| **DATABASE MANAGEMENT SYSTEMS LAB** | | | |
| --- | --- | --- | --- |
| **Course code:** | 22CDL45 | **Credits:** | 1 |
| **L: T:P:** | 0:0:2 | **CIE Marks:** | 50 |
| **Exam Hours:** | 03 | **SEE Marks:** | 50 |
| **Total Hours:** | 15 |  |  |

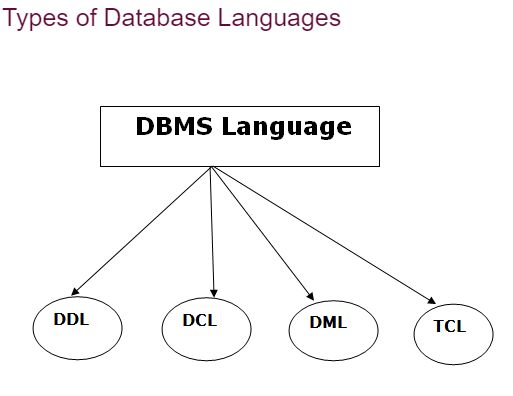
| **Course Objectives** | |
| --- | --- |
| **1** | To Create, Update and query on the database. |
| **2** | Demonstrate the working of different concepts of DBMS |
| **3** | To gain knowledge on importance of applying normalization and assigning keys to the attributes |
| **4** | To Implement, analyze and evaluate the project developed for an application using DBMS. |

| **Course Outcomes: At the end of the course, student will be able to** | |
| --- | --- |
| **CO1** | **Ability** to design and implement a database schema for given problem |
| **CO2** | **Apply** the normalization techniques for development of application software to realistic problems. |
| **CO3** | **Demons**trate various advanced queries execution such as relational constraints, joins, set operations, aggregate functions, trigger, views and embedded SQL. |
| **CO4** | Develop database applications using front-end tools and back-end DBMS |

| **Mapping of Course outcomes to Program outcomes:** | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | 3 | 1 | 1 |  | 3 |  |  |  |  |  |  |  | 2 | 1 | 2 |
| **CO2** | 3 | 2 | 2 | 1 | 3 |  |  |  |  |  |  |  | 2 | 2 | 2 |
| **CO3** | 3 | 2 | 2 | 2 | 3 |  |  |  |  |  |  |  | 2 | 2 | 2 |
| **CO4** | 3 | 2 | 2 | 2 | 3 |  |  |  |  |  |  |  | 2 | 2 | 2 |

|  |  |  |
| --- | --- | --- |
| **Expt** | **Content of the Experiment** | **Hours** |
| 1 | Consider the following schema for a Library Database:  BOOK (Book\_id, Title, Publisher\_Name, Pub\_Year)  BOOK\_AUTHORS (Book\_id, Author\_Name)  PUBLISHER (Pub\_id,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  i. Create the above tables by properly specifying the primary keys and the foreign keys.  ii. Enter at least five tuples for each relation.  iii. Retrieve details of all books in the library – book-id, title, publisher name,  authors, number of copies, etc.  iv. Get the details of borrowers who have borrowed more than 3 book from Jan  2024 to may 2024  v. Demonstrate the DELETE operation by deleting a book details in BOOK  table.  vi. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.  vii. Create a view of all books and its number of copies that are currently available in the Library. | 2 |
| 2 | Consider the following schema for Order Database:  SALESMAN (Salesman\_id, Name, City, Commission)  CUSTOMER (Customer\_id, Cust\_Name, City, Grade, Salesman\_id)  ORDERS (Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id)  Write SQL queries to  i. Create the above tables by properly specifying the primary keys and the  foreign keys.  ii. Enter at least five tuples for each relation.  iii. Count the customers with grades above Bangalore’s average.  iv. Find the name and numbers of all salesmen who had more than one  customer.  v. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)  vi. Create a view that finds the salesman who has the customer with the highest order of a day.  vii. Demonstrate the DELETE operation by removing salesmen with id 1000.  All his orders must also be deleted. | 2 |
| 3 | Consider the schema for Movie Database:  ACTOR (Act\_id, Act\_Name, Act\_Gender)  DIRECTOR (Dir\_id, Dir\_Name, Dir\_Phone)  MOVIES (Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id)  MOVIE\_CAST (Act\_id, Mov\_id, Role)  RATING (Mov\_id, Rev\_Stars)  Write SQL queries to2  i. Create the above tables by properly specifying the primary keys and the  foreign keys.  ii. Enter at least five tuples for each relation.  iii. List the titles of all movies directed by ‘Mr. Dwarakesh’.  iv. Find the movie names where actors acted in two or more movies.  v. List all actors who acted in a movie before 2005 and also in a movie after 2015 (use JOIN operation).  vi. Find the title of movie and number of stars rated for each movie that has at least one rating and also find the highest number of stars that movie received. Sort the result by movie title.  vii. Update rating of all movies directed by ‘Lankesh’ to 3. | 2 |
| 4 | Consider the schema for College Database:  STUDENT(USN, SName, Address, Phone, Gender)  SEMSEC(SSID, Sem, Sec)  CLASS(USN, SSID)  COURSE(Subcode, Title, Sem, Credits)  IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)  Write SQL queries to  i. List all the student details studying in the fourth semester ‘C’ section.  ii. Compute the total number of male and female students in each semester and in each section.  iii. Create a view of Test1 marks of student with USN ‘1DS18IS101’ in all  Courses.  iv. Calculate the FinalIA (average of best two test marks) and update the  corresponding table for all students.  v. 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. | 2 |
| 5 | Consider the following database for a banking enterprise  BRANCH(branch-name:string, branch-city:string, assets:real)  ACCOUNT(accno:int, branch-name:string, balance:real)  DEPOSITOR(customer-name:string, accno:int)  CUSTOMER(customer-name:string, customer-street:string,  customer-city:string)  LOAN(loan-number:int, branch-name:string, amount:real)  BORROWER(customer-name:string, loan-number:int)  Write each of the following queries in SQL.  i. Create the above tables by properly specifying the primary keys and the  foreign keys  ii. Enter at least five tuples for each relation  iii. Find all the customers who have at least two accounts at the Main branch.  iv. Find all the customers who have an account at all the branches located in a specific city.  v. Demonstrate how you delete all account tuples at every branch located in a specific city.  vi. Find the names of all depositors of a specific branch.  vii. Find the details of all loan holders of a specific branch. | 2 |
| 6 | **Part B: Mini project**  Each student has to carry out a mini project on the problem identified individually or in a  group. For the problem identified :  1) List the set of requirements  2) Design an ER Diagram by identifying the following:  i. Entities(Minimum 5) and attributes  ii. Relationships and key for each entity  iii. Relationship Constraints: Cardinality Ratio and Participation.  3) Draw the Schema Diagram with Referential Integrity Constraints displayed.  4) Normalize the relations up to BCNF or 3rd Normal Form.  5) Create the database.  6) Insert suitable records (at least 5 records for each table) in the database.  7) Execute any two distinctive queries on the database.  8) Create and execute at least one trigger on the database.  9) Create and execute at least one stored procedure on the database.  10) Generate at least one typical report on the database.  The code developed during the project will be reviewed by internal faculties during the  semester. At the completion of a project the student will submit a project report, which will be evaluated by duly appointed examiner(s). |  |

**BASIC CONCEPTS OF DBMS**



## 1. Data Definition Language (DDL)

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

## 2. Data Manipulation Language (DML)

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

## 3. Data Control Language (DCL)

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

## 4. Transaction Control Language (TCL)

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

#### ****Introduction to SQL****

SQL stands for “Structured Query Language” and can be pronounced as “SQL” or “sequel – (Structured English Query Language)”. It is a query language used for accessing and modifying information in the database. IBM first developed SQL in 1970s. Also it is an ANSI/ISO standard. It has become a Standard Universal Language used by most of the relational database management systems (RDBMS). Some of the RDBMS systems are: Oracle, Microsoft SQL server, Sybase etc. Most of these have provided their own implementation thus enhancing its feature and making it a powerful tool. Few of the SQL commands used in SQL programming are SELECT Statement, UPDATE Statement, INSERT INTO Statement, DELETE Statement, WHERE Clause, ORDER BY Clause, GROUP BY Clause, ORDER Clause, Joins, Views, GROUP Functions, Indexes etc.

#### SQL Commands

SQL commands are instructions used to communicate with the database to perform specific task that work with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users.

**CREATE TABLE Statement**

The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key and foreign key can be defined for the columns while creating the table. The integrity constraints can be defined at column level or table level. The implementation and the syntax of the CREATE Statements differs for different RDBMS.

**The Syntax for the CREATE TABLE Statement is:**

|  |  |  |  |
| --- | --- | --- | --- |
| CREATE TABLE  table\_name | |  | |
| (column\_name1 datatype constraint, | | | |
| column\_name2 datatype,... | | |  |
| column\_nameNdatatype); |  | | |

**SQL Data Types:**

|  |  |
| --- | --- |
| char(size) | Fixed-length character string. Size is specified in parenthesis. Max 255 bytes. |
| Varchar2(siz e) | Variable-length character string. Max size is specified in parenthesis. |
| number(size  )  or int | Number value with a max number of column digits specified in parenthesis. |
| Date | Date value in „dd-mon-yy‟. Eg., ‟07-jul-2004‟ |
| number(size, d) or real | Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal. |

#### SQL Integrity Constraints:

Integrity Constraints are used to apply business rules for the database tables.The constraints available in SQL are **Foreign Key, Primary key, Not Null, Unique, Check.**

Constraints can be defined in two ways:

1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table- level definition.
   1. **Primary key:**

This constraint defines a column or combination of columns which uniquely identifies each row in the table.

**Syntax to define a Primary key at column level:**

Column\_namedatatype [CONSTRAINT constraint\_name] PRIMARY KEY

**Syntax to define a Primary key at table level:**

column\_name2,..)

[CONSTRAINT constraint\_name] PRIMARY KEY(column\_name1,

* 1. **Foreign key or Referential Integrity:**

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as foreign key.

**Syntax to define a Foreign key at column level:**

[CONSTRAINT constraint\_name] REFERENCES

referenced\_table\_name(column\_name)

* 1. **Not Null Constraint:**

This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

**Syntax to define a Not Null constraint:**

[CONSTRAINT constraint name] NOT NULL

* 1. **Unique Key:**

This constraint ensures that a column or a group of columns in each row have a distinct value.

A column(s) can have a null value but the values cannot be duplicated.

**Syntax to define a Unique key at column level:**

[CONSTRAINT constraint\_name] UNIQUE

**Syntax to define a Unique key at table level:**

[CONSTRAINT constraint\_name] UNIQUE(column\_name)

* 1. **Check Constraint:**

This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

**Syntax to define a Check constraint:**

[CONSTRAINT constraint\_name] CHECK (condition)

**ALTER TABLE Statement**

The SQL ALTER TABLE command is used to modify the definition structure) of a table by modifying the definition of its columns. The ALTER command is used to perform the following functions.

* + 1. Add, drop, modify table columns
    2. Add and drop constraints
    3. Enable and Disable constraints

**The HAVING clause**

The HAVING clause can be used to restrict the display of grouped rows. The result of the grouped query is passed on to the HAVING clause for output filtration.

**The INSERT INTO Statement**

The INSERT INTO statement is used to insert a new row in a table.

**The UPDATE Statement**

The UPDATE statement is used to update existing records in a table.

**The DELETE Statement**

The DELETE statement is used to delete rows in a table. SQL DELETE

**Commit command**

Commit command is used to permanently save any transaaction into database

#### Rollback command

This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction.

#### Savepoint command

**savepoint** command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

**Examples**

CREATE TABLE Employee(  
    EmployeeID int Primary Key,  
    LastName varchar(255),  
    FirstName varchar(255) Not Null,  
    Address varchar(255),  
    City varchar(255)  
);

CREATE TABLE Department(EmployeeID References Employee(EmployeeID),

DeptID int primary key,

DeptName varchar(20) Not Null,

DeptBranch varchar(20));

Insert Into Tablename values(12,’East Boston’,’Boston’);

ALTER TABLE

ALTER TABLE Employee  
ADD Email varchar(255);

ALTER TABLE Customers  
DROP COLUMN Email;

DROP TABLE

DROP TABLE Tablename;

TRUNCATE TABLE

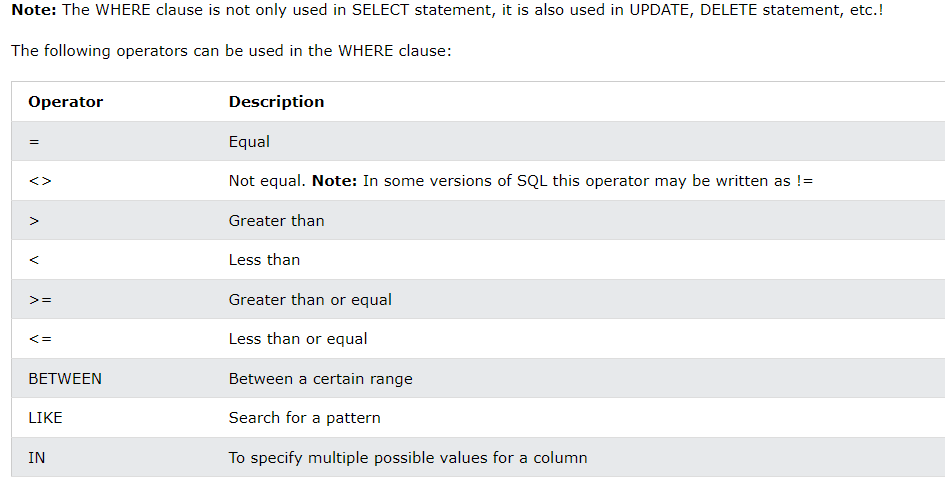
TRUNCATE TABLE Tablename;

Select Statements

SELECT TOP 3 \* FROM Customers;

SELECT \* FROM Customers  
WHERE Country='Mexico';

SELECT \* FROM Customers  
WHERE CustomerID=1;



**Lab Program 1:**

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

i. Create the above tables by properly specifying the primary keys and the foreign keys.

ii. Enter at least five tuples for each relation.

iii. Retrieve details of all books in the library – book-id, title, publisher name,

authors, number of copies, etc.

iv. Get the details of borrowers who have borrowed more than 3 book from Jan

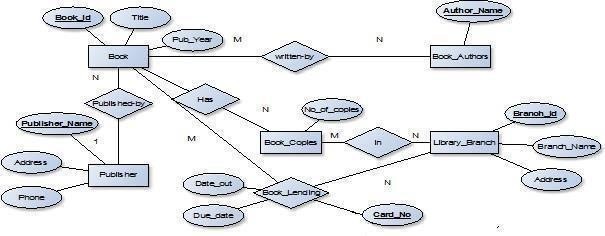
2024 to may 2024

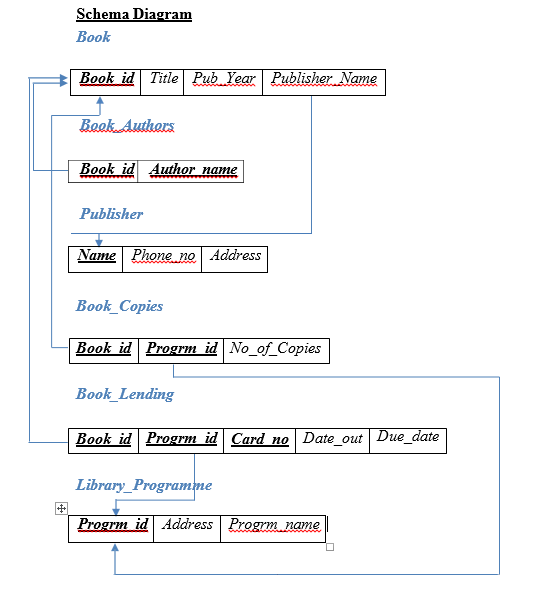
v. Demonstrate the DELETE operation by deleting a book details in BOOK

table.

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

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





**Table Creation**

CREATE TABLE PUBLISHER

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

ADDRESS VARCHAR (20));

CREATE TABLE BOOK

(BOOK\_ID INTEGER 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 INTEGER,

FOREIGN KEY(BOOK\_ID) REFERENCES BOOK(BOOK\_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK\_ID, AUTHOR\_NAME));

CREATE TABLE LIBRARY\_PROGRAMME (PROGRAMME\_ID INTEGER PRIMARY KEY, PROGRAMME\_NAME VARCHAR(50), ADDRESS VARCHAR2 (50));

CREATE TABLE BOOK\_COPIES(NO\_OF\_COPIES INTEGER, BOOK\_ID INTEGER ,

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

FOREIGN KEY(PROGRAMME\_ID) REFERENCES LIBRARY\_ PROGRAMME (PROGRAMME \_ID) ON DELETECASCADE,

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

CREATE TABLE CARD

(CARD\_NO INTEGER PRIMARY KEY);

CREATE TABLE BOOK\_LENDING (DATE\_OUT DATE,

DUE\_DATE DATE, BOOK\_ID INTEGER,

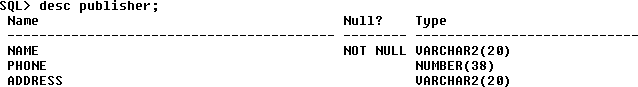
FOREIGN KEY(BOOK\_ID) REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE,

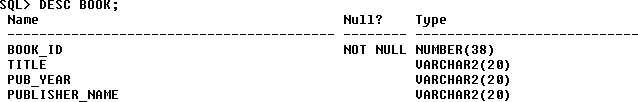
PROGRAMME\_ID INTEGER,

FOREIGN KEY(PROGRAMME\_ID) REFERENCESLIBRARY\_ PROGRAMME (PROGRAMME \_ID) ON DELETECASCADE,

CARD\_NO INTEGER);

**Table Descriptions**

DESC PUBLISHER

DESC BOOK;

**Insertion of Values to Tables**

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

INSERT INTO PUBLISHER VALUES (‘RANDOM HOUSE’, 7455679345, ‘HYDRABAD’);

INSERT INTO PUBLISHER VALUES (‘HACHETTE LIVRE’, 8970862340, ‘CHENAI’);

INSERT INTO PUBLISHER VALUES (‘GRUPO PLANETA’, 7756120238, ‘BANGALORE’);

INSERT INTO PUBLISHER VALUES (

'WILEY','PUNE','9856784563');

INSERT INTO PUBLISHER VALUES (

'MONGEY WILL','MEXICO','8836786577');

INSERT INTO BOOK VALUES (1,’DBMS’,’JAN-2024’, ‘MCGRAW-HILL’);

INSERT INTO BOOK VALUES (2,’ADBMS’,’JUN-2023’, ‘MCGRAW-HILL’);

INSERT INTO BOOK VALUES (3,’CN’,’SEP-2024’, ‘PEARSON’);

INSERT INTO BOOK VALUES (4,’CG’,’SEP-2023’, ‘GRUPO PLANETA’);

INSERT INTO BOOK VALUES (5,’OS’,’MAY-2024’, ‘PEARSON’);

INSERT INTO BOOK\_AUTHORS VALUES (’NAVATHE’, 1);

INSERT INTO BOOK\_AUTHORS VALUES (’NAVATHE’, 2);

INSERT INTO BOOK\_AUTHORS VALUES (’TANENBAUM’, 3);

INSERT INTO BOOK\_AUTHORS VALUES (’EDWARD ANGEL’, 4);

INSERT INTO BOOK\_AUTHORS VALUES (’GALVIN’, 5);

INSERT INTO LIBRARY\_ PROGRAMME VALUES(10,’RRNAGAR’,’BANGALORE’);

INSERT INTO LIBRARY\_ PROGRAMME VALUES (11,’APSCE’,’BANGALORE’); INSERT INTO LIBRARY\_ PROGRAMME VALUES (12,’N R COLONY’, ’BANGALORE’);

INSERT INTO LIBRARY\_ PROGRAMME VALUES (13,’NITTE’,’MANGALORE’); INSERT INTO LIBRARY\_ PROGRAMME VALUES (14,’MANIPAL’,’UDUPI’);

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 (’01-JAN-23’,’01-JUN-23’, 1, 10, 101);

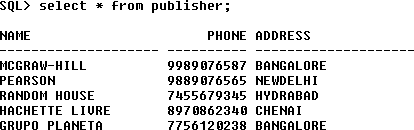
INSERT INTO BOOK\_LENDING VALUES (’11-JAN-23’,’11-MAR-23’, 3, 14, 101);

INSERT INTO BOOK\_LENDING VALUES (’21-FEB-24’,’21-APR-24’, 2, 13, 101);

INSERT INTO BOOK\_LENDING VALUES (’15-MAR-23’,’15-JUL-23’, 4, 11, 101);

INSERT INTO BOOK\_LENDING VALUES (‘12-APR-23’,’12-MAY-23’, 1, 11, 104);

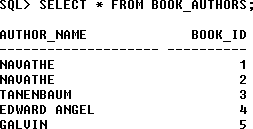
SELECT \* FROM PUBLISHER;



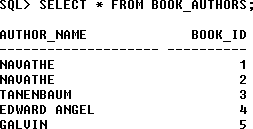
SELECT \* FROM BOOK;



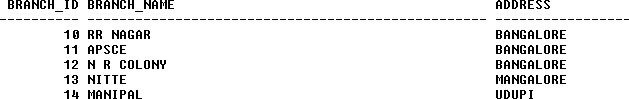
SELECT \* FROM BOOK\_AUTHORS;



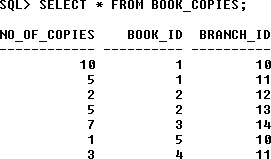
SELECT \* FROM BOOK\_AUTHORS;



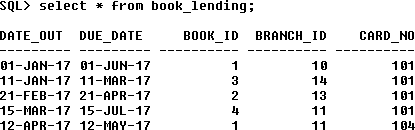
SELECT \* FROM LIBRARY\_BRANCH;



SELECT \* FROM BOOK\_COPIES;



SELECT \* FROM BOOK\_LENDING;



**Queries:**

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

//Here We are relating tables BOOK , BOOK\_AUTHORS, BOOK\_COPIES and LIBRARY\_PROGRAMME

# 2.Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2024 to Jun 2024.

SELECT CARD\_NO FROM BOOK\_LENDING

WHERE DATE\_OUT BETWEEN '01-JAN-2023' AND '01-JUL-2023' GROUP BY CARD\_NO

HAVING COUNT (\*)>3;

In Oracle **GROUP BY** clause is used with SELECT statement to collect data from multiple records and group the results by one or more columns.

**Syntax:**

**SELECT** expression1, expression2, ... expression\_n,

aggregate\_function (aggregate\_expression)

**FROM** tables

**WHERE** conditions

**GROUP** **BY** expression1, expression2, ... expression\_n;

## Parameters:

**expression1, expression2, ... expression\_n:** It specifies the expressions that are not encapsulated within aggregate function. These expressions must be included in GROUP BY clause.

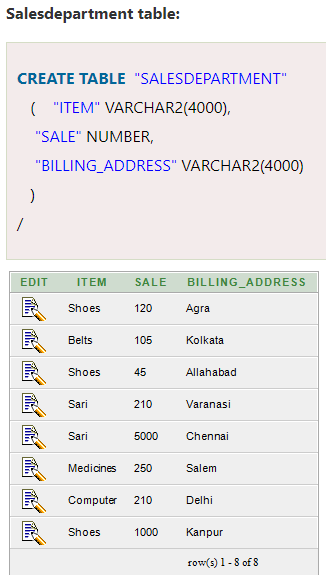
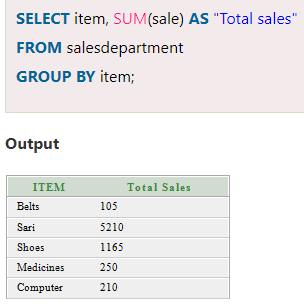
**aggregate\_function:** It specifies the aggregate functions i.e. SUM, COUNT, MIN, MAX or AVG functions.

**aggregate\_expression:** It specifies the column or expression on that the aggregate function is based on.

**tables:** It specifies the table from where you want to retrieve records.

**conditions:** It specifies the conditions that must be fulfilled for the record to be selected.

Example

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

DELETE FROM BOOK WHERE BOOK\_ID=3;

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

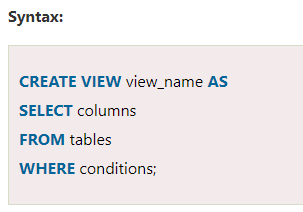
CREATE VIEW V\_PUBLICATION AS SELECT PUB\_YEAR

FROM BOOK;

**Views**

In Oracle, view is a virtual table that does not physically exist. It is stored in Oracle data dictionary and do not store any data. It can be executed when called.

A view is created by a query joining one or more tables.



Example :

**CREATE** **TABLE**  "SUPPLIERS"

    (    "SUPPLIER\_ID" NUMBER,

     "SUPPLIER\_NAME" VARCHAR2(4000),

     "SUPPLIER\_ADDRESS" VARCHAR2(4000)

    )

CREATE VIEW SUPPLIERS\_VIEW AS SELECT SUPPLIER\_ID, SUPPLIER\_NAME FROM SUPPLIERS

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

CREATE VIEW V\_BOOKS AS

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

LIBRARY\_PROGRAMME L

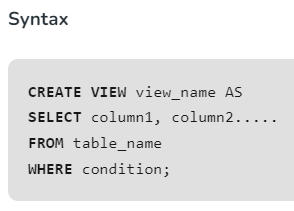
WHERE B.BOOK\_ID=C.BOOK\_ID AND

C.PROGRAMME\_ID=L.PROGRAMME\_ID;

**Views**

**Views** are a kind of virtual table. A view also has rows and columns like tables, but a view doesn’t store data on the disk like a table. View defines a customized query that retrieves data from one or more tables, and represents the data as if it was coming from a single source.

Views can be created using one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain conditions



## ****Uses of a View****

A good database should contain views for the given reasons:

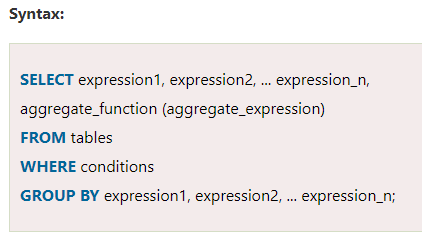
* **Restricting data access –** Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.
* **Hiding data complexity –** A view can hide the complexity that exists in multiple joined tables.
* **Simplify commands for the user –** Views allow the user to select information from multiple tables without requiring the users to actually know how to perform a join.
* **Store complex queries –** Views can be used to store complex queries.
* **Rename Columns –** Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in a select statement. Thus, renaming helps to hide the names of the columns of the base tables.
* **Multiple view facility –** Different views can be created on the same table for different users.

## Key differences between Table and View

* A table is a database object that holds information used in applications and reports. On the other hand, a view is also a database object utilized as a table and can also link to other tables.
* A table consists of rows and columns to store and organized data in a structured format, while the view is a result set of SQL statements.
* A table is **structured** with columns and rows, while a view is a virtual table **extracted** from a database.
* The table is an independent data object while views are usually depending on the table.
* The table is an **actual or real table** that exists in physical locations. On the other hand, views are the **virtual or logical table** that does not exist in any physical location.
* A table allows to performs add, update or delete operations on the stored data. On the other hand, we cannot perform add, update, or delete operations on any data from a view. If we want to make any changes in a view, we need to update the data in the source tables.
* We cannot **replace** the table object directly because it is stored as a physical entry. In contrast, we can easily use the replace option to recreate the view because it is a pseudo name to the SQL statement running behind on the database server.

# GROUP BY Clause

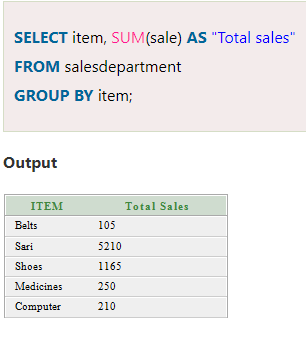
In Oracle GROUP BY clause is used with SELECT statement to collect data from multiple records and group the results by one or more columns.



## Parameters:

**expression1, expression2, ... expression\_n:** It specifies the expressions that are not encapsulated within aggregate function. These expressions must be included in GROUP BY clause.

**aggregate\_function:** It specifies the aggregate functions i.e. SUM, COUNT, MIN, MAX or AVG functions.



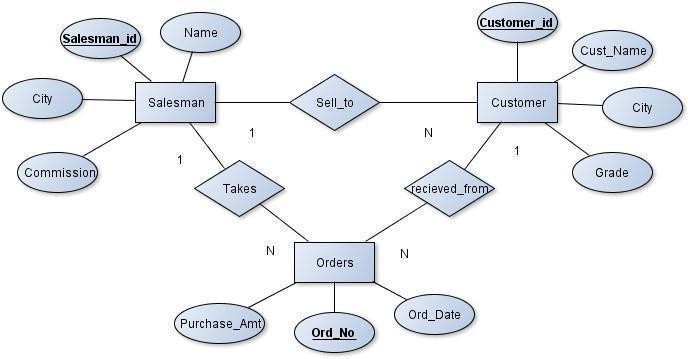
**Lab Program 2 : SALES Management**

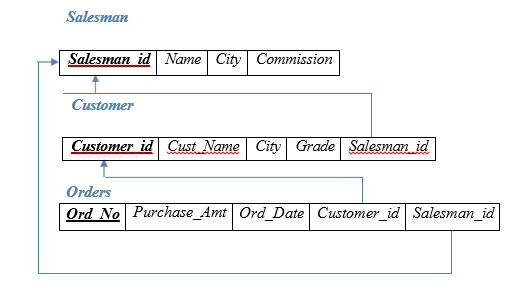
**SALESMAN (*Salesman\_id, Name, City, Commission*) CUSTOMER (*Customer\_id, Cust\_Name, City, Grade, Salesman\_id*)**

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

1. **Count the customers with grades above Bangalore’s average.**
2. **Find the name and numbers of all salesmen who had more than one customer.**
3. **List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)**
4. **Find all orders with above average amount. Return ord\_no, purch\_amt, ord\_date, customer\_id and salesman\_id.**
5. **Create a view that finds the salesman who has the customer with the highest order of a day.**
6. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

**Entity-Relationship 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 CUSTOMER (CUSTOMER\_ID NUMBER (4),

CUST\_NAME VARCHAR2 (20),

CITY VARCHAR2 (20), GRADE NUMBER (3),

PRIMARY KEY (CUSTOMER\_ID), SALESMAN\_ID INTEGER,

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,

PRIMARY KEY (ORD\_NO),CUSTOMER\_ID INTEGER,

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

SALESMAN\_ID INTEGER,

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

**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-24’, 10, 1000);

INSERT INTO ORDERS VALUES (51, 450, ‘20-JAN-24’, 10, 2000);

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

INSERT INTO ORDERS VALUES (53, 3500, ‘13-APR-24’, 14, 3000);

INSERT INTO ORDERS VALUES (54, 550, ‘09-MAR-24’, 12, 2000);

**Queries:**

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

GROUP BY GRADE

HAVING GRADE > (SELECT AVG(GRADE) FROM CUSTOMER1

WHERE CITY='BANGALORE');



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

-- Selecting the 'salesman\_id' and 'name' columns from the 'salesman' table (aliased as 'a')

SELECT salesman\_id, name

-- Specifying the table to retrieve data from ('salesman' as 'a')

FROM salesman a

-- Filtering the results based on the condition that the count of customers associated with the salesman is greater than 1

WHERE 1 <

(SELECT COUNT(\*)

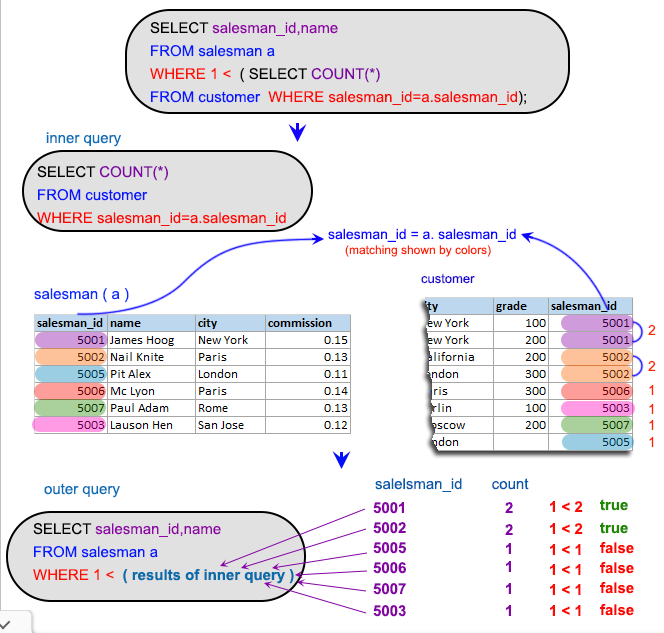
-- Subquery: Counting the number of rows in the 'customer' table where 'salesman\_id' matches the outer query's 'salesman\_id'

FROM customer

WHERE salesman\_id = a.salesman\_id);

**Explanation:**

In the said SQL query the subquery in the WHERE clause counts the number of rows in the 'customer' table where the salesman\_id matches the salesman\_id in the 'salesman' table. The outer query then selects the salesman\_id and name columns from the 'salesman' table where the count returned by the subquery is greater than 1. This results in a list of salesman\_id and name of all salesmen who have more than one customer.



1. **List all salesmen and indicate those who have and don’t have customers in their**

**cities (Use UNION operation.)**

-- Selecting columns (salesman.salesman\_id, name, cust\_name, commission) from tables 'salesman' and 'customer'

SELECT salesman.salesman\_id, name, cust\_name, commission

FROM salesman, customer

-- Filtering rows where the city in the 'salesman' table matches the city in the 'customer' table

WHERE salesman.city = customer.city

-- Performing a UNION operation with the result set of a subquery

UNION

-- Selecting specific columns (salesman\_id, name, 'NO MATCH', commission) from the 'salesman' table

SELECT salesman\_id, name, 'NO MATCH', commission

FROM salesman

-- Filtering rows where the city in the 'salesman' table is not equal to any city in the 'customer' table

WHERE NOT city = ANY

(SELECT city

FROM customer)

-- Ordering the result set based on the second column (name) in descending order

ORDER BY 2 DESC

**Explanation:**

The said query in SQL that retrieves data from the 'salesman' and 'customer' tables, and outputs a list of salespeople and their commissions based on the city where they are located and the city of their customers.  
The query first joins the 'salesman' and 'customer' tables based on the condition that the salesman's city matches the customer's city. It then uses the UNION operator to combine this result with another query that selects salespeople who do not have any customers in any city. For these salespeople, the query outputs "NO MATCH" instead of the customer name.  
The ORDER BY clause sorted the output by the second column in descending order

# 4. Find all orders with above average amount. Return ord\_no, purch\_amt, ord\_date, customer\_id and salesman\_id.

-- Selecting all columns from the 'orders' table (aliased as 'a')

SELECT \*

-- Specifying the table to retrieve data from ('orders' as 'a')

FROM orders a

-- Filtering the results based on the condition that 'purch\_amt' is greater than the average 'purch\_amt' for the same 'customer\_id'

WHERE purch\_amt >

-- Subquery: Calculating the average 'purch\_amt' from the 'orders' table (aliased as 'b') for the same 'customer\_id'

(SELECT AVG(purch\_amt) FROM orders b

WHERE b.customer\_id = a.customer\_id);

**Explanation:**

The said SQL query is selecting all columns from the 'orders' table where the value in the "purch\_amt" column is greater than the average "purch\_amt" of orders associated with the same customer.  
It is using a subquery in the WHERE clause to calculate the average "purch\_amt" of orders for each customer by matching the "customer\_id" in the 'orders' table with the "customer\_id" in the 'orders' table in the subquery. The outer query then selects all columns from the 'orders' table where the value in the "purch\_amt" column is greater than the average "purch\_amt" returned by the subquery for that specific customer.  
This results in a list of all orders with a purchase amount greater than the average purchase amount of all orders for that specific customer.

**5.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);

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

DELETE FROM SALESMAN WHERE SALESMAN\_ID=1000

# Lab Program 3 :

## Consider the schema for Movie Database:

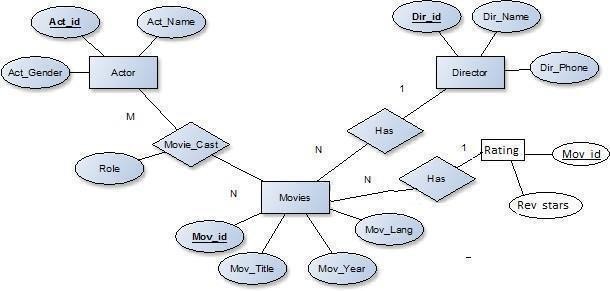
ACTOR (*Act\_id, Act\_Name, Act\_Gender*) DIRECTOR (*Dir\_id, Dir\_Name, Dir\_Phone*)

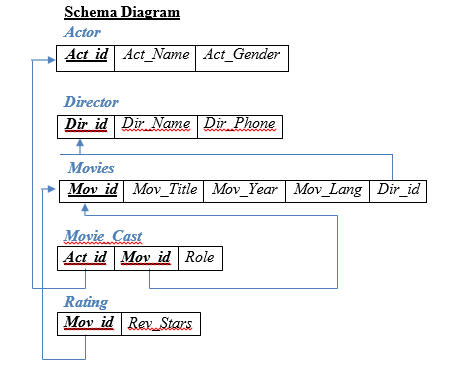
MOVIES (*Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id*) MOVIE\_CAST (*Act\_id, Mov\_id, Role*)

*RATING (Mov\_id, Rev\_Stars)*

*Write SQL queries to*

* 1. List the titles of all movies directed by ‘Hitchcock’.
  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 movie title.
  5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

Entity-Relationship Diagram



**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), DIR\_ID INTEGER,

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));

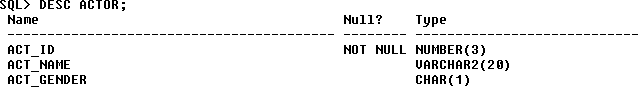
CREATE TABLE RATING ( MOV\_ID NUMBER (4),

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

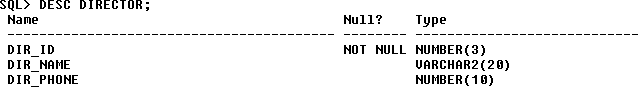
FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

**Table Descriptions**

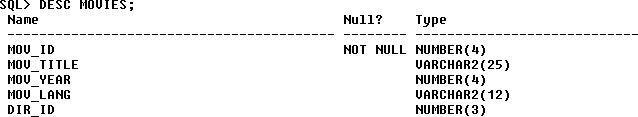
DESC ACTOR;



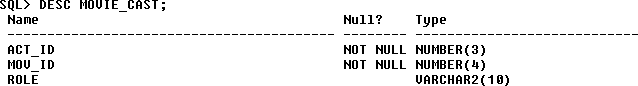
DESC DIRECTOR;



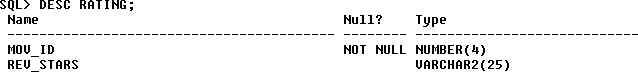
DESC MOVIES;



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’);

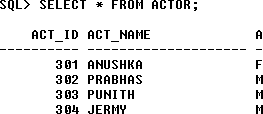
INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

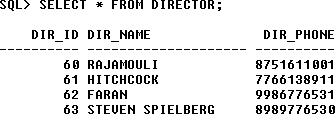
INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

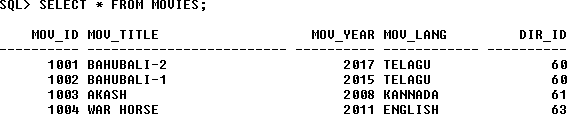
SELECT \* FROM ACTOR;



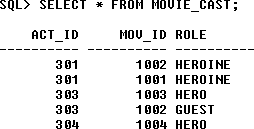
SELECT \* FROM DIRECTOR;



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’)

# 2.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;

# 

## 3.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

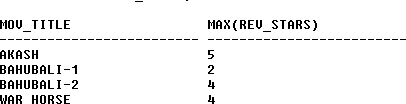
# AND C.MOV\_YEAR NOT BETWEEN 2000 AND 2015;

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

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

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

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



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



**Lab Program 4:**

Consider the schema for College Database:

STUDENT(USN, SName, Address, Phone, Gender)

SEMSEC(SSID, Sem, Sec)

CLASS(USN, SSID)

COURSE(Subcode, Title, Sem, Credits)

IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

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

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

iii. Create a view of Test1 marks of student with USN ‘1DS22ICD70’ in all

Courses.

iv. Calculate the FinalIA (average of best two test marks) and update the

corresponding table for all students.

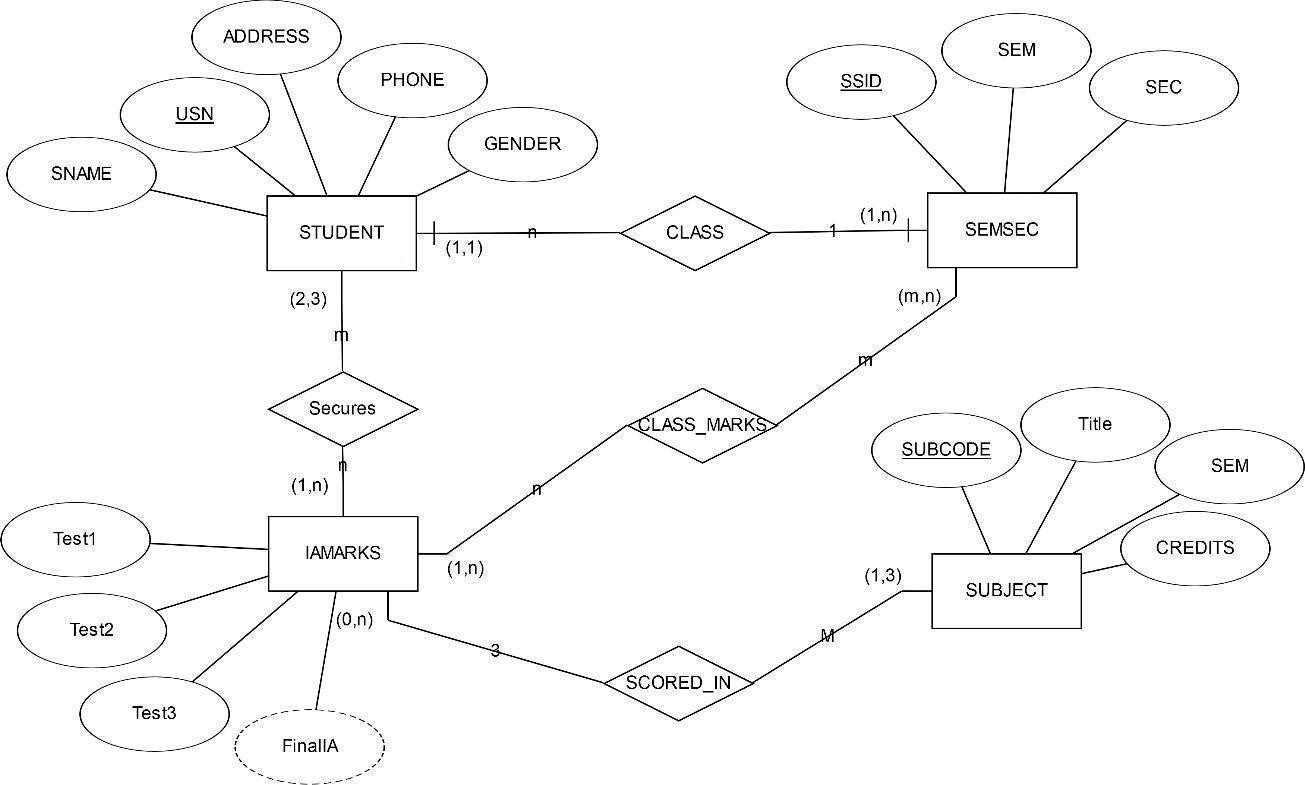
v. Categorize students based on the following criterion:

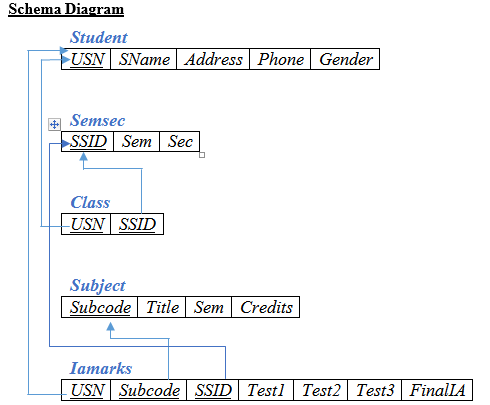
If FinalIA = 17 to 20 then CAT = ‘Outstanding’

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.

**Entity - Relationship 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),

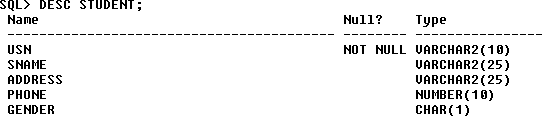
PRIMARY KEY (USN, SUBCODE, SSID),

FOREIGN KEY (USN) REFERENCES STUDENT (USN),

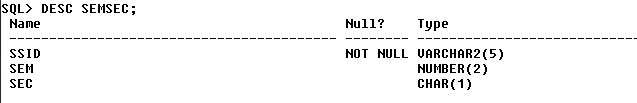
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE), FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));

**Table Descriptions**

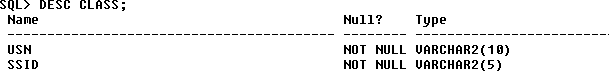
DESC STUDENT;



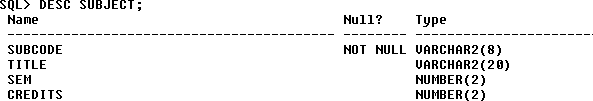
DESC SEMSEC;



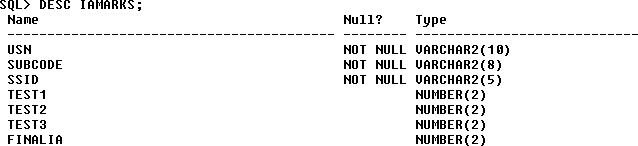
DESC CLASS;



DESC SUBJECT;



DESC IAMARKS;



**Insertion of values to tables**

INSERT INTO STUDENT VALUES ('1AP13CS020','AKSHAY','BELAGAVI', 8877881122,'M');

INSERT INTO STUDENT VALUES ('1AP13CS062','SANDHYA','BENGALURU', 7722829912,'F');

INSERT INTO STUDENT VALUES ('1AP13CS091','TEESHA','BENGALURU', 7712312312,'F');

INSERT INTO STUDENT VALUES ('1AP13CS066','SUPRIYA','MANGALURU', 8877881122,'F');

INSERT INTO STUDENTVALUES ('1AP14CS010','ABHAY','BENGALURU', 9900211201,'M');

INSERT INTO STUDENT VALUES ('1AP14CS032','BHASKAR','BENGALURU', 9923211099,'M');

INSERT INTO STUDENTVALUES ('1 AP14CS025','ASMI','BENGALURU', 7894737377,'F'); INSERT INTO STUDENT VALUES ('1AP15CS011','AJAY','TUMKUR', 9845091341,'M'); INSERT INTO STUDENT VALUES ('1AP15CS029','CHITRA','DAVANGERE', 7696772121,'F');

INSERT INTO STUDENT VALUES ('1AP15CS045','JEEVA','BELLARY', 9944850121,'M'); INSERT INTO STUDENT VALUES ('1AP15CS091','SANTOSH','MANGALURU', 8812332201,'M');

INSERT INTO STUDENT VALUES ('1AP16CS045','ISMAIL','KALBURGI', 9900232201,'M');

INSERT INTO STUDENT VALUES ('1AP16CS088','SAMEERA','SHIMOGA', 9905542212,'F');

INSERT INTO STUDENT VALUES ('1AP16CS122','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 (‘1AP13CS020’,’CSE8A’); INSERT INTO CLASS VALUES (‘1AP13CS062’,’CSE8A’); INSERT INTO CLASS VALUES (‘1AP13CS066’,’CSE8B’); INSERT INTO CLASS VALUES (‘1AP13CS091’,’CSE8C’);

INSERT INTO CLASS VALUES (‘1AP14CS010’,’CSE7A’); INSERT INTO CLASS VALUES (‘1AP14CS025’,’CSE7A’); INSERT INTO CLASS VALUES (‘1AP14CS032’,’CSE7A’);

INSERT INTO CLASS VALUES (‘1AP15CS011’,’CSE4A’); INSERT INTO CLASS VALUES (‘1AP15CS029’,’CSE4A’); INSERT INTO CLASS VALUES (‘1AP15CS045’,’CSE4B’); INSERT INTO CLASS VALUES (‘1AP15CS091’,’CSE4C’);

INSERT INTO CLASS VALUES (‘1AP16CS045’,’CSE3A’); INSERT INTO CLASS VALUES (‘1AP16CS088’,’CSE3B’); INSERT INTO CLASS VALUES (‘1AP16CS122’,’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 ('1AP13CS091','10CS81','CSE8C', 15, 16, 18);

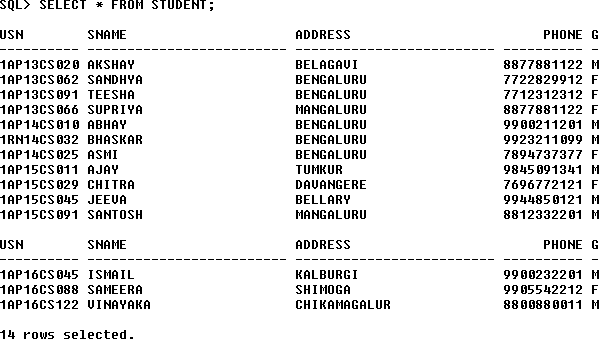
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1AP13CS091','10CS82','CSE8C', 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1AP13CS091','10CS83','CSE8C', 19, 15, 20);

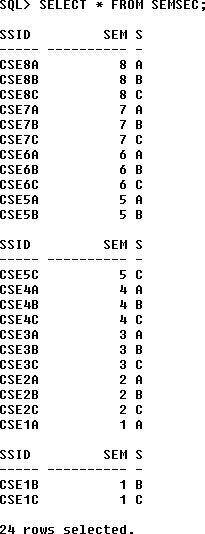
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1AP13CS091','10CS84','CSE8C', 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1AP13CS091','10CS85','CSE8C', 15, 15, 12);

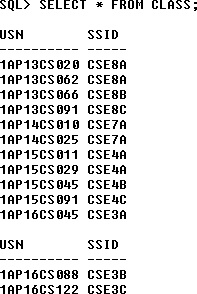
SELECT \* FROM STUDENT;



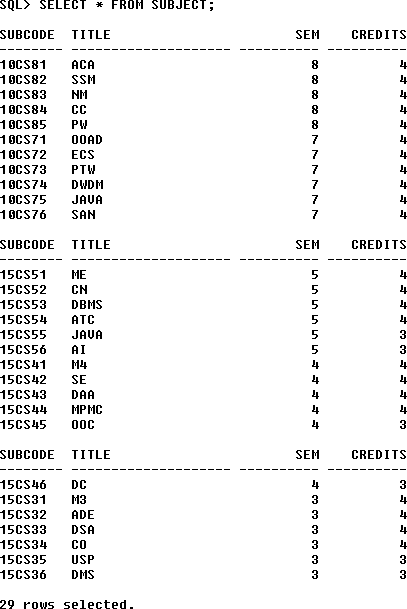
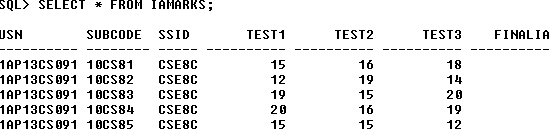
SELECT \* FROM SEMSEC;



SELECT \* FROM CLASS;



SELECT \* FROM SUBJECT;



**Queries:**

## 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’;

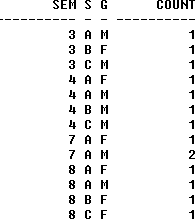


## Compute the total number of male and female students in each semester and in each section.

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;

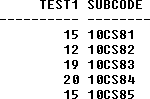


## Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.

CREATE VIEW STU\_TEST1\_MARKS\_VIEW AS

SELECT TEST1, SUBCODE FROM IAMARKS

WHERE USN = '1RN13CS09



## 4.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);

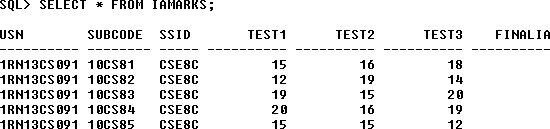
--DBMS\_OUTPUT.PUT\_LINE('AVERAGE = '||C\_AV);

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

CLOSE C\_IAMARKS; END;

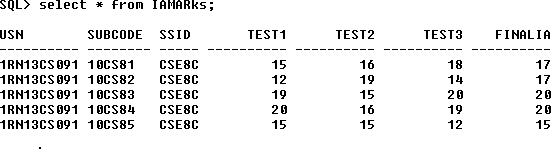
/

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



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

BEGIN AVGMARKS; END;



## 5.Categorize students based on the following criterion:

## If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’

## If FinalIA< 12 then CAT = ‘Weak’

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

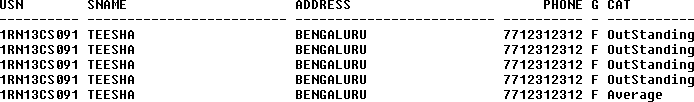
## SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, (CASE

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

## END) AS CAT

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

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



**Lab Program 5:**

Consider the following database for a banking enterprise

BRANCH(branch-name:string, branch-city:string, assets:real)

ACCOUNT(accno:int, branch-name:string, balance:real)

DEPOSITOR(customer-name:string, accno:int)

CUSTOMER(customer-name:string, customer-street:string,

customer-city:string)

LOAN(loan-number:int, branch-name:string, amount:real)

BORROWER(customer-name:string, loan-number:int)

Write each of the following queries in SQL.

i. Create the above tables by properly specifying the primary keys and the

foreign keys

ii. Enter at least five tuples for each relation

iii. Find all the customers who have at least two accounts at the Main branch.

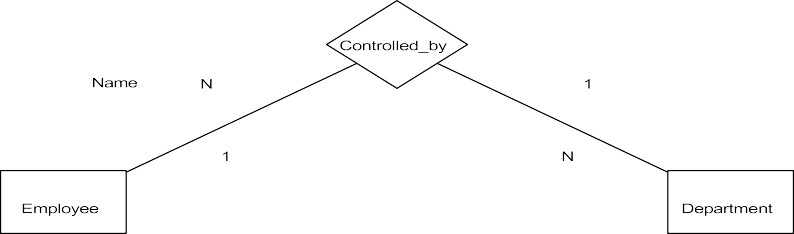
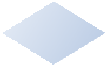
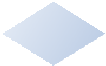
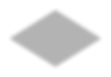
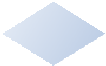
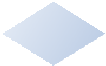
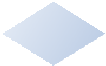
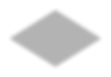
iv. Find all the customers who have an account at all the branches located in a specific city.

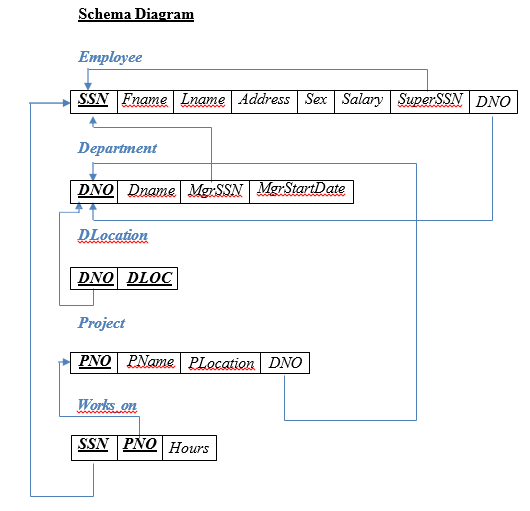
v. Demonstrate how you delete all account tuples at every branch located in a specific city.

vi. Find the names of all depositors of a specific branch.

vii. Find the details of all loan holders of a specific branch.

**Entity-Relationship Diagram**





**Table Creation**

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

CREATE TABLE EMPLOYEE

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

ADDRESS VARCHAR (20),

SEX CHAR (1), SALARY INTEGER, SUPERSSN VARCHAR (20),

FOREIGN KEY(SUPERSSN) REFERENCES EMPLOYEE (SSN),

DNO VARCHAR(20),

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 MGRSSN VARCHAR (20), FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE (SSN);

CREATE TABLE DLOCATION (DLOC VARCHAR (20),

DNO VARCHAR(20),

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

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

PLOCATION VARCHAR(20), DNO VARCHAR(20),

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

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

SSN VARCHAR(20),

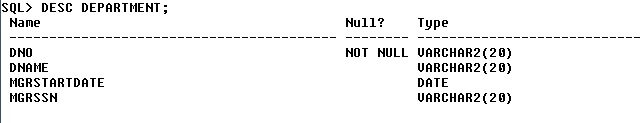
FOREIGN KEY(SSN) REFERENCES EMPLOYEE (SSN), PNO INTEGER,

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

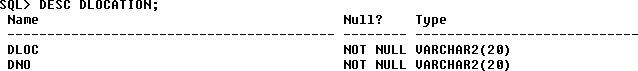
**Table Descriptions**

DESC EMPLOYEE;

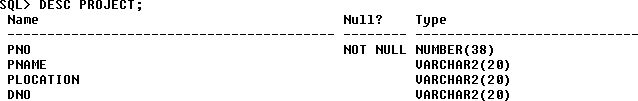
DESC DEPARTMENT;



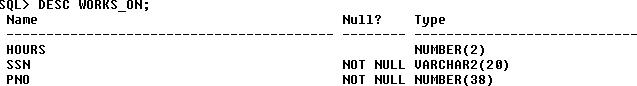
DESC DLOCATION;



DESC PROJECT;



DESC WORKS\_ON;



**Insertion of values to tables**

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSECE01’,’JOHN’,’SCOTT’,’BANGALORE’,’M’, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE01’,’JAMES’,’SMITH’,’BANGALORE’,’M’, 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE02’,’HEARN’,’BAKER’,’BANGALORE’,’M’, 700000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE03’,’EDWARD’,’SCOTT’,’MYSORE’,’M’, 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE04’,’PAVAN’,’HEGDE’,’MANGALORE’,’M’, 650000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE05’,’GIRISH’,’MALYA’,’MYSORE’,’M’, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSCSE06’,’NEHA’,’SN’,’BANGALORE’,’F’, 800000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSACC01’,’AHANA’,’K’,’MANGALORE’,’F’, 350000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSACC02’,’SANTHOSH’,’KUMAR’,’MANGALORE’,’M’, 300000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSISE01’,’VEENA’,’M’,’MYSORE’,’M’, 600000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES (‘APSIT01’,’NAGESH’,’HR’,’BANGALORE’,’M’, 500000);

INSERT INTO DEPARTMENT VALUES (‘1’,’ACCOUNTS’,’01-JAN-01’,’APSACC02’); INSERT INTO DEPARTMENT VALUES (‘2’,’IT’,’01-AUG-16’,’APSIT01’);

INSERT INTO DEPARTMENT VALUES (‘3’,’ECE’,’01-JUN-08’,’APSECE01’); INSERT INTO DEPARTMENT VALUES (‘4’,’ISE’,’01-AUG-15’,’APSISE01’); INSERT INTO DEPARTMENT VALUES (‘5’,’CSE’,’01-JUN-02’,’APSCSE05’);

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

UPDATE EMPLOYEE SET SUPERSSN=NULL, DNO=’3’ WHERE SSN=’APSECE01’;

UPDATE EMPLOYEE SET SUPERSSN=’APSCSE02’, DNO=’5’ WHERE SSN=’APSCSE01’;

UPDATE EMPLOYEE SET SUPERSSN=’APSCSE03’, DNO=’5’ WHERE SSN=’APSCSE02’;

UPDATE EMPLOYEE SET SUPERSSN=’APSCSE04’, DNO=’5’ WHERE SSN=’APSCSE03’;

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=’APSCSE05’ WHERE SSN=’APSCSE04’

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=’APSCSE06’ WHERE SSN=’APSCSE05’;

UPDATE EMPLOYEE SET DNO=’5’, SUPERSSN=NULL WHERE SSN=’APSCSE06’;

UPDATE EMPLOYEE SET DNO=’1’, SUPERSSN=’APSACC02’ WHERE SSN=’APSACC01’;

UPDATE EMPLOYEE SET DNO=’1’, SUPERSSN=NULL WHERE SSN=’APSACC02’;

UPDATE EMPLOYEE SET DNO=’4’, SUPERSSN=NULL WHERE SSN=’APSISE01’;

UPDATE EMPLOYEE SET DNO=’2’, SUPERSSN=NULL WHERE SSN=’APSIT01’;

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’);

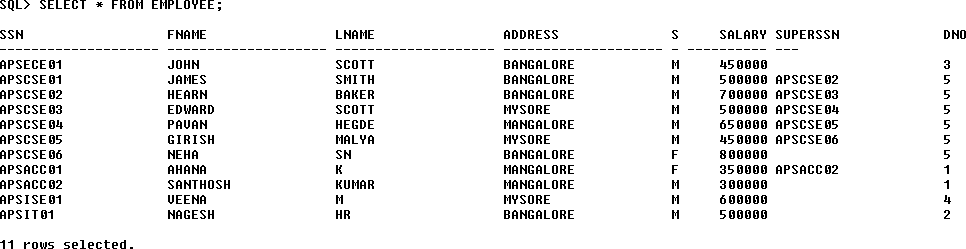
INSERT INTO PROJECT VALUES (100,’IOT’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (101,’CLOUD’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (102,’BIGDATA’,’BANGALORE’,’5’); INSERT INTO PROJECT VALUES (103,’SENSORS’,’BANGALORE’,’3’);

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’);

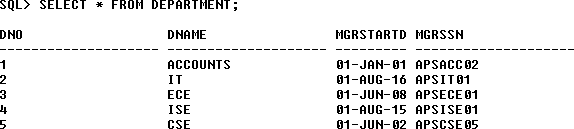
INSERT INTO PROJECT VALUES (107,’SMART CITY’,’BANGALORE’,’2’);

INSERT INTO WORKS\_ON VALUES (4, ‘APSCSE01’, 100); INSERT INTO WORKS\_ON VALUES (6, ‘APSCSE01’, 101); INSERT INTO WORKS\_ON VALUES (8, ‘APSCSE01’, 102); INSERT INTO WORKS\_ON VALUES (10, ‘APSCSE02’, 100); INSERT INTO WORKS\_ON VALUES (3, ‘APSCSE04’, 100); INSERT INTO WORKS\_ON VALUES (4, ‘APSCSE05’, 101); INSERT INTO WORKS\_ON VALUES (5, ‘APSCSE06’, 102); INSERT INTO WORKS\_ON VALUES (6, ‘APSCSE03’, 102); INSERT INTO WORKS\_ON VALUES (7, ‘APSECE01’, 103); INSERT INTO WORKS\_ON VALUES (5, ‘APSACC01’, 104); INSERT INTO WORKS\_ON VALUES (6, ‘APSACC02’, 105); INSERT INTO WORKS\_ON VALUES (4, ‘APSISE01’, 106); INSERT INTO WORKS\_ON VALUES (10, ‘APSIT01’, 107);

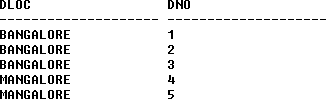
SELECT \* FROM EMPLOYEE;



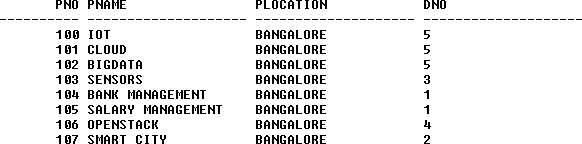
SELECT \* FROM DEPARTMENT;



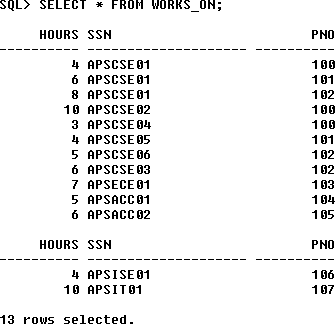
SELECT \* FROM DLOCATION;



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’);

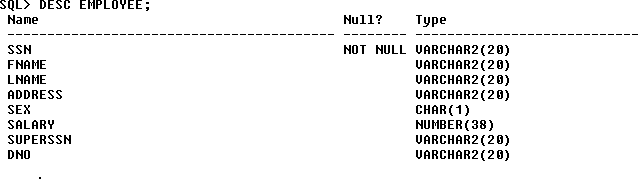


## 2.Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.

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’;





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

**SELECT SUM (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’;**

****

## 4.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));



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

SELECT 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**

## 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 integrityconstraints.
  + 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 onlypart of entire database.

## Define the "integrity rules"

There are two Integrityrules.

* + EntityIntegrity: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 entitytype in the database.

## What is an Extension of entity type?

The collections of entities of a particular entitytype 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 entitytypes.

## What is degree of Relationship type?

It is the number of entitytype 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 queryevaluation 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?

* + Everydependency 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-keyof R.
  + A is a prime attribute of R.

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

## 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 keyof 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.

## What are partial, alternate,, artificial, compound and natural key?

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

Alternate Key:

ArtificialKey:

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

If no obvious key, either stand alone or compound is available, then the last

resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

CompoundKey**:**If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

NaturalKey**:**When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

## 48.What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found.

* + Binarysearch style indexing
  + B-Tree indexing
  + Inverted list indexing
  + Memoryresident table
  + Table indexing