

Faculty of Engineering & Technology (coed)

****

**DBMS Laboratory (22CS43)**

**Manual**

**Department of Computer Science and Engineering**

**DBMS LABORATORY MANUAL**

## SEMESTER IV

**LAB EXPERIMENTS**

**PART A: SQL PROGRAMMING**

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, Programme\_id, No-of\_Copies)**

**BOOK\_LENDING(Book\_id, Programme\_id, Card\_No, Date\_Out, Due\_Date)**

**LIBRARY\_PROGRAMME(Programme\_id, Programme\_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 Programme, etc.**
  2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017**
  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.**

**Program Objectives:**

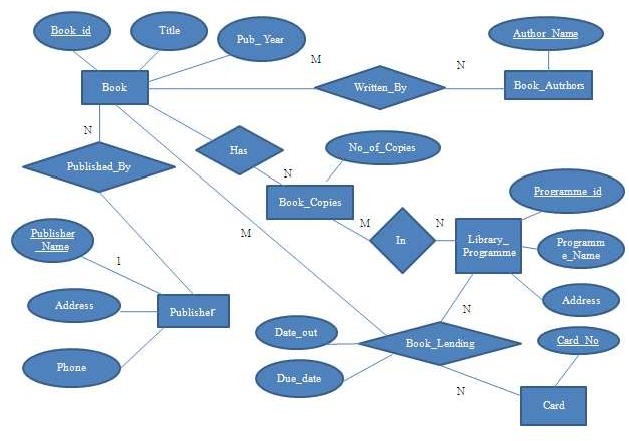
**This course will enable students to**

Foundation knowledge in database concepts, technology and practice to groomstudents into well-informed database application developers.

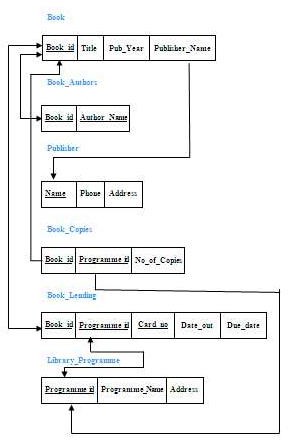
Strong practice in SQL programming through a variety of database problems. Develop database applications using front-end tools and back-end DBMS.

**Solution:**

**Entity-Relationship Diagram**

****

**Schema Diagram**

****

**Table Creation**

CREATE TABLE BOOK ( BOOK\_ID INT (10) PRIMARY KEY, TITLE VARCHAR (20),

PUB\_YEAR VARCHAR (20),

PUBLISHER\_NAME VARCHAR (20),

FOREIGN KEY (PUBLISHER\_NAME) REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLE BOOK\_AUTHORS ( AUTHOR\_NAME VARCHAR (20),

BOOK\_ID INT (10),

PRIMARY KEY (BOOK\_ID, AUTHOR\_NAME),

FOREIGN KEY (BOOK\_ID) REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE);

CREATE TABLE PUBLISHER (

NAME VARCHAR (20) PRIMARY KEY, PHONE BIGINT (20),

ADDRESS VARCHAR (100));

CREATE TABLE BOOK\_COPIES ( NO\_OF\_COPIES INT (5),

BOOK\_ID INT (10),

PROGRAMME\_ID INT (10),

PRIMARY KEY (BOOK\_ID,PROGRAMME\_ID),

FOREIGN KEY (BOOK\_ID) REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE, FOREIGN KEY (PROGRAMME\_ID) REFERENCES LIBRARY\_PROGRAMME (PROGRAMME\_ID) ON DELETE CASCADE);

CREATE TABLE BOOK\_LENDING ( DATE\_OUT DATE,

DUE\_DATE DATE, BOOK\_ID INT (10),

PROGRAMME\_ID INT (10),

CARD\_NO INT (10),

PRIMARY KEY (BOOK\_ID,PROGRAMME\_ID, CARD\_NO),

FOREIGN KEY (BOOK\_ID) REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE, FOREIGN KEY (PROGRAMME\_ID) REFERENCES LIBRARY\_PROGRAMME(PROGRAMME\_ID) ON DELETE CASCADE,

FOREIGN KEY (CARD\_NO) REFERENCES CARD (CARD\_NO) ON DELETE CASCADE);

CREATE TABLE CARD (

CARD\_NO INT (10) PRIMARY KEY);

CREATE TABLE LIBRARY\_PROGRAMME ( PROGRAMME\_ID INT (10) PRIMARY KEY, PROGRAMME\_NAME VARCHAR (50),

ADDRESS VARCHAR (100));

**Table Descriptions**

DESC BOOK

;



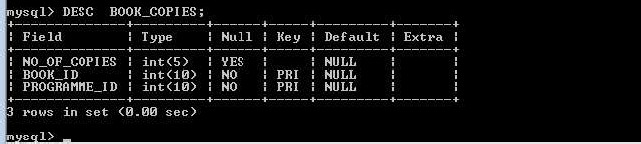
DESC BOOK\_AUTHORS;



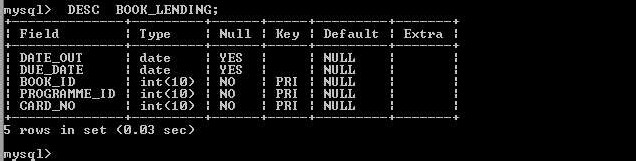
DESC PUBLISHER;



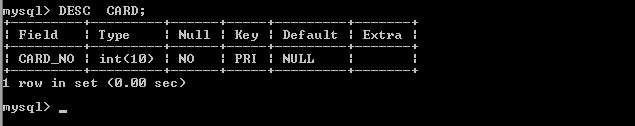
DESC BOOK\_COPIES



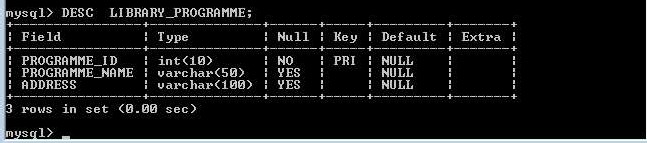
DESC BOOK\_LENDING;



DESC CARD;



DESC LIBRARY\_PROGRAMME



**Insertion of Values to Tables**

INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL'); INSERT INTO BOOK VALUES (2,'ADBMS','JUN-2016','MCGRAW-HILL'); INSERT INTO BOOK VALUES (3, 'CD','SEP-2016','PEARSON');

INSERT INTO BOOK VALUES (4,' ALGORITHMS ','SEP-2015',' MIT'); INSERT INTO BOOK VALUES (5,'OS','MAY-2016','PEARSON');

INSERT INTO BOOK\_AUTHORS VALUES ('NAVATHE', 1); INSERT INTO BOOK\_AUTHORS VALUES ('NAVATHE', 2); INSERT INTO BOOK\_AUTHORS VALUES ('ULLMAN',3); INSERT INTO BOOK\_AUTHORS VALUES ('CHARLES', 4); INSERT INTO BOOK\_AUTHORS VALUES('GALVIN', 5);

INSERT INTO PUBLISHER VALUES ('MCGRAW-HILL', 9989076587,'BANGALORE'); INSERT INTO PUBLISHER VALUES ('PEARSON', 9889076565,'NEWDELHI');

INSERT INTO PUBLISHER VALUES ('PRENTICE HALL', 7455679345,'HYEDRABAD'); INSERT INTO PUBLISHER VALUES ('WILEY', 8970862340,'CHENNAI');

INSERT INTO PUBLISHER VALUES ('MIT',7756120238,'BANGALORE');

INSERT INTO BOOK\_COPIES VALUES (10, 1, 10);

INSERT INTO BOOK\_COPIES VALUES (5, 1, 11);

INSERT INTO BOOK\_COPIES VALUES (2, 2, 12);

INSERT INTO BOOK\_COPIES VALUES (5, 2, 13);

INSERT INTO BOOK\_COPIES VALUES (7, 3, 14);

INSERT INTO BOOK\_COPIES VALUES (1, 5, 10);

INSERT INTO BOOK\_COPIES VALUES (3, 4, 11);

INSERT INTO BOOK\_LENDING VALUES ('2017-01-01','2017-06-01', 1, 10, 101);

INSERT INTO BOOK\_LENDING VALUES ('2017-01-11 ','2017-03-11', 3, 14, 101);

INSERT INTO BOOK\_LENDING VALUES ('2017-02-21','2017-04-21', 2, 13, 101);

INSERT INTO BOOK\_LENDING VALUES ('2017-03-15 ','2017-07-15', 4, 11, 101);

INSERT INTO BOOK\_LENDING VALUES ('2017-04-12','2017-05-12', 1, 11, 104);

INSERT INTO CARD VALUES (100); INSERT INTO CARD VALUES (101); INSERT INTO CARD VALUES (102); INSERT INTO CARD VALUES (103); INSERT INTO CARD VALUES (104);

INSERT INTO LIBRARY\_PROGRAMME VALUES (10,'VIJAY NAGAR','MYSURU'); INSERT INTO LIBRARY\_PROGRAMME VALUES (11,'VIDYANAGAR','HUBLI'); ; INSERT INTO LIBRARY\_PROGRAMME VALUES(12,'KUVEMPUNAGAR','MYSURU'); INSERT INTO LIBRARY\_PROGRAMME VALUE(13,'RAJAJINAGAR','BANGALORE'); INSERT INTO LIBRARY\_PROGRAMME VALUES (14,'MANIPAL','UDUPI');

SELECT \* FROM BOOK;

|  |  |  |  |
| --- | --- | --- | --- |
| **BOOK\_ID** | **TITLE** | **PUB\_YEAR** | **PUBLISHER\_NAME** |
| 1 | DBMS | Jan-2017 | MCGRAW-HILL |
| 2 | ADBMS | Jun-2017 | MCGRAW-HILL |
| 3 | CD | Sep-2016 | PEARSON |
| 4 | ALGORITHMS | Sep-2015 | MIT |
| 5 | OS | May-2016 | PEARSON |

SELECT \* FROM BOOK\_AUTHORS;

|  |  |
| --- | --- |
| **AUTHOR\_NAME** | **BOOK\_ID** |
| NAVATHE | 1 |
| NAVATHE | 2 |
| ULLMAN | 3 |
| CHARLES | 4 |
| GALVIN | 5 |

SELECT \* FROM PUBLISHER;

|  |  |  |
| --- | --- | --- |
| **NAME** | **PHONE** | **ADDRESS** |
| MCGRAW-HILL | 9989076587 | BANGALORE |
| MIT | 7756120238 | BANGALORE |
| PEARSON | 9889076565 | NEWDELHI |
| PRENTICE HALL | 7455679345 | HYEDRABAD |
| WILEY | 8970862340 | CHENNAI |

SELECT \* FROM BOOK\_COPIES;

|  |  |  |
| --- | --- | --- |
| **NO\_OF\_COPIES** | **BOOK\_ID** | **PROGRAMME\_ID** |
| 10 | 1 | 10 |
| 5 | 1 | 11 |
| 2 | 2 | 12 |
| 5 | 2 | 13 |
| 7 | 3 | 14 |
| 1 | 5 | 10 |
| 3 | 4 | 11 |

SELECT \* FROM BOOK\_LENDING;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DATEOUT** | **DUEDATE** | **BOOKID** | **PROGRAMME\_ID** | **CARDNO** |
| 2017-01-01 | 2017-06-01 | 1 | 10 |  |
| 2017-01-11 | 2017-03-11 | 3 | 4 | 101 |
| 2017-02-21 | 2017-04-21 | 2 | 13 | 101 |
| 2017-03-15 | 2017-07-15 | 4 | 11 | 101 |
| 2017-04-12 | 2017-05-12 | 1 | 11 | 104 |

SELECT \* FROM CARD;

|  |
| --- |
| **CARDNO** |
| 101 |
| 102 |
| 103 |
| 104 |
| 105 |

SELECT \* FROM LIBRARY\_PROGRAMME;

|  |  |  |
| --- | --- | --- |
| **PROGRAMME\_ID** | **PROGRAMME\_NAME** | **ADDRESS** |
| 10 | VIJAY NAGAR | MYSURU |
| 11 | VIDYANAGAR | HUBLI |
| 12 | KUVEMPUNAGAR | MYSURU |
| 13 | RAJAJINAGAR | BANGALORE |
| 14 | MANIPAL | UDUPI |

**Queries:**

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

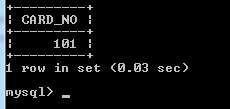
SELECT B.BOOK\_ID, B.TITLE, B.PUBLISHER\_NAME, A.AUTHOR\_NAME, C.NO\_OF\_COPIES, L.PROGRAMME\_ID FROM BOOK B, BOOK\_AUTHORS A, BOOK\_COPIES

C, LIBRARY\_PROGRAMME L WHERE B.BOOK\_ID=A.BOOK\_ID AND B.BOOK\_ID=C.BOOK\_ID AND L.PROGRAMME\_ID=C.PROGRAMME\_ID;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BOOK\_ ID** | **TITLE** | **PUBLISHER\_ NAME** | **AUTHOR\_ NAME** | **NO\_ OF\_COPIES** | **PROGRAMME**  **\_ID** |
| 1 | DBMS | MCGRAW-HILL | NAVATHE | 10 | 10 |
| 1 | DBMS | MCGRAW-HILL | NAVATHE | 5 | 11 |
| 2 | ADBMS | MCGRAW-HILL | NAVATHE | 2 | 12 |
| 2 | ADBMS | MCGRAW-HILL | NAVATHE | 5 | 13 |
| 3 | CD | PEARSON | ULLMAN | 7 | 14 |
| 4 | ALGORITHMS | MIT | CHARLES | 1 | 11 |
| 5 | OS | PEARSON | GALVIN | 3 | 10 |

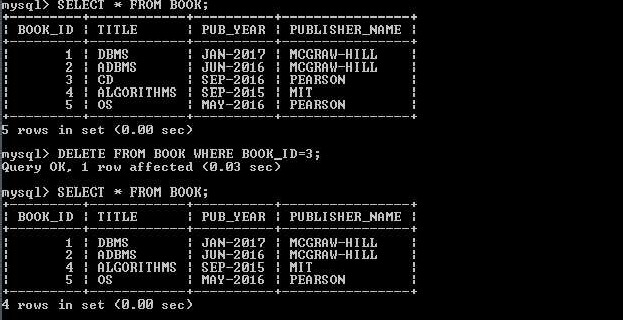
1. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.**

SELECT CARD\_NO FROM BOOK\_LENDING WHERE DATE\_OUT BETWEEN '2017-01-01'AND '2017-07-01' GROUP BY CARD\_NO HAVING COUNT(\*)>3;



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

DELETE FROM BOOK WHERE BOOK\_ID=3;



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

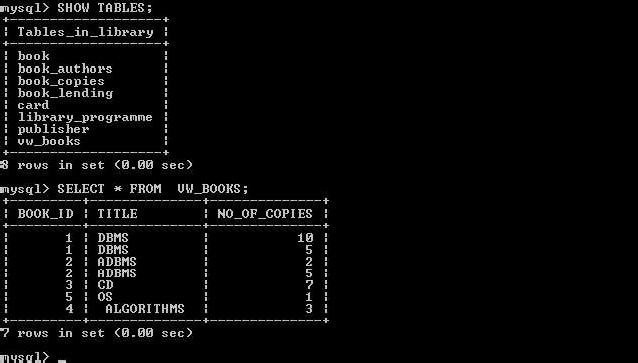
CREATE VIEW VW\_PUBLICATION AS SELECT PUB\_YEAR FROM BOOK; SELECT \* FROM VW\_PUBLICATION



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

CREATE VIEW VW\_BOOKS AS SELECT B.BOOK\_ID, B.TITLE, C.NO\_OF\_COPIES FROM BOOK B, BOOK\_COPIES C, LIBRARY\_PROGRAMME L WHERE B.BOOK\_ID=C.BOOK\_ID AND C.PROGRAMME\_ID=L.PROGRAMME\_ID;

SELECT \* FROM VW\_BOOKS;



**Program Outcomes:**

**The students are able to**

Create, Update and query on the database.

Demonstrate the working of different concepts of DBMS

Implement, analyze and evaluate the project developed for an application.

1. **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, Salesman\_id)**

**Write SQL queries to**

* 1. **** **average.**
  2. **Find the name and numbers of all salesmen who had more than one customer.**
  3. **** **their**

**cities (Use UNION operation.)**

* 1. **Create a view that finds the salesman who has the customer with the highest order of a day.**
  2. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

**Program Objectives:**

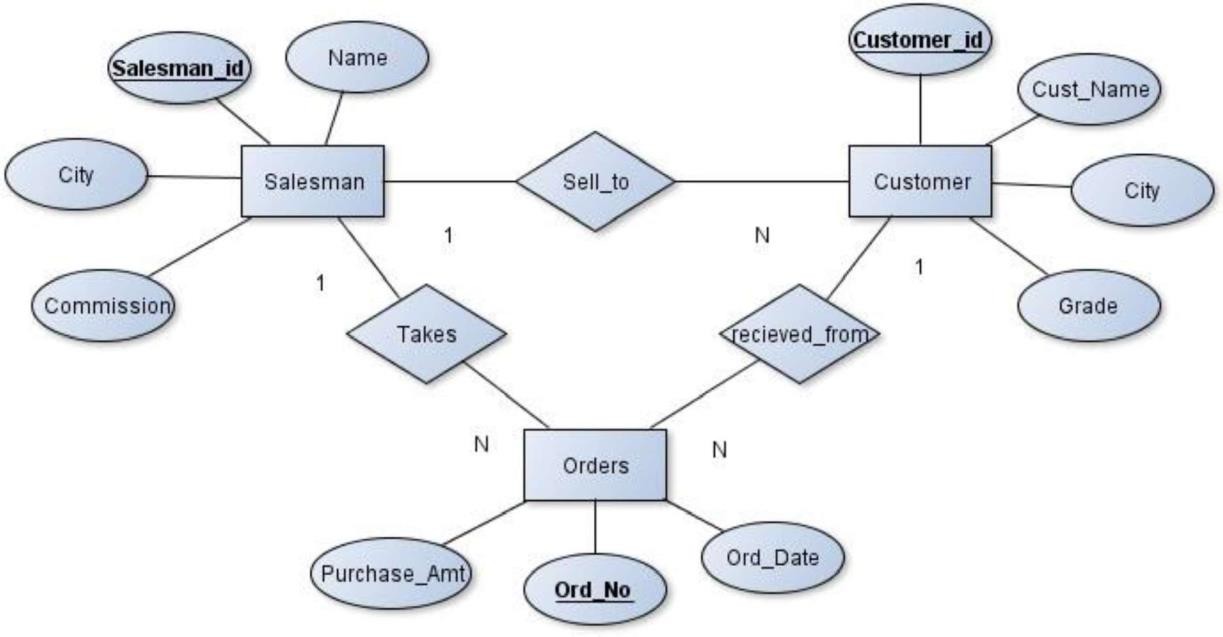
**This course will enable students to**

Foundation knowledge in database concepts, technology and practice to groomstudents into well-informed database application developers.

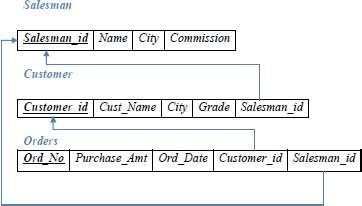
Strong practice in SQL programming through a variety of database problems.

Develop database applications using front-end tools and back-end DBMS.

**Solution:**

**Entity-Relationship Diagram**

**Schema Diagram**

****

**Table Creation**

CREATE TABLE SALESMAN ( SALESMAN\_ID INT (4) PRIMARY KEY, NAME VARCHAR (20),

CITY VARCHAR (20),

COMMISSION VARCHAR (20));

CREATE TABLE CUSTOMER ( CUSTOMER\_ID INT (5) PRIMARY KEY, CUST\_NAME VARCHAR (20),

CITY VARCHAR (20), GRADE INT (4),

SALESMAN\_ID INT (6),

FOREIGN KEY (SALESMAN\_ID) REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE SET NULL);

CREATE TABLE ORDERS ( ORD\_NO INT (5) PRIMARY KEY, PURCHASE\_AMT DECIMAL (10, 2), ORD\_DATE DATE,

CUSTOMER\_ID INT (4),

SALESMAN\_ID INT (4),

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER (CUSTOMER\_ID) ON DELETE CASCADE,

FOREIGN KEY (SALESMAN\_ID) REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE CASCADE);

**Table Descriptions**

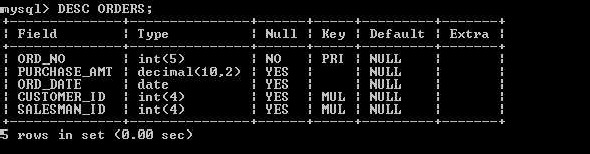
DESC SALESMAN;



DESC CUSTOMER;



DESC ORDERS;



Insertion of Value

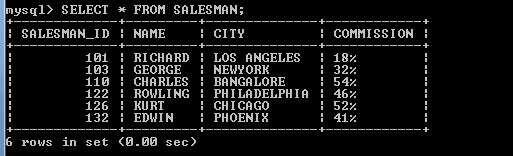
INSERT INTO SALESMAN VALUES(101,'RICHARD','LOS ANGELES','18%'); INSERT INTO SALESMAN VALUES(103,'GEORGE','NEWYORK','32%'); INSERT INTO SALESMAN VALUES(110,'CHARLES','BANGALORE','54%'); INSERT INTO SALESMAN VALUES(122,'ROWLING','PHILADELPHIA','46%'); INSERT INTO SALESMAN VALUES(126,'KURT','CHICAGO','52%');

INSERT INTO SALESMAN VALUES(132,'EDWIN','PHOENIX','41%');

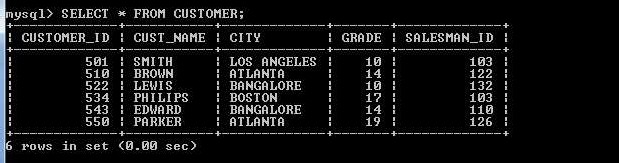
INSERT INTO CUSTOMER VALUES(501,'SMITH','LOS ANGELES',10,103); INSERT INTO CUSTOMER VALUES(510,'BROWN','ATLANTA',14,122); INSERT INTO CUSTOMER VALUES(522,'LEWIS','BANGALORE',10,132); INSERT INTO CUSTOMER VALUES(534,'PHILIPS','BOSTON',17,103); INSERT INTO CUSTOMER VALUES(543,'EDWARD','BANGALORE',14,110); INSERT INTO CUSTOMER VALUES(550,'PARKER','ATLANTA',19,126);

INSERT INTO ORDERS VALUES(1,1000, '2017-05-04',501,103); INSERT INTO ORDERS VALUES(2,4000,'2017-01-,522,132); INSERT INTO ORDERS VALUES(3,2500, '2017-02-24',550,126); INSERT INTO ORDERS VALUES(5,6000,'2017-04-13',522,103); INSERT INTO ORDERS VALUES(6,7000, '2017-03-09',550,126); INSERT INTO ORDERS VALUES (7,3400,'2017-01-20',501,122);

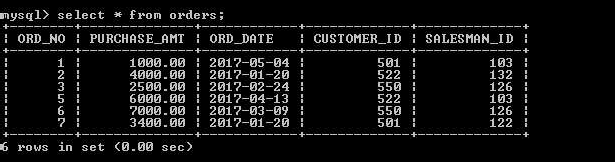
SELECT \* FROM SALESMAN;



SELECT \* FROM CUSTOMER;



SELECT \* FROM ORDERS;

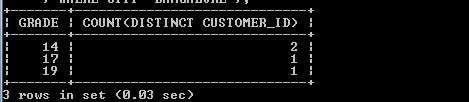


**Queries**

1. **** **average.**

SELECT GRADE, COUNT (CUSTOMER\_ID) FROM CUSTOMER GROUP BY GRADE

HAVING GRADE > (SELECT AVG (GRADE) FROM CUSTOMER WHERE CITY='BANGALORE');



SELECT GRADE,COUNT(DISTINCT CUSTOMER\_ID) FROM CUSTOMER GROUP BY GRADE

HA NG GRADE >( E ECT AVG(GRADE) FROM CUST W E CITY=

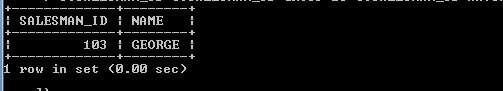
1. **Find the name and numbers of all salesmen who had more than one customer.**

SELECT SALESMAN\_ID,NAME FROM SALESMAN A

WHERE 1 <(SELECT COUNT(\*) FROM CUSTOMER WHERE SALESMAN\_ID=A.SALESMAN\_ID)

## OR

SELECT S.SALESMAN\_ID,NAME, FROM CUSTOMER C,SALESMAN S WHERE S.SALESMAN\_ID=C.SALESMAN\_ID GROUP BY C.SALESMAN\_ID HAVING COUNT(\*)>1



1. **** **their cities**

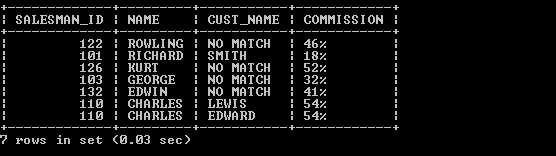
**(Use UNION operation.)**

SELECT S.SALESMAN\_ID,NAME,CUST\_NAME,COMMISSION FROM SALESMAN S,CUSTOMER C

WHERE S.CITY = C.CITY UNION

SELECT SALESMAN\_ID, NAME, 'NO MATCH',COMMISSION FROM SALESMAN WHERE NOT CITY = ANY (SELECT CITY

FROM CUSTOMER) ORDER BY 2 DESC;



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

CREATE VIEW VW\_ELITSALESMAN AS

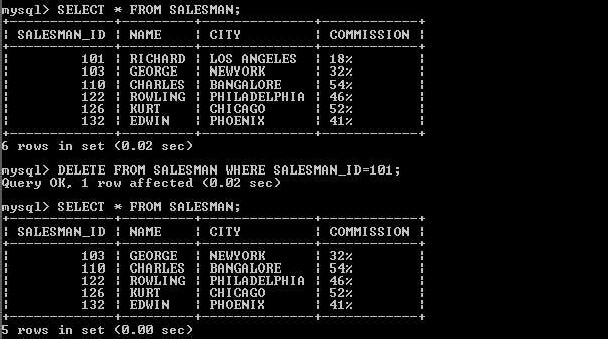
SELECT B.ORD\_DATE,A.SALESMAN\_ID,A.NAME FROM SALESMAN A, ORDERS B WHERE A.SALESMAN\_ID = B.SALESMAN\_ID AND B.PURCHASE\_AMT=(SELECT MAX(PURCHASE\_AMT) FROM ORDERS C

WHERE C.ORD\_DATE = B.ORD\_DATE); SELECT \* FROM VW\_ELITSALESMAN

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

Use ON DELETE CASCADE at the end of foreign key definitions while creating child table orders and then execute the following:

DELETE FROM SALESMAN WHERE SALESMAN\_ID=101;



**Program Outcomes:**

**The students are able to**

Create, Update and query on the database.

Demonstrate the working of different concepts of DBMS

Implement, analyze and evaluate the project developed for an application.

1. **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 **
  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 2000 and also in a movie after 2015 (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 movietitle.**
  5. **** **5.**

**Program Objectives:**

**This course will enable students to**

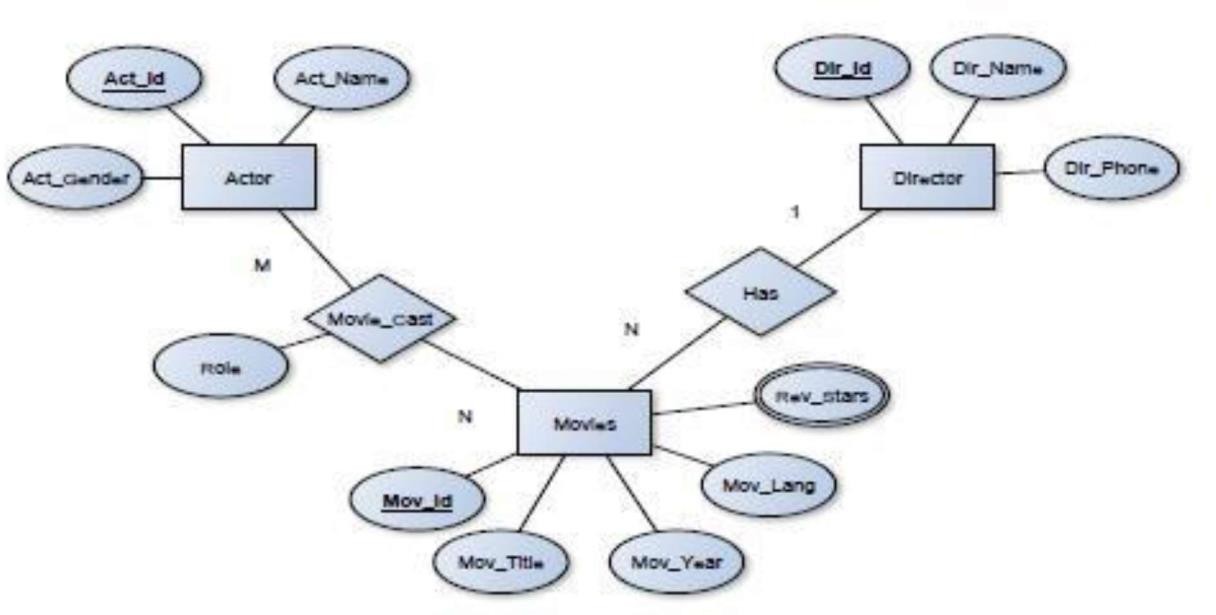
Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.

Strong practice in SQL programming through a variety of database problems.

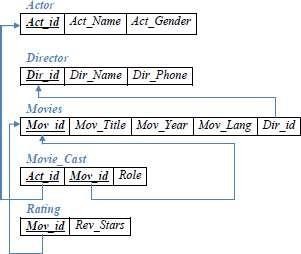
Develop database applications using front-end tools and back-end DBMS.

**Solution:**

**Entity-Relationship Diagram**

****

**Schema Diagram**

****

**Table Creation**

CREATE TABLE ACTOR ( ACT\_ID INT (5) PRIMARY KEY, ACT\_NAME VARCHAR (20),

ACT\_GENDER CHAR (1));

CREATE TABLE DIRECTOR ( DIR\_ID INT (5) PRIMARY KEY, DIR\_NAME VARCHAR (20), DIR\_PHONE BIGINT);

CREATE TABLE MOVIES (MOV\_ID INT (4) PRIMARY KEY, MOV\_TITLE VARCHAR (50),

MOV\_YEAR INT (4),

MOV\_LANG VARCHAR (20),

DIR\_ID INT (5),

FOREIGN KEY (DIR\_ID) REFERENCES DIRECTOR(DIR\_ID));

CREATE TABLE MOVIES\_CAST ( ACT\_ID INT (5),

MOV\_ID INT (5),

ROLE VARCHAR (20),

PRIMARY KEY (ACT\_ID, MOV\_ID),

FOREIGN KEY (ACT\_ID) REFERENCES ACTOR (ACT\_ID), FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

CREATE TABLE RATING ( MOV\_ID INT (5) PRIMARY KEY, REV\_STARS VARCHAR (25),

FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

**Table Descriptions**

DESC ACTOR;



DESC DIRECTOR;



DESC MOVIES;



DESC MOVIES\_CAST;



DESC RATING;



**Insertion of Values to Tables**

INSERT INTO ACTOR VALUES (  INSERT INTO ACTOR VALUES ( INSERT INTO ACTOR VALUES ( INSERT INTO ACTOR VALUES (

 9563400156); INSERT INTO DIRECTOR VALUES(102,'ALAN TAYLOR',9971960035);

75); INSERT INTO DIRECTOR VALUES (105,'HITCHCOCK',7766138911); INSERT INTO DIRECTOR VALUES (106,'STEVEN SPIELBERG',9966138934);

25);

INSERT INTO MOVIES VALUES (501,'JAB HARRY MET SEJAL',2017,'HINDI',104); INSERT INTO MOVIES VALUES (502,'RAJAKUMARA',2017,'KANNADA',103); INSERT INTO MOVIES VALUES (503,'JOLLY LLB 2', 2013,'HINDI', 100);

INSERT INTO MOVIES VALUES (504,'TERMINATOR GENESYS',2015,'ENGLISH',102); INSERT INTO MOVIES VALUES (505,'JAWS',1975,'ENGLISH',106);

INSERT INTO MOVIES VALUES (506,'BRIDGE OF SPIES',2015,'ENGLISH', 106); INSERT INTO MOVIES VALUES (507,'VERTIGO',1943,'ENGLISH',105);

INSERT IN

INSERT INTO MOVIES\_CAST VALUES (1, 501,'HEROINE'); INSERT INTO MOVIES\_CAST VALUES (1, 502,'HEROINE'); INSERT INTO MOVIES\_CAST VALUES (3, 503,'COMEDIAN'); INSERT INTO MOVIES\_CAST VALUES (4, 504,'GUEST'); INSERT INTO MOVIES\_CAST VALUES (4, 501,'HERO');

INSERT INTO RATING VALUES (501, 4);

INSERT INTO RATING VALUES (502, 2);

INSERT INTO RATING VALUES (503, 5);

INSERT INTO RATING VALUES (504, 4);

INSERT INTO RATING VALUES (505, 3);

INSERT INTO RATING VALUES (506, 2);

SELECT \* FROM ACTOR;

|  |  |  |
| --- | --- | --- |
| **ACT\_ID** | **ACT\_NAME** | **ACT** |
| 1 | MADHURI DIXIT | F |
| 2 | AAMIR KHAN | M |
| 3 | JUHI CHAWLA | F |
| 4 | SRIDEVI | F |

SELECT \* FROM DIRECTOR;

|  |  |  |
| --- | --- | --- |
| **DIR\_ID** | **DIR\_NAME** | **DIR\_PHONE** |
| 100 | SUBHASH KAPOOR | 56340015 |
| 102 | ALAN TAYLOR | 719600310 |
| 103 | SANTHOSH ANANDDRAM | 99346111 |
| 104 | IMTIAZ ALI | 85399209 |
| 105 | HITCHCOCK | 7766138911 |
| 106 | STEVEN SPIELBERG | 9966138934 |

SELECT \* FROM MOVIES;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOV\_ID** | **MOV\_TITLE** | **MOV\_YEAR** | **MOV\_LANG** | **DIR\_ID** |
| 501 | JAB HARRY MET SEJAL | 2017 | HINDI | 104 |
| 502 | RAJAKUMARA | 2017 | KANNADA | 103 |
| 503 | JOLLY LLB 2 | 2013 | HINDI | 100 |
| 504 | TERMINATOR GENESYS | 2015 | ENGLISH | 102 |
| 505 | JAWS | 1975 | ENGLISH | 106 |
| 506 | BRIDGE OF SPIES | 2015 | ENGLISH | 106 |
| 507 | VERTIGO | 1958 | ENGLISH | 105 |
| 508 | SHADOW OF A DOUBT | 1943 | ENGLISH | 105 |

SELECT \* FROM MOVIE\_CAST;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOV\_ID** | **MOV\_TITLE** | **MOV\_YEAR** | **MOV\_LANG** | **DIR\_ID** |
| 501 | JAB HARRY MET SEJAL | 2017 | HINDI | 104 |
| 502 | RAJAKUMARA | 2017 | KANNADA | 103 |
| 503 | JOLLY LLB 2 | 2013 | HINDI | 100 |
| 504 | TERMINATOR GENESYS | 2015 | ENGLISH | 102 |
| 505 | JAWS | 1975 | ENGLISH | 106 |
| 506 | BRIDGE OF SPIES | 2015 | ENGLISH | 106 |
| 507 | VERTIGO | 1958 | ENGLISH | 105 |
| 508 | SHADOW OF A DOUBT | 1943 | ENGLISH | 105 |

SELECT \* FROM RATING;

|  |  |
| --- | --- |
| **MOV\_ID** | **REV\_STARS** |
| 501 | 4 |
| 502 | 2 |
| 503 | 5 |
| 504 | 4 |
| 505 | 3 |
| 506 | 2 |
| 507 | 2 |
| 508 | 4 |

**Queries:**

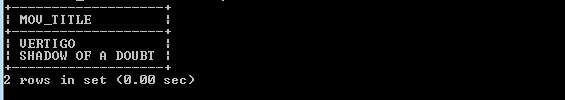
1. **List the titles of all movies directed by **

SELECT MOV\_TITLE FROM MOVIES WHERE DIR\_ID IN (SELECT DIR\_ID FROM



## OR

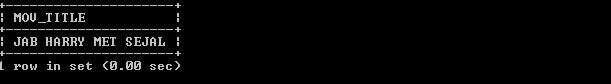
SELECT MOV\_TITLE FROM MOVIES M, DIRECTOR D WHERE M.DIR\_ID=D.DIR\_ID AND DIR\_NAME='HITCHCOCK';



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

SELECT MOV\_TITLE FROM MOVIES M,MOVIES\_CAST MV

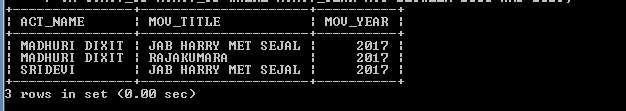
WHERE M.MOV\_ID=MV.MOV\_ID AND ACT\_ID IN(SELECT ACT\_ID FROM MOVIES\_CAST GROUP BY ACT\_ID HAVING COUNT(ACT\_ID)>1) GROUP BY MOV\_TITLE HAVING COUNT(\*)>1;



1. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

SELECT ACT\_NAME, MOV\_TITLE, MOV\_YEAR FROM ACTOR A JOIN MOVIE\_CAST C ON A.ACT\_ID=C.ACT\_ID INNER JOIN MOVIES M

ON C.MOV\_ID=M.MOV\_ID WHERE M.MOV\_YEAR NOT BETWEEN 2000 AND 2015;



1. **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,MAX(REV\_STARS) FROM MOVIES M ,RATING R WHERE

M.MOV\_ID=R.MOV\_ID GROUP BY MOV\_TITLE HAVING MAX(REV\_STARS)>0 ORDER BY MOV\_TITLE;



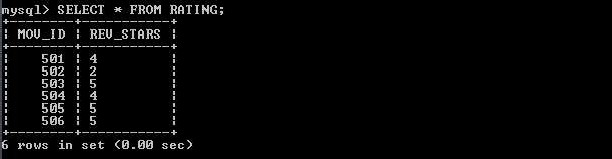
1. **** **5**

UPDATE RATING SET REV\_STARS=5 WHERE MOV\_ID IN(SELECT MOV\_ID FROM MOVIES WHERE DIR\_ID IN(SELECT DIR\_ID FROM DIRECTOR

WHERE DIR\_NAME='STEVEN SPIELBERG'));

## OR

UPDATE RATING R, MOVIES M, DIRECTOR D SET REV\_STARS=5 WHERE R.MOV\_ID=M.MOV\_ID AND M.DIR\_ID=D.DIR\_ID AND DIR\_NAME='STEVEN SPIELBERG';



**Program Outcomes:**

**The students are able to**

Create, Update and query on the database.

Demonstrate the working of different concepts of DBMS

Implement, analyze and evaluate the project developed for an application.

1. **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 ** **section.**
  2. **Compute the total number of male and female students in each semester and in each section.**
  3. **** **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 =**

**Give these details only for 8th semester A, B, and C section students.**

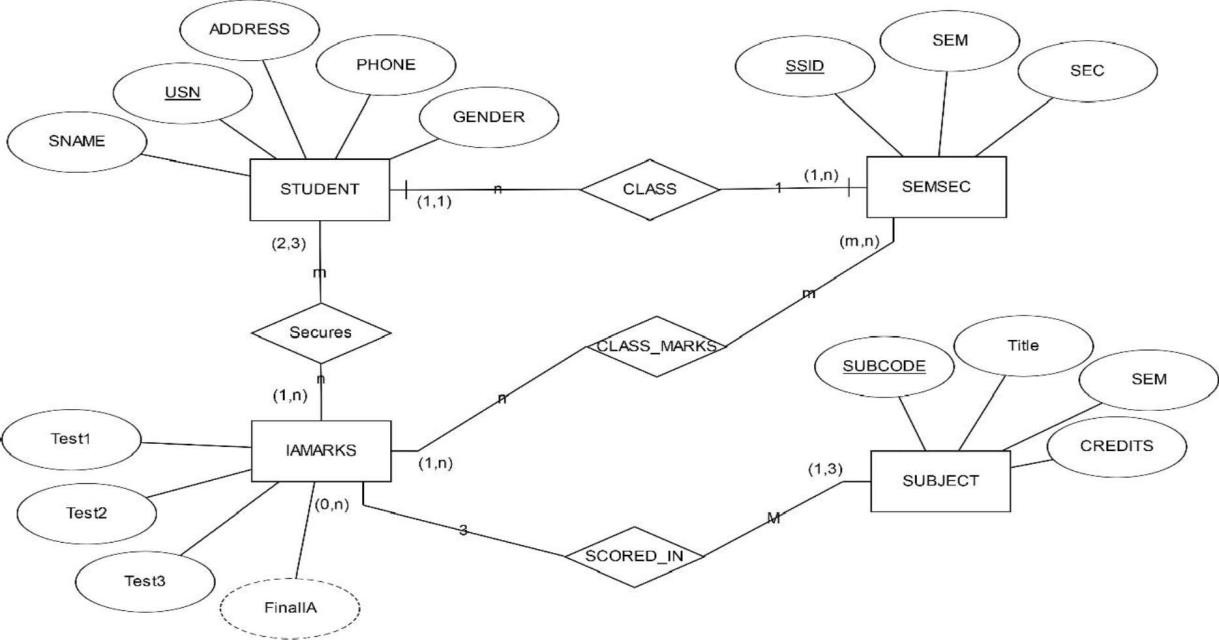
**Program Objectives:**

**This course will enable students to**

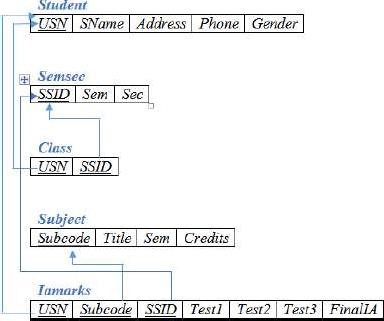
Foundation knowledge in database concepts, technology and practice to groomstudents into well-informed database application developers.

Strong practice in SQL programming through a variety of database problems. Develop database applications using front-end tools and back-end DBMS.

**Solution:**

**Entity - Relationship Diagram**

**Schema Diagram**

****

**Table Creation**

CREATE TABLE STUDENT (

USN VARCHAR (10) PRIMARY KEY, SNAME VARCHAR (25),

ADDRESS VARCHAR (25),

PHONE BIGINT (10),

GENDER CHAR (1));

CREATE TABLE SEMSEC (

SSID VARCHAR (5) PRIMARY KEY, SEM INT (5),

SEC CHAR (1));

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));

CREATE TABLE SUBJECT( SUBCODE VARCHAR(10) PRIMARY KEY,

TITLE VARCHAR(20), SEM INT,

CREDITS INT);

CREATE TABLE IAMARKS ( USN VARCHAR (10),

SUBCODE VARCHAR (8),

SSID VARCHAR (5),

TEST1 INT (2),

TEST2 INT (2),

TEST3 INT (2),

FINALIA INT (2),

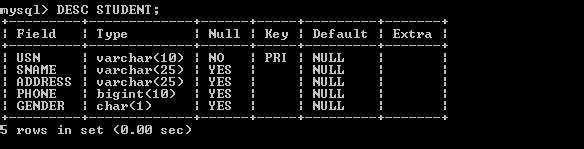
PRIMARY KEY (USN, SUBCODE, SSID),

FOREIGN KEY (USN) REFERENCES STUDENT (USN),

FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE), FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));

**Table Descriptions**

DESC STUDENT;



DESC SEMSEC;



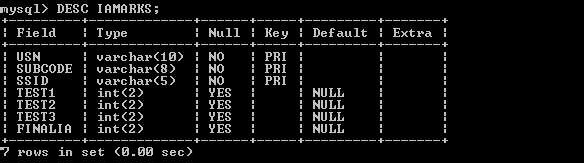
DESC CLASS;



DESC SUBJECT;



DESC IAMARKS;



**Insertion of values to tables**

INSERT INTO STUDENT VALUES ('4AD13CS020','AKSHAY','BELAGAVI', 8877881122,'M'); INSERT INTO STUDENT VALUES ('4AD13CS062','SANDHYA','BENGALURU', 7722829912,'F');

INSERT INTO STUDENT VALUES ('4AD13CS091','TEESHA','BENGALURU', 7712312312,'F'); INSERT INTO STUDENT VALUES ('4AD13CS066','SUPRIYA','MANGALURU', 8877881122,'F');

INSERT INTO STUDENT VALUES ('4AD14CS010','ABHAY','BENGALURU', 9900211201,'M'); INSERT INTO STUDENT VALUES ('4AD14CS032','BHASKAR','BENGALURU', 9923211099,'M');

INSERT INTO STUDENT VALUES ('4AD14CS025','ASMI','BENGALURU', 7894737377,'F'); INSERT INTO STUDENT VALUES ('4AD15CS011','AJAY','TUMKUR', 9845091341,'M'); INSERT INTO STUDENT VALUES ('4AD15CS029','CHITRA','DAVANGERE', 7696772121,'F'); INSERT INTO STUDENT VALUES ('4AD15CS045','JEEVA','BELLARY', 9944850121,'M'); INSERT INTO STUDENT VALUES ('4AD15CS091','SANTOSH','MANGALURU', 8812332201,'M')

INSERT INTO STUDENT VALUES ('4AD16CS045','ISMAIL','KABURGI', 9900232201,'M'); INSERT INTO STUDENT VALUES ('4AD16CS088','SAMEERA','SHIMOGA', 9905542212,'F'); INSERT INTO STUDENT VALUES ('4AD16CS122','VINAYAKA','CHIKAMAGALUR', 8800880011,'M');

INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A'); INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B'); INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C'); INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A'); INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B'); INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C'); INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A'); INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B'); INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C'); INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A'); INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B'); INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C'); INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A'); INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B'); INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C'); INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A'); INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B'); INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C'); INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A'); INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B'); INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C'); INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A'); INSERT INTO SEMSEC VALUES ('CSE1B', 1,'B'); INSERT INTO SEMSEC VALUES ('CSE1C', 1,'C');

INSERT INTO CLASS VALUES ('4AD13CS020','CSE8A'); INSERT INTO CLASS VALUES ('4AD13CS062','CSE8A'); INSERT INTO CLASS VALUES ('4AD13CS066','CSE8B'); INSERT INTO CLASS VALUES ('4AD13CS091','CSE8C'); INSERT INTO CLASS VALUES ('4AD14CS010','CSE7A'); INSERT INTO CLASS VALUES ('4AD14CS025','CSE7A'); INSERT INTO CLASS VALUES ('4AD14CS032','CSE7A'); INSERT INTO CLASS VALUES ('4AD15CS011','CSE4A'); INSERT INTO CLASS VALUES ('4AD15CS029','CSE4A'); INSERT INTO CLASS VALUES ('4AD15CS045','CSE4B'); INSERT INTO CLASS VALUES ('4AD15CS091','CSE4C'); INSERT INTO CLASS VALUES ('4AD16CS045','CSE3A'); INSERT INTO CLASS VALUES ('4AD16CS088','CSE3B'); INSERT INTO CLASS VALUES ('4AD16CS122','CSE3C');

INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4); INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4); INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4); INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4); INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4); INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4); INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4); INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4); I INSERT INTO SUBJECT VALUES ('10CS75','JAVA', 7, 4); INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4); INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4); INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4); INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4); INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4); INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3); INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3); INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4); INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4); INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4); INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4); INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3); INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3); INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4); INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4); INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4); INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4); INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3); INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);

INSERT INTO IAMARKS VALUES ('4AD13CS091','10CS81','CSE8C', 15, 16, 18,0);

INSERT INTO IAMARKS VALUES ('4AD13CS091','10CS82','CSE8C', 12, 19, 14,0);

INSERT INTO IAMARKS VALUES ('4AD13CS091','10CS83','CSE8C', 19, 15, 20,0);

INSERT INTO IAMARKS VALUES ('4AD13CS091','10CS84','CSE8C', 20, 16, 19,0);

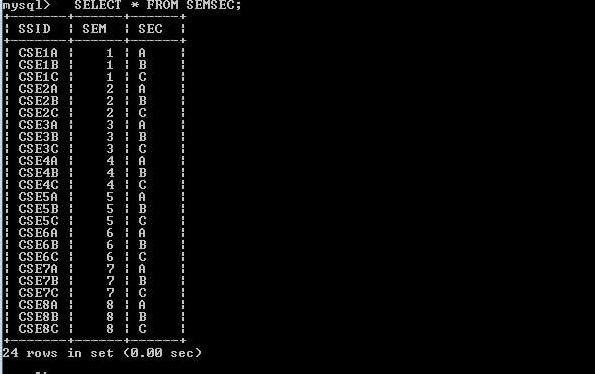
INSERT INTO IAMARKS VALUES ('4AD13CS091','10CS85','CSE8C', 15, 15, 12,0);

SELECT \* FROM STUDENT;



S LECT \* FROM

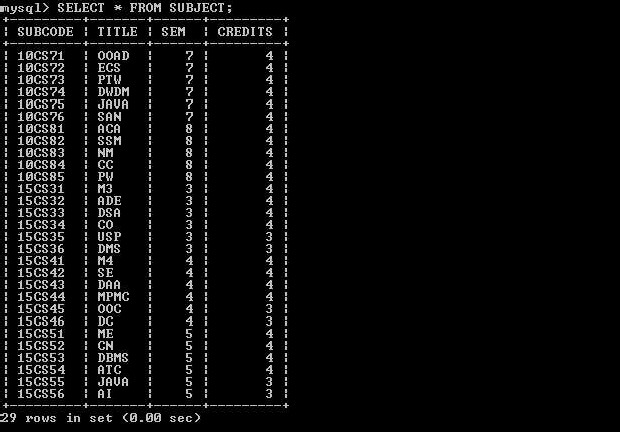
SELECT \* FROM SEMSEC;



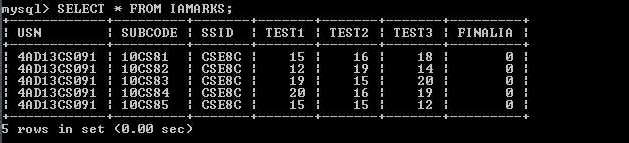
SELECT \* FROM CLASS;



SELECT \* FROM SUBJECT;



SELECT \* FROM IAMARKS;



**Queries:**

1. **** **section.**

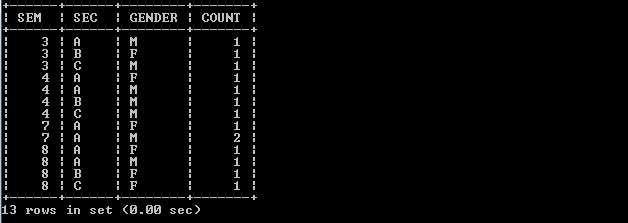
SELECT S.\*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C WHERE



1. **Compute the total number of male and female students in each semester and ineach section.** SELECT SS.SEM, SS.SEC, S.GENDER, COUNT (S.GENDER) AS COUNT FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND SS.SSID = C.SSID

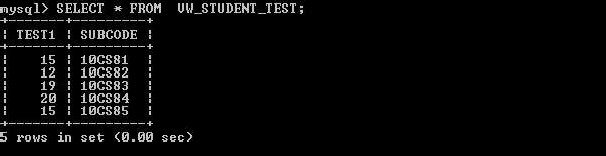
GROUP BY SS.SEM, SS.SEC, S.GENDER ORDER BY SEM;



1. **** **subjects.**

CREATE VIEW VW\_STUDENT\_TEST AS SELECT TEST1,SUBCODE FROM IAMARKS WHERE USN= 4AD13CS091';

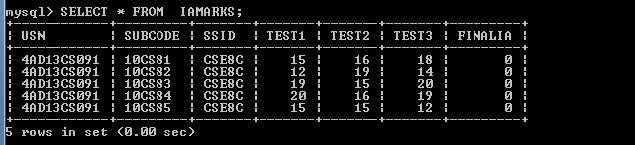
SELECT \* FROM VW\_STUDENT\_TEST



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

UPDATE IAMARKS

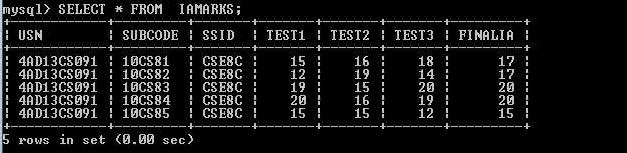
SET FINALIA=GREATEST(TEST1+TEST2,TEST2+TEST3,TEST1+TEST3)/2;

**Note:** Before execution above SQL statement, IAMARKS table contents are: SELECT \* FROM IAMARKS;

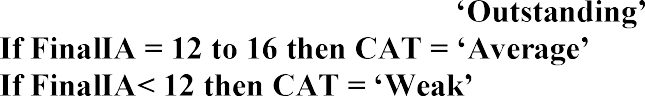
UPDATE IAMARKS

SET FINALIA=GREATEST(TEST1+TEST2,TEST2+TEST3,TEST1+TEST3)/2;

After executing above SQL statement, IAMARKS table contents are:



1. **Categorize students based on the following criterion:**



**If FinalIA = 17 to 20 then CAT =**

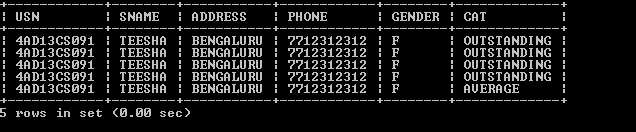
**Give these details only for 8th semester A, B, and C section students.**

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 = 8;



**Program Outcomes:**

**The students are able to**

Create, Update and query on the database.

Demonstrate the working of different concepts of DBMS

Implement, analyze and evaluate the project developed for an application.

1. **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, Hours)**

**Write SQL queries to**

1. **Make a list of all project numbers for projects that involve an employee whose last**

**project.**

1. **Show the resulting salaries if every employee ** **percent raise.**
2. ****



**the maximum salary, the minimum salary, and the average salary in this department**

1. **Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).**
2. **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.**

**Program Objectives:**

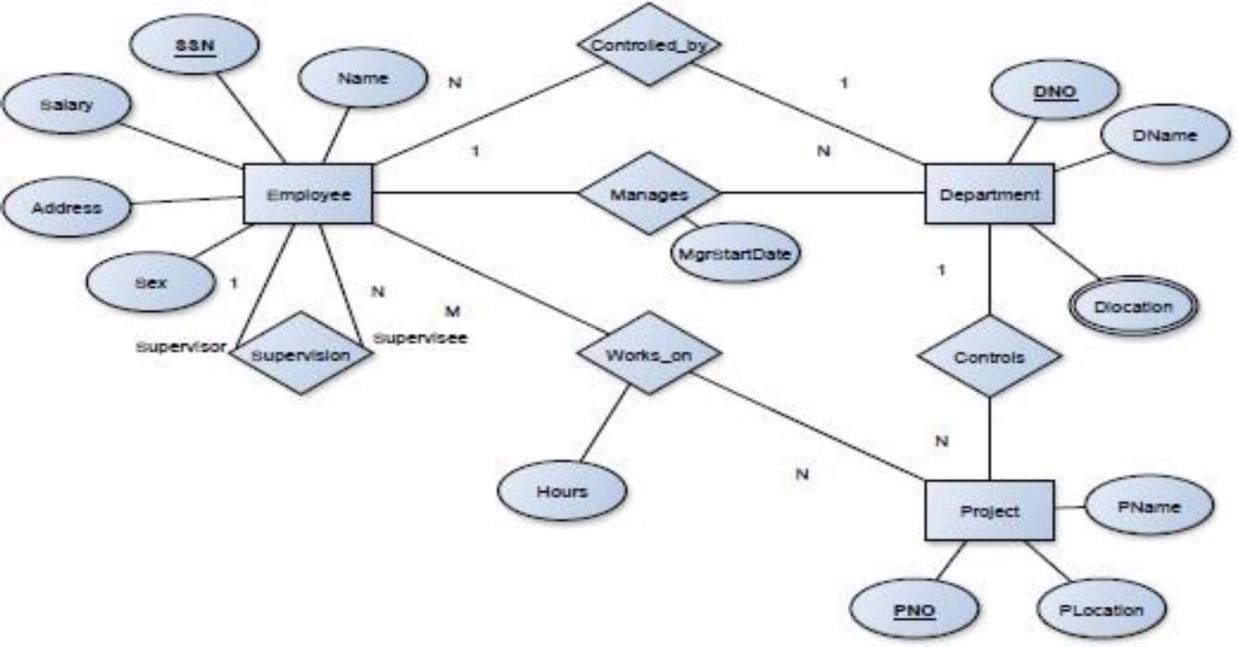
**This course will enable students to**

Foundation knowledge in database concepts, technology and practice to groomstudents

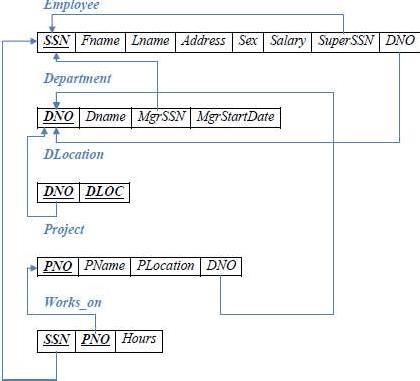
into well-informed database application developers.

Strong practice in SQL programming through a variety of database problems. Develop database applications using front-end tools and back-end DBMS.

**Solution:**

**Entity-Relationship Diagram**

**Schema Diagram**

****

**Table Creation**

CREATE TABLE DEPARTMENT ( DNO VARCHAR (20) PRIMARY KEY, DNAME VARCHAR (20), MGRSTARTDATE DATE,

MGRSSN VARCHAR (20));

CREATE TABLE EMPLOYEE (

SSN VARCHAR (20) PRIMARY KEY, FNAME VARCHAR (20),

LNAME VARCHAR (20),

ADDRESS VARCHAR (100),

SEX CHAR (1),

SALARY INT (10),

SUPERSSN VARCHAR (20),

DNO VARCHAR (20),

FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN), FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

**NOTE:** Once DEPARTMENT and EMPLOYEE tables are created we must alter department table to add foreign constraint MGRSSN using sql command

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

CREATE TABLE DLOCATION ( DLOC VARCHAR (20),

DNO VARCHAR (20), PRIMARY KEY (DNO, DLOC),

FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

CREATE TABLE PROJECT ( PNO INT (10) PRIMARY KEY, PNAME VARCHAR (20),

PLOCATION VARCHAR (20),

DNO VARCHAR (20),

FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

CREATE TABLE WORKS\_ON ( HOURS INT (4),

SSN VARCHAR (20),

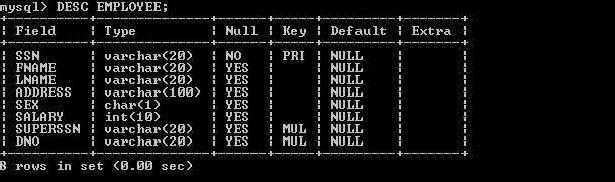
PNO INT (10),

PRIMARY KEY (SSN, PNO),

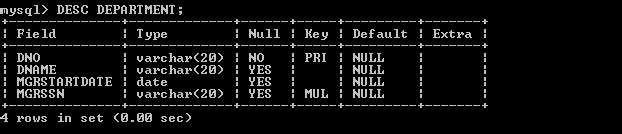
FOREIGN KEY (SSN) REFERENCES EMPLOYEE (SSN), FOREIGN KEY (PNO) REFERENCES PROJECT (PNO));

**Table Descriptions**

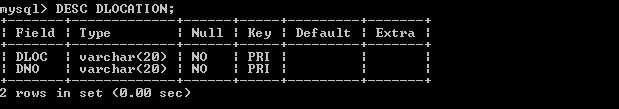
DESC EMPLOYEE;



DESC DEPARTMENT;



DESC DLOCATION;



DESC PROJECT;

DESC PROJECT;



DESC WORKS\_ON;



**Insertion of values to tables**

INSERT INTO EMPLOYEE VALUES ('ATMEECE01','JOHN','SCOTT','BANGALORE','M', 450000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE01','JAMES','SMITH','BANGALORE','M', 500000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE02','HEARN','BAKER','BANGALORE','M', 700000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE03','EDWARD','SCOTT','MYSORE','M', 500000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE04','PAVAN','HEGDE','MANGALORE','M', 650000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE05','GIRISH','MALYA','MYSORE','M', 450000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMECSE06','NEHA','SN','BANGALORE','F', 800000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMEACC01','AHANA','K','MANGALORE','F', 350000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMEACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000,NULL,NULL); INSERT INTO EMPLOYEE VALUES ('ATMEISE01','VEENA','M','MYSORE','F', 600000,NULL,NULL);

INSERT INTO EMPLOYEE VALUES ('ATMEIT01','NAGESH','HR','BANGALORE','M', 500000,NULL,NULL);

INSERT INTO DEPARTMENT VALUES ('1','ACCOUNTS','2001-01-01','ATMEACC02'); INSERT INTO DEPARTMENT VALUES ('2','IT','2016-08-01','ATMEIT01');

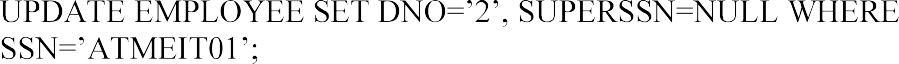
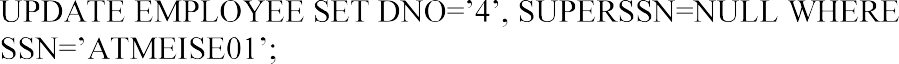
INSERT INTO DEPARTMENT VALUES ('3','ECE','2008-6-01','ATMEECE01'); INSERT INTO DEPARTMENT VALUES ('4','ISE','2015-06-01','ATMEISE01'); INSERT INTO DEPARTMENT VALUES ('5','CSE','2002-06-01','ATMECSE05');

**Note: update entries of employee table to fill missing fields SUPERSSN and DNO**

UPDATE EMPLOYEE SET SUPE



UPDATE EMPLOYEE 



INSERT INTO DLOCATION VALUES ('BANGALORE', '1'); INSERT INTO DLOCATION VALUES ('BANGALORE', '2'); INSERT INTO DLOCATION VALUES ('BANGALORE', '3'); INSERT INTO DLOCATION VALUES ('MANGALORE', '4'); INSERT INTO DLOCATION VALUES ('MANGALORE', '5');

INSERT INTO PROJECT VALUES (100,'IOT','BANGALORE','5'); INSERT INTO PROJECT VALUES (101,'CLOUD','BANGALORE','5'); INSERT INTO PROJECT VALUES (102,'BIGDATA','BANGALORE','5'); INSERT INTO PROJECT VALUES (103,'SENSORS','BANGALORE','3');

INSERT INTO PROJECT VALUES (104,'BANK MANAGEMENT','BANGALORE','1'); INSERT INTO PROJECT VALUES (105,'SALARY MANAGEMENT','BANGALORE','1');

INSERT INTO PROJECT VALUES (106,'OPENSTACK','BANGALORE','4'); INSERT INTO PROJECT VALUES (107,'SMART CITY','BANGALORE','2');

INSERT INTO WORKS\_ON VALUES (4, 'ATMECSE01', 100); INSERT INTO WORKS\_ON VALUES (6, 'ATMECSE01', 101); INSERT INTO WORKS\_ON VALUES (8, 'ATMECSE01', 102); INSERT INTO WORKS\_ON VALUES (10, 'ATMECSE02', 100);

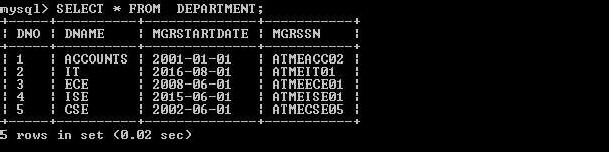


INSERT INTO WORKS\_ON VALUES (4, 'ATMECSE05', 101); INSERT INTO WORKS\_ON VALUES (5, 'ATMECSE06', 102); INSERT INTO WORKS\_ON VALUES (6, 'ATMECSE03', 102); INSERT INTO WORKS\_ON VALUES (7, 'ATMEECE01', 103); INSERT INTO WORKS\_ON VALUES (5, 'ATMEACC01', 104); INSERT INTO WORKS\_ON VALUES (6, 'ATMEACC02', 105); INSERT INTO WORKS\_ON VALUES (4, 'ATMEISE01', 106); INSERT INTO WORKS\_ON VALUES (10, 'ATMEIT01', 107);

SELECT \* FROM EMPLOYEE;



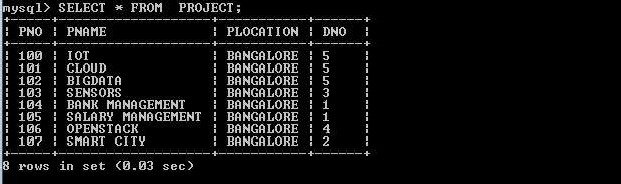
SELECT \* FROM DEPARTMENT ;



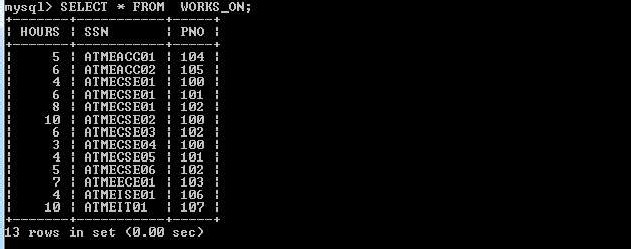
SELECT \* FROM DLOCATION ;



SELECT \* FROM PROJECT ;



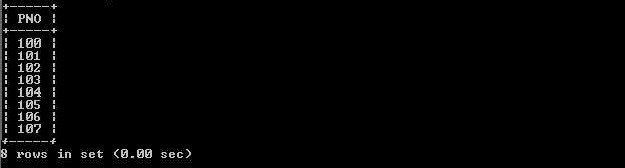
SELECT \* FROM WORKS\_ON



**Queries:**

1. **Make a list of all project numbers for projects that involve an employee whose last name**

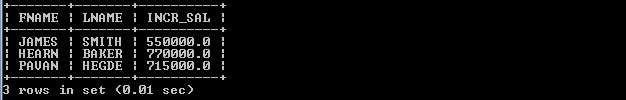
(SELECT DISTINCT P.PNO FROM PROJECT P, DEPARTMENT D, EMPLOYEE E UNION

(SELECT DISTINCT P1.PNO FROM PROJECT P1, WORKS\_ON W, EMPLOYEE E1 WHERE

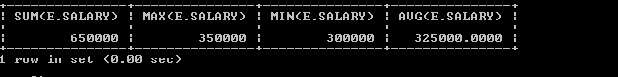


1. ****

**percent raise.**

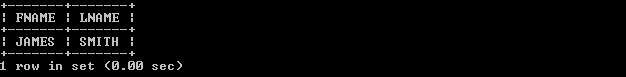
SELECT E.FNAME, E.LNAME, 1.1\*E.SALARY AS INCR\_SAL FROM EMPLOYEE E, WORKS\_ON W, PROJECT P WHERE E.SSN=W.SSN AND W.PNO=P.PNO AND

1. ****

**maximum salary, the minimum salary, and the average salary in this department** SELECT SUM (E.SALARY), MAX (E.SALARY), MIN (E.SALARY), AVG (E.SALARY) FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DNO AND

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

SELECT E.FNAME,E.LNAME FROM EMPLOYEE E WHERE NOT EXISTS (SELECT PNO FROM PROJECT P WHERE DNO=5 AND PNO NOT IN (SELECT PNO FROM WORKS\_ON W WHERE E.SSN=SSN));



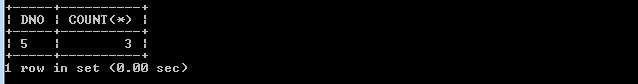
1. **For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6, 00,000.** SELECT D.DNO, COUNT (\*)

FROM DEPARTMENT D, EMPLOYEE E WHERE D.DNO=E.DNO

AND E.SALARY>600000

AND D.DNO IN (SELECT E1.DNO FROM EMPLOYEE E1

GROUP BY E1.DNO HAVING COUNT (\*)>5) GROUP BY D.DNO;



**Program Outcomes:**

**The students are able to**

Create, Update and query on the database.

Demonstrate the working of different concepts of DBMS

Implement, analyze and evaluate the project developed for an application.

# PART B

# Develop PL/SQL programs using PROCEDURE

**//Create a Procedure to Add a New Employee** [ORACLE@LOCALHOST] $ sql hr/oracle@localhost:1521/freepdb1 SQL> set serveroutput on

**//Create a table called employees:**

CREATE TABLE employees( emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

Salary NUMBER

);

**//Create a Procedure**

CREATE OR REPLACE PROCEDURE add\_employee( p\_emp\_id IN employees.emp\_id%TYPE,

p\_emp\_name IN employees.emp\_name%TYPE, p\_salary IN employees.salary%TYPE

) IS

BEGIN

INSERT INTO employees(emp\_id, emp\_name, salary) VALUES(p\_emp\_id, p\_emp\_name, p\_salary);

DBMS\_OUTPUT.PUT\_LINE(‘Employee added successfully.’);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE(‘Error: Employee ID already exists.’);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(‘Error: ’ || SQLERRM);

END;

/

**//Execute the Procedure**

BEGIN

add\_employee(101, ‘John Doe’, 500000);

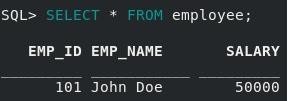
End;

/

**// View the table**

SELECT \* FROM employees;

**OUTPUT:**

****

# Develop PL/SQL program using FUNCTIONS

**//Illustrate function returns the total number of customers in the customers table.**

[ORACLE@LOCALHOST] $ sql hr/oracle@localhost:1521/freepdb1

SQL> set serveroutput on

**//Create a table called customers**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY, name VARCHAR(50),

age NUMBER,

address VARCHAR(50)

);

**//Insert Values into table customer**

INSERT INTO customers (customer\_id, name, age, address) VALUES (101, 'John', 30, 'Bengaluru'),

(102, 'Mark', 35, 'Mumbai'),

(103, 'Arun', 28, 'Pune'),

(104, 'Ram', 32, 'Mysuru');

**//Create a function**

CREATE OR REPLACE FUNCTION totalCustomers RETURN number

IS

total number(2) := 0; BEGIN

SELECT count(\*) into total FROM customers;

RETURN total;

END;

/

**//Calling a Function**

DECLARE

c number(2); BEGIN

c := totalCustomers();

dbms\_output.put\_line('Total no. of Customers: ' || c);

END;

/

## OUTPUT:

Total no. of Customers: 4

# Develop PL/SQL program using CURSORS

**// PL/SQL program that lists all employees with a salary greater than 50000.**

[ORACLE@LOCALHOST] $ sql hr/oracle@localhost:1521/freepdb1

SQL> set serveroutput on

**//Create a table called employees:**

CREATE TABLE EMPLOYEES( EMP\_ID NUMBER PRIMARY KEY, EMP\_NAME VARCHAR2(100), SALARY NUMBER

);

**//Insert Values into table employees**

INSERT INTO EMPLOYEES (EMP\_ID, EMP\_NAME, SALARY) VALUES (101, 'John', 30000,),

(102, 'Mark', 60000),

(103, 'Arun', 55000),

(104, 'Ram', 58000);

DECLARE

//Define variables to hold the data fetched from the cursor v\_emp\_id EMPLOYEES.EMP\_ID%TYPE; v\_emp\_name EMPLOYEES.EMP\_NAME%TYPE; v\_salary EMPLOYEES.SALARY%TYPE;

//Declare the cursor CURSOR emp\_cursor IS

SELECT EMP\_ID, EMP\_NAME, SALARY FROM EMPLOYEES

WHERE SALARY > 50000;

BEGIN

//Open the cursor OPEN emp\_cursor;

LOOP

//Fetch data into variables

FETCH emp\_cursor INTO v\_emp\_id, v\_emp\_name, v\_salary;

//Exit the loop when no more rows are found EXIT WHEN emp\_cursor%NOTFOUND;

//Display the data (in real apps, you might process or store it) DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_emp\_id || ', Name: ' || v\_emp\_name || ', Salary: ' || v\_salary);

END LOOP;

//Close the cursor CLOSE emp\_cursor;

END;

/

## OUTPUT:

**EMP\_ID EMP\_NAME SALARY**

102 Mark 60000

103 Arun 55000

104 Ram 58000

1. **Develop PL/SQL program using TRIGGERS**

//We have a table named employees, and we want to automatically log changes whenever an employee table is inserted, deleted and employee's salary is updated.

[ORACLE@LOCALHOST] $ sql hr/oracle@localhost:1521/freepdb1 SQL> set serveroutput on

**//Main table**

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100), salary NUMBER

);

**//Audit log table**

CREATE TABLE salary\_audit (

audit\_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

emp\_id NUMBER, old\_salary NUMBER, new\_salary NUMBER, changed\_on DATE,

changed\_by VARCHAR2(100), operation\_type VARCHAR2(10)

);

**//Trigger for INSERT**

CREATE OR REPLACE TRIGGER trg\_salary\_insert AFTER INSERT ON employees

FOR EACH ROW BEGIN

INSERT INTO salary\_audit (emp\_id, old\_salary, new\_salary, changed\_on, changed\_by, operation\_type) VALUES (:NEW.emp\_id, NULL, :NEW.salary, SYSDATE, USER, 'INSERT');

END;

/

**//Trigger for UPDATE**

CREATE OR REPLACE TRIGGER trg\_salary\_update BEFORE UPDATE OF salary ON employees

FOR EACH ROW BEGIN

INSERT INTO salary\_audit (emp\_id, old\_salary, new\_salary, changed\_on, changed\_by, operation\_type) VALUES (:OLD.emp\_id, :OLD.salary, :NEW.salary, SYSDATE, USER, ‘UPDATE’);

END;

/

**//Trigger for DELETE**

CREATE OR REPLACE TRIGGER trg\_salary\_delete BEFORE DELETE ON employees

FOR EACH ROW BEGIN

INSERT INTO salary\_audit (emp\_id, old\_salary, new\_salary, changed\_on, changed\_by, operation\_type) VALUES (:OLD.emp\_id, :OLD.salary, NULL, SYSDATE, USER, 'DELETE');

END;

/

**//INSERT operation**

INSERT INTO employees (emp\_id, emp\_name, salary) VALUES (1, 'Bob', 60000);

**//UPDATE operation** UPDATE employees SET salary = 65000 WHERE emp\_id = 1;

**//DELETE operation**

DELETE FROM employees WHERE emp\_id = 1;

**//View audit logs**

SELECT \* FROM salary\_audit ORDER BY audit\_id;

## OUTPUT:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **AUDIT\_ID** | **EMP\_ID** | **OLD\_SALARY** | **NEW\_SALARY** | **CHANGED\_ON** | **CHANGED\_BY** | **OPERATION\_TYPE** |
| 1 | 1 | NULL | 60000 | 22-JUN-2025 10:00 | HR | INSERT |
| 2 | 1 | 60000 | 65000 | 22-JUN-2025 10:01 | HR | UPDATE |
| 3 | 1 | 65000 | NULL | 22-JUN-2025 10:02 | HR | DELETE |

1. **Develop PL/SQL program using PACKAGES**

//We'll build a package for managing **employee salary operations**: inserting employees, updating salary, and getting salary info.

[ORACLE@LOCALHOST] $ sql hr/oracle@localhost:1521/freepdb1 SQL> set serveroutput on

**//Create the employees Table**

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100), salary NUMBER

);

**//Create the Package Specification**

CREATE OR REPLACE PACKAGE emp\_pkg IS

PROCEDURE add\_employee(p\_id NUMBER, p\_name VARCHAR2, p\_salary NUMBER); PROCEDURE update\_salary(p\_id NUMBER, p\_new\_salary NUMBER);

FUNCTION get\_salary(p\_id NUMBER) RETURN NUMBER;

END emp\_pkg;

/

**//Create the Package Body**

CREATE OR REPLACE PACKAGE BODY emp\_pkg IS

PROCEDURE add\_employee(p\_id NUMBER, p\_name VARCHAR2, p\_salary NUMBER) IS BEGIN

INSERT INTO employees (emp\_id, emp\_name, salary) VALUES (p\_id, p\_name, p\_salary);

END add\_employee;

PROCEDURE update\_salary(p\_id NUMBER, p\_new\_salary NUMBER) IS BEGIN

UPDATE employees

SET salary = p\_new\_salary WHERE emp\_id = p\_id;

END update\_salary;

FUNCTION get\_salary(p\_id NUMBER) RETURN NUMBER IS v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary FROM employees

WHERE emp\_id = p\_id; RETURN v\_salary;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN NULL;

END get\_salary;

END emp\_pkg;

/

**//Test the Package**

**//Add employees**

BEGIN

emp\_pkg.add\_employee(101, 'Alice', 50000);

emp\_pkg.add\_employee(102, 'Bob', 60000); END;

/

**//Update salary**

BEGIN

emp\_pkg.update\_salary(102, 65000); END;

/

**//Get salary**

DECLARE

v\_sal NUMBER; BEGIN

v\_sal := emp\_pkg.get\_salary(102); DBMS\_OUTPUT.PUT\_LINE('Bob's Salary: ' || v\_sal);

END;

/

## OUTPUT:

Bob's Salary: 65000