HKBK COLLEGE OF ENGINEERING
(Affiliated to VTU, Belgaum and Approved by AICTE)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NBA Accredited Programme

LABORATORY MANUAL
DBMS LABORATORY WITH MINI PROJECT

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2021-2022)
18CSL58

HKBK COLLEGE OF ENGINEERING
Nagawara, Bangaluru -560 045
www.hkbkeducation.org

HKBK COLLEGE OF ENGINEERING
(Affiliated to VTU, Belgaum and Approved by AICTE)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Mission and Vision of the Institution

Mission

- To achieve academic excellence in science, engineering and technology through dedication to duty, innovation in teaching and faith in human values.
- To enable our students to develop into outstanding professional with high ethical standards to face the challenges of 21st century.
- To provide educational opportunities to the deprived and weaker section of the society to uplift their socio-economic status.

Vision

To empower students through wholesome education and enable the students to develop into highly qualified and trained professionals with ethics and emerge as responsible citizen with broad outlook to build a vibrant nation.

Mission and Vision of the CSE Department

Mission

- To provide excellent technical knowledge and computing skills to make the graduates globally competitive with professional ethics.
- To involve in research activities and be committed to lifelong learning to make positive contributions to the society.

Vision

To advance the intellectual capacity of the nation and the international community by imparting knowledge to graduates who are globally recognized as innovators, entrepreneur and competent professionals.

HKBK COLLEGE OF ENGINEERING
(Affiliated to VTU, Belgaum and Approved by AICTE)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Programme Educational Objectives

PEO-1	To provide students with a strong foundation in engineering fundamentals and in the computer science and engineering to work in the global scenario.
PEO-2	To provide sound knowledge of programming and computing techniques and good communication and interpersonal skills so that they will be capable of analyzing, designing and building innovative software systems.
PEO-3	To equip students in the chosen field of engineering and related fields to enable him to work in multidisciplinary teams.
PEO-4	To inculcate in students professional, personal and ethical attitude to relate engineering issues to broader social context and become responsible citizen.
PEO-5	To provide students with an environment for life-long learning which allow them to successfully adapt to the evolving technologies throughout their professional carrier and face the global challenges.

Programme Outcomes	
a.	**Engineering Knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
b.	**Problem Analysis: Identify,** formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences
c.	**Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
d.	**Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.
e.	**Modern Tool Usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an under- standing of the limitations.
f.	**The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.
g.	**Environment and Sustainability**: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.
h.	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.
i.	**Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and in multi disciplinary settings.

j.	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
k.	**Life-long Learning**: Recognize the need for and have the preparation and ability to engage in independent and life- long learning in the broadest context of technological change.
l.	**Project Management and Finance**: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
Programme Specific Outcomes	
m.	**Problem-Solving Skills:** An ability to investigate and solve a problem by analysis, interpretation of data, design and implementation through appropriate techniques,tools and skills.
n.	**Professional Skills**: An ability to apply algorithmic principles, computing skills and computer science theory in the modelling and design of computer-based systems.
o.	**Entrepreneurial Ability:** An ability to apply design, development principles and management skills in the construction of software product of varying complexity to become an entrepreneur

HKBK College of Engineering

Department of Computer Sciences and Engineering
Bangalore-560045
DBMS LABORATORY WITH MINI PROJECT

Course Objectives:

This course will enable students to

- Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.
- Strong practice in SQL programming through a variety of database problems.
- Develop database applications using front-end tools and back-end DBMS.

Descriptions (if any):

PART-A: SQL Programming (Max. Exam Mks. 50)

• Design, develop, and implement the specified queries for the following problems using Oracle, MySQL, MS SQL Server, or any other DBMS under LINUX/Windows environment.
• Create Schema and insert at least 5 records for each table. Add appropriate database constraints.

PART-B: Mini Project (Max. Exam Mks. 30)

• Use Java, C#, PHP, Python, or any other similar front-end tool. All applications must be demonstrated on desktop/laptop as a stand-alone or web based application (Mobile apps on Android/IOS are not permitted.)

Lab Experiments:

Part A: SQL Programming

1. Consider the following schema for a Library Database:
   BOOK(Book_id, Title, Publisher_Name, Pub_Year)
   BOOK_AUTHORS(Book_id, Author_Name)
   PUBLISHER(Name, Address, Phone)
   BOOK_COPIES(Book_id, Branch_id, No-of_Copies)
   BOOK_LENDING(Book_id, Branch_id, Card_No, Date_Out, Due_Date)
   LIBRARY_BRANCH(Branch_id, Branch_Name, Address)
   Write SQL queries to
   1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
   2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

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

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.

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.

5. 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).
   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.

Part B: Mini project

- For any problem selected, write the ER Diagram, apply ER-mapping rules, normalize the relations, and follow the application development process.
- Make sure that the application should have five or more tables, at least one trigger and one stored procedure, using suitable frontend tool.
- Indicative areas include; health care, education, industry, transport, supply chain, etc.

Course Outcomes:

The students should be able to:

- Create, Update and query on the database.
- Demonstrate the working of different concepts of DBMS
- Implement, analyze and evaluate the project developed for an application.

Conduction of Practical Examination:

1. All laboratory experiments from part A are to be included for practical examination.
2. Mini project has to be evaluated for 30 Marks as per 6(b).
3. Report should be prepared in a standard format prescribed for project work.
4. Students are allowed to pick one experiment from the lot.
5. Strictly follow the instructions as printed on the cover page of answer script.

6. Marks distribution:
    a. Part A: Procedure + Conduction + Viva:10 + 35 +5 =50 Marks
    b. Part B: Demonstration + Report + Viva voce = 15+10+05 = 30 Marks
7. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

PART-A: SQL Programming

Introduction

Data are known facts that can be recorded and that have implicit meaning. A **Database** is collection of related data. A **database management system (DBMS)** is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd. A relational database system contains one or more objects called tables. The data or information for the database are stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. Columns contain the column name, data type, and any other attributes for the column. Rows contain the records or data for the columns.

The name **SQL** is presently expanded as Structured Query Language. Originally, SQL was called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research as the interface for an experimental relational database system called SYSTEM R. SQL is now the standard language for commercial relational DBMSs.

SQL is a comprehensive database language: It has statements for data definitions, queries, and updates. Hence, it is both a DDL *and* a DML. In addition, it has facilities for defining views on the database, for specifying security and authorization, for defining integrity constraints, and for specifying transaction controls. It also has rules for embedding SQL statements into a general-purpose programming language such as Java, COBOL, or C/C++.

SQL Data Definition and Data Types

SQL uses the terms **table, row, and column** for the formal relational model terms **relation, tuple, and attribute** respectively.

The main SQL command for data definition is the CREATE statement, which can be used to create schemas, tables (relations) as well as other constructs such as views, assertions, and triggers.

A schema is created via the CREATE SCHEMA statement, which can include all the schema elements definitions.

Create schema company **authorization** ‘Jsmith’;

CREATE TABLE Command:

The **CREATE TABLE** command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints.

Create table table_name (
Attribute_name1 datatype [constraints],
Attribute_name2 datatype [constraints],
....
Attribute_namen datatype[constraints],
[integrity constraint1, … integrity_constaintk]
);

Create table department
(
Dname varchar(15) **not null**,
Dnumber int **not null**,
Mgr_ssn char(9) **not null**,
Mgr_start_date date,
Primary key (Dnumber),
Unique (Dname),
Foreign key (Mgr_ssn) **references** employee(Ssn)
);

Data Types:

- Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL
- Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
- Bit String: BLOB, CLOB
- Boolean: true, false, and null
- Date and Time: DATE (YYYY-MM-DD) TIME(HH:MM:SS)
- Timestamp: DATE + TIME

SQL Constraints:

1. **NOT NULL Constraint:** By default, a column can hold NULL values. The **NOT NULL** constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

   Dname VARCHAR(15) **NOT NULL**

2. **DEFAULT Constraint:** The DEFAULT constraint is used to provide a default value for a column. The default value will be added to all new records IF no other value is specified.

   college varchar(25) **DEFAULT 'HKBKCE'**

3. **UNIQUE Constraint:** The UNIQUE constraint ensures that all values in a column are different. Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns. A PRIMARY KEY constraint automatically has a UNIQUE constraint. However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

   ID int **NOT NULL UNIQUE**

4. **CHECK Constraint:** The CHECK constraint is used to limit the value range that can be placed in a column.

   age int **CHECK (age >=18)**

5. **PRIMARY KEY Constraint*:*** The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only one primary key, which may consist of single or multiple fields.

   PRIMARY KEY (Dnumber)

6. **FOREIGN KEY Constraint:** A FOREIGN KEY is a key used to link two tables together. A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

   FOREIGN KEY (Mgr_ssn) **REFERENCES** EMPLOYEE(Ssn)

   Referential Triggered Action Clause:

   Set NULL: Sets the column value to **NULL** when you delete the parent table row.

   Set DEFAULT: sets the column value to DEFAULT when you delete the parent table row.

   CASCADE: CASCADE will **propagate the change** when the parent changes. If you delete a row, rows in constrained tables that reference that row **will also be deleted**, etc.

   RESTRICT: RESTRICT causes you cannot delete a given **parent row** if a **child row** exists that references the value for that parent row.

   NO ACTION: NO ACTION and RESTRICT are very much alike. When an UPDATE or DELETE statement is executed on the referenced table, the DBMS verifies at the end of the statement execution that **none of the referential** relationships are **violated**. in short child row no concern if parent row **delete** or **update**.

An option must be qualified with either ON DELETE or ON UPDATE.

FOREIGN KEY (Mgr_ssn) **REFERENCES** EMPLOYEE(Ssn) **ON DELETE** SET DEFAULT **ON UPDATE** CASCADE);

The DROP Command:

The DROP command can be used to drop *named* schema elements, such as tables, domains, or constraints. One can also drop a schema. For example, if a whole schema is no longer needed, the DROP SCHEMA command can be used. There are two *drop* behavior options:

CASCADE and RESTRICT. For example, to remove the COMPANY database schema and all its tables, domains, and other elements, the CASCADE option is used as follows:

drop schema company cascade;

If the RESTRICT option is chosen in place of CASCADE, the schema is dropped only if it has no elements in it; otherwise, the DROP command will not be executed. To use the RESTRICT option, the user must first individually drop each element in the schema, then drop the schema itself.

Notice that the DROP TABLE command not only deletes all the records in the table if successful, but also removes the table definition from the catalog.

The ALTER Command

The definition of a base table or of other named schema elements can be changed by using the ALTER command. For base tables, the possible **alter table actions** include adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints.

Alter table employee **add column** job varchar(12);
Alter table employee **drop column** address **cascade**;
Alter table department **alter column** mgr_ssn **drop default**;
Alter table department **alter column** mgr_ssn **set default** '333445555';
Alter table employee **drop constraint** empsuperfk **cascade**;

Basic Retrieval Queries in SQL:

SQL has one basic statement for retrieving information from a database: the **SELECT** statement.

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

where
■ <attribute list> is a list of attribute names whose values are to be retrieved by the query.
■ <table list> is a list of the relation names required to process the query.
■ <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

In SQL, the basic logical comparison operators for comparing attribute values with one another and with literal constants are =, <, <=, >, >=, and <>.

Retrieve the birth date and address of the employee(s) whose nameis 'John B. Smith'.

select Bdate, Address
from employee
where Fname='John' and Minit='B' and Lname='Smith';

This query involves only the EMPLOYEE relation listed in the FROM clause. The query *selects* the individual EMPLOYEE tuples that satisfy the condition of the WHERE clause, then *projects* the result on the Bdate and Address attributes listed in the SELECT clause. The SELECT clause of SQL specifies the attributes whose values are to be retrieved, which are called the **projection attributes**, and the WHERE clause specifies the boolean condition that must be true for any retrieved tuple, which is known as the **selection condition**.

A query that involves only selection and join conditions plus projection attributes is known as a **select-project-join** query.

Retrieve the name and address of all employees who work for the 'Research' department.

```
select Fname, Lname, Address
from employee, department
where Dname='Research' and Dnumber=Dno;
```

In the WHERE clause, the condition Dname = 'Research' is a **selection condition** that chooses the particular tuple of interest in the DEPARTMENT table, because Dname is an attribute of DEPARTMENT. The condition Dnumber = Dno is called a **join condition**, because it combines two tuples: one from DEPARTMENT and one from EMPLOYEE, whenever the value of Dnumber in DEPARTMENT is equal to the value of Dno in EMPLOYEE.

For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
select E.Fname, E.Lname, S.Fname, S.Lname
from EMPLOYEE AS E, EMPLOYEE AS S
where E.Super_ssn=S.Ssn;
```

In this case, we are required to declare alternative relation names E and S, called **aliases** or **tuple variables**, for the EMPLOYEE relation. An alias can follow the keyword **AS,** or it can directly follow the relation name.

Unspecified WHERE Clause and Use of the Asterisk

A *missing* WHERE clause indicates no condition on tuple selection; hence, *all tuples* of the relation specified in the FROM clause qualify and are selected for the query result.

```
select Ssn
from employee;
```

To retrieve all the attribute values of the selected tuples, we do not have to list the attribute names explicitly in SQL; we just specify an *asterisk* (*), which stands for *all the attributes.*

```
select *
from employee
where Dno=5;
```

DISTINCT:

DISTINCT keyword in the SELECT clause, meaning that only distinct tuples should remain in the result.

select distinct Salary
from employee;

Set Operations:

SQL has directly incorporated some of the set operations from mathematical *set theory*. There are set union (UNION), set difference (**EXCEPT)** and set intersection (**INTERSECT**) operations.

The relations resulting from these set operations are sets of tuples; that is, *duplicate tuples are eliminated from the result.* These set operations apply only to *union-compatible relations*, so we must make sure that the two relations on which we apply the operation have the same attributes and that the attributes appear in the same order in both relations.

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

(**select distinct** Pnumber
from project, department, employee
where Dnum=Dnumber **and** Mgr_ssn=Ssn
and Lname='Smith')
union
(**select distinct** Pnumber
from project, works_on, employee
where Pnumber=Pno **and** Essn=Ssn
and Lname='Smith');

The first SELECT query retrieves the projects that involve a 'Smith' as manager of the department that controls the project, and the second retrieves the projects that involve a 'Smith' as a worker on the project. Notice that if several employees have the last name 'Smith', the project names involving any of them will be retrieved. Applying the UNION operation to the two SELECT queries gives the desired result.

SQL also has corresponding multiset operations, which are followed by the keyword **ALL** (UNION ALL, EXCEPT ALL, INTERSECT ALL). Their results are multisets (duplicates are not eliminated).

Substring Pattern Matching and Arithmetic Operators:

The first feature allows comparison conditions on only parts of a character string, using the **LIKE** comparison operator. This can be used for string **pattern matching**. Partial strings are specified using two reserved characters: % replaces an arbitrary number of zero or more characters, and the underscore (_) replaces a single character.

Retrieve all employees whose address is in Houston, Texas.

select Fname, Lname
from employee
where Address **like** '%Houston,TX%';

To retrieve all employees who were born during the 1950s, '5'must be the third character of the string (according to our format for date), so we use the value '_ _ 5 _____ ', with each underscore serving as a placeholder for an arbitrary character.

Find all employees who were born during the 1950s.

select Fname, Lname
from employee
where Bdate **like** '_ _ 5 _____ ';

If an underscore or % is needed as a literal character in the string, the character should be preceded by an *escape character*, which is specified after the string using the keyword ESCAPE.

Comparison operator Between

Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

select *
from employee
where (Salary **BETWEEN** 30000 **AND** 40000) and Dno = 5;

The condition (Salary **BETWEEN** 30000 **AND** 40000) is equivalent to the condition ((Salary >= 30000) **AND** (Salary <= 40000)).

Ordering of Query Results

SQL allows the user to order the tuples in the result of a query by the values of one or more of the attributes that appear in the query result, by using the **ORDER BY** clause.

The default order is in ascending order of values. We can specify the keyword **DESC** if we want to see the result in a descending order of values. The keyword **ASC** can be used to specify ascending order explicitly.

Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

select D.Dname, E.Lname, E.Fname, P.Pname
from department d, employee e, works_on w, project p
where D.Dnumber= E.Dno and E.Ssn= W.Essn and W.Pno= P.Pnumber
ORDER BY D.Dname, E.Lname, E.Fname;

GROUP BY

In many cases, we want to apply the aggregate functions to subgroups of tuples in a relation. Each subgroup of tuples consists of the set of tuples that have the same value for the grouping attribute(s). The function is applied to each subgroup independently. SQL has a GROUP BY-clause for specifying the grouping attributes, which must also appear in the SELECT-clause.

For each department, retrieve the department number, the number of employees in the department, and their average salary.

select dno, count (*), avg (salary)
from employee **group by** dno

The EMPLOYEE tuples are divided into groups. Each group having the same value for the grouping attribute DNO. The COUNT and AVG functions are applied to each such group of tuples separately. The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples. A join condition can be used in conjunction with grouping

THE HAVING-CLAUSE

Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions. The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project.

select pnumber, pname, count (*) **from** project,
works_on **where** pnumber=pno **group by** pnumber,
pname
having count (*) > 2

Nested Queries

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query.

Retrieve the name and address of all employees who work for the 'Research' department.

select fname, lname, address
from employee
where dno **in** (**select** dnumber
from department
where dname='Research');

Note: The nested query selects the number of the 'Research' department. The outer query selects an EMPLOYEE tuple if its DNO value is in the result of either nested query. The

comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V

In general, we can have several levels of nested queries. A reference to an unqualified attribute refers to the relation declared in the innermost nested query.

Correlated Queries

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated.

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated. The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

Retrieve the name of each employee who has a dependent with the same first name as the employee.

select e.fname, e.lname from employee as e where e.ssn **in** (select essn from dependent where essn=e.ssn and e.fname=dependent_name)

The nested query has a different result in the outer query. A query written with nested SELECT... FROM… WHERE... blocks and using the **= or IN** comparison operators can *always* be expressed as a single block query.

THE EXISTS FUNCTION

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. We can formulate the above Query in an alternative form that uses EXIST.

select fname, lname from employee
where **exists** (select * from dependent where ssn=essn and fname=dependent_name)

Retrieve the names of employees who have no dependents.

select fname, lname from employee
where **not exists**
(select * from dependent where ssn=essn)

Note: In this correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

EXPLICIT SETS

It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query.

Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

select distinct essn from works_on where pno **in (1, 2, 3)**

NULLS IN SQL QUERIES

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

Retrieve the names of all employees who do not have supervisors.

select fname, lname from employee
where superssn is null

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result.

Aggregate functions

Include COUNT, SUM, MAX, MIN, and AVG.

Find the maximum salary, the minimum salary, and the average salary among all employees.

SELECT **MAX (SALARY), MIN(SALARY), AVG(SALARY)**
FROM EMPLOYEE

Note: Some SQL implementations may not allow more than one function in the SELECT-clause

INSERT, DELETE, and UPDATE Statements in SQL:

In SQL, three commands can be used to modify the database: INSERT, DELETE, and UPDATE.

The INSERT Command

In its simplest form, INSERT is used to add a single tuple to a relation. We must specify the relation name and a list of values for the tuple. The values should be listed *in the same order* in which the corresponding attributes were specified in the CREATE TABLE command.

insert into employee
values ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);

A second form of the INSERT statement allows the user to specify explicit attribute names that correspond to the values provided in the INSERT command. This is useful if a relation has many attributes but only a few of those attributes are assigned values in the new tuple.

Insert into employee (Fname, Lname, Dno, Ssn)
values ('Richard', 'Marini', 4, '653298653');

The DELETE Command

The DELETE command removes tuples from a relation. It includes a WHERE clause, similar to that used in an SQL query, to select the tuples to be deleted. Tuples are explicitly deleted from only one table at a time. However, the deletion may propagate to tuples in other relations if *referential triggered actions* are specified in the referential integrity constraints of the DDL.

delete from employee
where Lname='Brown';

Depending on the number of tuples selected by the condition in the WHERE clause, zero, one, or several tuples can be deleted by a single DELETE command. A missing WHERE clause specifies that all tuples in the relation are to be deleted; however, the table remains in the database as an empty table.

DELETE FROM EMPLOYEE;

The UPDATE Command

The **UPDATE** command is used to modify attribute values of one or more selected tuples. As in the DELETE command, a WHERE clause in the UPDATE command selects the tuples to be modified from a single relation. However updating a primary key value may propagate to the foreign key values of tuples in other relations if such a *referential triggered action* is specified in the referential integrity constraints of the DDL.

An additional **SET** clause in the UPDATE command specifies the attributes to be modified and their new values.

For example, to change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

Update project
set Plocation = 'Bellaire', Dnum = 5
where Pnumber=10;

Several tuples can be modified with a single UPDATE command. An example is to give all employees in the 'Research' department a 10 percent raise in salary,

Update employee
set Salary = Salary * 1.1
where Dno = 5;

It is also possible to specify NULL or DEFAULT as the new attribute value. Notice that each UPDATE command explicitly refers to a single relation only. To modify multiple relations, we must issue several UPDATE commands.

VIEWS IN SQL

1. A view is a single *virtual table* that is derived from other tables. The other tables could be base tables or previously defined view.
2. Allows for limited update operations Since the table may not physically be stored
3. Allows full query operations
4. A convenience for expressing certain operations
5. A view does not necessarily exist in physical form, which limits the possible update operations that can be applied to views.

Additional features of SQL

SQL has various techniques for writing programs in various programming languages that include SQL statements to access one or more databases. These include embedded (and dynamic) SQL, SQL/CLI (Call Level Interface) and its predecessor ODBC (Open Data Base Connectivity), and SQL/PSM (Persistent Stored Modules).

SQL has transaction control commands. These are used to specify units of database processing for concurrency control and recovery purposes. SQL has language constructs for specifying the *granting and revoking of privileges* to users. Privileges typically correspond to the right to use certain SQL commands to access certain relations. Each relation is assigned an owner, and either the owner or the DBA staff can grant to selected users the privilege to use an SQL statement—such as SELECT INSERT, DELETE, or UPDATE—to access the relation. In addition, the DBA staff can grant the privileges to create schemas, tables, or views to certain users. These SQL commands—called **GRANT** and **REVOKE**.

SQL has language constructs for creating triggers. These are generally referred to as **active database** techniques, since they specify actions that are automatically triggered by events such as database updates.

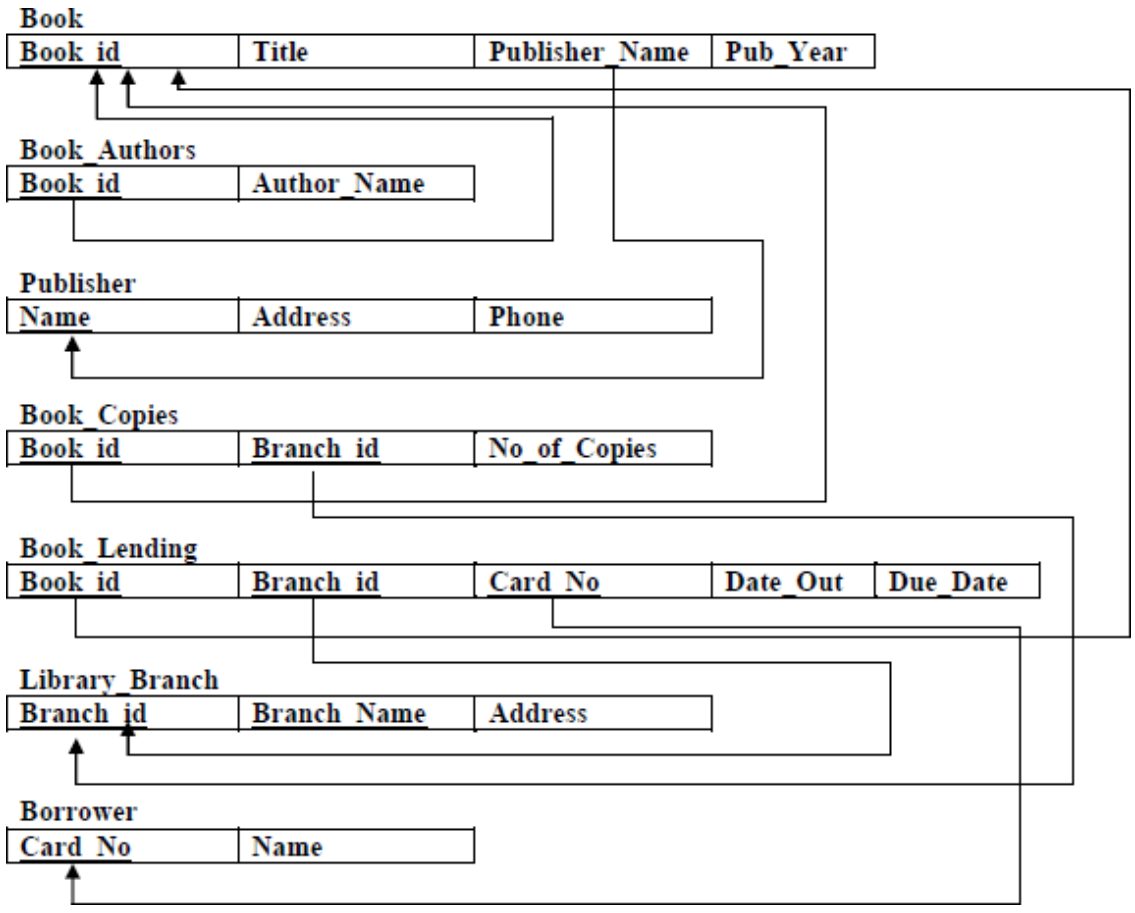
1. Consider the following schema for a Library Database:

BOOK(Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS(Book_id, Author_Name)
PUBLISHER(Name, Address, Phone)
BOOK_COPIES(Book_id, Branch_id, No-of_Copies)
BOOK_LENDING(Book_id, Branch_id, Card_No, Date_Out, Due_Date)
LIBRARY_BRANCH(Branch_id, Branch_Name, Address)

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

Schema Diagram:




```
CREATE TABLE Publisher (
Name varchar(200) PRIMARY KEY,
Address varchar(400),
Phone int );

CREATE TABLE Book (
 Book_id int PRIMARY KEY,
Title varchar(200),
Publisher_Name varchar(200),
Pub_Year int,
```

```
FOREIGN KEY (Publisher_Name) REFERENCES Publisher(Name) ON DELETE SET
NULL );

CREATE TABLE Book_Authors (
Book_id int NOT NULL,
Author_Name varchar(200) NOT NULL,
PRIMARY KEY (Book_id, Author name),
FOREIGN KEY (Book_id) REFERENCES Book(Book_id) ON DELETE CASCADE );

CREATE TABLE Library_Branch (
Branch_id int PRIMARY KEY,
Branch_name varchar(200) NOT NULL,
Address varchar(400) );

CREATE TABLE Book_Copies (
Book_id int NOT NULL,
Branch_id int NOT NULL,
No_of_copies int DEFAULT 1,
PRIMARY KEY (Book_id, Branch_id),
FOREIGN KEY (Book_id) REFERENCES Book(Book_id) ON DELETE CASCADE,
FOREIGN KEY (Branch_id) REFERENCES Library_Branch(Branch_id) ON DELETE
CASCADE);

CREATE TABLE Borrower (
Card_no int PRIMARY KEY,
Name varchar(200) NOT NULL
);

CREATE TABLE Book_Lending (
Book_id int NOT NULL,
Branch_id int NOT NULL,
Card_no int NOT NULL,
Date_out Date,
Due_date Date,
PRIMARY KEY (Book_id, Branch_id, Card no),
FOREIGN KEY (Book_id) REFERENCES Book(Book_id) ON DELETE CASCADE,
FOREIGN KEY (Branch_id) REFERENCES Library Branch(Branch_id) ON DELETE
CASCADE,
FOREIGN KEY (Card_no) REFERENCES Borrower(Card_no) ON DELETE CASCADE
);
```

Publisher table:

```
INSERT INTO `publisher` (`Name`, `Address`, `Phone`) VALUES ('Pearson', 'New Delhi',
'8976789543');
INSERT INTO `publisher` (`Name`, `Address`, `Phone`) VALUES ('PHI', 'New Delhi',
'9876785333');
INSERT INTO `publisher` (`Name`, `Address`, `Phone`) VALUES ('Tata McGraw-Hill',
'New Delhi', '9876785432');
INSERT INTO `publisher` (`Name`, `Address`, `Phone`) VALUES ('Technical
Publications', 'Pune', '9823451678');
```

INSERT INTO `publisher` (`Name`, `Address`, `Phone`) VALUES ('Universities Press', 'Hyderabad', '9123214566');

Book table:

INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('1', 'Web Engineering', 'Tata McGraw-Hill', '2008');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('2', 'Unix System Programming', 'Technical Publications', '2013');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('3', 'Design of UNIX OS', 'PHI', '2008');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('4', 'Formal Languages & Automata Theory', 'Technical Publications', '2012');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('5', 'Fundamentals of DS in C', 'Universities Press', '2008');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('6', 'Introduction to Automata Theory', 'Pearson', '2006');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('7', 'Operating Systems', 'Pearson', '2003');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('8', 'OOAD', 'Tata McGraw-Hill', '2012');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('9', 'OOAD', 'Universities Press', '2012');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('10', 'OS', 'Technical Publications', '2012');
INSERT INTO `book` (`Book_id`, `Title`, `Publisher_Name`, `Pub_Year`) VALUES ('11', 'PCD', 'Pearson', '2013');

book_authors table:

INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('1', 'Roger S Pressman');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('2', 'I A Dhotre');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('3', 'Maurice J Bach');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('4', 'A A Puntambekar');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('5', 'Horowitz');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('6', 'Hopcroft');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('7', 'Deitel');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('8', 'Simon Bennett');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('9', 'Brahma Dathan');
INSERT INTO `book_authors` (`Book_id`, `Author_Name`) VALUES ('10', 'I A Dhotre');

library_branch table:

INSERT INTO `library_branch` (`Branch_id`, `Branch_name`, `Address`) VALUES ('101', 'BranchA', 'Nagawara');

INSERT INTO `library_branch` (`Branch_id`, `Branch_name`, `Address`) VALUES ('102', 'BranchB', 'KG Halli');
INSERT INTO `library_branch` (`Branch_id`, `Branch_name`, `Address`) VALUES ('103', 'BranchC', 'Indira Nagar');
INSERT INTO `library_branch` (`Branch_id`, `Branch_name`, `Address`) VALUES ('104', 'BranchD', 'M G Road');
INSERT INTO `library_branch` (`Branch_id`, `Branch_name`, `Address`) VALUES ('105', 'BranchE', 'K R Circle');

book_copies:

INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('1', '101', '2');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('1', '104', '3');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('2', '102', '10');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('3', '103', '5');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('4', '105', '6');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('5', '101', '7');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('6', '102', '10');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('6', '105', '8');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('7', '104', '12');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('8', '103', '15');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('9', '102', '6');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('10', '103', '2');
INSERT INTO `book_copies` (`Book_id`, `Branch_id`, `No_of_copies`) VALUES ('10', '101', '4');

Borrower table:

INSERT INTO `borrower` (`Card_no`, `Name`) VALUES ('201', 'John Smith');
INSERT INTO `borrower` (`Card_no`, `Name`) VALUES ('202', 'Franklin');
INSERT INTO `borrower` (`Card_no`, `Name`) VALUES ('203', 'Joyce');
INSERT INTO `borrower` (`Card_no`, `Name`) VALUES ('204', 'James');
INSERT INTO `borrower` (`Card_no`, `Name`) VALUES ('205', 'Rose');

book_lending table:

INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('1', '101', '201', '2017-07-26', '2017-08-04');

```
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('2', '102', '201', '2017-01-17', '2017-07-27');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('5', '101', '202', '2017-04-11', '2017-04-21');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('6', '102', '204', '2017-03-06', '2017-07-16');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('7', '104', '203', '2016-12-05', '2016-12-15');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('10', '101', '205', '2017-07-04', '2017-07-14');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('4', '105', '202', '2017-01-02', '2017-07-12');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('8', '103', '201', '2017-05-09', '2017-05-19');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('3', '103', '203', '2017-06-01', '2017-06-10');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('10', '103', '205', '2017-02-01', '2017-07-10');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('6', '105', '204', '2016-11-02', '2016-11-12');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('1', '104', '202', '2017-02-10', '2017-02-20');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('9', '102', '201', '2017-04-01', '2017-04-11');
INSERT INTO `book_lending` (`Book_id`, `Branch_id`, `Card_no`, `Date_out`, `Due_date`) VALUES ('2', '102', '202', '2017-05-02', '2017-05-12');
```

Table: Book

Book_id	Title	Publisher_Name	Pub_Year
1	Web Engineering	Tata McGraw-Hill	2008
2	Unix System Programming	Technical Publications	2013
3	Design of UNIX OS	PHI	2008
4	Formal Languages & Automata Theory	Technical Publications	2012
5	Fundamentals of DS in C	Universities Press	2008
6	Introduction to Automata Theory	Pearson	2006
7	Operating Systems	Pearson	2003
8	OOAD	Tata McGraw-Hill	2012
9	OOAD	Universities Press	2012
10	OS	Technical Publications	2012
11	PCD	Pearson	2013

Table: Book_Authors

Book_id	Author_Name
1	Roger S Pressman
2	I A Dhotre
3	Maurice J Bach
4	A A Puntambekar
5	Horowitz
6	Hopcroft
7	Deitel
8	Simon Bennett
9	Brahma Dathan
10	I A Dhotre
11	Dennis Ritchie

Table:Book_copies

Book_id	Branch_id	No_of_copies
1	101	2
1	104	3
2	102	10
3	103	5
4	105	6
5	101	7
6	102	10
6	105	8
7	104	12
8	103	15
9	102	6
10	101	4
10	103	2

Table: Book_lending

Book_id	Branch_id	Card_no	Date_out	Due_date
1	101	201	2017-07-26	2017-08-04
1	104	202	2017-02-10	2017-07-20
2	102	201	2017-01-17	2017-07-27
2	102	202	2017-05-02	2017-05-12
3	103	203	2017-06-01	2017-06-10
4	105	202	2017-01-02	2017-07-12
5	101	202	2017-04-11	2017-04-21
6	102	204	2017-03-06	2017-07-16
6	105	204	2016-11-02	2016-11-12
7	104	203	2016-12-05	2016-12-15
8	103	201	2017-05-09	2017-05-19
9	102	201	2017-04-01	2017-04-11
10	101	205	2017-07-04	2017-07-14
10	103	205	2017-02-01	2017-02-10

Table:Borrower

Card_no	Name
201	John Smith
202	Franklin
203	Joyce
204	James
205	Rose

Table: library_branch

Branch_id	Branch_name	Address
101	BranchA	Nagawara
102	BranchB	KG Halli
103	BranchC	Indira Nagar
104	BranchD	M G Road
105	BranchE	K R Circle

Table: Publisher

Name	Address	Phone
Pearson	New Delhi	8976789543
PHI	New Delhi	9876785333
Tata McGraw-Hill	New Delhi	9876785432
Technical Publications	Pune	9823451678
Universities Press	Hyderabad	9123214566

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

select branch_name,c.Book_id,title,publisher_name,author_name, No_of_copies FROM book_copies c, book b, book_authors a,library_branch l where b.book_id=c.book_id and a.book_id=b.book_id and l.branch_id=c.Branch_id group by c.Branch_id,c.Book_id;

branch_name	Book_id	title	publisher_name	author_name	No_of_copies
BranchA	1	Web Engineering	Tata McGraw-Hill	Roger S Pressman	2
BranchA	5	Fundamentals of DS in C	Universities Press	Horowitz	7
BranchA	10	OS	Technical Publications	I A Dhotre	4
BranchB	2	Unix System Programming	Technical Publications	I A Dhotre	10
BranchB	6	Introduction to Automata Theory	Pearson	Hopcroft	10
BranchB	9	OOAD	Universities Press	Brahma Dathan	6
BranchC	3	Design of UNIX OS	PHI	Maurice J Bach	5
BranchC	8	OOAD	Tata McGraw-Hill	Simon Bennett	15
BranchC	10	OS	Technical Publications	I A Dhotre	2
BranchD	1	Web Engineering	Tata McGraw-Hill	Roger S Pressman	3
BranchD	7	Operating Systems	Pearson	Deitel	12
BranchE	4	Formal Languages & Automata Theory	Technical Publications	A A Puntambekar	6
BranchE	6	Introduction to Automata Theory	Pearson	Hopcroft	8

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

SELECT b.card_no,b.name FROM book_lending l,borrower b where b.card_no=l.card_no and date_out between '2017-01-01' and '2017-07-01' group by l.card_no having count(l.card_no)>3

card_no	name
202	Franklin

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=11;
select * from book;

Book_id	Title	Publisher_Name	Pub_Year
1	Web Engineering	Tata McGraw-Hill	2008
2	Unix System Programming	Technical Publications	2013
3	Design of UNIX OS	PHI	2008
4	Formal Languages & Automata Theory	Technical Publications	2012
5	Fundamentals of DS in C	Universities Press	2008
6	Introduction to Automata Theory	Pearson	2006
7	Operating Systems	Pearson	2003
8	OOAD	Tata McGraw-Hill	2012
9	OOAD	Universities Press	2012
10	OS	Technical Publications	2012

select * from book_authors;

Book_id	Author_Name
1	Roger S Pressman
2	I A Dhotre
3	Maurice J Bach
4	A A Puntambekar
5	Horowitz
6	Hopcroft
7	Deitel
8	Simon Bennett
9	Brahma Dathan
10	I A Dhotre

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

select * from book where Pub_Year = '2012';

Book_id	Title	Publisher_Name	Pub_Year
4	Formal Languages & Automata Theory	Technical Publications	2012
8	OOAD	Tata McGraw-Hill	2012
9	OOAD	Universities Press	2012
10	OS	Technical Publications	2012

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

create view books as select c.book_id,title,author_name,publisher_name,pub_year, sum(No_of_copies) from book b,book_authors a,book_copies c where c.book_id=a.book_id and a.book_id=b.book_id group by book_id;

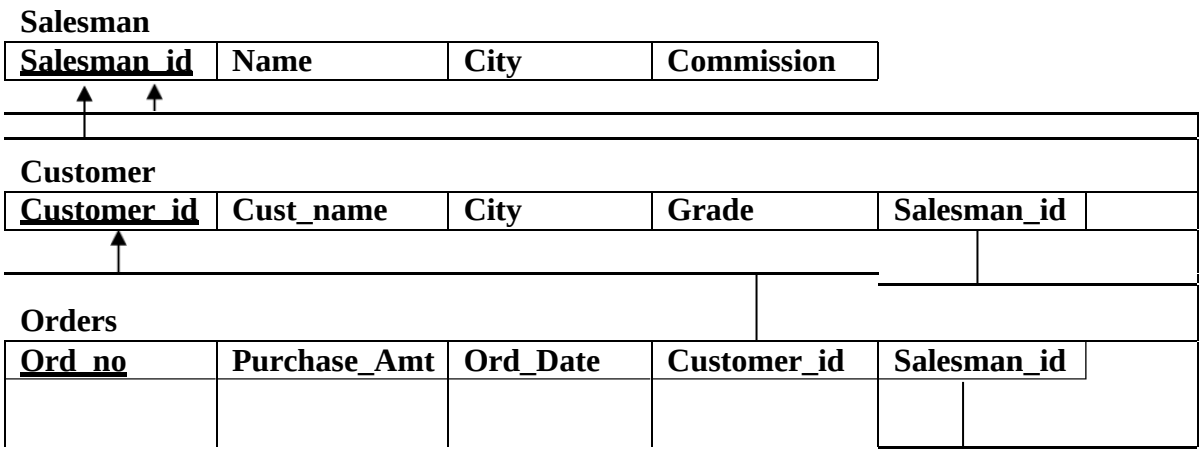
select * from books;

book_id	title	author_name	publisher_name	pub_year	sum(No_of_copies)
1	Web Engineering	Roger S Pressman	Tata McGraw-Hill	2008	5
2	Unix System Programming	I A Dhotre	Technical Publications	2013	10
3	Design of UNIX OS	Maurice J Bach	PHI	2008	5
4	Formal Languages & Automata Theory	A A Puntambekar	Technical Publications	2012	6
5	Fundamentals of DS in C	Horowitz	Universities Press	2008	7
6	Introduction to Automata Theory	Hopcroft	Pearson	2006	18
7	Operating Systems	Deitel	Pearson	2003	12
8	OOAD	Simon Bennett	Tata McGraw-Hill	2012	15
9	OOAD	Brahma Dathan	Universities Press	2012	6
10	OS	I A Dhotre	Technical Publications	2012	6

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

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

Schema Diagram:




```
create table salesman(
salesman_id int primary key,
name varchar(45),
city varchar(30),
commission float
);

Create table customer(
Customer_id int primary key,
Cust_name varchar(35),
City varchar(25),
Grade int,
Salesman_id int,
Foreign key (salesman_id) REFERENCES salesman(salesman_id) ON DELETE
CASCADE
);

Create table orders
(
Ord_no int primary key,
```

```
Purchase_amt float,
Ord_date date,
Customer_id int,
Salesman_id int,
Foreign key (salesman_id) REFERENCES salesman(salesman_id) ON DELETE
CASCADE,
Foreign key (customer_id) REFERENCES customer_id(customer_id) ON DELETE
CASCADE
);
```

Salesman table:

```
INSERT INTO salesman (salesman_id, name, city, commission) VALUES (5001, 'James',
'Bangalore', 0.15);
INSERT INTO salesman (salesman_id, name, city, commission) VALUES (5002, 'John',
'Bangalore', 0.13);
INSERT INTO salesman (salesman_id, name, city, commission) VALUES (5003, 'Alex',
'Chennai', 0.11);
INSERT INTO salesman (salesman_id, name, city, commission) VALUES (5004, 'Lyon',
'Delhi', 0.14);
INSERT INTO salesman (salesman_id, name, city, commission) VALUES (5001, 'Nail',
'Delhi', 0.12);
```

Customer table:

```
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3001, 'Davis', 'Bangalore', 200, 5001);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3002, 'Rimando', 'Bangalore', 100, 5001);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3003, 'Johns', 'Delhi', 300, 5005);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3004, 'Brainard', 'Chennai', 200, 5002);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3005, 'Zusi', 'Chennai', 100, 5004);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3006, 'Shaini', 'Bangalore', 200, 5004);
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id) VALUES
(3007, 'Smith', 'Mumbai', 400, 5001);
```

Orders table:

```
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7001, 150.5, '2017-01-05', 3005, 5004);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7002, 270.5, '2017-02-10', 3001, 5001);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7003, 110.5, '2016-08-17', 3002, 5001);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7004, 948.5, '2016-09-10', 3005, 5004);
```

INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7005, 2400, '2017-02-10', 3002, 5001);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7006, 349, '2017-02-10', 3004, 5002);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7007, 5500, '2017-02-10', 3006, 5004);
INSERT INTO orders (ord_no, purchase_amt, ord_date, customer_id, salesman_id)
VALUES (7008, 65, '2017-01-05', 3004, 5002);

Table: Salesman

salesman_id	name	city	commission
5001	James	Bangalore	0.15
5002	John	Bangalore	0.13
5003	Alex	Chennai	0.11
5004	Lyon	Delhi	0.14
5005	Nail	Delhi	0.12

Table: Customer

customer_id	cust_name	city	grade	salesman_id
3001	Davis	Bangalore	200	5001
3002	Rimando	Bangalore	100	5001
3003	Johns	Delhi	300	5005
3004	Brainard	Chennai	200	5002
3005	Zusi	Chennai	100	5004
3006	Shaini	Bangalore	200	5004
3007	Smith	Mumbai	400	5001

Table: Orders

ord_no	purchase_amt	ord_date	customer_id	salesman_id
7001	150.5	2017-01-05	3005	5004
7002	270.65	2017-02-10	3001	5001
7003	110.5	2016-08-17	3002	5001
7004	948.5	2016-09-10	3005	5004
7005	2400	2017-02-10	3002	5001
7006	349	2017-02-10	3004	5002
7007	5500	2017-02-10	3006	5004
7008	65	2017-01-05	3004	5002

1. **Count the customers with grades above Bangalore's average.**

select count(*)No_of_Customers from customer where grade > (select avg(grade) from customer where city='Bangalore');

No_of_Customers
5

2. **Find the name and numbers of all salesman who had more than one customer.**

select salesman_id,name from salesman where salesman_id in (select salesman_id from customer group by salesman_id having count(*) >1);

salesman_id	name
5001	James
5004	Lyon

3. **List all the salesman and indicate those who have and don't have customers in their cities (Use UNION operation.)**

   select s.salesman_id from salesman s,customer c where s.salesman_id=c.salesman_id and s.city=c.city UNION select s.salesman_id from salesman s,customer c where s.salesman_id=c.salesman_id and s.city!=c.city;

salesman_id
5001
5005
5002
5004

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

   create view maxorder as select`ord_date`,max(`purchase_amt`)Highest_Sales,o.salesman_id,name from orders o,salesman s where o.`salesman_id`=s.salesman_id group by`ord_date`;

   select * from maxorder;

ord_date	Highest_Sales	salesman_id	name
2016-08-17	110.5	5001	James
2016-09-10	948.5	5004	Lyon
2017-01-05	150.5	5002	John
2017-02-10	5500	5001	James

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

INSERT INTO `salesman` (`salesman_id`, `name`, `city`, `commission`) VALUES ('1000', 'Livingstone', 'Hyderabad', '0.09');
INSERT INTO `customer` (`customer_id`, `cust_name`, `city`, `grade`, `salesman_id`) VALUES ('3008', 'Gabriel', 'Hyderabad', '150', '1000');
INSERT INTO `orders` (`ord_no`, `purchase_amt`, `ord_date`, `customer_id`, `salesman_id`) VALUES ('7009', '400', '2017-07-28', '3008', '1000');

Delete from salesman where salesman_id=1000;

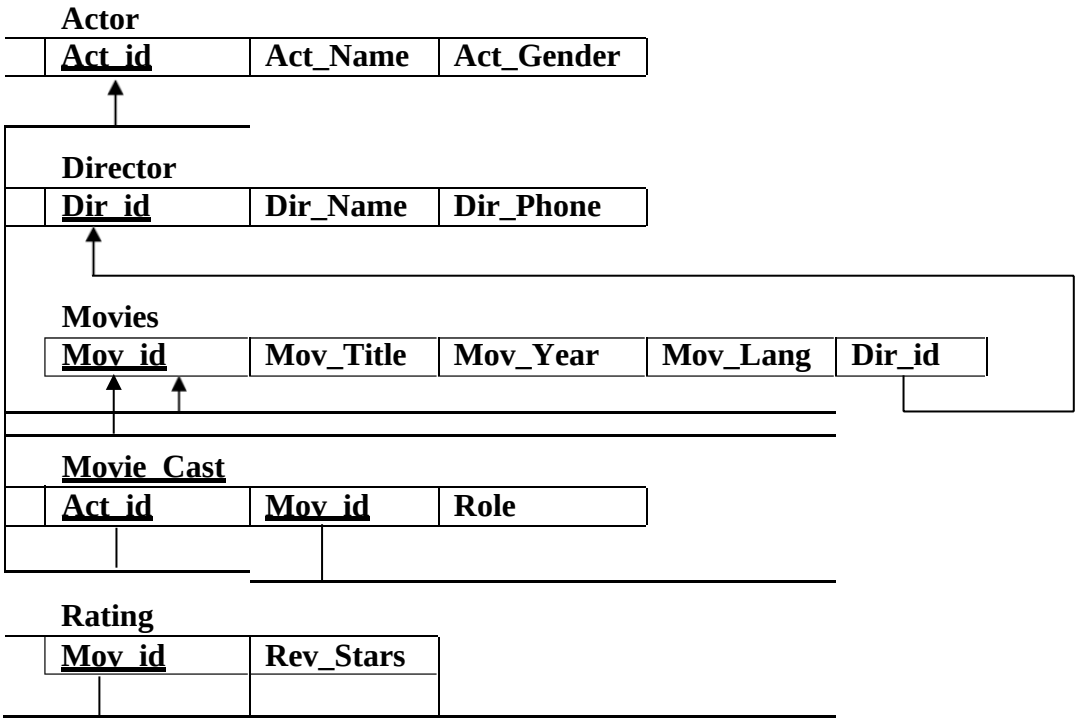
3. Consider the schema for Movie Database:

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.

Schema Diagram:




```
CREATE TABLE Actor
(
        Act_Id INT,
        Act_Name VARCHAR(25),
        Act_Gender CHAR(1),
        PRIMARY KEY(Act_Id)
);
```

```
CREATE TABLE Director
 (
        Dir_Id INT NOT NULL,
        Dir_Name VARCHAR(25) NOT NULL,
        Dir_Phone INT NOT NULL,
        PRIMARY KEY (Dir_Id)
);

CREATE TABLE Movies
(
        Mov_Id INT NOT NULL,
        Mov_Title VARCHAR(50) NOT NULL,
        Mov_Year INT NOT NULL,
        Mov_Lang VARCHAR(15) NOT NULL,
        Dir_Id INT NOT NULL,
        PRIMARY KEY (Mov_Id),
        FOREIGN key (Dir_Id) references Director(Dir_Id) on DELETE CASCADE
);


CREATE TABLE Movie_Cast
(
        Act_id INT NOT NULL ,
        Mov_Id INT NOT NULL,
        Role VARCHAR(45) NOT NULL ,
        PRIMARY KEY (Act_id, Mov_Id),
        FOREIGN KEY (Act_Id) REFERENCES Actor(Act_Id) on DELETE CASCADE,
        FOREIGN KEY (Mov_Id) REFERENCES movies(Mov_Id) on DELETE
CASCADE
);


CREATE TABLE Rating
(
        Mov_Id  INT NOT NULL ,
        Rev_Stars INT NOT NULL,
        PRIMARY KEY (Mov_Id),
        FOREIGN KEY (Mov_Id) REFERENCES Movies(Mov_Id) on DELETE
        CASCADE
);
```

Actor table:

```
INSERT INTO `actor` (`Act_Id`, `Act_Name`, `Act_Gender`) VALUES ('101', 'James',
'M'), ('102', 'Deborah', 'F'), ('103', 'Peter', 'M'), ('104', 'Robert', 'M'), ('105', 'Murray', 'M');
INSERT INTO `actor` (`Act_Id`, `Act_Name`, `Act_Gender`) VALUES ('106', 'Harrison',
'M'), ('107', 'Nicole', 'F'), ('108', 'Stephen', 'M'), ('109', 'Jack', 'M'), ('110', 'Kate', 'F');
```

Director table:

INSERT INTO `director` (`Dir_Id`, `Dir_Name`, `Dir_Phone`) VALUES ('201', 'Alfred', '675409'), ('202', 'Jack', '689543'), ('203', 'David', '660908'), ('204', 'Michael', '656432'), ('205', 'Milos', '600944');
INSERT INTO `director` (`Dir_Id`, `Dir_Name`, `Dir_Phone`) VALUES ('206', 'Stanley', '677543'), ('207', 'Roman', '660089');

Movies table:

INSERT INTO `movies` (`Mov_Id`, `Mov_Title`, `Mov_Year`, `Mov_Lang`, `Dir_Id`) VALUES ('1', 'Vertigo', '1994', 'English', '201'), ('2', 'Innocents', '1997', 'English', '201'), ('3', 'Deer Hunter', '1972', 'English', '202'), ('4', 'Eyes Wid Shut', '2002', 'English', '202'), ('5', 'Wings', '2016', 'English', '203'), ('6', 'Usual Suspects', '2006', 'English', '204'), ('7', 'Samurai', '2017', 'English', '205'), ('8', 'The Prestige', '2016', 'English', '206'), ('9', 'American Beauty', '2015', 'English', '201'), ('10', 'Walls', '2000', 'English', '207');

Movie_cast table:

INSERT INTO `movie_cast` (`Act_id`, `Mov_Id`, `Role`) VALUES ('101', '1', 'James'), ('101', '6', 'Fero'), ('101', '2', 'Eddie'), ('102', '5', 'July'), ('103', '3', 'John'), ('104', '7', 'Adam'), ('105', '8', 'Manus'), ('106', '1', 'Rick'), ('107', '8', 'Rose'), ('107', '9', 'Sam'), ('108', '1', 'Rock'), ('108', '5', 'Bobby'), ('109', '10', 'Ed'), ('110', '4', 'Cathie');

Rating table:

INSERT INTO `rating` (`Mov_Id`, `Rev_Stars`) VALUES ('1', '3'), ('3', '2'), ('4', '5'), ('5', '4'), ('6', '3'), ('7', '2'), ('8', '1'), ('9', '5'), ('10', '4');

Table Actor:

Act_Id	Act_Name	Act_Gender
101	James	M
102	Deborah	F
103	Peter	M
104	Robert	M
105	Murray	M
106	Harrison	M
107	Nicole	F
108	Stephen	M
109	Jack	M
110	Kate	F

Table: Director

Dir_Id	Dir_Name	Dir_Phone
201	Hitchcock	675409
202	Jack	689543
203	David	660908
204	Michael	656432
205	Milos	600944
206	Stanley	677543
207	Roman	660089

Table: Movies

Mov_Id	Mov_Title	Mov_Year	Mov_Lang	Dir_Id
1	Vertigo	1994	English	201
2	Innocents	1997	English	201
3	Deer Hunter	1972	English	202
4	Eyes Wid Shut	2002	English	202
5	Wings	2016	English	203
6	Usual Suspects	2006	English	204
7	Samurai	2017	English	205
8	The Prestige	2016	English	206
9	American Beauty	2015	English	201
10	Walls	2000	English	207

Table: Movie_cast

Act_id	Mov_Id	Role
101	1	James
101	2	Eddie
101	6	Fero
102	5	July
103	3	John
104	7	Adam
105	8	Manus
106	1	Rick
107	8	Rose
107	9	Sam
108	1	Rock
108	5	Bobby
109	10	Ed
110	4	Cathie

Table: rating

Mov_Id	Rev_Stars
1	3
3	2
4	5
5	4
6	3
7	2
8	1
9	5
10	4

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

```
select m.mov_title from movies m,director d where d.dir_id=m.dir_id and d.Dir_Name
='Hitchcock';
```

mov_title
Vertigo
Innocents
American Beauty

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

```
select mov_title from movies where mov_id in( select mov_id from movie_cast where
act_id in( select act_id from movie_cast group by act_id having count(*)>1));
```

(OR)

```
select distinct(mov_title) from movies join movie_cast on movies.mov_id =
movie_cast.mov_id and act_id in( select act_id from movie_cast group by act_id having
count(*)>1);
```

mov_title
Vertigo
Innocents
Wings
Usual Suspects
The Prestige
American Beauty

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

```
SELECT A.act_id FROM
    (SELECT act_id FROM movie_cast join movies on movie_cast.Mov_Id =
    movies.Mov_Id WHERE mov_year<2000 )A ,
    (SELECT act_id FROM movie_cast join movies on
    movie_cast.Mov_Id=movies.Mov_Id WHERE mov_year>2015) B where A.act_id
    =B.act_id;
```

act_id
108

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,rev_stars from movies m,rating r where m.mov_id=r.mov_id order by mov_title;

mov_title 1	rev_stars
American Beauty	5
Deer Hunter	2
Eyes Wid Shut	5
Samurai	2
The Prestige	1
Usual Suspects	3
Vertigo	3
Walls	4
Wings	4

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

update rating r set rev_stars=5 where r.mov_id = (select m.mov_id from movies m,director d where m.dir_id=d.dir_id and Dir_Name='Steven Spielberg');

1 row affected

4. Consider the schema for College Database:

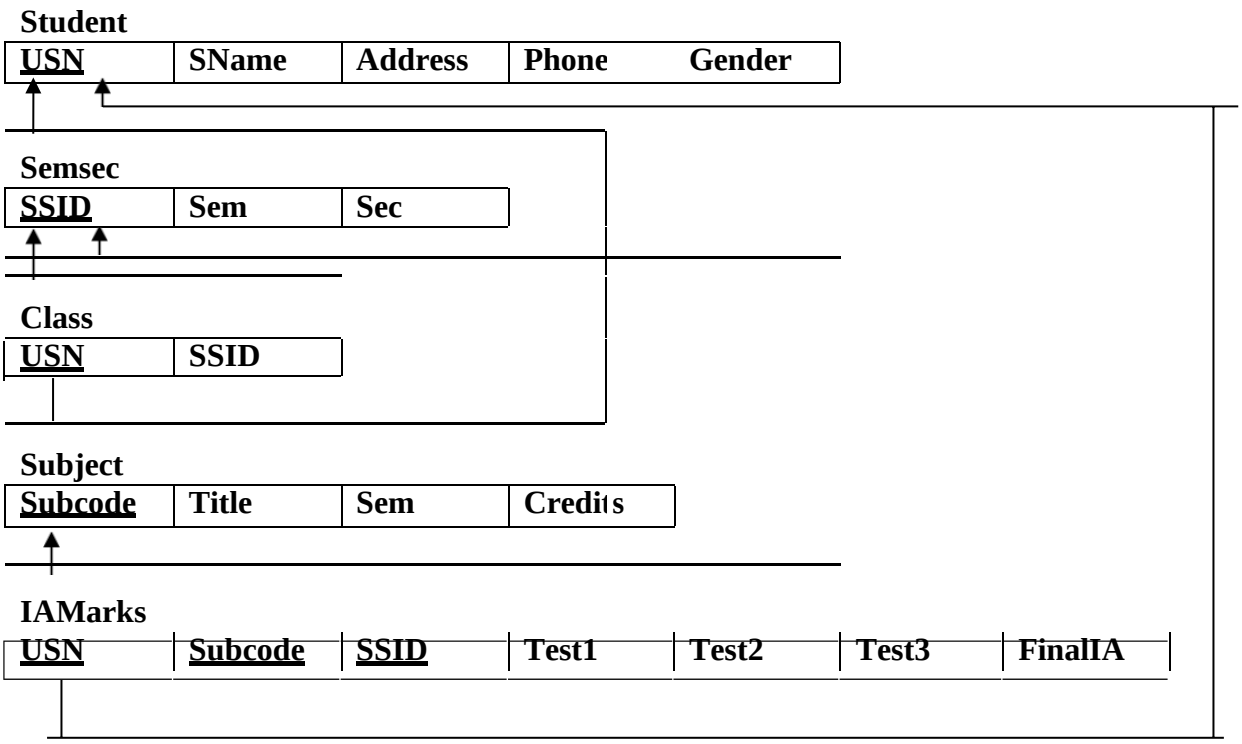
STUDENT(USN, SName, Address, Phone, Gender)
SEMSEC(SSID, Sem, Sec)
CLASS(USN, SSID)
SUBJECT(Subcode, Title, Sem, Credits)
IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

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.

Schema Diagram:



```
create table student(
   usn varchar(20)not null,
   sname varchar(20),
   address varchar(20),
   phone int,
   gender char(1),
   primary key(usn)
);

create table semsec(
   ssid int not null,
   sem int,
   section char(1),
   PRIMARY key(ssid)
);

create table class(
   usn varchar(20) primary key,
   ssid int,
   foreign key(ssid) references semsec(ssid) on delete cascade,
   foreign key(usn) references student(usn) on delete cascade
);

create table subject(
   subcode varchar(20) primary key,
   title varchar(30),
   sem int,
   credits int
);

create table iamarks(
   usn varchar(20),
   subcode varchar(20),
   ssid int,
   test1 int,
   test2 int,
   test3 int,
   final int,
   primary key(usn,subcode,ssid),
   foreign key(usn) references student(usn) on delete cascade,
   foreign key(subcode) references subject(subcode) on delete cascade,
   foreign key(ssid) references semsec(ssid) on delete cascade
);
```

Student table:

```
insert into student (usn, sname, address, phone, gender) values
("1hk17CS01","Ajith","Bangalore",1010101010,'M'),
("1hk17CS02","Barat","Mysore",2020202020,'M'),
("1hk17CS03","Clara","Delhi",303030303,'F'),
```

("1hk17CS04","Denis","Pune",4040404040,'M'),
("1hk17CS05","Elisa","Patna",505050505,'F'),
("1hk17CS06","Franc","Mumbai",6060606060,'M'),
("1hk17CS07","Gaury","Bangalore",077777777,'F'),
("1hk17CS08","Harry","Kolkatta",88888870,'M'),
("1hk17CS09","Isaac","Dehradun",9090909090,'M'),
("1hk17CS010","Jay","Sikkim",111110000,'M');

insert into student(usn, sname, address, phone, gender) values
("1hk16CS41","Aman","Manipal",110110110,'M'),
("1hk16CS42","Barka","Hyderabad",1212121212,'F'),
("1hk16CS43","Chahana","Chennai",1313131313,'F'),
("1hk16CS44","Deepa","Coimbatore",1414141141,'F'),
("1hk16CS45","Emad","Mumbai",15151515,'M'),
("1hk16CS46","Faroq","Chennai",1616161616,'M'),
("1hk16CS47","Ganga","Coimbatore",17171717,'F'),
("1hk16CS48","Hemalatha","Hyderabad",1818181818,'F'),
("1hk16CS49","Inder","Bangalore",1919191919,'M'),
("1hk16CS50","Jamal","Patna",2202020202,'M');

insert into student(usn, sname, address, phone, gender) values
("1hk15CS61","Arti","Pune",2121212121,'F'),
("1hk15CS62","Barti","Delhi",2202202201,'F'),
("1hk15CS63","Ceaser","Mysore",212121212,'M'),
("1hk15CS64","Dia","Mysore",222222222,'F'),
("1hk15CS65","Elmsri","Bangalore",2323232323,'M'),
("1hk15CS67","Geetha","Bangalore",252525252,'F'),
("1hk15CS68","Harish","Bangalore",565656565,'M'),
("1hk15CS69","Iman","Bangalore",2929292929,'M'),
("1hk15CS70","Jibin","Mysore",990990909,'M'),
("1hk15CS71","Kruthi","Mysore",6767676767,'F'),
("1hk15CS72","Lara","Pune",3030303030,'F');

insert into student(usn, sname, address, phone, gender) values
("1hk14CS81", "Athiya", "Patna", 3131313131, 'F'),
("1hk14CS82", "Beema", "New Delhi", 3232323232, 'M'),
("1hk14CS83", "Chitra", "Pune",3333333333, 'F'),
("1hk14CS84", "Dipika", "Patna", 3434343434, 'F'),
("1hk14CS85", "Elizabeth", "Mumbai", 2353535355, 'F'),
("1hk14CS86", "Fakruddin", "Mumbai", 363663663, 'M'),
("1hk14CS87", "Gary", "Bangalore", 3737373737, 'M'),
("1hk14CS88", "Hema", "Kolkatta", 3838383838, 'M'),
("1hk14CS89", "Ishana", "Dehradun", 3939393939, 'F'),
("1hk14CS90", "Jason", "Manipal", 4040404040, 'M'),
("1hk14CS91", "Kirana", "Hyderabad", 4949494949, 'F'),
("1hk14CS92", "Lucky", "Chennai", 454545545, 'F');

Semsec table:

insert into semsec (ssid, sem, section) values (1,2,'A'), (2,2,'B'), (3,2,'C'), (4,4,'A'), (5,4,'B'), (6,4,'C'), (7,6,'A'), (8,6,'B'), (9,6,'C'), (10,8,'A'), (11,8,'B'), (12,8,'C');

class table:

insert into class(usn,ssid) values ("1hk17CS01",'1'), ("1hk17CS02",'1'), ("1hk17CS03",'1'), ("1hk17CS04",'2'),("1hk17CS05",'2'),("1hk17CS06",'2'), ("1hk17CS07",'3'),("1hk17CS08",'3'),("1hk17CS09",'3');

insert into class(usn,ssid) values ("1hk16CS41",'4'), ("1hk16CS42",'4'), ("1hk16CS43",'4'), ("1hk16CS44",'4'),("1hk16CS45",'5'),("1hk16CS46",'5'),("1hk16CS47",'6'),("1hk16CS48",'6'),("1hk16CS49",'6'),("1hk16CS50",'6');

insert into class(usn,ssid) values ("1hk15CS61",'7'), ("1hk15CS62",'7'), ("1hk15CS63",'7'), ("1hk15CS64",'8'),("1hk15CS65",'8'),("1hk15CS67",9),("1hk15CS68",9),("1hk15CS69",9),("1hk15CS70",9),("1hk15CS71",9),("1hk15CS72",9),("1hk14CS81",10),("1hk14CS82",10),("1hk14CS83",10),("1hk14CS84",10),("1hk14CS85",11),("1hk14CS86",11),("1hk14CS87",11),("1hk14CS88",12),("1hk14CS89",12),("1hk14CS90",12),("1hk14CS91",12),("1hk14CS92",12);

subject table:

insert into subject(subcode,title,sem,credits) values ("15PCD23","PCD",2,4), ("15CHE21","CHEM",2,4), ("15ELN22","Basic Electronics",2,4), ("15MAT24","Maths",2,4), ("15CS42","SE",4,4), ("15CS43","DAA",4,4), ("15CS44","MP",4,4), ("15CS46","DC",4,4), ("15CS61","Cryptography",6,4), ("15CS62","CGV",6,4), ("15CS63","SS",6,4), ("15CS64","OS",6,4), ("15CS81","IOT",8,4), ("15CS82","Big Data Analytics",8,4), ("15CS834","SMS",8,4), ("15CS86","Seminar",8,4);

iamarks table:

insert into iamarks(usn,subcode,ssid,test1,test2,test3) values("1hk17CS01","15PCD23",1,10,12,14), ("1hk17CS01","15CHE21",1,11,12,13), ("1hk17CS01","15ELN22",1,13,14,15),("1hk17CS01","15MAT24",1,16,17,18),("1hk16CS41","15CS42",4,19,20,19),("1hk16CS41","15CS43",4,20,20,20),("1hk16CS41","15CS44",4,7,9,10),("1hk16CS41","15CS46",4,10,15,20),("1hk15CS61","15CS61",7,8,12,16),("1hk15CS61","15CS62",7,9,13,17),("1hk15CS61","15CS63",7,10,14,18),("1hk15CS61","15CS64",7,11,15,19),("1hk14CS81","15CS81",10,16,14,20),("1hk14CS81","15CS82",10,20,12,13),("1hk14CS81","15CS834",10,15,16,20),("1hk14CS81","15CS86",10,20,19,18);

insert into iamarks(usn,subcode,ssid,test1,test2,test3) values("1hk14CS82","15CS81",10,12,15,18), ("1hk14CS82","15CS82",10,13,20,12), ("1hk14CS82","15CS834",10,15,16,10),("1hk14CS82","15CS86",10,12,9,8);

insert into iamarks(usn,subcode,ssid,test1,test2,test3) values("1hk14CS83","15CS81",10,2,5,8),("1hk14CS83","15CS82",10,3,12,2),("1hk14CS83","15CS834",10,5,6,10),("1hk14CS83","15CS86",10,2,19,18);

```
insert into iamarks(usn,subcode,ssid,test1,test2,test3)
values("1hk14CS84","15CS81",10,12,15,8),("1hk14CS84","15CS82",10,13,0,1),("1hk14
CS84","15CS834",10,15,6,1),("1hk14CS84","15CS86",10,2,19,8);

insert into iamarks(usn,subcode,ssid,test1,test2,test3)
values("1hk14CS85","15CS81",11,12,5,11),("1hk14CS85","15CS82",11,13,2,13),("1hk14
CS85","15CS834",11,14,16,19),("1hk14CS85","15CS86",11,2,14,8),("1hk14CS86","15C
S81",11,12,14,13),("1hk14CS86","15CS82",11,3,2,5),("1hk14CS86","15CS834",11,4,6,9)
,("1hk14CS86","15CS86",11,13,17,18),("1hk14CS87","15CS81",11,13,15,19),("1hk14CS
87","15CS82",11,11,2,20),("1hk14CS87","15CS834",11,11,10,6),("1hk14CS87","15CS86
",11,12,4,15),("1hk14CS88","15CS81",12,9,13,19),("1hk14CS88","15CS82",12,1,2,20),("
1hk14CS88","15CS834",12,10,11,12),("1hk14CS88","15CS86",12,13,14,15),("1hk14CS8
9","15CS81",12,19,3,9),("1hk14CS89","15CS82",12,13,12,20),("1hk14CS89","15CS834"
,12,16,17,18),("1hk14CS89","15CS86",12,3,11,15),("1hk14CS90","15CS81",12,16,13,19)
,("1hk14CS90","15CS82",12,20,2,20),("1hk14CS90","15CS834",12,6,7,20),("1hk14CS90
","15CS86",12,6,12,5),("1hk14CS91","15CS81",12,19,10,20),("1hk14CS91","15CS82",1
2,2,20,20),("1hk14CS91","15CS834",12,20,12,15),("1hk14CS91","15CS86",12,12,5,20),(
"1hk14CS92","15CS81",12,9,10,20),("1hk14CS92","15CS82",12,20,19,19),("1hk14CS92
","15CS834",12,20,15,13),("1hk14CS92","15CS86",12,20,20,20);
```

Table: Student

usn	sname	address	phone	gender
1hk14CS81	Athiya	Patna	2147483647	F
1hk14CS82	Beema	New Delhi	2147483647	M
1hk14CS83	Chitra	Pune	2147483647	F
1hk14CS84	Dipika	Patna	2147483647	F
1hk14CS85	Elizabeth	Mumbai	2147483647	F
1hk14CS86	Fakruddin	Mumbai	363663663	M
1hk14CS87	Gary	Bangalore	2147483647	M
1hk14CS88	Hema	Kolkatta	2147483647	M
1hk14CS89	Ishana	Dehradun	2147483647	F
1hk14CS90	Jason	Manipal	2147483647	M
1hk14CS91	Kirana	Hyderabad	2147483647	F
1hk14CS92	Lucky	Chennai	454545545	F
1hk15CS61	Arti	Pune	2121212121	F
1hk15CS62	Barti	Delhi	2147483647	F
1hk15CS63	Ceaser	Mysore	212121212	M
1hk15CS64	Dia	Mysore	222222222	F
1hk15CS65	Elmsri	Bangalore	2147483647	M
1hk15CS67	Geetha	Bangalore	252525252	F
1hk15CS68	Harish	Bangalore	565656565	M
1hk15CS69	Iman	Bangalore	2147483647	M
1hk15CS70	Jibin	Mysore	990990909	M
1hk15CS71	Kruthi	Mysore	2147483647	F
1hk15CS72	Lara	Pune	2147483647	F
1hk16CS41	Aman	Manipal	110110110	M
1hk16CS42	Barka	Hyderabad	1212121212	F
1hk16CS43	Chahana	Chennai	1313131313	F
1hk16CS44	Deepa	Coimbatore	1414141141	F
1hk16CS45	Emad	Mumbai	15151515	M
1hk16CS46	Faroq	Chennai	1616161616	M
1hk16CS47	Ganga	Coimbatore	17171717	F
1hk16CS48	Hemalatha	Hyderabad	1818181818	F
1hk16CS49	Inder	Bangalore	1919191919	M
1hk16CS50	Jamal	Patna	2147483647	M
1hk17CS01	Ajith	Bangalore	1010101010	M
1hk17CS010	Jay	Sikkim	111110000	M
1hk17CS02	Barat	Mysore	2020202020	M
1hk17CS03	Clara	Delhi	303030303	F
1hk17CS04	Denis	Pune	2147483647	M
1hk17CS05	Elisa	Patna	505050505	F
1hk17CS06	Franc	Mumbai	2147483647	M
1hk17CS07	Gaury	Bangalore	77777777	F
1hk17CS08	Harry	Kolkatta	88888870	M
1hk17CS09	Isaac	Dehradun	2147483647	M

Table: Semsec

ssid	sem	section
1	2	A
2	2	B
3	2	C
4	4	A
5	4	B
6	4	C
7	6	A
8	6	B
9	6	C
10	8	A
11	8	B
12	8	C

Table: Subject

subcode	title	sem	credits
15CHE21	CHEM	2	4
15CS42	SE	4	4
15CS43	DAA	4	4
15CS44	MP	4	4
15CS46	DC	4	4
15CS61	Cryptography	6	4
15CS62	CGV	6	4
15CS63	SS	6	4
15CS64	OS	6	4
15CS81	IOT	8	4
15CS82	Big Data Analytics	8	4
15CS834	SMS	8	4
15CS86	Seminar	8	4
15ELN22	Basic Electronics	2	4
15MAT24	Maths	2	4
15PCD23	PCD	2	4

Table: Class

usn	ssid
1hk17CS01	1
1hk17CS02	1
1hk17CS03	1
1hk17CS04	2
1hk17CS05	2
1hk17CS06	2
1hk17CS07	3
1hk17CS08	3
1hk17CS09	3
1hk16CS41	4
1hk16CS42	4
1hk16CS43	4
1hk16CS44	4
1hk16CS45	5
1hk16CS46	5
1hk16CS47	6
1hk16CS48	6
1hk16CS49	6
1hk16CS50	6
1hk15CS61	7
1hk15CS62	7
1hk15CS63	7
1hk15CS64	8
1hk15CS65	8
1hk15CS67	9
1hk15CS68	9
1hk15CS69	9
1hk15CS70	9
1hk15CS71	9
1hk15CS72	9
1hk14CS81	10
1hk14CS82	10
1hk14CS83	10
1hk14CS84	10
1hk14CS85	11
1hk14CS86	11
1hk14CS87	11
1hk14CS88	12
1hk14CS89	12
1hk14CS90	12
1hk14CS91	12
1hk14CS92	12

iamarks

usn	subcode	ssid	test1	test2	test3	final
1hk14CS81	15CS81	10	16	14	20	*NULL*
1hk14CS81	15CS82	10	20	12	13	*NULL*
1hk14CS81	15CS834	10	15	16	20	*NULL*
1hk14CS81	15CS86	10	20	19	18	*NULL*
1hk14CS82	15CS81	10	12	15	18	*NULL*
1hk14CS82	15CS82	10	13	20	12	*NULL*
1hk14CS82	15CS834	10	15	16	10	*NULL*
1hk14CS82	15CS86	10	12	9	8	*NULL*
1hk14CS83	15CS81	10	2	5	8	*NULL*
1hk14CS83	15CS82	10	3	12	2	*NULL*
1hk14CS83	15CS834	10	5	6	10	*NULL*
1hk14CS83	15CS86	10	2	19	18	*NULL*
1hk14CS84	15CS81	10	12	15	8	*NULL*
1hk14CS84	15CS82	10	13	0	1	*NULL*
1hk14CS84	15CS834	10	15	6	1	*NULL*
1hk14CS84	15CS86	10	2	19	8	*NULL*
1hk14CS85	15CS81	11	12	5	11	*NULL*
1hk14CS85	15CS82	11	13	2	13	*NULL*
1hk14CS85	15CS834	11	14	16	19	*NULL*
1hk14CS85	15CS86	11	2	14	8	*NULL*
1hk14CS86	15CS81	11	12	14	13	*NULL*
1hk14CS86	15CS82	11	3	2	5	*NULL*
1hk14CS86	15CS834	11	4	6	9	*NULL*
1hk14CS86	15CS86	11	13	17	18	*NULL*
1hk14CS87	15CS81	11	13	15	19	*NULL*

usn	subcode	ssid	test1	test2	test3	final
1hk14CS87	15CS82	11	11	2	20	*NULL*
1hk14CS87	15CS834	11	11	10	6	*NULL*
1hk14CS87	15CS86	11	12	4	15	*NULL*
1hk14CS88	15CS81	12	9	13	19	*NULL*
1hk14CS88	15CS82	12	1	2	20	*NULL*
1hk14CS88	15CS834	12	10	11	12	*NULL*
1hk14CS88	15CS86	12	13	14	15	*NULL*
1hk14CS89	15CS81	12	19	3	9	*NULL*
1hk14CS89	15CS82	12	13	12	20	*NULL*
1hk14CS89	15CS834	12	16	17	18	*NULL*
1hk14CS89	15CS86	12	3	11	15	*NULL*
1hk14CS90	15CS81	12	16	13	19	*NULL*
1hk14CS90	15CS82	12	20	2	20	*NULL*
1hk14CS90	15CS834	12	6	7	20	*NULL*
1hk14CS90	15CS86	12	6	12	5	*NULL*
1hk14CS91	15CS81	12	19	10	20	*NULL*
1hk14CS91	15CS82	12	2	20	20	*NULL*
1hk14CS91	15CS834	12	20	12	15	*NULL*
1hk14CS91	15CS86	12	12	5	20	*NULL*
1hk14CS92	15CS81	12	9	10	20	*NULL*
1hk14CS92	15CS82	12	20	19	19	*NULL*
1hk14CS92	15CS834	12	20	15	13	*NULL*
1hk14CS92	15CS86	12	20	20	20	*NULL*
1hk15CS61	15CS61	7	8	12	16	*NULL*
1hk15CS61	15CS62	7	9	13	17	*NULL*

usn	subcode	ssid	test1	test2	test3	final
1hk15CS61	15CS63	7	10	14	18	*NULL*
1hk15CS61	15CS64	7	11	15	19	*NULL*
1hk16CS41	15CS42	4	19	20	19	*NULL*
1hk16CS41	15CS43	4	20	20	20	*NULL*
1hk16CS41	15CS44	4	7	9	10	*NULL*
1hk16CS41	15CS46	4	10	15	20	*NULL*
1hk17CS01	15CHE21	1	11	12	13	*NULL*
1hk17CS01	15ELN22	1	13	14	15	*NULL*
1hk17CS01	15MAT24	1	16	17	18	*NULL*
1hk17CS01	15PCD23	1	10	12	14	*NULL*

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

   select s.usn,sname,address,phone,gender from student s,semsec m,class c where m.ssid=c.ssid and c.usn=s.usn and sem=4 and section='C';

usn	sname	address	phone	gender
1hk16CS47	Ganga	Coimbatore	17171717	F
1hk16CS48	Hemalatha	Hyderabad	1818181818	F
1hk16CS49	Inder	Bangalore	1919191919	M
1hk16CS50	Jamal	Patna	2147483647	M

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

   select sem,section,count(case when gender='m' then 1 end) as male_cnt, count(case when gender='f' then 1 end) as female_cnt from student st,class c,semsec s where s.ssid=c.ssid and c.usn=st.usn group by sem,section;

sem	section	Male_cnt	Female_cnt
2	A	2	1
2	B	2	1
2	C	2	1
4	A	1	3
4	B	2	0
4	C	2	2
6	A	1	2
6	B	1	1
6	C	3	3
8	A	1	3
8	B	2	1
8	C	2	3

3. **Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.**

   create view test1_mark as select s.subcode as Subject_Code, title as Subject_Name, test1 as Test1_Marks from iamarks i,subject s where s.subcode=i.subcode and usn ='1hk15CS61';

   select * from test1_marks;

Subject_Code	Subject_Name	Test1_Marks
15CS61	Cryptography	8
15CS62	CGV	9
15CS63	SS	10
15CS64	OS	11

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

create view finalia as (select usn, subcode, greatest(avg(test1+test2)/2, avg(test1+test3)/2, avg(test2+test3)/2) as finalia from iamarks GROUP BY usn, subcode)

select * from finalia;

usn	subcode	finalia
1hk14CS81	15CS81	18.00000000
1hk14CS81	15CS82	16.50000000
1hk14CS81	15CS834	18.00000000
1hk14CS81	15CS86	19.50000000
1hk14CS82	15CS81	16.50000000
1hk14CS82	15CS82	16.50000000
1hk14CS82	15CS834	15.50000000
1hk14CS82	15CS86	10.50000000
1hk14CS83	15CS81	6.50000000
1hk14CS83	15CS82	7.50000000
1hk14CS83	15CS834	8.00000000
1hk14CS83	15CS86	18.50000000
1hk14CS84	15CS81	13.50000000
1hk14CS84	15CS82	7.00000000
1hk14CS84	15CS834	10.50000000
1hk14CS84	15CS86	13.50000000
1hk14CS85	15CS81	11.50000000
1hk14CS85	15CS82	13.00000000
1hk14CS85	15CS834	17.50000000
1hk14CS85	15CS86	11.00000000
1hk14CS86	15CS81	13.50000000
1hk14CS86	15CS82	4.00000000
1hk14CS86	15CS834	7.50000000
1hk14CS86	15CS86	17.50000000
1hk14CS87	15CS81	17.00000000

UPDATE iamarks i set final=(SELECT finalia from finalia f where i.usn=f.usn and i.subcode=f.subcode);

60 rows affected

Select * from iamarks;

usn	subcode	ssid	test1	test2	test3	final
1hk14CS81	15CS81	10	16	14	20	18
1hk14CS81	15CS82	10	20	12	13	17
1hk14CS81	15CS834	10	15	16	20	18
1hk14CS81	15CS86	10	20	19	18	20
1hk14CS82	15CS81	10	12	15	18	17
1hk14CS82	15CS82	10	13	20	12	17
1hk14CS82	15CS834	10	15	16	10	16
1hk14CS82	15CS86	10	12	9	8	11
1hk14CS83	15CS81	10	2	5	8	7
1hk14CS83	15CS82	10	3	12	2	8
1hk14CS83	15CS834	10	5	6	10	8
1hk14CS83	15CS86	10	2	19	18	19
1hk14CS84	15CS81	10	12	15	8	14
1hk14CS84	15CS82	10	13	0	1	7
1hk14CS84	15CS834	10	15	6	1	11
1hk14CS84	15CS86	10	2	19	8	14
1hk14CS85	15CS81	11	12	5	11	12
1hk14CS85	15CS82	11	13	2	13	13
1hk14CS85	15CS834	11	14	16	19	18
1hk14CS85	15CS86	11	2	14	8	11
1hk14CS86	15CS81	11	12	14	13	14
1hk14CS86	15CS82	11	3	2	5	4
1hk14CS86	15CS834	11	4	6	9	8
1hk14CS86	15CS86	11	13	17	18	18
1hk14CS87	15CS81	11	13	15	19	17

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 i.usn,s.sem,s.section,i.subcode, CASE
   WHEN final >=17 AND final <=20 THEN 'Outstanding'
   WHEN final >=12 AND final <=16 THEN 'Average'
   ELSE 'Weak' END AS CAT FROM iamarks i, class c, semsec s where i.usn=c.usn and c.ssid=s.ssid and s.sem=8 and s.section in('A','B','C');

usn	sem	section	subcode	CAT
1hk14CS81	8	A	15CS81	Outstanding
1hk14CS81	8	A	15CS82	Outstanding
1hk14CS81	8	A	15CS834	Outstanding
1hk14CS81	8	A	15CS86	Outstanding
1hk14CS82	8	A	15CS81	Outstanding
1hk14CS82	8	A	15CS82	Outstanding
1hk14CS82	8	A	15CS834	Average
1hk14CS82	8	A	15CS86	Weak
1hk14CS83	8	A	15CS81	Weak
1hk14CS83	8	A	15CS82	Weak
1hk14CS83	8	A	15CS834	Weak
1hk14CS83	8	A	15CS86	Outstanding
1hk14CS84	8	A	15CS81	Average
1hk14CS84	8	A	15CS82	Weak
1hk14CS84	8	A	15CS834	Weak
1hk14CS84	8	A	15CS86	Average
1hk14CS85	8	B	15CS81	Average
1hk14CS85	8	B	15CS82	Average
1hk14CS85	8	B	15CS834	Outstanding
1hk14CS85	8	B	15CS86	Weak
1hk14CS86	8	B	15CS81	Average
1hk14CS86	8	B	15CS82	Weak
1hk14CS86	8	B	15CS834	Weak
1hk14CS86	8	B	15CS86	Outstanding
1hk14CS87	8	B	15CS81	Outstanding
1hk14CS87	8	B	15CS82	Average
1hk14CS87	8	B	15CS834	Weak
1hk14CS87	8	B	15CS86	Average
1hk14CS88	8	C	15CS81	Average
1hk14CS88	8	C	15CS82	Weak
1hk14CS88	8	C	15CS834	Average
1hk14CS88	8	C	15CS86	Average
1hk14CS89	8	C	15CS81	Average
1hk14CS89	8	C	15CS82	Outstanding
1hk14CS89	8	C	15CS834	Outstanding
1hk14CS89	8	C	15CS86	Average
1hk14CS90	8	C	15CS81	Outstanding
1hk14CS90	8	C	15CS82	Outstanding
1hk14CS90	8	C	15CS834	Average
1hk14CS90	8	C	15CS86	Weak

usn	sem	section	subcode	CAT
1hk14CS91	8	C	15CS81	Outstanding
1hk14CS91	8	C	15CS82	Outstanding
1hk14CS91	8	C	15CS834	Outstanding
1hk14CS91	8	C	15CS86	Average
1hk14CS92	8	C	15CS81	Average
1hk14CS92	8	C	15CS82	Outstanding
1hk14CS92	8	C	15CS834	Outstanding
1hk14CS92	8	C	15CS86	Outstanding

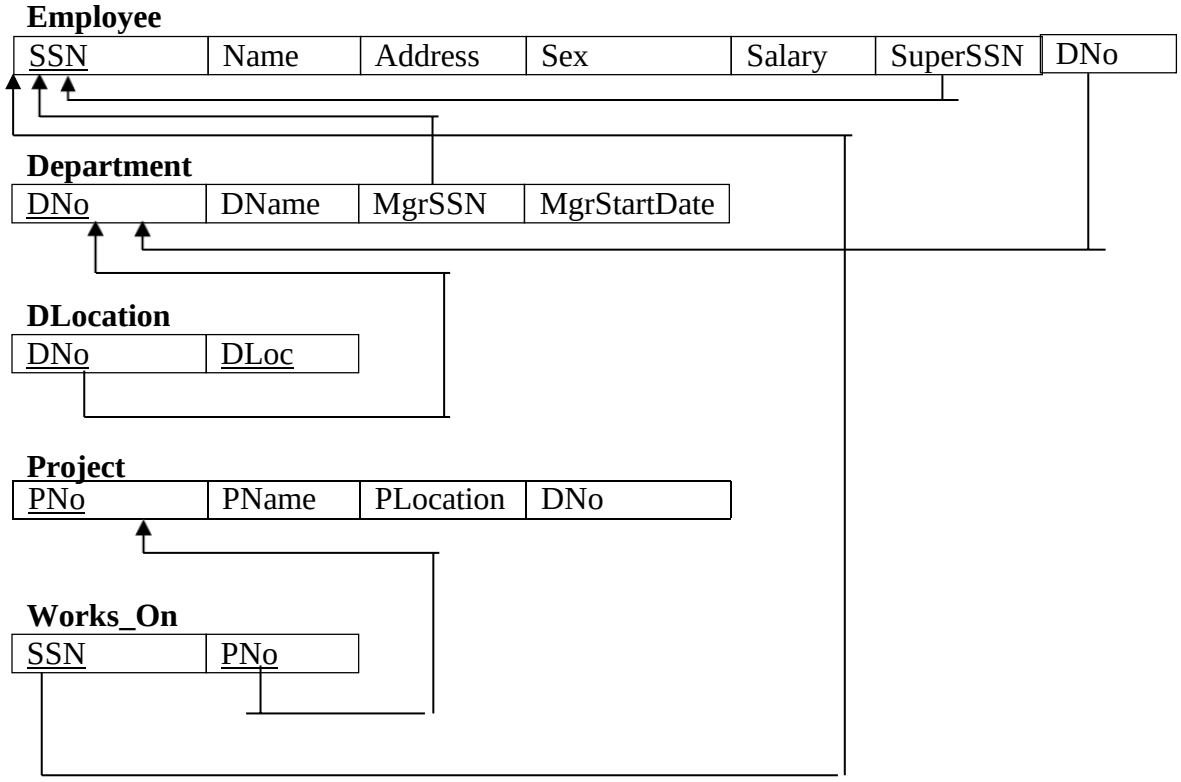
5. Consider the schema for Company Database:

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).
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.

Schema Diagram:




```
create table Employee(
   ssn int not null ,
   name varchar(25),
   address varchar(25),
```

```
    sex varchar(10),
    salary int,
    superssn int,
    dno int,
    primary key ( ssn),
    FOREIGN KEY (superssn) REFERENCES Employee(ssn)
);

create table department(
    dno int not null ,
    dname varchar(25),
    mgrssn int(25),
    mgrstrtdate date,
    primary key (dno),
    FOREIGN key (mgrssn) REFERENCES employee(ssn)
);

alter table employee add FOREIGN key (dno) REFERENCES department (dno);

create table dlocation (
    dno int not null ,
    dloc varchar(25) not null ,
    primary key ( dno,dloc),
    FOREIGN KEY (dno) REFERENCES department(dno)
);

create table  project(
    pno int not null ,
    pname varchar(25) ,
    plocation varchar(25),
    dno int,
    primary key (pno),
    FOREIGN KEY (dno) REFERENCES department(dno)
);

create table works_on (
    ssn int not null ,
    pno int not null ,
    hours int not null ,
    primary key (ssn, pno),
    FOREIGN KEY (pno) REFERENCES project(pno)
);
```

Employee table:

```
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('23412356', 'JENNIFER', 'PARIS', 'FEMALE','700000', '4');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('23456781', 'JAMES', 'NEWYORK', 'MALE','300000', '4');
```

```
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('33344555', 'FRANKLIN', 'CALIFORNIA','MALE', '600000', '5');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('67891234', 'JOYCE', 'WASHINGTON','FEMALE', '400000', '5');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('123456789', 'JOHN', 'TEXAS', 'MALE','300000', '5');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('888666555', 'AHMAD', 'CALIFORNIA','MALE', '700000', '4');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('984600445', 'MARK', 'WASHINGTON','MALE', '800000', '5');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('56789012', 'ALEENA', 'LONDON', 'FEMALE', '700000', '5');
INSERT INTO `employee` (`SSN`, `NAME`, `ADDRESS`, `SEX`, `SALARY`, `DNO`)
VALUES ('89012345', 'ALICE', 'STAFFORD', 'FEMALE', '1200000', '5');
UPDATE `employee` SET `superssn` = '984600445' WHERE `ssn` = 23412356;
UPDATE `employee` SET `superssn` = '67891234' WHERE `ssn` = 23456781;
UPDATE `employee` SET `superssn` = '123456789' WHERE `ssn` = 33344555;
UPDATE `employee` SET `superssn` = '89012345' WHERE `ssn` = 56789012;
UPDATE `employee` SET `superssn` = '888666555' WHERE `ssn` = 67891234;
UPDATE `employee` SET `superssn` = '56789012' WHERE `ssn` = 89012345;
UPDATE `employee` SET `superssn` = '33344555' WHERE `ssn` = 123456789;
UPDATE `employee` SET `superssn` = '23456781' WHERE `ssn` = 888666555;
UPDATE `department` SET `mgrssn` = '888666555' WHERE `dno` = 1;
UPDATE `department` SET `mgrssn` = '123456789' WHERE `dno` = 4;
UPDATE `department` SET `mgrssn` = '33344555' WHERE `dno` = 5;
```

Dlocation table:

```
INSERT INTO `dlocation` (`DNO`, `DLOC`) VALUES ('1', 'WASHINGTON');
INSERT INTO `dlocation` (`DNO`, `DLOC`) VALUES ('4', 'CALIFORNIA');
INSERT INTO `dlocation` (`DNO`, `DLOC`) VALUES ('5', 'NEW YORK');
INSERT INTO `dlocation` (`DNO`, `DLOC`) VALUES ('5', 'WASHINGTON');
```

Project table:

```
INSERT INTO project (PNAME, PNO, PLOCATION, DNO) VALUES ('PRODUCTA', 1,
'HOUSTON', 5);
INSERT INTO project (`PNAME`, `PNO`, `PLOCATION`, `DNO`) VALUES
('PRODUCTB', 2, 'WASHINGTON', 5);
INSERT INTO project (`PNAME`, `PNO`, `PLOCATION`, `DNO`) VALUES
('PRODUCTC', 3, 'CALIFORNIA', 5);
INSERT INTO project (`PNAME`, `PNO`, `PLOCATION`, `DNO`) VALUES
('COMPUTERIZATION',10,'NEW YORK',4);
INSERT INTO project (`PNAME`, `PNO`, `PLOCATION`, `DNO`) VALUES
('IOT',20,'PARIS',1);
INSERT INTO project (`PNAME`, `PNO`, `PLOCATION`, `DNO`) VALUES
('REORGANIZATION',30,'STAFFORD',4);
```

Works_on table:

```
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('123456789', '1', '33');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('123456789', '2', '8');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('984600445', '3', '40');
INSERT INTO `works_on`(`SSN`, `PNO`, `HOURS`) VALUES ('984600445', '1', '50');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('888666555', '1', '45');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('888666555', '2', '54');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('33344555', '2', '10');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('33344555', '3', '10');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('33344555', '10', '10');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('33344555', '20', '10');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('67891234', '30', '20');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('67891234', '10', '20');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('23412356', '30', '12');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('23412356', '20', '14');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('23456781', '20', '20');
INSERT INTO `works_on` (`SSN`, `PNO`, `HOURS`) VALUES ('23456781', '1', '34');
```

Table: Employee

ssn	name	address	sex	salary	superssn	dno
23412356	JENNIFER	PARIS	FEMALE	700000	984600445	4
23456781	JAMES	NEW YORK	MALE	300000	67891234	4
33344555	FRANKLIN	CALIFORNIA	MALE	600000	123456789	5
56789012	ALEENA	LONDON	FEMALE	700000	89012345	5
67891234	JOYCE	WASHINGTON	FEMALE	400000	888666555	5
89012345	ALICE	STAFFORD	FEMALE	1200000	56789012	5
123456789	JOHN	TEXAS	MALE	300000	33344555	5
888666555	AHMAD	CALIFORNIA	MALE	700000	23456781	4
984600445	MARK	WASHINGTON	MALE	800000	*NULL*	5

Table: Department

dno	dname	mgrssn	mgrstrtdate
1	ADMINISTRATION	888666555	1992-06-25
4	RESEARCH	23456781	1990-04-23
5	ACCOUNTS	33344555	1988-05-22

Table: Dlocation

dno	dloc
1	WASHINGTON
4	CALIFORNIA
5	NEW YORK
5	WASHINGTON

Table:Project

pno	pname	plocation	dno
1	PRODUCTA	HOUSTON	5
2	PRODUCTB	WASHINGTON	5
3	PRODUCTC	CALIFORNIA	5
10	COMPUTERIZATION	NEW YORK	4
20	IOT	PARIS	1
30	REORGANIZATION	STAFFORD	4

Table:Works_on

ssn	pno	hours
23412356	20	14
23412356	30	12
23456781	1	34
23456781	20	20
33344555	2	10
33344555	3	10
33344555	10	10
33344555	20	10
67891234	10	20
67891234	30	20
123456789	1	33
123456789	2	8
888666555	1	45
888666555	2	54
888666555	3	12
984600445	1	50
984600445	3	40

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 p.dno=d.dno and d.mgrssn=e.ssn and lname="scott" union select distinct p.pno from project p, works_on w,employee e where w.pno=p.pno and w.ssn=e.ssn and lname="scott";

PNO
10
30
1
20

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

select fname, lname, 0.1*salary from employee e, works_on w, project p where e.ssn=w.ssn and p.pno=w.pno and p.pname="iot";

FNAME	LNAME	0.1*SALARY
JENNIFER	LOPEZ	70000.0
JAMES	SCOTT	30000.0
FRANKLIN	WILSON	60000.0

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(salary) as SUM, max(salary) as MAX, min(salary) as MIN, avg(salary) as AVG from employee e, department d where e.dno=d.dno and dname="accounts";

SUM	MAX	MIN	AVG
4000000	1200000	300000	666666.6667

4. 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 where not exists ((select Pnumber from project where Dnum=5) except (select pno from works_on where ssn=essn));

name
AHMAD

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 as D_NO,count(*) as No_Of_Employees from department d, employee e where d.dno=e.dno and e.salary>600000 and e.dno in (select dno from employee group by dno having count(*)>5) group by d.dno

D_NO	No_Of_Employees
5	3

DBMS viva questions

1. Characteristic of a database management system

- It represents complex relation relationship between data.
- Keep a tight control of data redundancy
- Ensures that data can be shared across the applications
- Enforces data access authorization
- Has automatic, intelligent back up and recovery procedure for data.

2. Characteristic of a Relation DBMS models

- The relationship data management model eliminate all parent –child relationships and instead represented al, data in the database as sample row/column tables of data values.
- Each table is a n independent entity and there is no physical relationship between tables.
- Most of the DBMS models based on the relational models.
- Relational model of data management is based on set theory. Built in query language is designed in the RDBMS, so that it can manipulates sets of data.
- The user interface used with relational models is non-procedural because only what needs to be done is specified and not how it has to be done. Using any of the other methods, you have not only to specify what needs to be done but how it has to be done as well.

3. Roles of DBA

- Updating the database.
- Retrieving information from the database.
- Accepting the query language statements
- Enforcing security specifications.
- Enforcing data integrity specifications.
- Enforcing transaction consistency.
- Managing data sharing.
- Optimizing queries.
- Managing system catalogs.
- All the permissions are granted by the DBA only

4. 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.

5. 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.

6. What is a Database system?

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

7. Advantages of DBMS?

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

8. Disadvantage in File Processing System?

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

9. Describe the three levels of data abstraction?

The are three levels of abstraction:

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

10. Define the "integrity rules"

There are two Integrity rules.

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

11. 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.

14. 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

15. 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.

16. What is Data Model?

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

17. 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.

18. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in 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.

19. What is an Entity?

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

20. What is an Entity type?

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

21. What is an Entity set?

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

22. What is an Extension of entity type?

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

23. What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

24. What is an attribute?

It is a particular property, which describes the entity.

25. 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).

26. What is degree of a Relation?

It is the number of attribute of its relation schema.

27. What is Relationship?

It is an association among two or more entities.

28. What is Relationship set?

The collection (or set) of similar relationships.

29. What is Relationship type*?*

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

30. What is degree of Relationship type?

It is the number of entity type participating.

31. What is DDL (Data Definition Language)?

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

32. What is VDL (View Definition Language)?

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

33. What is SDL (Storage Definition Language)?

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

34. 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.

35. What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organised 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.

36. What is DML Compiler?

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

37. What is Query evaluation engine?

It executes low-level instruction generated by compiler.

38. What is DDL Interpreter?

It interprets DDL statements and record them in tables containing metadata.

39. What is Record-at-a-time?

The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

40. What is Set-at-a-time or Set-oriented?

The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set- oriented.

41. What is Relational Algebra?

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

44. What is normalization?

It is a process of analysing 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.

45. What is Functional Dependency?

A Functional dependency is denoted by X $\rightarrow$ 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.

46. What is Lossless join property?

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

47. What is 1 NF (Normal Form)?

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

48. 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.

49. What is 3NF?

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

- X is a Super-key of R.

- A is a prime attribute of R.

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

50. What is BCNF (Boyce-Codd Normal Form)?

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

51. 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.

52. What is 5NF?

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

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

53. What is Domain-Key Normal Form?

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

54. 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:

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

Artificial Key:

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.

*Compound Key**:***

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.

*Natural Key**:***

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

55. What is system catalog or catalog relation? How is better known as?

A RDBMS maintains a description of all the data that it contains, information

about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

57. What is durability in DBMS?

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

58. What is a checkpoint and When does it occur?

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

59. What do you mean by flat file database?

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

60. What is "transparent DBMS"?

It is one, which keeps its Physical Structure hidden from user.

61. Brief theory of Network, Hierarchical schemas and their properties

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

62. What is a query?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

66. How do you communicate with an RDBMS?

You communicate with an RDBMS using Structured Query Language (SQL)

67. Define SQL and state the differences between SQL and other conventional programming Languages

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

69. Tables derived from the ERD

a) Are totally unnormalised
b) Are always in 1NF
c) Can be further denormalised
d) May have multi-valued attributes

(b) Are always in 1NF

70. In mapping of ERD to DFD
    a) entities in ERD should correspond to an existing entity/store in DFD
    b) entity in DFD is converted to attributes of an entity in ERD
    c) relations in ERD has 1 to 1 correspondence to processes in DFD
    d) relationships in ERD has 1 to 1 correspondence to flows in DFD
    e) entities in ERD should correspond to an existing entity/store in DFD

SQL

1. Which is the subset of SQL commands used to manipulate Oracle Database structures, including tables?
   Data Definition Language (DDL)

- What operator performs pattern matching?
  LIKE operator

- What operator tests column for the absence of data?
  IS NULL operator
- What are the wildcards used for pattern matching?
  _ for single character substitution and % for multi-character substitution

- State true or false. EXISTS, SOME, ANY are operators in SQL.
  True

- State true or false. !=, <>, ^= all denote the same operation.
  True

- What are the privileges that can be granted on a table by a user to others?
  Insert, update, delete, select, references, index, execute, alter, all

- What command is used to get back the privileges offered by the GRANT command?
  REVOKE

- Which system tables contain information on privileges granted and privileges obtained?
  USER_TAB_PRIVS_MADE, USER_TAB_PRIVS_RECD

- Which system table contains information on constraints on all the tables created?
  USER_CONSTRAINTS

- TRUNCATE TABLE EMP;
  DELETE FROM EMP;

Will the outputs of the above two commands differ?

Both will result in deleting all the rows in the table EMP.

- What is the difference between TRUNCATE and DELETE commands?

TRUNCATE is a DDL command whereas DELETE is a DML command. Hence DELETE operation can be rolled back, but TRUNCATE operation cannot be rolled back. WHERE clause can be used with DELETE and not with TRUNCATE.

- What command is used to create a table by copying the structure of another table?

Answer :

CREATE TABLE .. AS SELECT command

- Which date function is used to find the difference between two dates?

MONTHS_BETWEEN

- What is the advantage of specifying WITH GRANT OPTION in the GRANT command?

The privilege receiver can further grant the privileges he/she has obtained from the owner to any other user.

- What is the use of the DROP option in the ALTER TABLE command?

It is used to drop constraints specified on the table.

sal = 11000, comm = 1000

- What is the use of DESC in SQL?

Answer :

DESC has two purposes. It is used to describe a schema as well as to retrieve rows from table in descending order.

- What is the use of CASCADE CONSTRAINTS?

When this clause is used with the DROP command, a parent table can be dropped even when a child table exists.

- Which function is used to find the largest integer less than or equal to a specific value?

FLOOR