**17CSL58: DBMS LABORATORY WITH MINI PROJECT**

**Course 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.

**Database:** A Database is a collection of interrelated data and a Database Management Systemis a a software system that enables users to define, create and maintain the database and which provides controlled access to the database

**SQL:** It is structured query language, basically used to pass the query to retrieve andmanipulate the information from database. Depending upon the nature of query, SQL is divided into different components:

* **DDL**(Data Definition Language )
* **DML**(Data Manipulation Language )
* **DCL**(Data Control Language )

**DDL:** The Data Definition Language (DDL) is used to create the database (i.e. tables, keys,relationships etc), maintain the structure of the database and destroy databases and database objects.

**Eg.** Create, Drop, Alter, Describe, Truncate

1. **CREATE** statements: It is used to create the table.

**Syntax:**

CREATE TABLE table\_name(columnName1 datatype(size), columnName2 datatype(size),.........);

1. **DROP statements:** To destroy an existing database, table, index, or view. If a table isdropped all records held within it are lost and cannot be recovered.

**Syntax:**

DROP TABLE table\_name;

3. **ALTER statements:** To modify an existing database object.

* **Adding new columns: Syntax:**

Alter table table\_name Add(New\_columnName1 datatype(size), New\_columnName2 datatype(size),.........)

* **Dropping a columns from a table : Syntax:**

Alter table table\_name DROP column columnName:

* **Modifying Existing columns:**

**Syntax:**

Alter table table\_name Modify (columnName1 Newdatatype(Newsize));

1. **Describe statements:** To describe the structure (column and data types) of an existingdatabase, table, index, or view.

**Syntax:**

DESC table\_name;

1. **Truncate statements:** To destroy the data in an existing database, table, index, or view.If a table is truncated all records held within it are lost and cannot be recovered but the table structure is maintained.

**Syntax :**

TRUNCATE TABLE table\_name;

**Data Manipulation Language (DML):**

* A Data Manipulation Language enables programmers and users of the database to retrieve insert, delete and update data in a database. e.g. INSERT, UPDATE, DELETE, SELECT.

**INSERT**: INSERT statement adds one or more records to any single table in a relationaldatabase.

**Syntax:**

INSERT INTO tablename VALUES (expr1,expr2........);

**UPDATE:** UPDATE statement that changes the data of one or more records in a table. Eitherall the rows can be updated, or a subset may be chosen using a condition.

**Syntax:**

UPDATE table\_name SET column\_name = value [, column\_name = value ...] [WHERE condition]

**DELETE:** DELETE statement removes one or more records from a table. A subset may bedefined for deletion using a condition, otherwise all records are removed.

**Syntax:**

DELETE FROM tablename WHERE condition:

**SELECT:** SELECT statement returns a result set of records from one or more tables.

The select statement has optional clauses:

* WHERE specifies which rows to retrieve
* GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group having group.
* HAVING selects among the groups defined by the GROUP BY clause.
* ORDER BY specifies an order in which to return the rows.

**Syntax:**

SELECT<attribute list> FROM<table list> WHERE<condition> Where

* Attribute list is a list of attribute name whose values to be retrieved by the query.
* Table list is a list of table name required to process query.
* Condition is a Boolean expression that identifies the tuples to be retrieved by query.

**Data Constraints** are the business Rules which are enforced on the data being stored in a tableare called Constraints.

Types of Data Constraints

1. I/O Constraint This type of constraint determines the speed at which data can be inserted

or extracted from an Oracle table. I/O Constraints is divided into two different types

●The Primary Key Constraint

●The Foreign Key Constraint

1. Business rule Constraint This type of constraint is applied to data prior the data being Inserted into table columns.

●Column level

●Table level

|  |  |  |  |
| --- | --- | --- | --- |
| **The PRIMARY KEY defined at column level** | | |  |
| **Syntax:** |  |  |
| CREATETABLEtablename  (Columnname1DATATYPE CONSTRAINT <constraintname1> PRIMARY KEY, Columnname2 DATATYPE, columnname3 DATATYPE,.....); |  |  |
|  |  |  |
|  |  |  |

**The PRIMARY KEY defined at table level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, **PRIMARY KEY (columnname1, columnname2));**

**The FOREIGN KEY defined at column level**

**Syntax**

CREATE TABLE tablename (Columnname1

tablename[(columnname)] [ON DELETE CASCADE],

columnname3 DATATYPE ,.....);

DATATYPE columnname2 REFERENCES DATATYPE ,

The table in which FOREIGN KEY is defined is called FOREIGN TABLE or DETAIL TABLE. The table in which PRIMARY KEY is defined and referenced by FOREIGN KEY is called PRIMARY TABLE or MASTER TABLE.

**ON DELETE CASCADE** is set then DELETE operation in master table will trigger theDELETE operation for corresponding records in the detail table.

**The FOREIGN KEY defined at table level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, PRIMARY KEY (columnname1, columnname2), FOREIGN KEY (columnname2) REFERENCES tablename2;

A CONSTRAINT can be given User Defined Name, the syntax is:

CONSTRAINT < constraint name><constraint definition>

**The CHECK Constraint defined at column level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE CHECK (logical expression), columnname2 DATATYPE, columnname3 DATATYPE,...);

**The CHECK Constraint defined at table level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, CHECK (logical expression1), CHECK (logical expression2));

**The UNIQUE Constraint defined at the column level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE UNIQUE, columnname2 DATATYPE UNIQUE, columnname3 DATATYPE ...);

**The UNIQUE Constraint defined at the the table level**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, UNIQUE(columnname1));

**NOT NULL constraint defined at column level :**

**Syntax:**

CREATE TABLE tablename (Columnname1 DATATYPE NOT NULL, columnname2 DATATYPE NOT NULL, columnname3 DATATYPE,...);

**Note:**

The NOT NULL constraint can only be applied at column level.

**ER- Diagram:** It is an Entity **–**Relationship diagram which is used to represent the relationshipbetween different entities. An entity is an object in the real world which is distinguishable from other objects. The overall logical structure of a database can be expressed graphically by an ER diagram, which is built up from following components.

* Rectangles: represent entity sets.
* Ellipses: represent attributes.
* Diamonds: represent relationships among entity sets.
* Lines: link attribute to entity sets and entity sets to relationships.

**Mapping Cardinalities:** It expresses the number of entities to which another entity can beassociated via a relationship set. For a binary relationship set R between entity sets A and B. The Mapping Cardinalities must be one of the following.

* One to one
* One to many
* Many to one
* Many to many

**LAB EXPERIMENTS**

*PART A: SQL PROGRAMMING*

# Consider the following schema for a Library Database:

**BOOK (*Book\_id, Title, Publisher\_Name, Pub\_Year*) BOOK\_AUTHORS (Book\_id, Author\_*Name*) PUBLISHER (*Name, Address, Phone*) BOOK\_COPIES (*Book\_id, Branch\_id, No-of\_Copies*)**

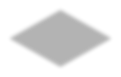
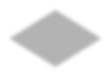
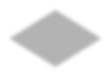
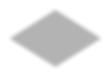
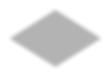
**BOOK\_LENDING (*Book\_id, Branch\_id, Card\_No, Date\_Out, Due\_Date*) LIBRARY\_BRANCH (*Branch\_id, Branch\_Name, Address*)**

# Write SQL queries to

* 1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**
  2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 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.**

**Solution:**

**Entity-Relationship Diagram**



Book

Book\_Authors

Publisher

**Author\_Name**

**Book\_id**

Title

Pub\_Year M

N

written-by

N

Published-by

Has

N

No\_of\_copies

**Branch\_id**

**Publisher\_Name**

1

M

In

Branch\_Name

Address

Address

Date\_out

N

Book\_Lending

Phone

**Card\_No**

Due\_date

N

Card

|  |  |
| --- | --- |
| Book\_Copies | M |
|  |

|  |  |
| --- | --- |
| N | Library\_Branch |
|  |

**Schema Diagram**

**Table Creation:**

**PUBLISHER**

SQL> **CREATE TABLE** PUBLISHER(

NAME VARCHAR(18) **PRIMARY KEY**,

ADDRESS VARCHAR(10),

PHONE VARCHAR(10));

Table created.

**BOOK**

SQL> **CREATE TABLE** BOOK(

BOOK\_ID INTEGER **PRIMARY KEY**,

TITLE VARCHAR(20),

PUBLISHER\_NAME VARCHAR(20)

PUB\_YEAR NUMBER(4),

**FOREIGN KEY**(PUBLISHER\_NAME) **REFERENCES** PUBLISHER(NAME)**ON DELETE** **CASADE**

);

Table created.

**BOOK\_AUTHORS**

SQL> **CREATE TABLE** BOOK\_AUTHORS(

BOOK\_ID INTEGER,

AUTHOR\_NAME VARCHAR(20),

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

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

Table created.

**LIBRARY\_BRANCH**

SQL> **CREATE TABLE** LIBRARY\_BRANCH(

BRANCH\_ID INTEGER **PRIMARY KEY**,

BRANCH\_NAME VARCHAR(18),

ADDRESS VARCHAR(15));

Table created.

**BOOK\_COPIES**

SQL> **CREATE TABLE** BOOK\_COPIES(

BOOK\_ID INTEGER,

BRANCH\_ID INTEGER,

NO\_OF\_COPIES INTEGER,

**FOREIGN KEY**(BOOK\_ID) **REFERENCES** BOOK(BOOK\_ID) **ON DELETE CASCADE**,

**FOREIGN KEY**(BRANCH\_ID) **REFERENCES** LIBRARY\_BRANCH(BRANCH\_ID) **ON DELETE CASCADE**,

**PRIMARY KEY**(BOOK\_ID,BRANCH\_ID));

Table created.

**BOOK\_LENDING**

SQL> **CREATE TABLE** BOOK\_LENDING(

BOOK\_ID INTEGER,

BRANCH\_ID INTEGER,

CARD\_NO INTEGER,

DATE\_OUT DATE,

DUE\_DATE DATE,

**PRIMARY KEY**(BOOK\_ID,BRANCH\_ID,CARD\_NO)**,**

**FOREIGN KEY**(BOOK\_ID) **REFERENCES** BOOK(BOOK\_ID) **ON DELETE CASCADE**,

**FOREIGN KEY**(BRANCH\_ID) **REFERENCES** LIBRARY\_BRANCH(BRANCH\_ID) **ON DELETE CASCADE**,

);Table created.

**Values for tables:**

**PUBLISHER**

SQL>INSERT INTO PUBLISHER VALUES('PEARSON','BANGALORE','9875462530'); SQL> INSERT INTO PUBLISHER VALUES('MCGRAW','NEWDELHI','7845691234'); SQL> INSERT INTO PUBLISHER VALUES('SAPNA','BANGALORE','7845963210');

**BOOK**

SQL> INSERT INTO BOOK VALUES(1111,'SE','PEARSON',2005);

SQL> INSERT INTO BOOK VALUES(2222,'DBMS','MCGRAW',2004);

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

**BOOK\_AUTHORS**

SQL> INSERT INTO BOOK\_AUTHORS VALUES(1111,'SOMMERVILLE'); SQL> INSERT INTO BOOK\_AUTHORS VALUES(2222,'NAVATHE'); SQL> INSERT INTO BOOK\_AUTHORS VALUES(3333,'HENRY GRAY'); SQL> INSERT INTO BOOK\_AUTHORS VALUES(4444,'THOMAS');

**LIBRARY\_BRANCH**

SQL> INSERT INTO LIBRARY\_BRANCH VALUES(11,'CENTRAL TECHNICAL','MG ROAD');

SQL> INSERT INTO LIBRARY\_BRANCH VALUES(22,'MEDICAL','BH ROAD');

SQL> INSERT INTO LIBRARY\_BRANCH VALUES(33,'CHILDREN','SS PURAM');

SQL> INSERT INTO LIBRARY\_BRANCH VALUES(44,'SECRETARIAT','SIRAGATE');

SQL> INSERT INTO LIBRARY\_BRANCH VALUES(55,'GENERAL','JAYANAGAR');

**BOOK\_COPIES**

SQL> INSERT INTO BOOK\_COPIES VALUES(1111,11,5);

SQL> INSERT INTO BOOK\_COPIES VALUES(3333,22,6);

SQL> INSERT INTO BOOK\_COPIES VALUES(4444,33,10);

SQL> INSERT INTO BOOK\_COPIES VALUES(2222,11,12);

SQL> INSERT INTO BOOK\_COPIES VALUES(4444,55,3);

**BOOK\_LENDING**

SQL> INSERT INTO BOOK\_LENDING VALUES(2222,11,1,'10-JAN-2017','20-AUG-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(3333,22,2,'09-JUL-2017','12-AUG-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(4444,55,1,'11-APR-2017','09-AUG-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(2222,11,5,'09-AUG-2017','19-AUG-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(4444,33,1,'10-JUN-2017','15-AUG-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(1111,11,1,'12-MAY-2017','10-JUN-2017');

SQL> INSERT INTO BOOK\_LENDING VALUES(3333,22,1,'10-JUL-2017','15-JUL-2017');

|  |  |  |  |
| --- | --- | --- | --- |
| SQL> SELECT \* FROM BOOK; | |  |  |
| BOOK\_ID | TITLE | PUBLISHER\_NAME PUB\_YEAR | |
| ---------- | ----------------- | ---------------- -------- | |
| 1111 | SE | PEARSON | 2005 |
| 2222 | DBMS | MCGRAW | 2004 |
| 3333 | ANOTOMY | PEARSON | 2010 |
| 4444 | ENCYCLOPEDIA | SAPNA | 2010 |

4 rows selected.

|  |  |  |
| --- | --- | --- |
| SQL> SELECT \* FROM BOOK\_AUTHORS; | |  |
| BOOK\_ID AUTHOR\_NAME | |  |
| ------- | ------------ |  |
| 1111 | SOMMERVILLE |  |
| 2222 | NAVATHE |  |
| 3333 | HENRY GRAY |  |
| 4444 | THOMAS |  |



4 rows selected.

SQL> SELECT \* FROM PUBLISHER;

NAME ADDRESS PHONE

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

PEARSON BANGALORE 9875462530

MCGRAW NEWDELHI 7845691234

SAPNA BANGALORE 7845963210

3 rows selected.

SQL> SELECT \* FROM BOOK\_COPIES;

BOOK\_ID BRANCH\_ID NO\_OF\_COPIES

|  |  |  |
| --- | --- | --- |
| ------- | --------- ------------ | |
| 1111 | 11 | 5 |
| 3333 | 22 | 6 |
| 4444 | 33 | 10 |
| 2222 | 11 | 12 |
| 4444 | 55 | 3 |

5 rows selected.

SQL> SELECT \* FROM BOOK\_LENDING;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BOOK\_ID | BRANCH\_ID CARD\_NO | | DATE\_OUT DUE\_DATE | |
| ------- | --------- -------- --------- --------- | | | |
| 2222 | 11 | 1 | 10-JAN-17 20-AUG-17 | |
| 3333 | 22 | 2 | 09-JUL-17 12-AUG-17 | |
| 4444 | 55 | 1 | 11-APR-17 09-AUG-17 | |
| 2222 | 11 | 5 | 09-AUG-17 | 19-AUG-17 |
| 4444 | 33 | 1 | 10-JUL-17 | 15-AUG-17 |
| 1111 | 11 | 1 | 12-MAY-17 | 10-JUN-17 |
| 3333 | 22 | 1 10-JUL-17 | | 15-JUL-17 |

7 rows selected.

|  |  |  |
| --- | --- | --- |
| SQL> SELECT \* FROM LIBRARY\_BRANCH; | |  |
| BRANCH\_ID BRANCH\_NAME | | ADDRESS |
| --------- ------------------- | | ---------- |
| 11 | CENTRAL TECHNICAL | MG ROAD |
| 22 | MEDICAL | BH ROAD |
| 33 | CHILDREN | SS PURAM |
| 44 | SECRETARIAT | SIRAGATE |
| 55 | GENERAL | JAYANAGAR |



5 rows selected.

Queries:

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

SELECT LB.BRANCH\_NAME, B.BOOK\_ID,TITLE, PUBLISHER\_NAME,AUTHOR\_NAME, NO\_OF\_COPIES

FROM BOOK B, BOOK\_AUTHORS BA, BOOK\_COPIES BC, LIBRARY\_BRANCH LB WHERE B.BOOK\_ID = BA.BOOK\_ID AND

BA.BOOK\_ID = BC.BOOK\_ID AND

BC.BRANCH\_ID = LB.BRANCH\_ID

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BRANCH\_NAME | BOOK\_ID | TITLE | PUBLISHER\_NAME | AUTHOR\_NAME | NO\_OF\_COPIES |
| ---------------- ------- | | ------------ ---------------- ------------- ------------ | | | |
| GENERAL | 4444 | ENCYCLOPEDIA SAPNA | | THOMAS | 3 |
| MEDICAL | 3333 | ANOTOMY | PEARSON | HENRY GRAY | 6 |
| CHILDREN | 4444 | ENCYCLOPEDIA SAPNA | | THOMAS | 10 |
| CENTRAL TECHNICAL | 1111 | SE | PEARSON | SOMMERVILLE | 5 |
| CENTRAL TECHNICAL | 2222 | DBMS | MCGRAW | NAVATHE | 12 |

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 '01-JAN-2017' AND '30-JUN-2017'

**GROUP BY** CARD\_NO

**HAVING** COUNT(\*) > 3;

CARD\_NO

-------

1

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 = '3333';

1 row deleted.

SQL> SELECT \* FROM BOOK;

BOOK\_ID TITLE PUBLISHER\_NAME PUB\_YEAR

|  |  |  |  |
| --- | --- | --- | --- |
| --------- | -------------------- | -------------------- -------- | |
| 1111 | SE | PEARSON | 2005 |
| 2222 | DBMS | MCGRAW | 2004 |
| 4444 | ENCYCLOPEDIA | SAPNA | 2010 |
| SQL> SELECT \* FROM BOOK\_COPIES; | | |  |
| BOOK\_ID BRANCH\_ID NO\_OF\_COPIES | | |  |
| ------- --------- ------------ | | |  |
| 1111 | 11 | 5 |  |
| 4444 | 33 | 10 |  |
| 2222 | 11 | 12 |  |
| 4444 | 55 | 3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| SQL> SELECT \* FROM BOOK\_LENDING; | | |  |
| BOOK\_ID BRANCH\_ID | | CARD\_NO DATE\_OUT | DUE\_DATE |
| ------- --------- --------- --------- | | | --------- |
| 2222 | 11 | 1 10-JAN-17 | 20-AUG-17 |
| 4444 | 55 | 1 11-APR-17 | 09-AUG-17 |
| 2222 | 11 | 5 09-AUG-17 | 19-AUG-17 |
| 4444 | 33 | 1 10-JUN-17 | 15-AUG-17 |
| 1111 | 11 | 1 12-MAY-17 | 10-JUN-17 |

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

**CREATE VIEW V\_PUBLICATION AS**

**SELECT** PUB\_YEAR

**FROM** BOOK;

SELECT \* FROM V\_PUBLICATIONS;

|  |
| --- |
| PUB\_YEAR |
| 2004 |
| 2005 |
| 2010 |
| 2010 |
|  |

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

**CREATE VIEW** BOOKS\_AVAILABLE **AS**

**SELECT** B.BOOK\_ID, B.TITLE, C.NO\_OF\_COPIES

**FROM** LIBRARY\_BRANCH L, BOOK B, BOOK\_COPIES C

**WHERE** B.BOOK\_ID = C.BOOK\_ID AND

L.BRANCH\_ID=C.BRANCH\_ID;

View created.

SQL> SELECT \* FROM BOOKS\_AVAILABLE;

|  |  |  |
| --- | --- | --- |
| BOOK\_ID | TITLE | NO\_OF\_COPIES |
| ------- | -------------------- ------------ | |
| 1111 | SE | 5 |
| 3333 | ANOTOMY | 6 |
| 4444 | ENCYCLOPEDIA | 10 |
| 2222 | DBMS | 12 |
| 4444 | ENCYCLOPEDIA | 3 |



# Consider the following schema for Order Database:

**SALESMAN (*Salesman\_id, Name, City, Commission*) CUSTOMER (*Customer\_id, Cust\_Name, City, Grade, Salesman\_id*)**

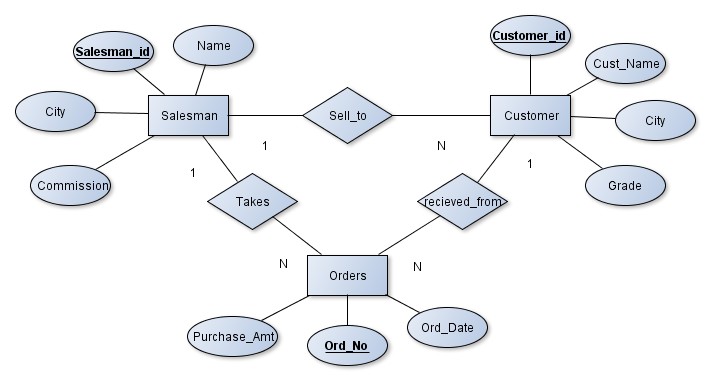
**ORDERS (*Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id*) Write SQL queries to**

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

* 1. **Find the name and numbers of all salesmen who had more than one customer.**
  2. **List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)**
  3. **Create a view that finds the salesman who has the customer with the highest order of a day.**
  4. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

**Solution:**

**Entity-Relationship Diagram**



**Schema Diagram**

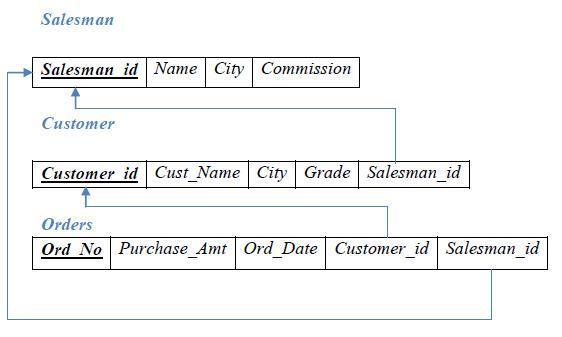
******

Table Creation

CREATE TABLE SALESMAN (SALESMAN\_ID NUMBER (4),

NAME VARCHAR2 (20),

CITY VARCHAR2 (20),

COMMISSION VARCHAR2 (20), PRIMARY KEY(SALESMAN\_ID));

CREATE TABLE CUSTOMER1 (CUSTOMER\_ID NUMBER (4),

CUST\_NAME VARCHAR2 (20),

CITY VARCHAR2 (20),

GRADE NUMBER (3),

SALESMAN\_ID NUMBER (4),

PRIMARY KEY (CUSTOMER\_ID),

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

CREATE TABLE ORDERS (ORD\_NO NUMBER (5),

PURCHASE\_AMT NUMBER (10, 2),

ORD\_DATE DATE,

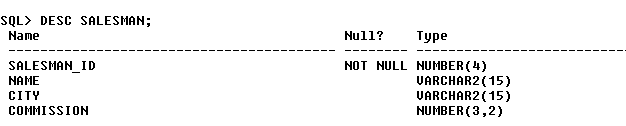
CUSTOMER\_ID NUMBER (4),

SALESMAN\_ID NUMBER (4),

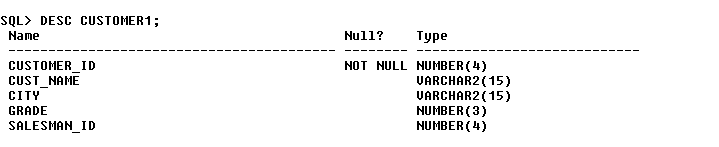
PRIMARY KEY (ORD\_NO),

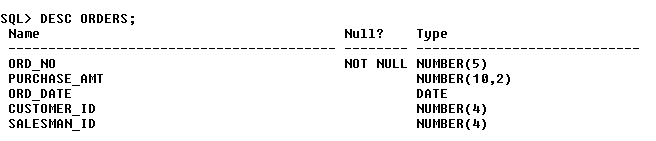
CUSTOMER\_ID REFERENCES CUSTOMER1 (CUSTOMER\_ID) ON DELETE CASCADE, SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE CASCADE);

# Table Descriptions

DESC SALESMAN;

DESC CUSTOMER1;



DESC ORDERS;

# Insertion of Values to Tables

INSERT INTO SALESMAN VALUES (1000, ‘JOHN’,’BANGALORE’,’25 %’); INSERT INTO SALESMAN VALUES (2000, ‘RAVI’,’BANGALORE’,’20 %’); INSERT INTO SALESMAN VALUES (3000, ‘KUMAR’,’MYSORE’,’15 %’); INSERT INTO SALESMAN VALUES (4000, ‘SMITH’,’DELHI’,’30 %’); INSERT INTO SALESMAN VALUES (5000, ‘HARSHA’,’HYDRABAD’,’15 %’);

INSERT INTO CUSTOMER1 VALUES (10, ‘PREETHI’,’BANGALORE’, 100, 1000);

INSERT INTO CUSTOMER1 VALUES (11, ‘VIVEK’,’MANGALORE’, 300, 1000);

INSERT INTO CUSTOMER1 VALUES (12, ‘BHASKAR’,’CHENNAI’, 400, 2000);

INSERT INTO CUSTOMER1 VALUES (13, ‘CHETHAN’,’BANGALORE’, 200, 2000);

INSERT INTO CUSTOMER1 VALUES (14, ‘MAMATHA’,’BANGALORE’, 400, 3000);

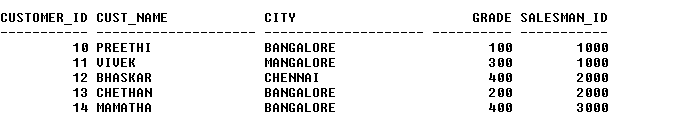
INSERT INTO ORDERS VALUES (50, 5000, ‘04-MAY-17’, 10, 1000);

INSERT INTO ORDERS VALUES (51, 450, ‘20-JAN-17’, 10, 2000);

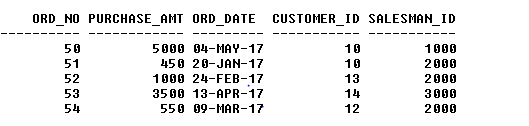
INSERT INTO ORDERS VALUES (52, 1000, ‘24-FEB-17’, 13, 2000);

INSERT INTO ORDERS VALUES (53, 3500, ‘13-APR-17’, 14, 3000);

INSERT INTO ORDERS VALUES (54, 550, ‘09-MAR-17’, 12, 2000); SELECT \* FROM SALESMAN;

SELECT \* FROM CUSTOMER1;

SELECT \* FROM ORDERS;

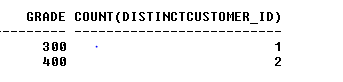


# Queries:

* + 1. **Count the customers with grades above Bangalore’s average.** SELECT GRADE, COUNT (DISTINCT CUSTOMER\_ID) FROM CUSTOMER1

GROUP BY GRADE

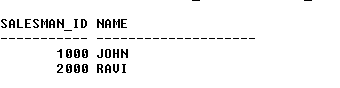
HAVING GRADE > (SELECT AVG(GRADE) FROM CUSTOMER1

WHERE CITY='BANGALORE');

# 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 CUSTOMER1

WHERE SALESMAN\_ID=A.SALESMAN\_ID);

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

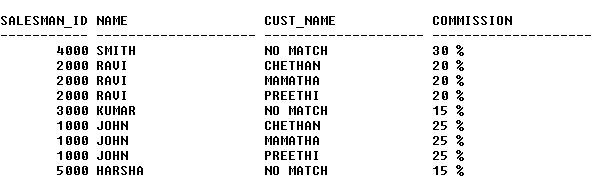
SELECT SALESMAN.SALESMAN\_ID, NAME, CUST\_NAME, COMMISSION FROM SALESMAN, CUSTOMER1

WHERE SALESMAN.CITY = CUSTOMER1.CITY UNION

SELECT SALESMAN\_ID, NAME, 'NO MATCH', COMMISSION FROM SALESMAN

WHERE NOT CITY = ANY (SELECT CITY

FROM CUSTOMER1) ORDER BY 2 DESC;



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

CREATE VIEW ELITSALESMAN AS

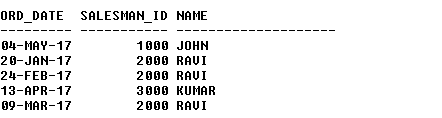
SELECT B.ORD\_DATE, A.SALESMAN\_ID, A.NAME FROM SALESMAN A, ORDERS B

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);

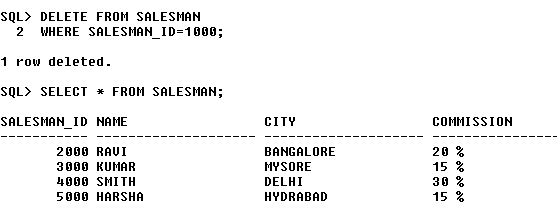


# Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

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

Use ON DELETE SET NULL at the end of foreign key definitions while creating child table customers and then executes the following:

DELETE FROM SALESMAN WHERE SALESMAN\_ID=1000;



# 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*)**

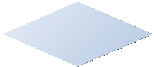
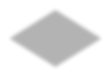
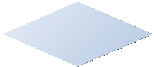
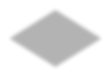
**RATING (*Mov\_id, Rev\_Stars*) Write SQL queries to**

# List the titles of all movies directed by ‘Hitchcock’.

* 1. **Find the movie names where one or more actors acted in two or more movies.**
  2. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
  3. **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.**
  4. **Update rating of all movies directed by ‘Steven Spielberg’ to 5.**

**Solution:**

**Entity-Relationship Diagram**



Actor

**Dir\_id**

Dir\_Name

**Act\_id**

Act\_Name

Dir\_Phone

Act\_Gender

1

M

Has

Movie\_Cast

N

Role

Rev\_Stars

N

Mov\_Lang

**Mov\_id**

Mov\_Title

Mov\_Year

Director

Movies

**Schema Diagram**

## Actor



***Director***

***Movie\_Cast***

|  |  |  |
| --- | --- | --- |
| ***Act\_id*** | *Act\_Name* | *Act\_Gender* |

|  |  |  |
| --- | --- | --- |
| ***Dir\_id*** | *Dir\_Name* | *Dir\_Phone* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Movies*** | | | | |  |
| ***Mov\_id*** | *Mov\_Title* | *Mov\_Year* | *Mov\_Lang* | *Dir\_id* | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Act\_id*** | ***Mov\_id*** | *Role* |

|  |  |
| --- | --- |
| ***Rating*** |  |
| ***Mov\_id*** | *Rev\_Stars* |
|  |  |

**Table Creation**

CREATE TABLE ACTOR ( ACT\_ID NUMBER (3),

ACT\_NAME VARCHAR (20),

ACT\_GENDER CHAR (1), PRIMARY KEY (ACT\_ID));

CREATE TABLE DIRECTOR ( DIR\_ID NUMBER (3),

DIR\_NAME VARCHAR (20),

DIR\_PHONE NUMBER (10), PRIMARY KEY (DIR\_ID));

CREATE TABLE MOVIES ( MOV\_ID NUMBER (4),

MOV\_TITLE VARCHAR (25),

MOV\_YEAR NUMBER (4),

MOV\_LANG VARCHAR (12),

DIR\_ID NUMBER (3), PRIMARY KEY (MOV\_ID),

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

CREATE TABLE MOVIE\_CAST ( ACT\_ID NUMBER (3),

MOV\_ID NUMBER (4),

ROLE VARCHAR (10),

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

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

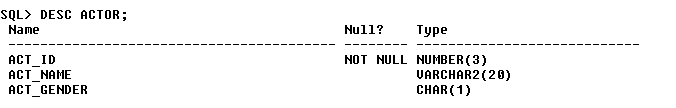
CREATE TABLE RATING ( MOV\_ID NUMBER (4),

REV\_STARS VARCHAR (25), PRIMARY KEY (MOV\_ID),

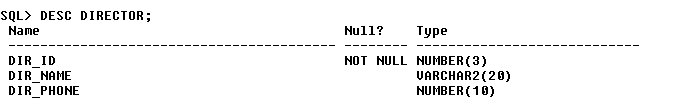
FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

# Table Descriptions

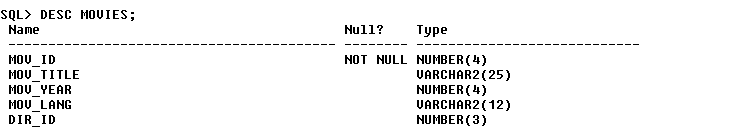
DESC ACTOR;



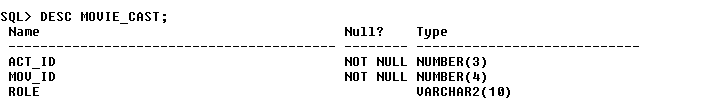
DESC DIRECTOR;



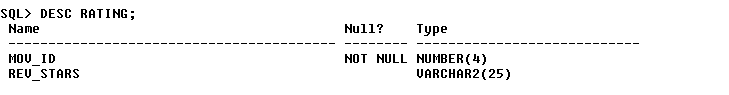
DESC MOVIES;



DESC MOVIE\_CAST;



DESC RATING;



# Insertion of Values to Tables

INSERT INTO ACTOR VALUES (301,’ANUSHKA’,’F’); INSERT INTO ACTOR VALUES (302,’PRABHAS’,’M’); INSERT INTO ACTOR VALUES (303,’PUNITH’,’M’); INSERT INTO ACTOR VALUES (304,’JERMY’,’M’);

INSERT INTO DIRECTOR VALUES (60,’RAJAMOULI’, 8751611001); INSERT INTO DIRECTOR VALUES (61,’HITCHCOCK’, 7766138911); INSERT INTO DIRECTOR VALUES (62,’FARAN’, 9986776531);

INSERT INTO DIRECTOR VALUES (63,’STEVEN SPIELBERG’, 8989776530);

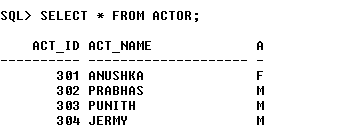
INSERT INTO MOVIES VALUES (1001,’BAHUBALI-2’, 2017, ‘TELAGU’, 60); INSERT INTO MOVIES VALUES (1002,’BAHUBALI-1’, 2015, ‘TELAGU’, 60); INSERT INTO MOVIES VALUES (1003,’AKASH’, 2008, ‘KANNADA’, 61); INSERT INTO MOVIES VALUES (1004,’WAR HORSE’, 2011, ‘ENGLISH’, 63);

INSERT INTO MOVIE\_CAST VALUES (301, 1002, ‘HEROINE’); INSERT INTO MOVIE\_CAST VALUES (301, 1001, ‘HEROINE’); INSERT INTO MOVIE\_CAST VALUES (303, 1003, ‘HERO’); INSERT INTO MOVIE\_CAST VALUES (303, 1002, ‘GUEST’); INSERT INTO MOVIE\_CAST VALUES (304, 1004, ‘HERO’);

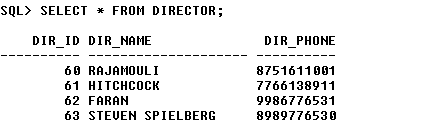
INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

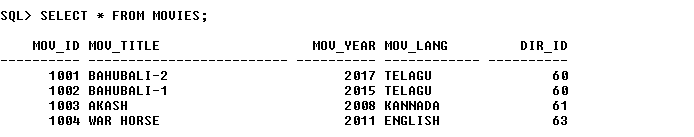
INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4); SELECT \* FROM ACTOR;

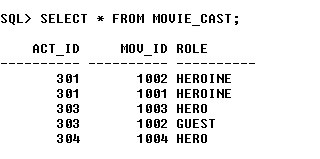
SELECT \* FROM DIRECTOR;

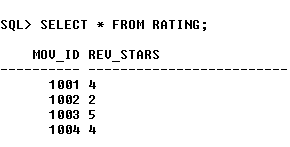


SELECT \* FROM MOVIES;



SELECT \* FROM MOVIE\_CAST;



SELECT \* FROM RATING;

# Queries:

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

SELECT MOV\_TITLE FROM MOVIES

WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME = ‘HITCHCOCK’);



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

SELECT MOV\_TITLE

FROM MOVIES M, MOVIE\_CAST MV

WHERE M.MOV\_ID=MV.MOV\_ID AND ACT\_ID IN (SELECT ACT\_ID

FROM MOVIE\_CAST GROUP BY ACT\_ID HAVING COUNT (ACT\_ID)>1)

GROUP BY MOV\_TITLE HAVING COUNT (\*)>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 JOIN MOVIES M

ON C.MOV\_ID=M.MOV\_ID

WHERE M.MOV\_YEAR NOT BETWEEN 2000 AND 2015; OR

SELECT A.ACT\_NAME, A.ACT\_NAME, C.MOV\_TITLE, C.MOV\_YEAR FROM ACTOR A, MOVIE\_CAST B, MOVIES C

WHERE A.ACT\_ID=B.ACT\_ID AND B.MOV\_ID=C.MOV\_ID

AND C.MOV\_YEAR NOT BETWEEN 2000 AND 2015;

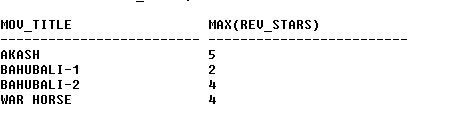


# Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

SELECT MOV\_TITLE, MAX (REV\_STARS) FROM MOVIES

INNER JOIN RATING USING (MOV\_ID) GROUP BY MOV\_TITLE

HAVING MAX (REV\_STARS)>0 ORDER BY MOV\_TITLE;



# Update rating of all movies directed by ‘Steven Spielberg’ to 5

KL

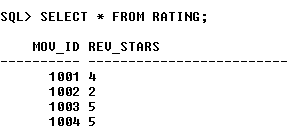
UPDATE RATING SET REV\_STARS=5

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

WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME = ‘STEVEN SPIELBERG’));



# 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**

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

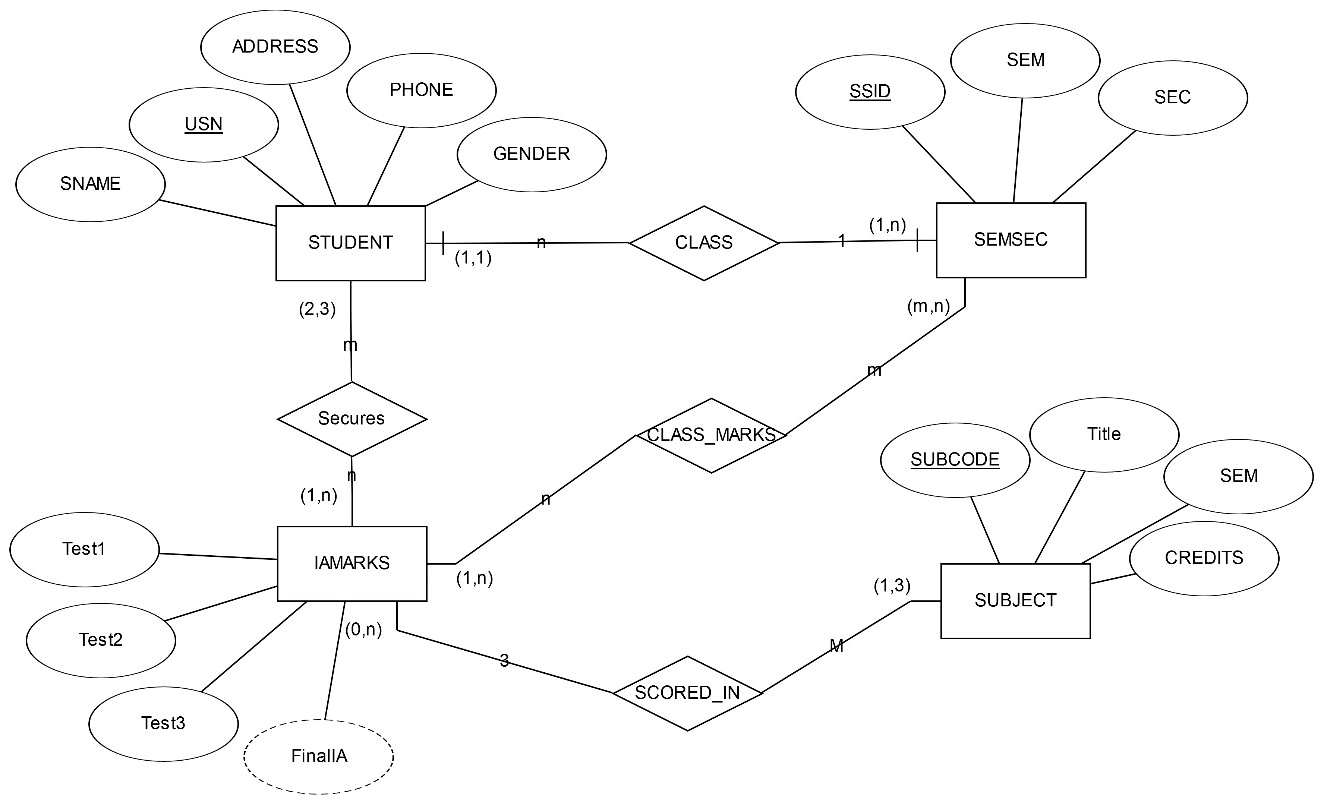
* 1. **Compute the total number of male and female students in each semester and in each section.**
  2. **Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.**
  3. **Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.**
  4. **Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = ‘Outstanding’**

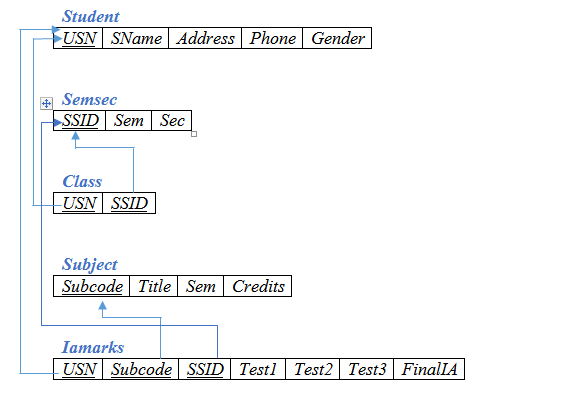
**If FinalIA = 12 to 16 then CAT = ‘Average’ If FinalIA< 12 then CAT = ‘Weak’**

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

**Solution:**

**Entity - Relationship Diagram**



**Schema Diagram**

**Table Creation**

CREATE TABLE STUDENT (

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

ADDRESS VARCHAR (25),

PHONE NUMBER (10),

GENDER CHAR (1));

CREATE TABLE SEMSEC (

SSID VARCHAR (5) PRIMARY KEY, SEM NUMBER (2),

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 (8),

TITLE VARCHAR (20),

SEM NUMBER (2),

CREDITS NUMBER (2), PRIMARY KEY (SUBCODE));

CREATE TABLE IAMARKS ( USN VARCHAR (10),

SUBCODE VARCHAR (8),

SSID VARCHAR (5),

TEST1 NUMBER (2),

TEST2 NUMBER (2),

TEST3 NUMBER (2),

FINALIA NUMBER (2),

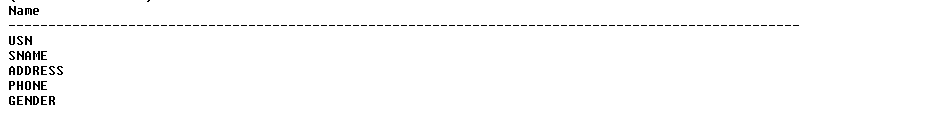
PRIMARY KEY (USN, SUBCODE, SSID),

FOREIGN KEY (USN) REFERENCES STUDENT (USN),

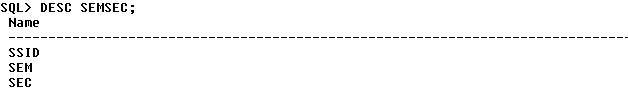
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE), FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));

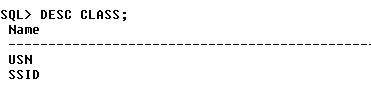
# Table Descriptions

DESC STUDENT;

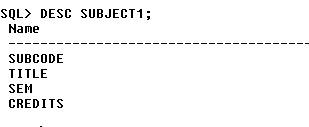


DESC SEMSEC;

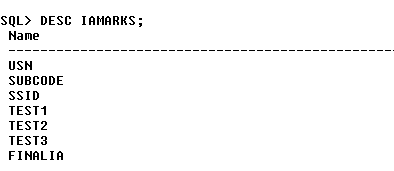


DESC CLASS;

DESC SUBJECT;



DESC IAMARKS;



# Insertion of values to tables

INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI', 8877881122,'M');

INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU', 7722829912,'F');

INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU', 7712312312,'F');

INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU', 8877881122,'F');

INSERT INTO STUDENTVALUES ('1RN14CS010','ABHAY','BENGALURU', 9900211201,'M');

INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU', 9923211099,'M');

INSERT INTO STUDENTVALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F'); INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');

INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE', 7696772121,'F');

INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M'); INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU', 8812332201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI', 9900232201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA', 9905542212,'F');

INSERT INTO STUDENT VALUES ('1RN16CS122','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 (‘1RN13CS020’,’CSE8A’); INSERT INTO CLASS VALUES (‘1RN13CS062’,’CSE8A’); INSERT INTO CLASS VALUES (‘1RN13CS066’,’CSE8B’); INSERT INTO CLASS VALUES (‘1RN13CS091’,’CSE8C’);

INSERT INTO CLASS VALUES (‘1RN14CS010’,’CSE7A’); INSERT INTO CLASS VALUES (‘1RN14CS025’,’CSE7A’); INSERT INTO CLASS VALUES (‘1RN14CS032’,’CSE7A’);

INSERT INTO CLASS VALUES (‘1RN15CS011’,’CSE4A’); INSERT INTO CLASS VALUES (‘1RN15CS029’,’CSE4A’); INSERT INTO CLASS VALUES (‘1RN15CS045’,’CSE4B’); INSERT INTO CLASS VALUES (‘1RN15CS091’,’CSE4C’);

INSERT INTO CLASS VALUES (‘1RN16CS045’,’CSE3A’); INSERT INTO CLASS VALUES (‘1RN16CS088’,’CSE3B’); INSERT INTO CLASS VALUES (‘1RN16CS122’,’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); 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);

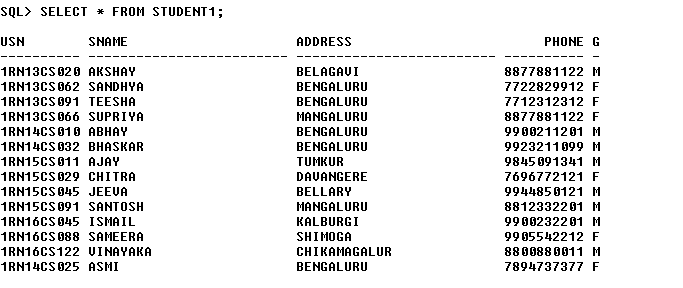
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS81','CSE8C', 15, 16, 18);

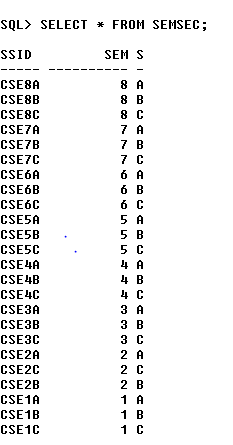
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS82','CSE8C', 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS83','CSE8C', 19, 15, 20);

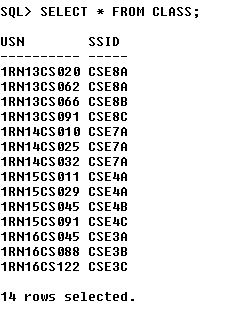
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS84','CSE8C', 20, 16, 19);

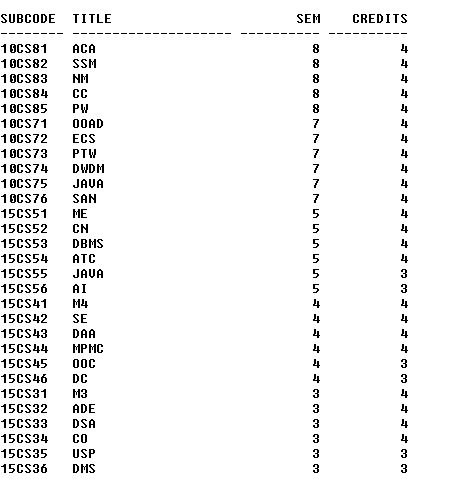
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS85','CSE8C', 15, 15, 12);

SELECT \* FROM STUDENT;

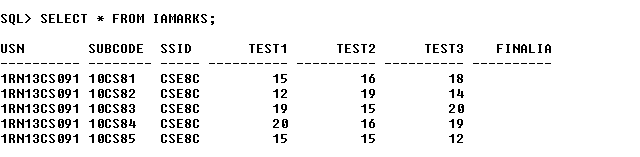
SELECT \* FROM SEMSEC;

SELECT \* FROM CLASS;



SELECT \* FROM SUBJECT;

SELECT \* FROM IAMARKS;



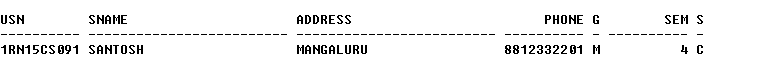
# Queries:

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

SELECT S.\*, SS.SEM, SS.SEC

FROM STUDENT S, SEMSEC SS, CLASS C WHERE S.USN = C.USN AND

SS.SSID = C.SSID AND SS.SEM = 4 AND

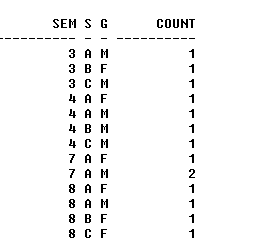
SS.SEc=’C’;

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

SELECT SS.SEM, SS.SEC, S.GENDER, COUNT (S.GENDER) AS COUNT FROM STUDENT S, SEMSEC SS, CLASS C

WHERES.USN = C.USN AND SS.SSID = C.SSID

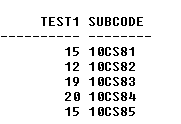
GROUP BY SS.SEM, SS.SEC, S.GENDER ORDER BY SEM;



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

CREATE VIEW STU\_TEST1\_MARKS\_VIEW AS

SELECT TEST1, SUBCODE FROM IAMARKS

WHERE USN = '1RN13CS091';

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

CREATE OR REPLACE PROCEDURE AVGMARKS IS

CURSOR C\_IAMARKS IS

SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2) AS C

FROM IAMARKS

WHERE FINALIA IS NULL FOR UPDATE;

C\_A NUMBER; C\_B NUMBER; C\_C NUMBER; C\_SM NUMBER; C\_AV NUMBER;

BEGIN

OPEN C\_IAMARKS; LOOP

FETCH C\_IAMARKS INTO C\_A, C\_B, C\_C; EXIT WHEN C\_IAMARKS%NOTFOUND;

--DBMS\_OUTPUT.PUT\_LINE(C\_A || ' ' || C\_B || ' ' || C\_C); IF (C\_A != C\_B) THEN

C\_SM:=C\_A+C\_B; ELSE

C\_SM:=C\_A+C\_C; END IF;

C\_AV:=C\_SM/2;

--DBMS\_OUTPUT.PUT\_LINE('SUM = '||C\_SM);

--DBMS\_OUTPUT.PUT\_LINE('AVERAGE = '||C\_AV);

UPDATE IAMARKS SET FINALIA=C\_AV WHERE CURRENT OF C\_IAMARKS;

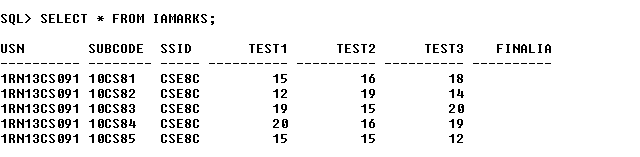
END LOOP;

CLOSE C\_IAMARKS; END;

/

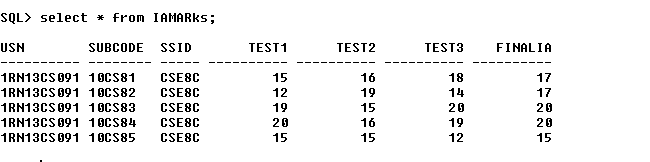
**Note:** Before execution of PL/SQL procedure, IAMARKS table contents are:

SELECT \* FROM IAMARKS;



# Below SQL code is to invoke the PL/SQL stored procedure from the command line:

BEGIN AVGMARKS; END;



# Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = ‘Outstanding’

**If FinalIA = 12 to 16 then CAT = ‘Average’ If FinalIA< 12 then CAT = ‘Weak’**

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

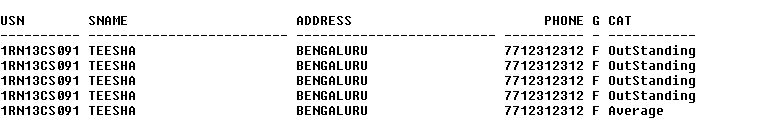
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, (CASE

WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING' WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE' ELSE 'WEAK'

END) AS CAT

FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB WHERE S.USN = IA.USN AND

SS.SSID = IA.SSID AND SUB.SUBCODE = IA.SUBCODE AND SUB.SEM = 8;



# Consider the schema for Company Database:

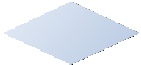
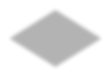
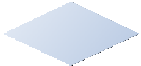
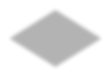
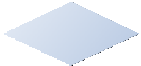
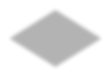
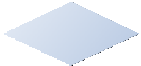
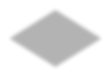
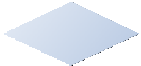
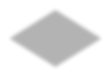
**EMPLOYEE (*SSN, Name, Address, Sex, Salary, SuperSSN, DNo*) DEPARTMENT (*DNo, DName, MgrSSN, MgrStartDate*) DLOCATION (*DNo,DLoc*)**

**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 name is ‘Scott’, either as a worker or as a manager of the department that controls the project.**
  2. **Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.**
  3. **Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department**
  4. **Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator). 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.**

**Entity-Relationship Diagram**



Employee

Department

Project

**SSN**

Controlled\_by

Name

N

1

**DNO**

Salary

DName

1

N

Manages

Address

MgrStartDate

1

Sex

1

N

M

Dlocation

Supervisee

Supervisor

Supervision

Works\_on

Controls

N

N

Hours

PName

**PNO**

PLocation

**Schema Diagram**

## Employee



***Project***



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***SSN*** | *Fname* | *Lname* | *Address* | *Sex* | *Salary* | *SuperSSN* | *DNO* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Depar*** | ***tment*** |  |  |
|  |  |  |  |
| ***DNO*** | *Dname* | *MgrSSN* | *MgrStartDate* |

|  |  |  |  |
| --- | --- | --- | --- |
| ***DLocation*** | | | |
|  | ***DNO*** | | ***DLOC*** |
|  | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***PNO*** | *PName* | *PLocation* | *DNO* |
| ***Work*** | ***s\_on*** |  |  |

|  |  |  |
| --- | --- | --- |
| ***SSN*** | ***PNO*** | *Hours* |

**Table Creation**

CREATE TABLE DEPARTMENT

(DNO VARCHAR2 (20) PRIMARY KEY, DNAME VARCHAR2 (20), MGRSTARTDATE DATE);

CREATE TABLE EMPLOYEE

(SSN VARCHAR2 (20) PRIMARY KEY, FNAME VARCHAR2 (20),

LNAME VARCHAR2 (20),

ADDRESS VARCHAR2 (20),

SEX CHAR (1), SALARY INTEGER,

SUPERSSN REFERENCES EMPLOYEE (SSN), 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 MGRSSN REFERENCES EMPLOYEE (SSN);

CREATE TABLE DLOCATION (DLOC VARCHAR2 (20),

DNO REFERENCES DEPARTMENT (DNO), PRIMARY KEY (DNO, DLOC));

CREATE TABLE PROJECT (PNO INTEGER PRIMARY KEY, PNAME VARCHAR2 (20),

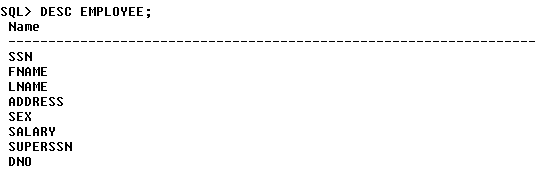
PLOCATION VARCHAR2 (20),

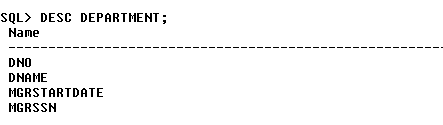
DNO REFERENCES DEPARTMENT (DNO));

CREATE TABLE WORKS\_ON (HOURS NUMBER (2),

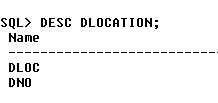
SSN REFERENCES EMPLOYEE (SSN), PNO REFERENCES PROJECT(PNO), PRIMARY KEY (SSN, PNO));

# Table Descriptions

DESC EMPLOYEE;

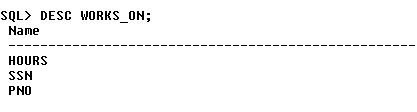
DESC DEPARTMENT;

DESC DLOCATION;



DESC PROJECT;

DESC WORKS\_ON;



# Insertion of values to tables

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSECE01’,’JOHN’,’SCOTT’,’BANGALORE’,’M’, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE01’,’JAMES’,’SMITH’,’BANGALORE’,’M’, 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE02’,’HEARN’,’BAKER’,’BANGALORE’,’M’, 700000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE03’,’EDWARD’,’SCOTT’,’MYSORE’,’M’, 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE04’,’PAVAN’,’HEGDE’,’MANGALORE’,’M’, 650000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE05’,’GIRISH’,’MALYA’,’MYSORE’,’M’, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSCSE06’,’NEHA’,’SN’,’BANGALORE’,’F’, 800000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSACC01’,’AHANA’,’K’,’MANGALORE’,’F’, 350000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSACC02’,’SANTHOSH’,’KUMAR’,’MANGALORE’,’M’, 300000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSISE01’,’VEENA’,’M’,’MYSORE’,’M’, 600000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘RNSIT01’,’NAGESH’,’HR’,’BANGALORE’,’M’, 500000);

INSERT INTO DEPARTMENT VALUES (‘1’,’ACCOUNTS’,’01-JAN-01’,’RNSACC02’); INSERT INTO DEPARTMENT VALUES (‘2’,’IT’,’01-AUG-16’,’RNSIT01’);

INSERT INTO DEPARTMENT VALUES (‘3’,’ECE’,’01-JUN-08’,’RNSECE01’); INSERT INTO DEPARTMENT VALUES (‘4’,’ISE’,’01-AUG-15’,’RNSISE01’); INSERT INTO DEPARTMENT VALUES (‘5’,’CSE’,’01-JUN-02’,’RNSCSE05’);

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

UPDATE EMPLOYEE SET SUPERSSN=NULL, DNO=’3’ WHERE SSN=’RNSECE01’;

UPDATE EMPLOYEE SET SUPERSSN=’RNSCSE02’, DNO=’5’ WHERE SSN=’RNSCSE01’;

UPDATE EMPLOYEE SET SUPERSSN=’RNSCSE03’, DNO=’5’ WHERE SSN=’RNSCSE02’;

UPDATE EMPLOYEE SET SUPERSSN=’RNSCSE04’, DNO=’5’ WHERE SSN=’RNSCSE03’;

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=’RNSCSE05’ WHERE SSN=’RNSCSE04’;

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=’RNSCSE06’ WHERE SSN=’RNSCSE05’;

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=NULL WHERE SSN=’RNSCSE06’;

UPDATE EMPLOYEE SET DNO=’1’, SUPERSSN=’RNSACC02’ WHERE SSN=’RNSACC01’;

UPDATE EMPLOYEE SET DNO=’1’, SUPERSSN=NULL WHERE SSN=’RNSACC02’;

UPDATE EMPLOYEE SET DNO=’4’, SUPERSSN=NULL WHERE SSN=’RNSISE01’;

UPDATE EMPLOYEE SET DNO=’2’, SUPERSSN=NULL WHERE SSN=’RNSIT01’;

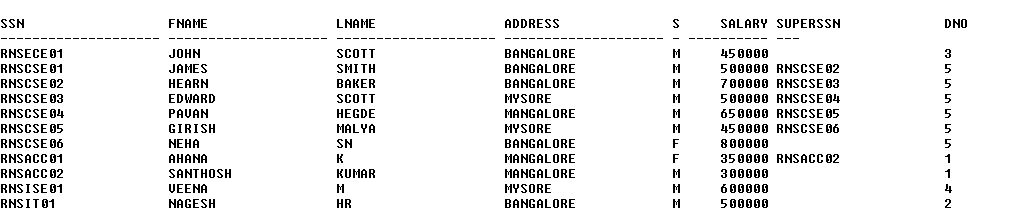
INSERT INTO DLOCATION VALUES (’BANGALORE’, ‘1’); INSERT INTO DLOCATION VALUES (’BANGALORE’, ‘2’); INSERT INTO DLOCATION VALUES (’BANGALORE’, ‘3’); INSERT INTO DLOCATION VALUES (’MANGALORE’, ‘4’); INSERT INTO DLOCATION VALUES (’MANGALORE’, ‘5’);

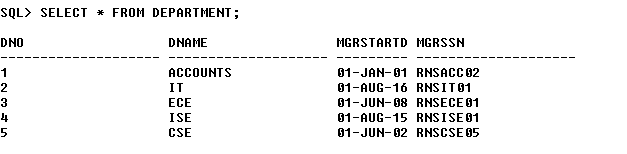
INSERT INTO PROJECT VALUES (100,’IOT’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (101,’CLOUD’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (102,’BIGDATA’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (103,’SENSORS’,’BANGALORE’,’3’);

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’);

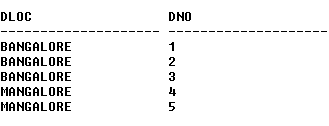
INSERT INTO PROJECT VALUES (107,’SMART CITY’,’BANGALORE’,’2’);

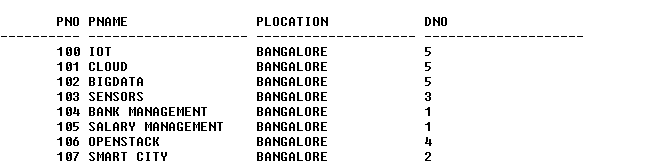
INSERT INTO WORKS\_ON VALUES (4, ‘RNSCSE01’, 100); INSERT INTO WORKS\_ON VALUES (6, ‘RNSCSE01’, 101); INSERT INTO WORKS\_ON VALUES (8, ‘RNSCSE01’, 102); INSERT INTO WORKS\_ON VALUES (10, ‘RNSCSE02’, 100); INSERT INTO WORKS\_ON VALUES (3, ‘RNSCSE04’, 100); INSERT INTO WORKS\_ON VALUES (4, ‘RNSCSE05’, 101); INSERT INTO WORKS\_ON VALUES (5, ‘RNSCSE06’, 102); INSERT INTO WORKS\_ON VALUES (6, ‘RNSCSE03’, 102); INSERT INTO WORKS\_ON VALUES (7, ‘RNSECE01’, 103); INSERT INTO WORKS\_ON VALUES (5, ‘RNSACC01’, 104); INSERT INTO WORKS\_ON VALUES (6, ‘RNSACC02’, 105); INSERT INTO WORKS\_ON VALUES (4, ‘RNSISE01’, 106); INSERT INTO WORKS\_ON VALUES (10, ‘RNSIT01’, 107);

SELECT \* FROM EMPLOYEE;

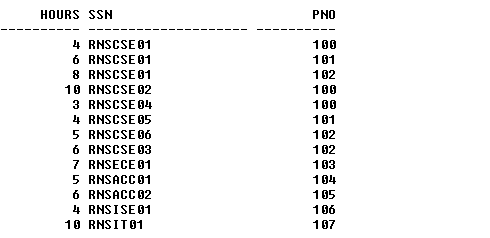
SELECT \* FROM DEPARTMENT;

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 is ‘Scott’, either as a worker or as a manager of the department that controls the project.**

(SELECT DISTINCT P.PNO

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

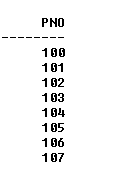
AND D.MGRSSN=E.SSN AND E.LNAME=’SCOTT’) UNION

(SELECT DISTINCT P1.PNO

FROM PROJECT P1, WORKS\_ON W, EMPLOYEE E1 WHERE P1.PNO=W.PNO

AND E1.SSN=W.SSN

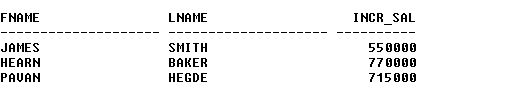
AND E1.LNAME=’SCOTT’);



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

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

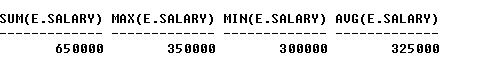
WHERE E.SSN=W.SSN AND W.PNO=P.PNO AND P.PNAME=’IOT’;



# Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department

SELECT SUM (E.SALARY), MAX (E.SALARY), MIN (E.SALARY), AVG (E.SALARY)

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

AND D.DNAME=’ACCOUNTS’;

# 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

WHERE DNO=’5’) MINUS (SELECT PNO FROM WORKS\_ON WHERE E.SSN=SSN));



# 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;



# Viva Questions

1. **What is SQL?**

Structured Query Language

# What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

# What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

# What is a Database system?

The database and DBMS software together is called as Database system.

# Advantages of DBMS?

* + Redundancy is controlled.
  + Unauthorized access is restricted.
  + Providing multiple user interfaces.
  + Enforcing integrity constraints.
  + Providing backup and recovery.

# Disadvantage in File Processing System?

* + Data redundancy & inconsistency.
  + Difficult in accessing data.
  + Data isolation.
  + Data integrity.
  + Concurrent access is not possible.
  + Security Problems.

# Describe the three levels of data abstraction?

There are three levels of abstraction:

* + Physical level: The lowest level of abstraction describes how data are stored.
  + Logical level*:* The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
  + View level:The highest level of abstraction describes only part of entire database.

# Define the "integrity rules"

There are two Integrity rules.

* + Entity Integrity:States that “Primary key cannot have NULL value”
  + Referential Integrity:States that “Foreign Key can be either a NULL value or should be Primary Key value of other relation.

# What is extension and intension?

Extension - It is the number of tuples present in a table at any instance. This is time dependent.

Intension -It is a constant value that gives the name, structure of table and the constraints laid on it.

# What is Data Independence?

Data independence means that “the application is independent of the storage structure and access strategy of data”. In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

* + Physical Data Independence: Modification in physical level should not affect the logical level.
  + Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

# What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

# What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

# What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

# What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

# What is an Entity?

It is an 'object' in the real world with an independent existence.

# What is an Entity type?

It is a collection (set) of entities that have same attributes.

# What is an Entity set?

It is a collection of all entities of particular entity type in the database.

# What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity

set.

# What is an attribute?

It is a particular property, which describes the entity.

# What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ...,tn). Each tuple is an ordered list of n- values t=(v1,v2, ..., vn).

# What is degree of a Relation?

It is the number of attribute of its relation schema.

# What is Relationship?

It is an association among two or more entities.

# What is Relationship set?

The collection (or set) of similar relationships.

## What is Relationship type?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

# What is degree of Relationship type?

It is the number of entity type participating.

# What is DDL (Data Definition Language)?

A data base schema is specified by a set of definitions expressed by a special language called DDL.

# What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

# What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

# What is Data Storage - Definition Language?

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage- definition language.

# What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organized by appropriate data model.

* + Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.
  + Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

# What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

1. **What is Relational Algebra?**

It is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

# What is Relational Calculus?

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL, ALPHA, QUEL.

# What is normalization?

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

* + Minimizing redundancy
  + Minimizing insertion, deletion and update anomalies.

# What is Functional Dependency?

A Functional dependency is denoted by X Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of

R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

# When is a functional dependency F said to be minimal?

* + Every dependency in F has a single attribute for its right hand side.
  + We cannot replace any dependency X A in F with a dependency Y A where Y is a proper subset of X and still have a set of dependency that is equivalent to F.
  + We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

# What is Multivalued dependency?

Multivalued dependency denoted by X Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

 t3[x] = t4[X] = t1[X] = t2[X]

 t3[Y] = t1[Y] and t4[Y] = t2[Y]

 t3[Z] = t2[Z] and t4[Z] = t1[Z]

where [Z = (R-(X U Y)) ]

# What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

# What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

# What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X Y is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

# What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

# What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true

* X is a Super-key of R.
* A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

# What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies additional constraints that for every FD X A, X must be a candidate key.

# What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true

* X is subset or equal to (or) XY = R.
* X is a super key.

# What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ...,Rn} that holds R, one the following is true

* Ri = R for some i.
* The join dependency is implied by the set of FD, over R in which the left side is key of R.

# What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.