KISHKINDA UNIVERSITY, BALLARI

Established under the Karnataka Act No. 20 of 2023 | Recognised by Govt. of Karnataka & UGC, New Delhi

T.E.H.R.D TRUST

KISHKINDA UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY(FET)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE TITLE: DATABASE MANAGEMENT SYSTEMS
COURSE CODE: 23CS43

LAB MANNUAL

DBMS LABORATORY

IV SEM

PREPARED BY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KISHKINDA UNIVERSITY

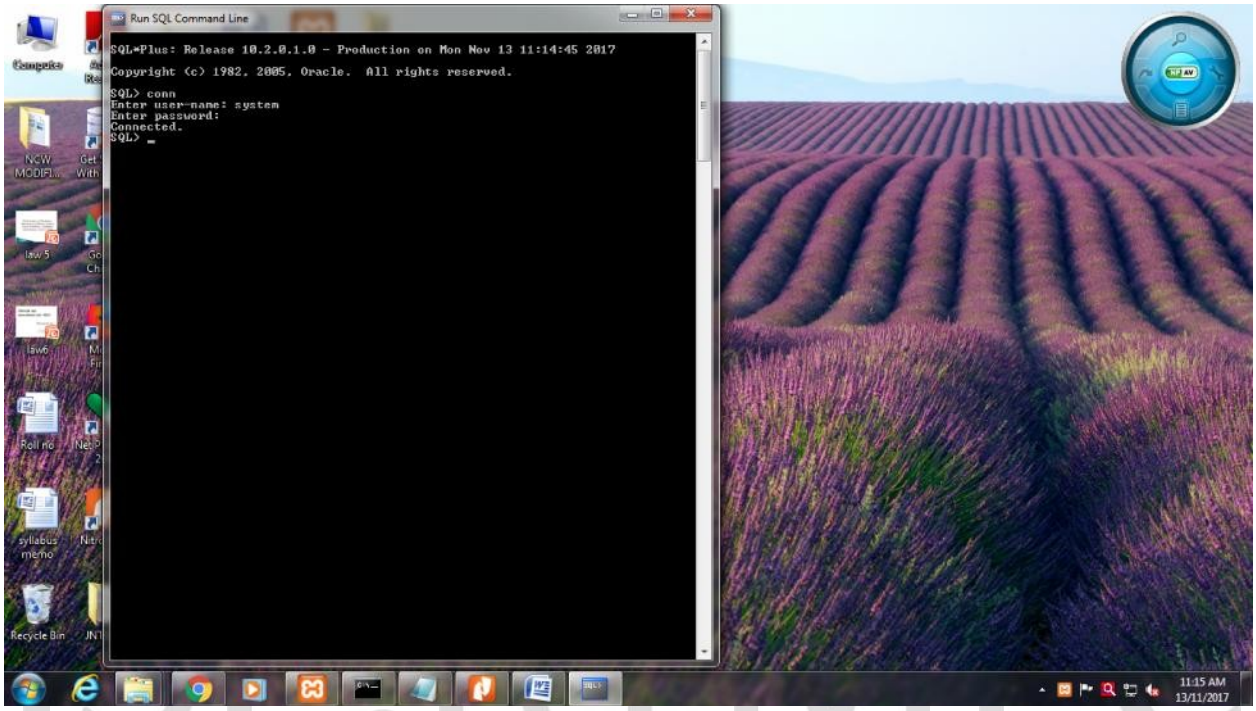
Main Campus

Mount View Campus, Off 28Kms, Ballari - Siruguppa Road,
Near Sindhigeri, GP NO.735, Hagaluru, Siruguppa Taluk,
Ballari - 583120, Karnataka

TABLE OF CONTENTS

`CONNECTING TO THE -SQL COMMAND LINE

Click on start->All Programs-> Oracle Database 10g Express Edition-> Run SQL Command Line



SQL> conn

Enter user-name: system

Enter password:12345/1234

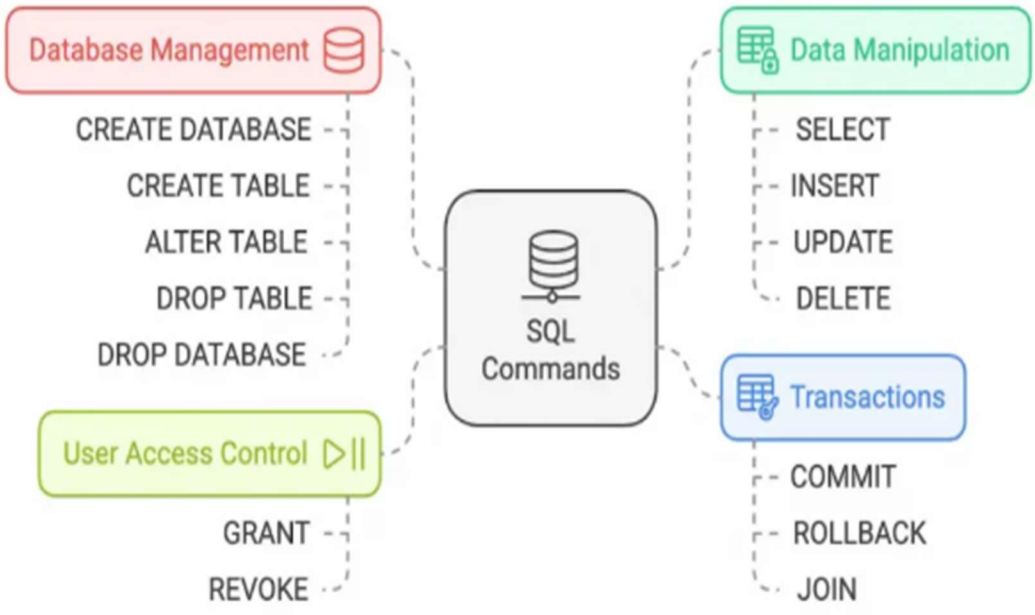
Connected.

Structured Query Language (SQL)

SQL, which stands for **Structured Query Language**, is a powerful language used for managing and manipulating relational databases.

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views



Categorization of SQL Commands

SQL commands can be categorized into five primary types, each serving a distinct purpose in database management.

Types of SQL Commands:

1. **DDL (Data Definition Language)**:
    - CREATE: Creates a new table or database.
    - ALTER: Modifies an existing database object.
    - DROP: Deletes an entire table, database, or other objects.
    - TRUNCATE: Removes all records from a table, deleting the space allocated for the records.
2. **DML (Data Manipulation Language)**:
    - SELECT: Retrieves data from the database.
    - INSERT: Adds new data to a table.
    - UPDATE: Modifies existing data within a table.
    - DELETE: Removes data from a table.
3. **DCL (Data Control Language)**:
    - GRANT: Gives users access privileges to the database.
    - REVOKE: Removes access privileges given with the GRANT command.
4. **TCL (Transaction Control Language)**:
    - COMMIT: Saves all changes made in the current transaction.
    - ROLLBACK: Restores the database to the last committed state.
    - SAVEPOINT: Sets a savepoint within a transaction.
    - SET TRANSACTION: Places a name on a transaction.

Data Definition Language (DDL) Commands

What is DDL?

DDL, which stands for Data Definition Language, is a subset of SQL (Structured Query Language) commands used to define and modify the database structure. These commands are used to create, alter, and delete database objects like tables, indexes, and schemas.

The primary DDL commands in SQL include:

1. **CREATE**: This command is used to create a new database object. For example, creating a new table, a view, or a database.
    - Syntax for creating a table: CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
2. **ALTER**: This command is used to modify an existing database object, such as adding, deleting, or modifying columns in an existing table.
    - Syntax for adding a column in a table: ALTER TABLE table_name ADD column_name datatype;
    - Syntax for modifying a column in a table: ALTER TABLE table_name MODIFY COLUMN column_name datatype;
3. **DROP**: This command is used to delete an existing database object like a table, a view, or other objects.
    - Syntax for dropping a table: DROP TABLE table_name;
4. **TRUNCATE**: This command is used to delete all data from a table, but the structure of the table remains. It's a fast way to clear large data from a table.
    - Syntax: TRUNCATE TABLE table_name;
5. **COMMENT**: Used to add comments to the data dictionary.
    - Syntax: COMMENT ON TABLE table_name IS 'This is a comment.';
6. **RENAME**: Used to rename an existing database object.
    - Syntax: RENAME TABLE old_table_name TO new_table_name;

DDL commands play a crucial role in defining the database schema.

Data Manipulation Language (DML) Commands in SQL

What is DML Commands in SQL?

Data Manipulation Language (DML) is a subset of SQL commands used for adding (inserting), deleting, and modifying (updating) data in a database. DML commands are crucial for managing the data within the tables of a database.

The primary DML commands in SQL include:

1. **INSERT**: This command is used to add new rows (records) to a table.
    - Syntax: INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);
2. **UPDATE**: This command is used to modify the existing records in a table.
    - Syntax: UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

- The WHERE clause specifies which records should be updated. Without it, all records in the table will be updated.

3. **DELETE**: This command is used to remove one or more rows from a table.
    - Syntax: DELETE FROM table_name WHERE condition;
    - Like with UPDATE, the WHERE clause specifies which rows should be deleted. Omitting the WHERE clause will result in all rows being deleted.
4. **SELECT**: Although often categorized separately, the SELECT command is sometimes considered part of DML as it is used to retrieve data from the database.
    - Syntax: SELECT column1, column2, ... FROM table_name WHERE condition;
    - The SELECT statement is used to query and extract data from a table, which can then be used for various purposes.

Data Control Language (DCL) Commands in SQL

What is DCL commands in SQL?

Data Control Language (DCL) is a subset of SQL commands used to control access to data in a database. DCL is crucial for ensuring security and proper data management, especially in multi-user database environments.

The primary DCL commands in SQL include:

1. **GRANT**: This command is used to give users access privileges to the database. These privileges can include the ability to select, insert, update, delete, and so on, over database objects like tables and views.
    - Syntax: GRANT privilege_name ON object_name TO user_name;
    - For example, GRANT SELECT ON employees TO user123; gives user123 the permission to read data from the employees table.
2. **REVOKE**: This command is used to remove previously granted access privileges from a user.
    - Syntax: REVOKE privilege_name ON object_name FROM user_name;
    - For example, REVOKE SELECT ON employees FROM user123; would remove user123's permission to read data from the employees table.

- Database administrators typically use DCL commands. When using these commands, it's important to carefully manage who has access to what data, especially in environments where data sensitivity and user roles vary significantly.
- In some systems, DCL functionality also encompasses commands like DENY (specific to certain database systems like Microsoft SQL Server), which explicitly denies specific permissions to a user, even if those permissions are granted through another role or user group.
- Remember, the application and syntax of DCL commands can vary slightly between different SQL database systems, so it's always good to refer to specific documentation for the database you are using.

Transaction Control Language (TCL) Commands in SQL

What are TCL commands in SQL?

Transaction Control Language (TCL) is a subset of SQL commands used to manage transactions in a database. Transactions are important for maintaining the integrity and consistency of data. They allow multiple database operations to be executed as a single unit of work, which either entirely succeeds or fails.

The primary TCL commands in SQL include:

1. **BEGIN TRANSACTION** (or sometimes just **BEGIN**): This command is used to start a new transaction. It marks the point at which the data referenced in a transaction is logically and physically consistent.
    - Syntax: BEGIN TRANSACTION;
    - Note: In many SQL databases, a transaction starts implicitly with any SQL statement that accesses or modifies data, so explicit use of BEGIN TRANSACTION is not always necessary.
2. **COMMIT**: This command is used to permanently save all changes made in the current transaction.
    - Syntax: COMMIT;
    - When you issue a COMMIT command, the database system will ensure that all changes made during the current transaction are saved to the database.
3. **ROLLBACK**: This command is used to undo changes that have been made in the current transaction.
    - Syntax: ROLLBACK;

- If you issue a ROLLBACK command, all changes made in the current transaction are discarded, and the state of the data reverts to what it was at the beginning of the transaction.

4. **SAVEPOINT**: This command creates points within a transaction to which you can later roll back. It allows for partial rollbacks and more complex transaction control.
    - Syntax: SAVEPOINT savepoint_name;
    - You can roll back to a savepoint using ROLLBACK TO savepoint_name;
5. **SET TRANSACTION**: This command is used to specify characteristics for the transaction, such as isolation level.
    - Syntax: SET TRANSACTION [characteristic];
    - This is more advanced usage and may include settings like isolation level which controls how transaction integrity is maintained and how/when changes made by one transaction are visible to other transactions.

TCL commands are crucial for preserving a database's ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring that all transactions are processed reliably. These commands play a key role in any database operation where data consistency and integrity are important.

Data Query Language (DQL) Commands in SQL

What are DQL commands in SQL?

Data Query Language (DQL) is a subset of SQL commands used primarily to query and retrieve data from existing database tables. In SQL, DQL is mostly centered around the SELECT statement, which is used to fetch data according to specified criteria. Here's an overview of the SELECT statement and its common clauses:

1. **SELECT**: The main command used in DQL, SELECT retrieves data from one or more tables.
    - Basic Syntax: SELECT column1, column2, ... FROM table_name;
    - To select all columns from a table, you use SELECT * FROM table_name;
2. **WHERE Clause**: Used with SELECT to filter records based on specific conditions.
    - Syntax: SELECT column1, column2, ... FROM table_name WHERE condition;
    - Example: SELECT * FROM employees WHERE department = 'Sales';

3. **JOIN Clauses**: Used to combine rows from two or more tables based on a related column between them.
    - Types include INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN.
    - Syntax: SELECT columns FROM table1 [JOIN TYPE] JOIN table2 ON table1.column_name = table2.column_name;
4. **GROUP BY Clause**: Used with aggregate functions (like COUNT, MAX, MIN, SUM, AVG) to group the result set by one or more columns.
    - Syntax: SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1;
5. **ORDER BY Clause**: Used to sort the result set in ascending or descending order.
    - Syntax: SELECT column1, column2 FROM table_name ORDER BY column1 [ASC|DESC], column2 [ASC|DESC];

Differentiating DDL, DML, DCL, TCL, and DQL Commands

Category	Full Form	Purpose	Common Commands
DDL	Data Definition Language	To define and modify database structure	CREATE, ALTER, DROP, TRUNCATE, RENAME
DML	Data Manipulation Language	To manipulate data within existing structures	SELECT, INSERT, UPDATE, DELETE
DCL	Data Control Language	To control access to data in the database	GRANT, REVOKE
TCL	Transaction Control Language	To manage transactions in the database	COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION
DQL	Data Query Language	To query and retrieve data from a database	SELECT (often used with WHERE, JOIN, GROUP BY, HAVING, ORDER BY)

Common DDL Commands

CREATE TABLE

The CREATE TABLE command is used to define a new table in the database. Here's an example:

```
CREATE TABLE Employees (

   EmployeeID INT PRIMARY KEY,

   FirstName VARCHAR(50),

   LastName VARCHAR(50).. );
```

This command defines a table called "Employees" with columns for employee ID, first name, last name, and more.

ALTER TABLE

The ALTER TABLE command allows you to modify an existing table. For instance, you can add a new column or modify the data type of an existing column:

```
ALTER TABLE Employees

ADD Email VARCHAR(100);
```

This adds an "Email" column to the "Employees" table.

DROP TABLE

The DROP TABLE command removes a table from the database:

```
DROP TABLE Employees;
```

This deletes the "Employees" table and all its data.

CREATE INDEX

The CREATE INDEX command is used to create an index on one or more columns of a table, improving query performance:

```
CREATE INDEX idx_LastName ON Employees(LastName);
```

This creates an index on the "LastName" column of the "Employees" table.

SQL Command	**Code Snippet**	**Output**
CREATE TABLE	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Department VARCHAR(50));	New "Employees" table created with specified columns.

SQL Command	Code Snippet	Output
ALTER TABLE	ALTER TABLE Employees ADD Email VARCHAR(100);	"Email" column added to the "Employees" table.
DROP TABLE	DROP TABLE Employees;	"Employees" table and its data deleted.

These examples illustrate the usage of DDL commands to create, modify, and delete database objects.

Data Manipulation Language (DML) Commands in SQL

What is DML?

DML, or Data Manipulation Language, is a subset of SQL used to retrieve, insert, update, and delete data in a database. DML commands are fundamental for working with the data stored in tables.

Common DML Commands in SQL

SELECT

The SELECT statement retrieves data from one or more tables based on specified criteria:

```
SELECT FirstName, LastName FROM Employees WHERE Department = 'Sales';
```

This query selects the first and last names of employees in the "Sales" department.

INSERT

The INSERT statement adds new records to a table:

```
INSERT INTO Employees (FirstName, LastName, Department) VALUES ('John', 'Doe', 'HR');
```

This inserts a new employee record into the "Employees" table.

UPDATE

The UPDATE statement modifies existing records in a table:

```
UPDATE Employees SET Salary = Salary * 1.1 WHERE Department = ‘Engineering’;
```

This increases the salary of employees in the “Engineering” department by 10%.

DELETE

The DELETE statement removes records from a table:

```
DELETE FROM Employees WHERE Department = 'Finance';
```

This deletes employees from the “Finance” department.

DML Commands in SQL with Examples

Here are code snippets and their corresponding outputs for DML commands:

SQL Command	**Code Snippet**	**Output**
SELECT	SELECT FirstName, LastName FROM Employees WHERE Department = 'Sales';	Retrieves the first and last names of employees in the “Sales” department.
INSERT	INSERT INTO Employees (FirstName, LastName, Department) VALUES ('John', 'Doe', 'HR');	New employee record added to the “Employees” table.
UPDATE	UPDATE Employees SET Salary = Salary * 1.1 WHERE Department = 'Engineering';	Salary of employees in the “Engineering” department increased by 10%.

SQL Command	Code Snippet	Output
DELETE	DELETE FROM Employees WHERE Department = 'Finance';	Employees in the "Finance" department deleted.

These examples demonstrate how to manipulate data within a database using DML commands.

Data Control Language (DCL) Commands in SQL

What is DCL?

DCL, or Data Control Language, is a subset of SQL used to manage database security and access control. DCL commands determine who can access the database and what actions they can perform.

Common DCL Commands

GRANT

The GRANT command is used to grant specific privileges to database users or roles:

```
GRANT SELECT, INSERT ON Employees TO HR_Manager;
```

This grants the "HR_Manager" role the privileges to select and insert data into the "Employees" table.

REVOKE

The REVOKE command is used to revoke previously granted privileges:

```
REVOKE DELETE ON Customers FROM Sales_Team;
```

This revokes the privilege to delete data from the "Customers" table from the "Sales_Team" role.

DCL Commands in SQL with Examples

Here are code snippets and their corresponding real-value outputs for DCL commands:

SQL Command	Code Snippet	Output (Real Value Example)
GRANT	GRANT SELECT, INSERT ON Employees TO HR_Manager;	"HR_Manager" role granted privileges to select and insert data in the "Employees" table.
REVOKE	REVOKE DELETE ON Customers FROM Sales_Team;	Privilege to delete data from the "Customers" table revoked from the "Sales_Team" role.

These examples illustrate how to control access and security in a database using DCL commands.

Transaction Control Language (TCL) Commands in SQL

What is TCL?

TCL, or Transaction Control Language, is a subset of SQL used to manage database transactions. TCL commands ensure data integrity by allowing you to control when changes to the database are saved permanently or rolled back.

Common TCL Commands in SQL

COMMIT

The COMMIT command is used to save changes made during a transaction to the database permanently:

```
BEGIN;

-- SQL statements

COMMIT;
```

This example begins a transaction, performs SQL statements, and then commits the changes to the database.

ROLLBACK

The ROLLBACK command is used to undo changes made during a transaction:

```
BEGIN;

-- SQL statements

ROLLBACK;
```

This example begins a transaction, performs SQL statements, and then rolls back the changes, restoring the database to its previous state.

SAVEPOINT

The SAVEPOINT command allows you to set a point within a transaction to which you can later roll back:

```
BEGIN;

-- SQL statements

SAVEPOINT my_savepoint;

-- More SQL statements

ROLLBACK TO my_savepoint;
```

This example creates a savepoint and later rolls back to that point, undoing some of the transaction's changes.

TCL Commands in SQL with Examples

Here are code snippets and their corresponding outputs for TCL commands:

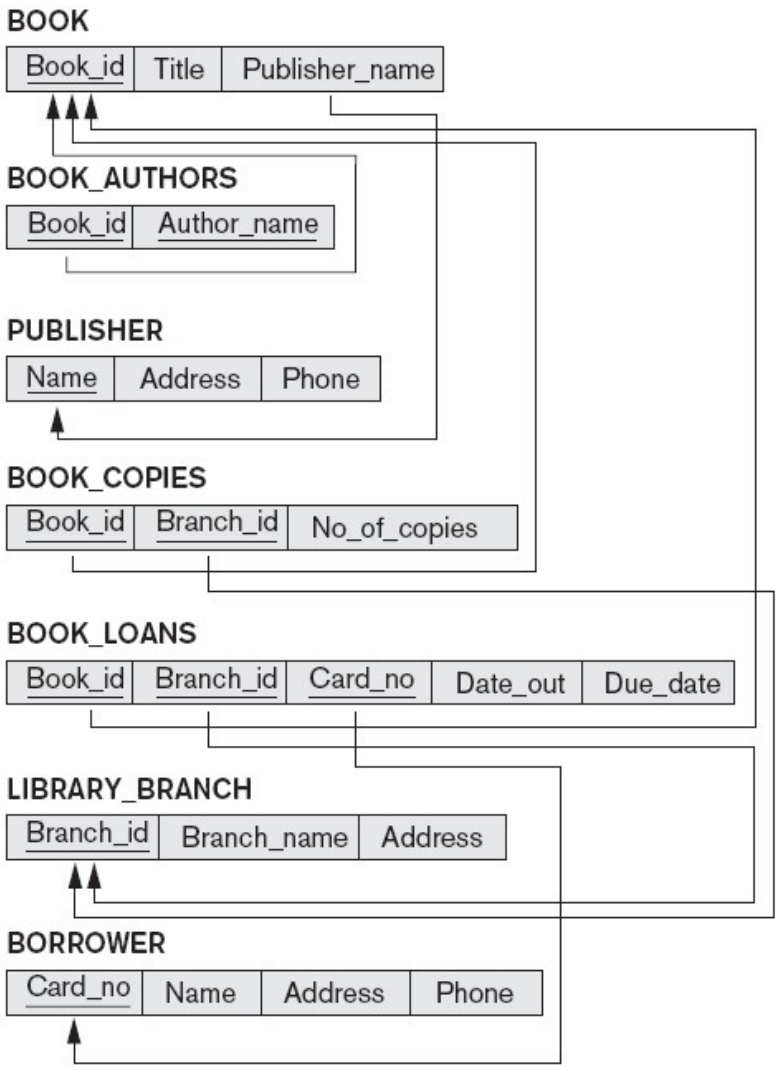
SQL Command	**Code Snippet**	**Output**

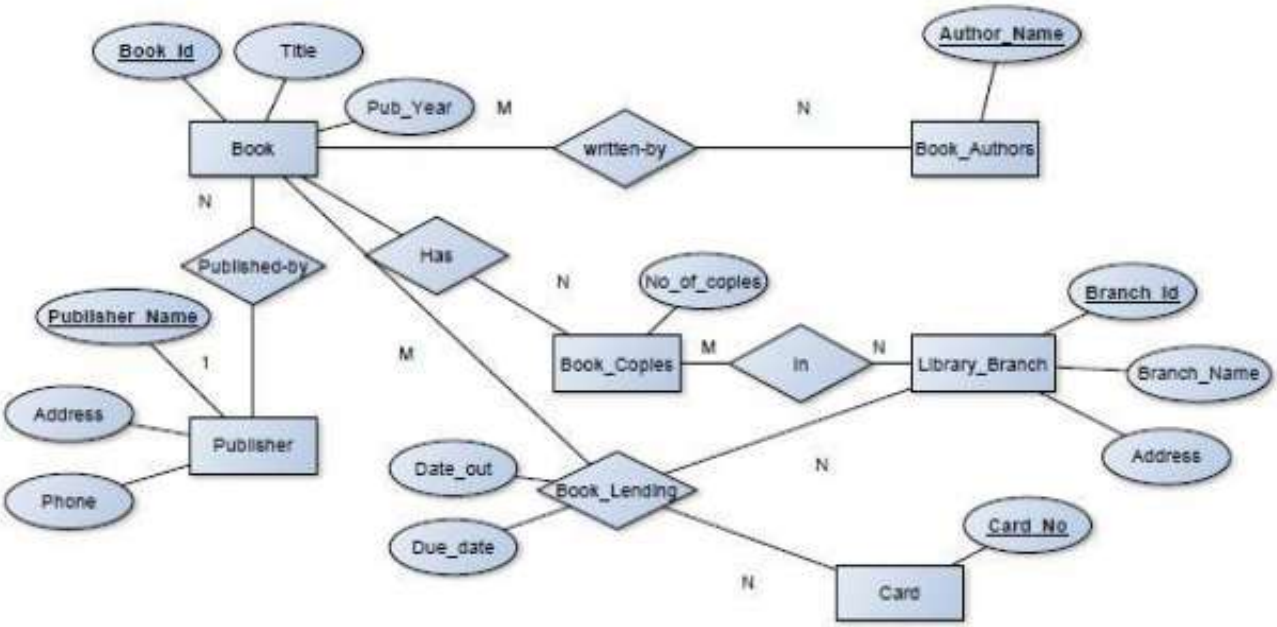
SQL Command	**Code Snippet**	**Output**
COMMIT	BEGIN; -- SQL statements COMMIT;	Changes made in the transaction saved permanently.
ROLLBACK	BEGIN; -- SQL statements ROLLBACK;	Changes made in the transaction rolled back.
SAVEPOINT	BEGIN; -- SQL statements SAVEPOINT my_savepoint; -- More SQL statements ROLLBACK TO my_savepoint;	Savepoint created and later used to roll back to a specific point in the transaction.

EXPERIMENT-1

Consider the following schema for a Library Database:
BOOK(Book_id, Title, 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)

SCHEMA DIAGRAM OF LIBRARY DATABASE





Create table

create table publisher(name varchar(18) primary key, address varchar(10), phone varchar(10));

create table book(bkid varchar(30) primary key, title varchar(20), name varchar(20) references publisher(name) on delete cascade,pyear number(4));

create table book_authors(bkid varchar(30) references book(bkid) on delete cascade, aname varchar(20),primary key(bkid));

create table library_branch(branch_id varchar(30) primary key, branch_name varchar(30), address varchar(30));

create table book_copies(bkid varchar(30)references book1(bkid) on delete cascade, branch_id varchar(30) references library_branch1(branch_id) on delete cascade, numcopies number);

create table book_lending(bkid varchar(30) references book1(bkid) on delete cascade,branch_id varchar(30) references library_branch(branch_id) on delete cascade,cardno number,date_out date,due_date date);

DESCRIPTION

desc publisher;

Name	Null?	Type
NAME	NOT NULL	VARCHAR2(18)
ADDRESS		VARCHAR2(10)
PHONE		VARCHAR2(10)

desc book;

Name	Null?	Type
BKID	NOT NULL	VARCHAR2(30)
TITLE		VARCHAR2(20)
NAME		VARCHAR2(20)
PYEAR		NUMBER(4)

desc book_authors;

Name	Null?	Type
BKID	NOT NULL	VARCHAR2(30)
ANAME		VARCHAR2(20)

desc library_branch;

Name	Null?	Type
BRANCHID	NOT NULL	VARCHAR2(30)
BRANCH_NAME		VARCHAR2(30)
ADDRESS		VARCHAR2(30)

desc book_copies;

Name	Null?	Type
BKID		VARCHAR2(30)
BRANCH_ID		VARCHAR2(30)
NUMCOPIES		NUMBER

desc book_lending;

Name	Null?	Type
BKID		VARCHAR2(30)
BRANCHID		VARCHAR2(30)
CARDNO		NUMBER
DATE_OUT		DATE
DUE_DATE		DATE

INSERT AND DISPLAY COMMANDS

```
insert into publisher values('&name','&address',&phone);
insert into book1 values('&bkid','&title','&name',&pyear);
insert into book_authors values('&bkid','&aname');
insert into library_branch values('&branchid','&branch_name','&address');
insert into book_copies values('&bkid','&branch_id',&numcopies);
insert into
book_lendingvalues('&bkid','&branchid',&cardno,'&date_out','&due_date');
```

```
select * from publisher;

NAME            ADDRESS    PHONE
------------------ ---------- ----------
PHI             Delhi      9877777777
pearson          delhi      9812345678
gold            bangalore  984412345
star            mysore     9898986666
vikas            Hubli      9986751169
```

```
SQL> select * from book;

BKID                 TITLE           NAME            PYEAR
------------------------------ -------------------- ------------------------------
CS111                C               PHI             2010

CS112                C++             pearson          2012

CS113                DBMS             gold            2015

CS114                CNS             gold            2020

CS115                python           vikas           2023
```

select * from book_authors;

BKID	ANAME
CS111	Navathe
CS112	Scott
CS113	kottur
CS114	subhash
CS115	terrance

select * from library_branch;

BRANCHID	BRANCH_NAME	ADDRESS
KUCSE01	CSE	BALLARI
KUEEE02	EEE	HOSPET
KUECE01	ECE	MYSORE
KUAIML01	AIML	HOSPET
KUDATA01	DATA	DELHI

select * from book_copies;

BKID	BRANCH_ID	NUMCOPIES
CS111	KUCSE01	20
CS112	KUEEE02	34
CS113	KUECE01	33
CS114	KUDATA01	70
CS115	KUAIML01	22

select * from book_lending;

BKID	BRANCHID	CARDNO	DATE_OUT	DUE_DATE
CS111	KUAIML01	111	17-AUG-16	20-SEP-16
CS112	KUEEE02	111	03-JUL-20	05-SEP-20
CS113	KUECE01	111	03-JUL-16	05-AUG-16
CS114	KUECE01	111	13-SEP-22	25-SEP-22
CS115	KUECE01	112	22-AUG-23	20-SEP-23
CS115	KUCSE01	1234	22-JAN-22	25-JAN-22

QUERIES

1.Retrieve details of all the books in the library as (BKID,TITLE,PUBLISHER NAME ,AUTHOR ,NO_OF_COPIES)

```
select b.bkid,b.title,b.name,ba.aname,bc.branch_id,bc.numcopies
from book b,book_authors ba,book_copies bc
where b.bkid=ba.bkid and
b.bkid=bc.bkid;
```

BKID	TITLE	NAME	NAME	BRANCH_ID	NUMCOPIES
CS111	C	PHI	Navathe	KUCSE01	20
CS112	C++	pearson	Scott	KUEEE02	34
CS113	DBMS	gold	kottur	KUECE01	33

2.Get the particulatrs of borrowers who have borrowed more than three books but from JAN-2017 to jun-2017

```
select b.bkid,bl.branchid,bl.cardno
from book b,book_lending bl
where b.bkid=bl.bkid and bl.cardno
in
(select cardno
from book_lending bl
where due_date between '1-JAN-2016' and '1-SEP-2022' and
date_out between  '1-JAN-2016' and '1-SEP-2022'
group by cardno having count(cardno)>=3);
```

BKID	BRANCHID	CARDNO
CS111	KUAIML01	111
CS112	KUEEE02	111
CS113	KUECE01	111
CS114	KUECE01	111

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

```
delete from book
where bkid='115';
```

1 row deleted.

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

```
select bkid,title,name,pyear
from book
group by pyear,bkid,title,name;
```

BKID	TITLE	NAME	PYEAR
CS114	CNS	gold	2020
CS113	DBMS	gold	2015

CS112 C++ pearson 2012

CS115 python vikas 2023

CS111 C PHI 2010

5.create a view of all books and its no of copies that are currently available in the library;

```
create view LIBRARY_BOOKS_DB
  as
  select b.bkid,b.title,bc.numcopies
  from book b,book_copies bc
  where b.bkid=bc.bkid;

select * from LIBRARY_BOOKS_DB;
```

BKID	TITLE	NUMCOPIES
CS111	C	20
CS112	C++	34
CS113	DBMS	33
CS114	CNS	70
CS115	python	22

EXPERIMENT-2

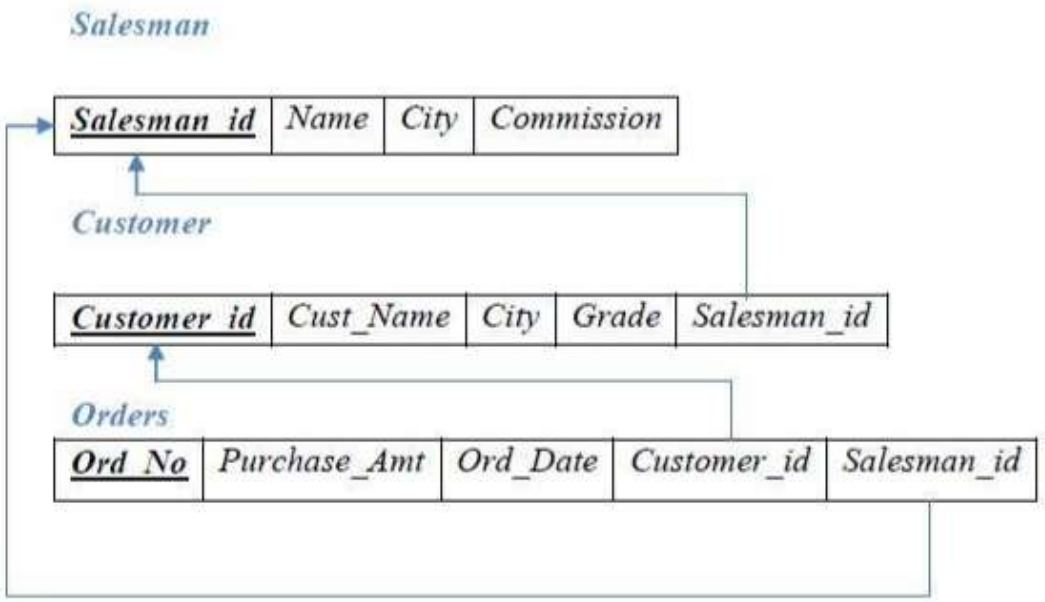
2) Consider the following schema for Order Database:

SALESMAN(Salesman_id, Name, City, Commission)

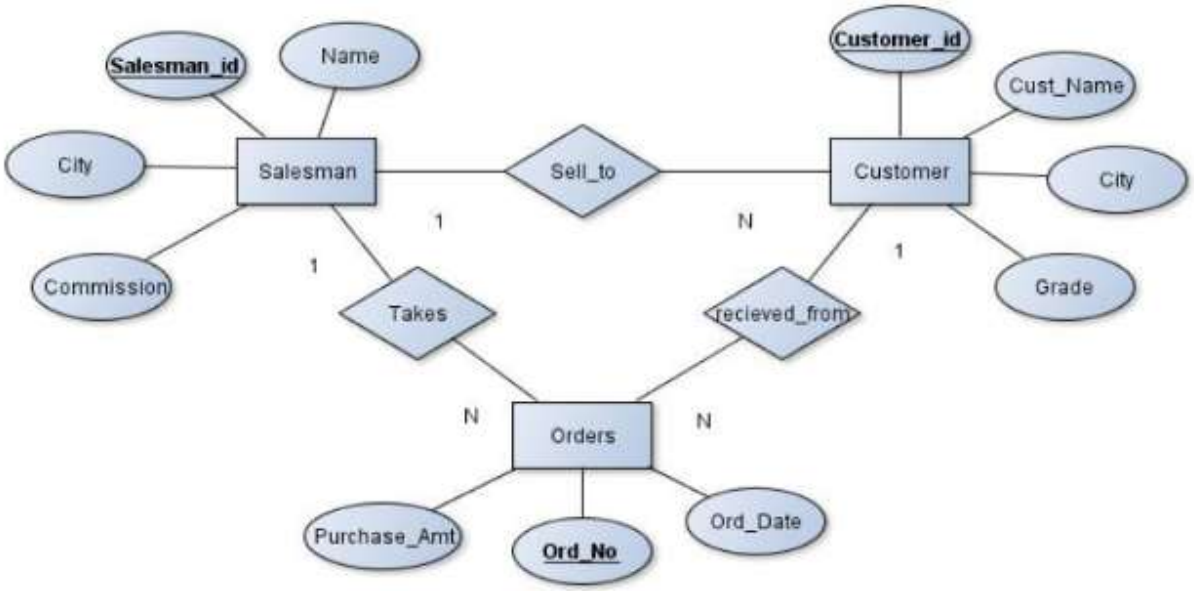
CUSTOMER(Customer_id, Cust_Name, City, Grade, Salesman_id)

ORDERS(Ord_No, Purchase_Amt, Ord_Date, Customer_id,Salesman_id)

SCHEMA DIAGRAM



E-R DIAGRAM



CREATION OF TABLES:

create table sales(sid number primary key, sname varchar(10),scity varchar(10),commission number);

create table customer(cid number primary key,cname varchar(10),city varchar(10),grade number(5),sid number references sales(sid) on delete cascade);

create table orders(ono number(5) primary key, pamount number, odate date, cid number references customer(cid) on delete cascade, sid number references sales(sid) on delete cascade);

Insert

insert into sales values(&sid,'&sname','&scity',&commission);

insert into customer values(&cid,'&cname','&city',&grade,&sid);

insert into orders values(&ono,&pamount,'&odate',&cid,&sid);

DESCRIPTION

desc sales;

Name	Null?	Type
SID	NOT NULL	NUMBER
SNAME		VARCHAR2(10)
SCITY		VARCHAR2(10)
COMMISSION		NUMBER

desc customer;

Name	Null?	Type
CID	NOT NULL	NUMBER
CNAME		VARCHAR2(10)
CITY		VARCHAR2(10)
GRADE		NUMBER(5)
SID		NUMBER

desc orders;

Name	Null?	Type
ONO	NOT NULL	NUMBER(5)
PAMOUNT		NUMBER
ODATE		DATE
CID		NUMBER
SID		NUMBER

select * from sales;

SID	SNAME	SCITY	COMMISSION
111	David	Bangalore	10
112	sam	mysore	20
113	sk	Ballari	20
114	pavan	mysore	30
115	ram	Hubli	40

select * from customer;

CID	CNAME	CITY	GRADE	SID
221	Priya	Bangalore	100	111
222	Suma	Mysore	200	111
223	Malli	Ballari	111	111
224	Raj	Hospet	100	111
225	Ravi	Raichur	500	112
226	Vijay	Bangalore	200	113

select * from orders;

ONO	PAMOUNT	ODATE	CID	SID
551	6574	02-JAN-17	221	111
552	43251	05-FEB-17	222	113
553	3526	06-MAR-17	224	111
554	3527	16-MAR-17	224	113
556	2300	02-JAN-16	225	111
557	3400	16-MAR-25	221	112
558	2435	20-JAN-17	224	114

Queries

Q1) Count the customers with grades above the bangalore's average.

```
select grade,count(*) as NO_OF_CUSTOMERS
 from customer
 group by grade
 having grade>(select avg(grade)
 from customer
 where city = 'Bangalore' );
```

GRADE	NO_OF_CUSTOMERS
500	1
200	2

Q2) find the name and no of all salesman who had more than one customer.

```
select sname,sid
from sales
where sid in
(select sid
 from orders
 group by sid
 having count(cid)>1);
```

David	111
sk	113

Q3) List all the salesman and indicate those who have and dont have in their city(use UNION operation)

```
(select s.sid ,s.sname,s.scity,c.cid,c.city,c.cname
 from sales s,customer c
 where s.sid=c.sid and c.city=s.scity)
    UNION
 (select s.sid ,s.sname,s.scity,c.cid,c.city,'NO CUSTOMERS IN CITY'
  from sales s,customer c
  where s.sid=c.sid and c.city!=s.scity);
```

```
     SID SNAME     SCITY          CID CITY      CNAME
---------- ---------- ---------- ---------- ---------- --------------------
     111 David     Bangalore       221 Bangalore  Priya
     111 David     Bangalore       222 Mysore     NO CUSTOMERS IN CITY
     111 David     Bangalore       223 Ballari    NO CUSTOMERS IN CITY
     111 David     Bangalore       224 Hospet     NO CUSTOMERS IN CITY
     112 sam       mysore          225 Raichur    NO CUSTOMERS IN CITY
     113 sk        Ballari         226 Bangalore  NO CUSTOMERS IN CITY

6 rows selected.
```

Q4) Create a view that finds the salesman who has the customer with the highest order of a day.

```
SQL> create view HIGHEST_ORDERS
    as select s.sid,s.sname,o1.odate,o1.pamount
    from  sales s,orders o1
    where s.sid=o1.sid and o1.pamount=
    (select max(o2.pamount) from orders o2 where
     o1.odate=o2.odate);

View created.
```

select * from HIGHEST_ORDERS;

```
     SID SNAME     ODATE       PAMOUNT
---------- ---------- --------- ----------
     111 David     02-JAN-17      6574
     113 sk        05-FEB-17     43251
```

```
111 David     06-MAR-17      3526
113 sk       16-MAR-17      3527
111 David     02-JAN-16      2300
112 sam       16-MAR-25      3400
114 pavan     20-JAN-17      2435
```

7 rows selected.

Q5) Demonstrate the DELETE operation by removing salesman with ID 1000.all his orders must also be deleted

```
delete from sales
  where sid=118;
```

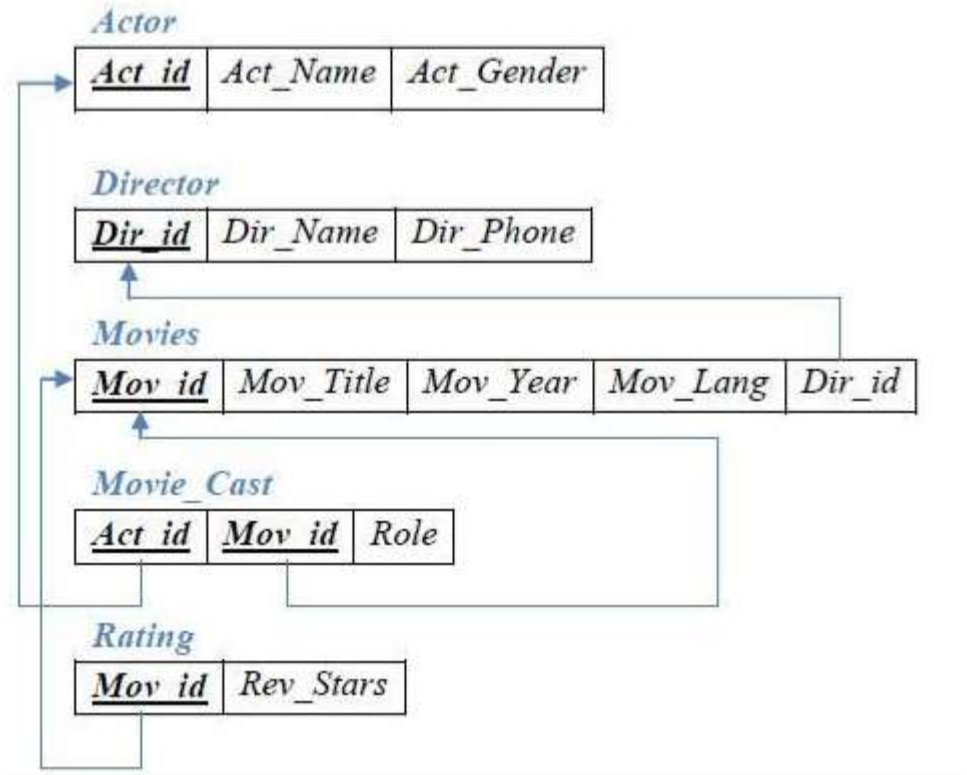
1 row deleted.

LAB PROGRAM 3: MOVIES DATABASE

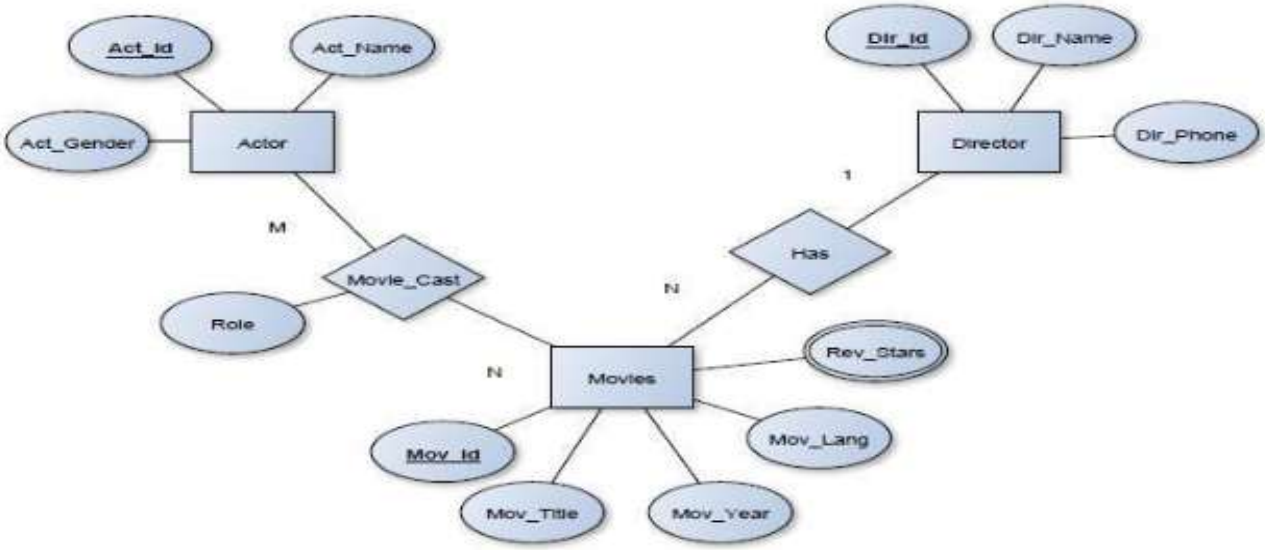
C. Consider the schema for Movie Database:

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.

SCHEMA -DIAGRAM



E-R DIAGRAM



CREATION OF TABLES

create table actor(actid number primary key, aname varchar(10),gender varchar(6));

create table director(did number primary key,dname varchar(30),phone number);

create table movies(mid number primary key, mtitle varchar(10),myear number,mlang varchar(10),did number(5) references director(did) on delete cascade);

create table movie_cast(actid number references actor(actid) on delete cascade, mid number references movies(mid) on delete cascade, role varchar(10));

create table rating(mid number references movies(mid) on delete cascade, revstars number);

Description:

desc actor;

Name	Null?	Type
ACTID	NOT NULL	NUMBER
ANAME		VARCHAR2(10)
GENDER		VARCHAR2(6)

desc director;

Name	Null?	Type
DID	NOT NULL	NUMBER
DNAME		VARCHAR2(30)
PHONE		NUMBER

desc movies;

Name	Null?	Type
MID	NOT NULL	NUMBER
MTITLE		VARCHAR2(10)
MYEAR		NUMBER
MLANG		VARCHAR2(10)
DID		NUMBER(5)

desc movie_cast;

Name	Null?	Type
ACTID		NUMBER
MID		NUMBER
ROLE		VARCHAR2(10)

desc rating;

Name	Null?	Type
MID		NUMBER
REVSTARS		NUMBER

Insert

insert into actor values(&aid,'&aname','&gender');

insert into director values(&did,'&dname',&phone);

insert into movies values(&mid,'&mtitle',&myear,'&mlang',&did);

insert into movie_cast values(&aid,&mid,'&role');

insert into rating values(&mid,&revstars);

Display Table

SQL> select * from actor;

ACTID	ANAME	GENDER
111	sam	male
112	bob	male
113	Ariyana	female
114	David	male
115	Jim	male
116	Kim	female
117	Puneeth	male

7 rows selected.

SQL> select * from director;

DID	DNAME	PHONE
2111	hitchcock	5647382
2112	steven	657484
2113	johnwatts	56767
2114	santosh	56764
2115	John	12345

SQL> select * from movies;

MID	MTITLE	MYEAR	MLANG	DID
500	jpark	2013	English	2112
501	rwindow	1954	English	2111
503	spiderman	2017	English	2113
504	Rajkumar	2020	Kannada	2114
505	kushi	2000	Telugu	2115

SQL> select * from movie_cast;

ACTID	MID	ROLE
111	501	mainlead
112	503	supporting
113	501	heroin
114	503	mainlead
115	503	negative
116	503	heroin
116	504	hero
111	503	supporting
112	501	hero
114	500	mainlead
115	500	supporting

11 rows selected.

select * from rating;

MID	REVSTARS
500	5
501	3
503	3
504	4
500	3
501	2
504	4
503	3
504	4
500	2
500	3

9 rows selected.

Queries:

Q1) List the titles of all the movies directed by "hitchcock".

```
select m.mid ,m.mtitle
from movies m,director d
 where m.did=d.did and d.dname='hitchcock';
```

MID	MTITLE
501	rwindow

Q2) Find the movie names where one or more actors acted in two or more movies.

```
select m.mtitle
from movies m, movie_cast mc
where m.mid=mc.mid and mc.actid in
 (select actid from movie_cast
  group by actid
  having count(actid)>=2)
  group by m.mtitle
  having count(*)>2;
```

MTITLE

---

spiderman

Q3) List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
select a.aname,m1.mtitle,m1.myear,m2.myear
from actor a, movies m1, movies m2, movie_cast c1, movie_cast c2
where m1.myear < 2000 and m2.myear > 2015 and a.actid = c1.actid
and c1.mid = m1.mid and a.actid = c2.actid and c2.mid = m2.mid;
```

ANAME	MTITLE	MYEAR	MYEAR
bob	rwindow	1954	2017
sam	rwindow	1954	2017

Q4) Find the title of movies and number of stars for each movie that has at least one rating, find the highest number of stars that movie received. Return the movie title and number of stars. Sort by movie title.

```
select m.mtitle,max(r.revstars)
from rating r,movies m
where m.mid=r.mid
 group by m.mtitle
 order by m.mtitle;
```

```
MTITLE    MAX(R.REVSTARS)
---------- ---------------
Rajkumar             4
jpark                5
rwindow              3
spiderman            3
```

Q5) Update rating of all movies directed by 'Steven Spielberg' to 5

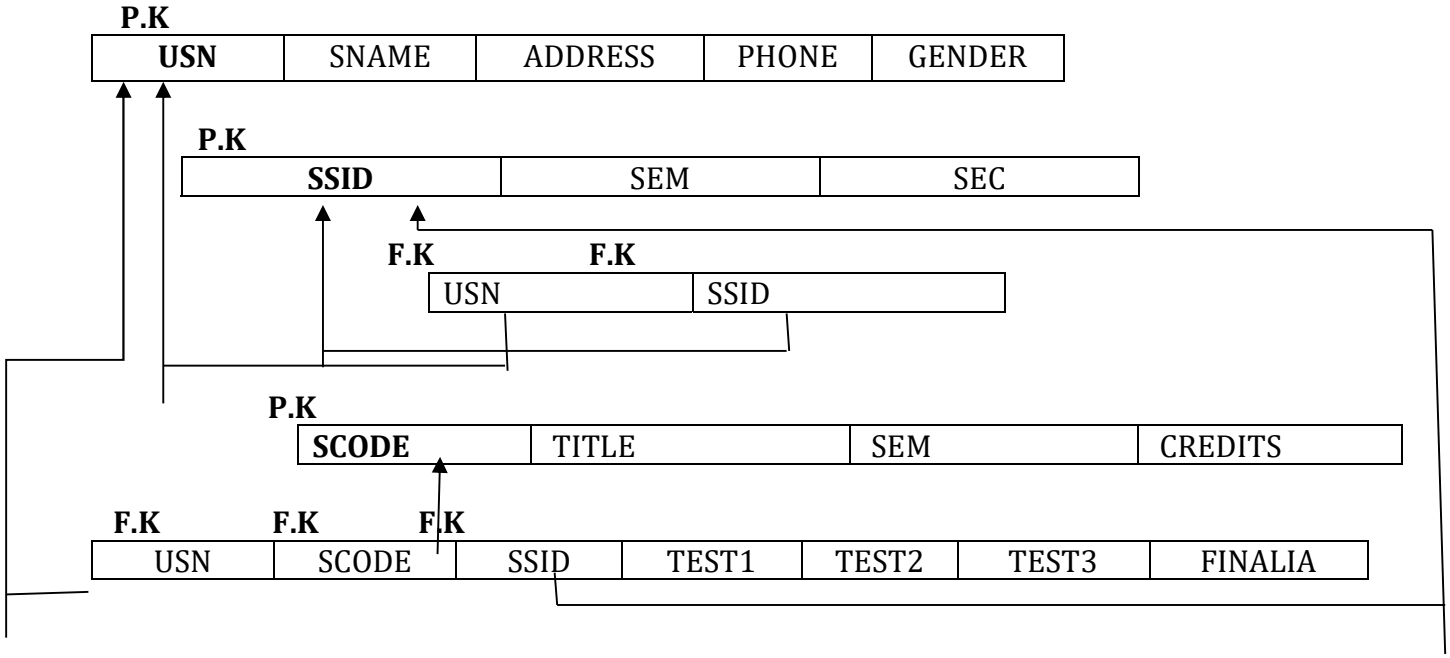
```
 update rating
set revstars='5'
where mid in
 (select m.mid
  from movies m,director d
  where m.did=d.did and dname='steven');
2 rows updated.
select * from rating;
  MID   REVSTARS
---------- ----------
     501         3
     503         3
     504         5
     501         2
     504         5
     503         3
     504         5
     500         1
     500         2
     500         3
```

LAB-PGM:4 COLLGE DATABASE

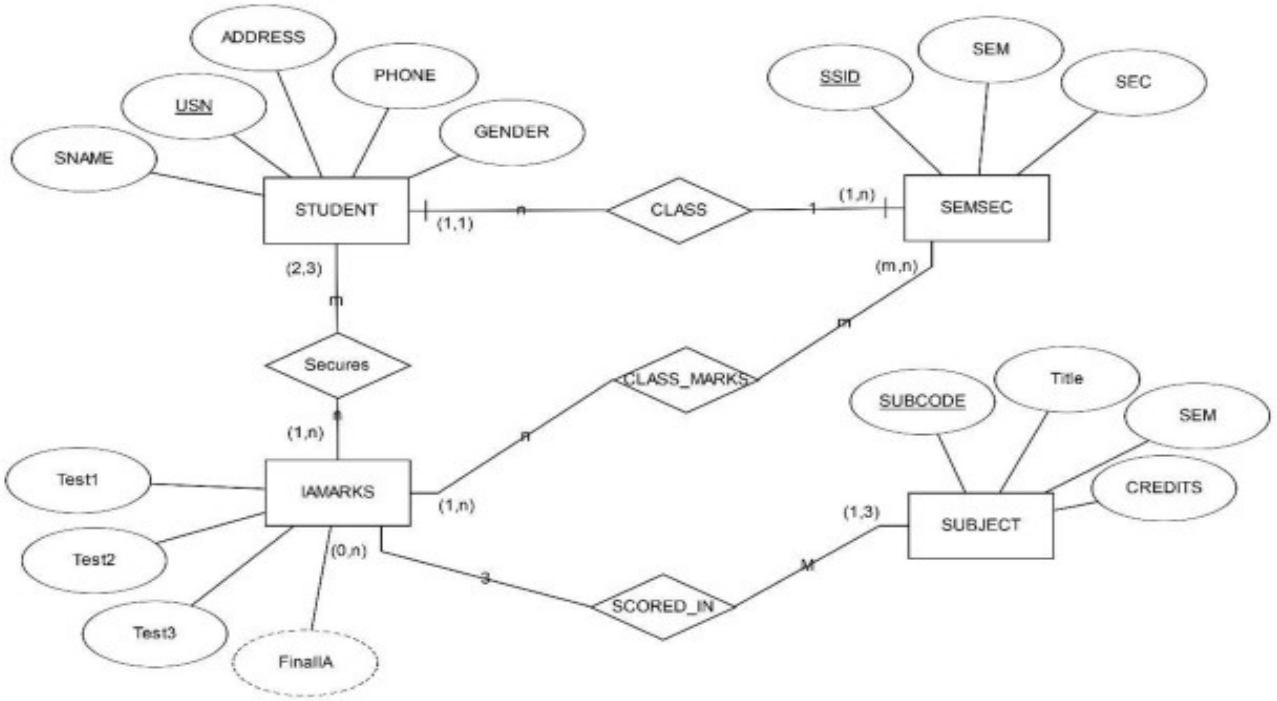
4. Consider the schema for College Database:

STUDENT (*USN, SName, Address, Phone, Gender*)
SEMSEC (*SSID, Sem, Sec*)
CLASS (*USN, SSID*)
SUBJECT (*Subcode, Title, Sem, Credits*)
IAMARKS (*USN, Subcode, SSID, Test1, Test2, Test3, FinalIA*)
Write SQL queries to
1. List all the student details studying in fourth semester 'C' section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA< 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students.

Solution:



E-R DIAGRAM



CREATION OF TABLES

1. **COLLEGE TABLE**

```
SQL> create table student
  2  (usn varchar(12) primary key,
  3  sname varchar(10),
  4  address varchar(10),phone number,gender varchar(10));

Table created.
```

2. SEMSEC

```
SQL> create table semsec
  2  (ssid number primary key,
  3  sem number,
  4  sec varchar(5)  );
```

```
Table created.
```

3. Class

```
SQL> create table class
  2  (usn varchar(12) references student(usn) on delete cascade,
  3  ssid number references semsec(ssid) on delete cascade);
Table created.
```

4. Subject

```
SQL> create table subject
  2  (scode number primary key,
  3  title varchar(10),sem number,
  4  credits number);
Table created.
```

5. IA Marks

```
SQL>  create table iamarks
  2    (usn varchar(12) references student(usn) on delete cascade,
  3     scode number references subject(scode) on delete cascade,
  4      ssid number references semsec(ssid) on delete cascade,
  5      test1 number,test2 number,test3 number,finalia number);
Table created.
```

Descriptons of tables

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
```

USN	**NOT NULL VARCHAR2(12)**
SNAME	**VARCHAR2(10)**
ADDRESS	**VARCHAR2(10)**
PHONE	**NUMBER**
GENDER	**NUMBER**

SQL> desc semsec;

Name	**Null?**	**Type**
SSID	**NOT NULL**	**NUMBER**
SEM		**NUMBER**
SEC		**VARCHAR2(5)**

SQL> desc class;

Name	**Null?**	**Type**
USN		**VARCHAR2(12)**
SSID		**NUMBER**

SQL> desc subject;

Name	**Null?**	**Type**
SCODE	**NOT NULL**	**NUMBER**
TITLE		**VARCHAR2(10)**
SEM		**NUMBER**
CREDITS		**NUMBER**

SQL> desc iamarks;

Name	**Null?**	**Type**
USN		**VARCHAR2(12)**
SCODE		**NUMBER**
SSID		**NUMBER**
TEST1		**NUMBER**
TEST2		**NUMBER**
TEST3		**NUMBER**

FINALIA	NUMBER

III INSERTION

1. Student

insert into student values('&usn','&name','&address',&phone,'&gender');

```
SQL> select * from student;

USN          SNAME     ADDRESS       PHONE        GENDER
------------ ---------- --------- - ---------- ------------------------------------
KUB24CSE637  Santosh   Ballari          9986751168     male
KUB24CSE613  Ananth    Bangalore        1234567        Male
KUB24CSE689  Omkar     Hubli            8866562314     male
KUB23CSE666  RAJESH     MYSORE     9620078963     MALE
KUB24CSE624  DEEPIKA    RAICHUR  11228934     FEMALE
KUB24CSE681  SNEHA      BALLARI    112233         FEMALE
KUB24CSE644  PAVAN      BALLARI    77886612       MALE
KUB24CSE678  VIJAY      BALLARI    1122467        MALE
```

2. Semsec

insert into semsec values(&ssid,&sem,'&sec');

```
SQL> select * from semsec;

     SSID        SEM SEC
---------- ---------- -----
      500          3 a
      501          3 b
      502          3 c
      503          4 c
      504          4 b
      505          4 a

6 rows selected.
```

3. Class
   insert into class values('&usn',&ssid);

```
SQL> select * from class;

USN             SSID
----------- ----------
KUB24CSE637     500
KUB24CSE613     500
KUB24CSE689     503
KUB24CSE624     502
KUB24CSE644     505
KUB24CSE678     504

6 rows selected.
```

4. Subject
   insert into subject values(&scode,'&title',&sem,&credits);

```
SQL> select * from subject;


    SCODE TITLE           SEM    CREDITS
---------- ---------- ---------- ----------
     2341 maths             4          3
     2332 co                3          4
     2333 os                3          4
     2342 mces              4          4
     2343 dbms              4          4
     2344 ada               4          3
     2351 python            5          3
     2361 bda               7          4
     2365 cn                5          4

9 rows selected.
```

5. Iamarks

```
insert into iamarks values('&usn',&scode,&ssid,&test1,&test2,&test3,&finalia);
```

```
SQL> select * from iamarks;

USN          SCODE      SSID     TEST1     TEST2     TEST3    FINALIA
---------- ---------- ---------- ---------- ---------- ---------- ----------
KUB24CSE637      2341        500         20         20         20         20
KUB24CSE613      2332        501         25         25         25         25
KUB24CSE689      2333        502         15         18         19         17
KUB23CSE666      2342        503         17         23         21         22
KUB24CSE624      2343        503         15         11         14         15
KUB24CSE681      2365        504         17         18          4         18
KUB24CSE644      2361        503         12         13         14         14
KUB24CSE644      2343        501         15         16         17         17
KUB24CSE678      2343        504         17         19         20         20

9 rows selected.
```

VI QUERIES

Q1) List all the student details studying in fourth semester ‘B’ section.

```
SQL>  select s.*
  2   from student s,semsec ss,class c
  3   where ss.ssid=c.ssid and s.usn=c.usn and
  4   ss.sem=4 and ss.sec='B';

USN         SNAME     ADDRESS        PHONE GENDER
---------- ---------- ---------- ---------- --------------------------
KUB24CSE678 VIJAY     BALLARI        1122467 MALE
```

Q2) Compute the total number of male and female students in each semester and in each section.

```
SQL> select ss.sem,ss.sec,s.gender,count(s.gender) as COUNT
  2  from student s,semsec ss,class c
  3  where c.usn=s.usn and ss.ssid=c.ssid
  4  group by ss.sem,ss.sec,s.gender
  5  order by sem;

SEM SEC   GENDER         COUNT
--------- ----- ---------- ----------
        3 A     Male                1
        3 A     male                1
        3 c     FEMALE              1
        4 A     MALE                1
        4 B     MALE                1
        4 C     male                1
6 rows selected.
```

Q3) Create a view of Test1 marks of student USN ‘KUB24CSE644’ in all subjects

```
SQL> create view Test1 as
  select scode,test1
  from iamarks
  where usn='KUB24CSE678';
```

```
View created.
SQL>
SQL>  select * from Test1;

    SCODE      TEST1
----------     ----------
     2343         17
```

Q4) Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

Before executing the procedure STUDENT_AVG(NOTE:FINALIA column contents are null)

```
SQL> select * from iamarks;

USN            SCODE      SSID     TEST1     TEST2     TEST3    FINALIA
-----------  ----------  ----------  ----------  ----------  ----------  ----------
KUB24CSE637      2341        500         20          20          20
KUB24CSE613      2332        501         25          25          25
KUB24CSE689      2333        502         15          18          19
KUB23CSE666      2342        503         17          23          21
KUB24CSE624      2343        503         15          11          14
KUB24CSE681      2365        504         17          18           4
KUB24CSE644      2361        503         12          13          14
KUB24CSE644      2343        501         15          16          17
KUB24CSE678      2343        504         17          19          20
```

CREATION OF STORED PROCEDURE STUDENT_AVG

```
SQL> CREATE OR REPLACE PROCEDURE STUDENT_AVG
  2  IS
  3  CURSOR C_IAMARKS IS
  4
  5  SELECT GREATEST(TEST1,TEST2) AS A,GREATEST(TEST1,TEST3) AS B,
  6  GREATEST(TEST3,TEST2) AS C
  7  FROM IAMARKS
  8  WHERE FINALIA IS NULL
```

```
 9  FOR UPDATE;
10  C_A NUMBER;
11  C_B NUMBER;
12  C_C NUMBER;
13  C_SM NUMBER;
14  C_AV NUMBER;
15
16  BEGIN
17  OPEN C_IAMARKS;
18
19  LOOP
20  FETCH C_IAMARKS INTO C_A,C_B,C_C;
21
22  EXIT WHEN C_IAMARKS%NOTFOUND;
23
24  DBMS_OUTPUT.PUT_LINE(C_A||' '||C_B||' '||C_C);
25  IF(C_A!=C_B) THEN
26  C_SM:=C_A+C_B;
27  ELSE
28  C_SM:=C_A+C_C;
29  END IF;
30
31  C_AV:=C_SM/2;
32
33  DBMS_OUTPUT.PUT_LINE('SUM='||C_SM);
34  DBMS_OUTPUT.PUT_LINE('AVERAGE='||C_AV);
35
36  UPDATE IAMARKS
37  SET FINALIA=C_AV
38
39  WHERE CURRENT OF C_IAMARKS;
40  END LOOP;
41
42  CLOSE C_IAMARKS;
43  END STUDENT_AVG;
44  /

Procedure created.
```

```
SQL> BEGIN
  2  STUDENT_AVG;
  3  END;
  4  /
```

PL/SQL procedure successfully completed.

SQL> SELECT * FROM IAMARKS;

SQL> select * from iamarks;

USN	SCODE	SSID	TEST1	TEST2	TEST3	FINALIA
KUB24CSE637	2341	500	20	20	20	20
KUB24CSE613	2332	501	25	25	25	25
KUB24CSE689	2333	502	15	18	19	17.33
KUB23CSE666	2342	503	17	23	21	20.33
KUB24CSE624	2343	503	15	11	14	8.33
KUB24CSE681	2365	504	17	18	4	13
KUB24CSE644	2361	503	12	13	14	13
KUB24CSE644	2343	501	15	16	17	16
KUB24CSE678	2343	504	17	19	20	56

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 6th 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.scode = ia.scode and
 sub.sem = 4;
```

USN	SNAME	ADDRESS	PHONE	GENDER	CAT
KUB24CSE637	Santosh	Ballari	9986751168		outstanding
KUB23CSE666	RAJESH	MYSORE	9620078963	MALE	weak
KUB24CSE624	DEEPIKA	RAICHUR	11228934	FEMALE	average
KUB24CSE644	PAVAN	BALLARI	77886612	MALE	outstanding
KUB24CSE678	VIJAY	BALLARI	1122467	MALE	outstanding

LAB PROGRAM -5 COMPANY DATABASE

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

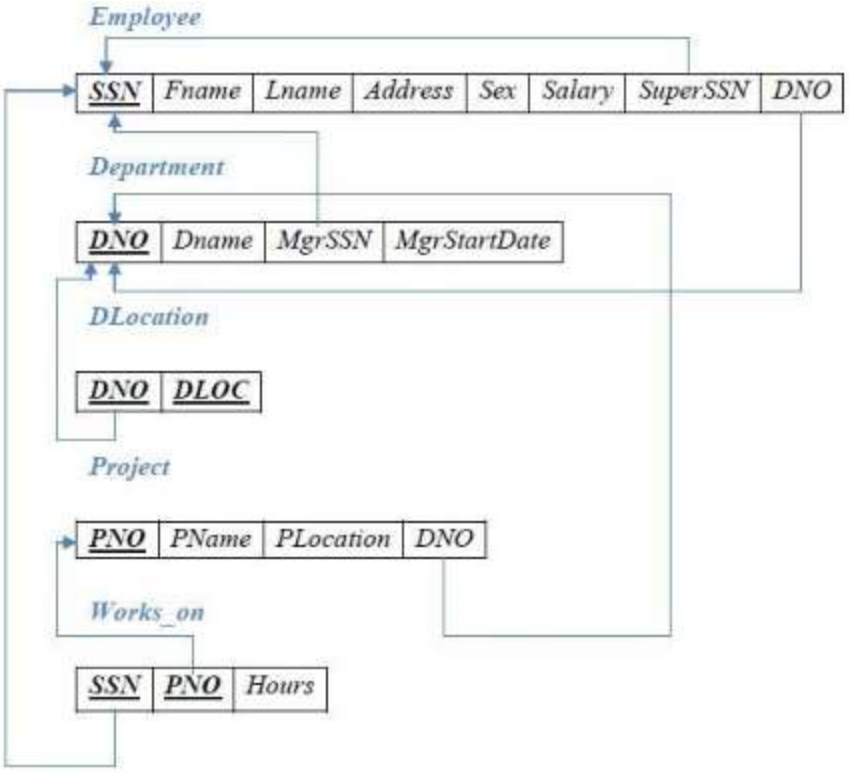
1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.

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.

SCHEMA-DIAGRAM



E-R DIAGRAM

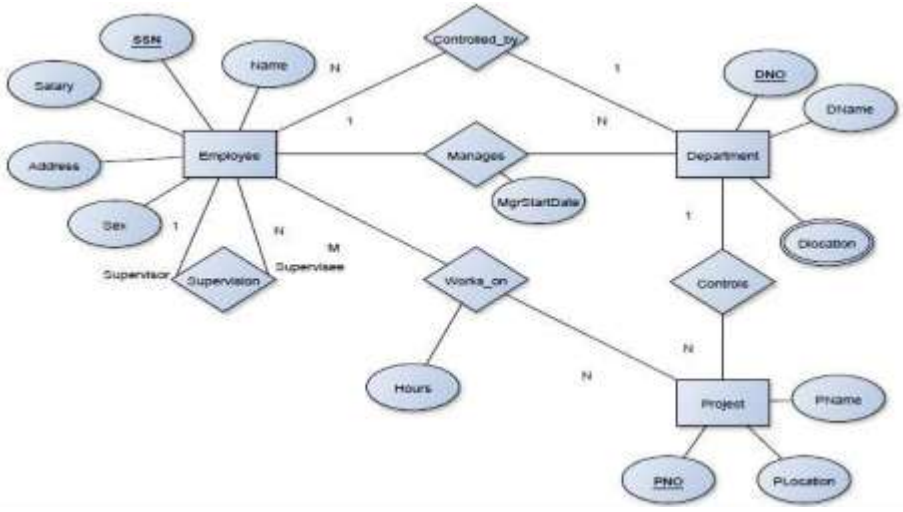


TABLE CREATION

TABLES CREATION

1. DEPARTMNET

```
SQL> create table department
  2  (dno number primary key,
  3  dname varchar(20)
  4  );

Table created.
```

2. EMPLOYEE

```
SQL> create table employee
  2  (ssn number primary key,
  3  name varchar(10),
  4  address varchar(20),
  5  gender varchar(10),
  6  salary number,
  7  dno number references department(dno) on delete cascade);

Table created.
```

Alter command to add foreign keys in employee and department tables

```
SQL> alter table department add(mgrssn number references employee(ssn)
  2  on delete cascade,
  3  mgr_start_date date);

Table altered.
```

```
SQL> alter table employee add
  2  (superssn number references employee(ssn) on delete cascade,
  3  dno number references department(dno) on delete cascade);

Table altered.
```

3. DLOCATION

```
SQL> create table dlocation
  2  (dno number references department(dno) on delete cascade,
  3  dlocation varchar(20) );

Table created.
```

4. PROJECT

```
SQL> create table project
  2  ( pno number primary key,
  3  pname varchar(20),
  4  plocation varchar(20),
  5   dno number references department(dno) on delete  cascade);

Table created.
```

5. Works_on

```
SQL>create table works_on
2(ssn number references employee(ssn) on delete cascade,
3 pno number references project(pno) on delete cascade,
 4.hrs number);
Table created
```

Description of tables

SQL> desc employee;

Name	Null?	Type
SSN	NOT NULL	NUMBER
NAME		VARCHAR2(10)
ADDRESS		VARCHAR2(20)
GENDER		VARCHAR2(10)
SALARY		NUMBER

SQL> desc department;

Name	Null?	Type
DNO	NOT NULL	NUMBER
DNAME		VARCHAR2(20)
MGRSSN		NUMBER
MSR_START_DATE		DATE

SQL> desc dlocation;

Name	Null?	Type
DNO		NUMBER
DLOCATION		VARCHAR2(20)

SQL> desc project;

Name	Null?	Type
PNO	NOT NULL	NUMBER
PNAME		VARCHAR2(20)

PLOCATION VARCHAR2(20) DNO NUMBER
SQL> desc works_on; Name Null? Type ---------- SSN NUMBER PNO NUMBER HRS NUMBER

INSERT COMMANDS

/ Note: use alter table commands to add mgrssn and mgrstart_date columns after insering values in employee table
INSERT INTO EMPLOYEE VALUES('&SSN','&NAME','&ADDRESS','&GENDER',&SALARY) Note: use alter table commands to add super_ssn and dno after entering the vales of ssn and dno
INSERT INTO DLOCATION VALUES(&DNO,'&DLOC');
INSERT INTO PROJECT VALUES(&PNO,'&PNAME','&PLOCATION','&DNO');
INSERT INTO WORKS_ON VALUES('&SSN',&PNO,&HRS);

TABLES –DISPLAY

EMPLOYEE TABLE

SQL> select * from employee;

SSN	NAME	ADDRESS	GENDER	SALARY	SUPERSSN	DNO
111	sam	blore	male	54637		
112	kim	mysore	female	65748		
113	bob	delhi	male	54637		
114	priya	mysore	female	65747		

115 smith blore male 53666
116 john bly male 54637
117 wong bly male 543656

Note Use update command to fill the superssn and dno columns

SQL> select * from employee;

SSN	NAME	ADDRESS	GENDER	SALARY	SUPERSSN	DNO
111	sam	Bangalore	male	54000	116	30
112	kishor	Mysore	male	65000	116	10
113	manisha	Delhi	female	54000	116	10
114	priya	mysore	female	66000	116	10
115	smith	mysore	male	53000	114	2
116	john	Bangalore	male	56444	117	10
117	krushi	Ballari	female	60000	null	40

SQL> select * from department;

DNO	DNAME	MGRSSN	MSR_START
10	sales	113	
20	HQ	115	
30	admin	111	
40	marketing	117	

SQL> update department set msr_start_date='12-mar-2000' where dno=20;

1 row updated.

SQL> update department set msr_start_date='1-may-2003' where dno=10;

1 row updated.

SQL> update department set msr_start_date='11-jan-2003' where dno=30;

```
1 row updated.

SQL> update department set msr_start_date='1-jan-2003' where dno=40;

1 row updated.
SQL> select * from department;

      DNO DNAME                MGRSSN MGR_START
---------- -------------------- ---------- ---------
        10 sales                   113 01-MAR-23
        20 HQ                      115 12-JAN-24
        30 Admin                   111 13-JAN-24
        40 marketing               117 23-FEB-24
        50 supermarket             112 13-JUN-24
```

```
insert into dlocation values(&dno,'&dlocation');
select  * from dlocation;

      DNO DLOCATION
---------- --------------------
        10 Bangalore
        20 Mumbai
        30 Bangalore
        40 Chennai
        50 Bangalore
        10 Ballari
```

```
insert into project values(&pno,'&pname','&plocation',&dno);
select * from project;
      PNO PNAME              PLOCATION            DNO
---------- -------------------- -------------------- ----------
      1234 productx            ballari                    10
         1 producty            Bangalore                  10
```

```
2 productz          Houston          20
3 IOT               Mysore           30
10 computer         Bangalore        30
20 IOT2             Mumbai           20
30 newbenefits      Stafford         40
```

```
SQL> select * from works_on;

     SSN        PNO        HRS
---------- ---------- ----------
     111         10          3
     112         20          3
     113         30          3
     114         20          3
     115       1234          3
     116         30          3
     117         20          3
     116          3          3
     114          1          4
     115          2          3
     116          3          2
     113         20          5

12 rows selected.
```

QUERIES

Q1) Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.

```
(select distinct pno
  from project p, department d,
  employee e where p.dno=d.dno and
```

```
 ssn=mgrssn and
 name='sam')
 union
 (select distinct p.pno
 from project p, works_on w,
 employee e where p.pno=w.pno and
  w.ssn=e.ssn and
   name='sam');
     PNO
----------
         3
         10
```

Q2) Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

```
select e.name, 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';
NAME        INCR_SAL
---------- ----------
john         62088.4
john         62088.4
```

Q3) Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
select sum(salary), max(salary), min(salary),avg(salary)
```

```
from employee e, department d
 where dname='marketing' and
  d.dno=e.dno;
SUM(SALARY)   MAX(SALARY)    MIN(SALARY)     AVG(SALARY)
---------- ---------- ---------- ----------------------------------------------
60000       60000                          60000                          60000
```

Q4) .Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

```
select name from
  employee e
  where not exists(
  (select pno
  from project
  where
  dno=5)
  minus
  (select pno
 from works_on w
 where e.ssn=w.ssn));
 NAME
----------
sam
kishor
manisha
priya
smith
john
krushi
```

Q5) 5. For each department that has more than two employees, retrieve the department number and the number of its employees who are making more than Rs. 60,000.

```
select dno,count(ssn)
from employee
where salary>60000 and dno
in(select dno
from employee
```

```
group by dno
having count(ssn)>=2)
group by dno;

 DNO COUNT(SSN)
---------- ----------
  10          2
```