2000

Lab 10 Queries:

1)

(I)

(i)

Create a query to list the salary > 300

Rithvik Ravilla
106121104

```
for $val in doc("../lab10/employee.xml")/EmployeeDetails/Employee
where $val/Salary > 300
return $val/EmpName
Krishna Singh Kumari Singh
```

(ii)

Get Employee numbers of employees whose last name starts with "R".

```
for $val in doc("../lab10/employee.xml")/EmployeeDetails/Employee
where starts-with($val/EmpName,"R")
return $val/EmpName
Rithvik Ravilla Rama Ravilla
```

(iii)

Get names of employees in the "CS" department.

```
let $d:=doc("../lab10/employee.xml")
for $e in $d/EmployeeDetails/Employee
where $e/Dept = "CS"
return $e/EmpName
Rithvik Ravilla Krishna Singh
```

(iv)

Get employees who are managers work more than 8 hours

```
let $d:=doc("../lab10/employee.xml")
for $e in $d/EmployeeDetails/Employee
where $e/Job = "Manager" and $e/WorkingHours >8
return $e/EmpName
Krishna Singh Kumari Singh
```

(v)

Display the salary in highest to lowest.

```
let $d:=doc("../lab10/employee.xml")
for $e in $d/EmployeeDetails/Employee
order by $e/Salary descending
return $e/EmpName
Kumari Singh Krishna Singh Vijaya Allu Rama Ravilla Rithvik Ravilla
```

(vi)

Display the Employee name in Alphabetical order.

```
let $d:=doc("../lab10/employee.xml")
for $e in $d/EmployeeDetails/Employee
order by $e/EmpName
```

Rithvik Ravilla
106121104

return \$e/EmpNo
4 5 2 1 3

1)
(II)

(i)
Create a xquery to list the Marks > 20

```
let $d:=doc("../lab10/student.xml")
for $e in $d/StudentDetails/Student
for $mark in $e/Marks
where $mark > 20
return $e
```

(ii)
Find the Avg Mark of a Student.

```
let $d:=doc("../lab10/student.xml")
for $e in $d/StudentDetails/Student
return avg(
  for $mark in $e/Marks
  return $mark
)
11 14 17 20 23
```

(iii)
Find the Total Marks of a Student.

```
let $d:=doc("../lab10/student.xml")
for $e in $d/StudentDetails/Student
return sum(for $mark in $e/Marks
return $mark)
33 42 51 60 69
```

(iv)
Find the Min and Max Mark of a student in a subject.

```
let $d:=doc("../lab10/student.xml")
for $e in $d/StudentDetails/Student
return max(for $mark in $e/Marks
return $mark)
12 15 18 21 24
```

```
let $d:=doc("../lab10/student.xml")
for $e in $d/StudentDetails/Student
return min(for $mark in $e/Marks
return $mark)
```

Rithvik Ravilla
106121104

12 15 18 21 24

2)

(I)

(i)

Create a xquery to list the price of journey < 300

```
let $d:=doc("../lab10/flight.xml")
for $e in $d/FlightDetails/Flight/Price
where $e < 300
return $e
```

100 200

(ii)

Create a xquery to find the departs Time of the particular flight on a 4.12.2020 from a particular city.

```
let $d:=doc("../lab10/flight.xml")
for $e in $d/FlightDetails/Flight
where $e/Date = "4.12.2020"
return $e/Departs
5:00
```

(iii)

Create a xquery to find the Flight Names handled by a particular Pilot.

```
let $d:=doc("../lab10/flight.xml")
for $e in $d/FlightDetails/Flight
where $e/PilotName = "Rithvik"
return $e/FIName
A D
```

(iv)

Create a xquery to find out number of Flight journeys of a particular flight on 30.11.2020

```
let $d:=doc("../lab10/flight.xml")
return count
(for $e in $d/FlightDetails/Flight
where $e/Date = "30.11.2020"
return $e)
2
```

(v)

Create a xquery to find Arrival Time of a particular flight on 25.11.2020 from a particular city.

```
let $d:=doc("../lab10/flight.xml")
for $e in $d/FlightDetails/Flight
```


Rithvik Ravilla
106121104

where \$e/Date = "25.11.2020"
return \$e/Arrives
23:00 14:45

2)
(II)
(i)

Create a xquery to list the employees in Dept ='Human Resources'.

```
let $d:=doc("../lab10/employee_2.xml")
for $e in $d/EmployeeDetails/Employee
where $e/Dept = "HR"
return $e/Ename
Vijaya Allu Kumari Singh
```

(ii)

Create a xquery to find the Employee who works in particular project and salary > 50000.

```
let $d:=doc("../lab10/employee_2.xml")
for $e in $d/EmployeeDetails/Employee
where $e/Dept = "HR" and $e/Salary > 400
return $e/Ename
Kumari Singh
```

(iii)

Create a xquery to find the Total salary of Employees in a particular department.

```
let $d:=doc("../lab10/employee_2.xml")
return sum (
for $e in $d/EmployeeDetails/Employee
where $e/Dept = "HR"
return $e/Salary
)
800
```

(iv)

Create a xquery to find the number of Employees working in a department.

```
let $d:=doc("../lab10/employee_2.xml")
return count (
for $e in $d/EmployeeDetails/Employee
where $e/Dept = "HR"
return $e
)
2
```

(v)

Create a xquery to find the highest salary of a manager in particular department.

Rithvik Ravilla
106121104

```
let $d:=doc("../lab10/employee_2.xml")
return max(
  for $e in $d/EmployeeDetails/Employee
  where $e/Dept = "HR"
  return $e/Salary
)
500
```