|| Ohm Shree Manjunathaya Namaha ||

SDM Educational Society's

S.D.M. Institute of Technology, Ujire-574240

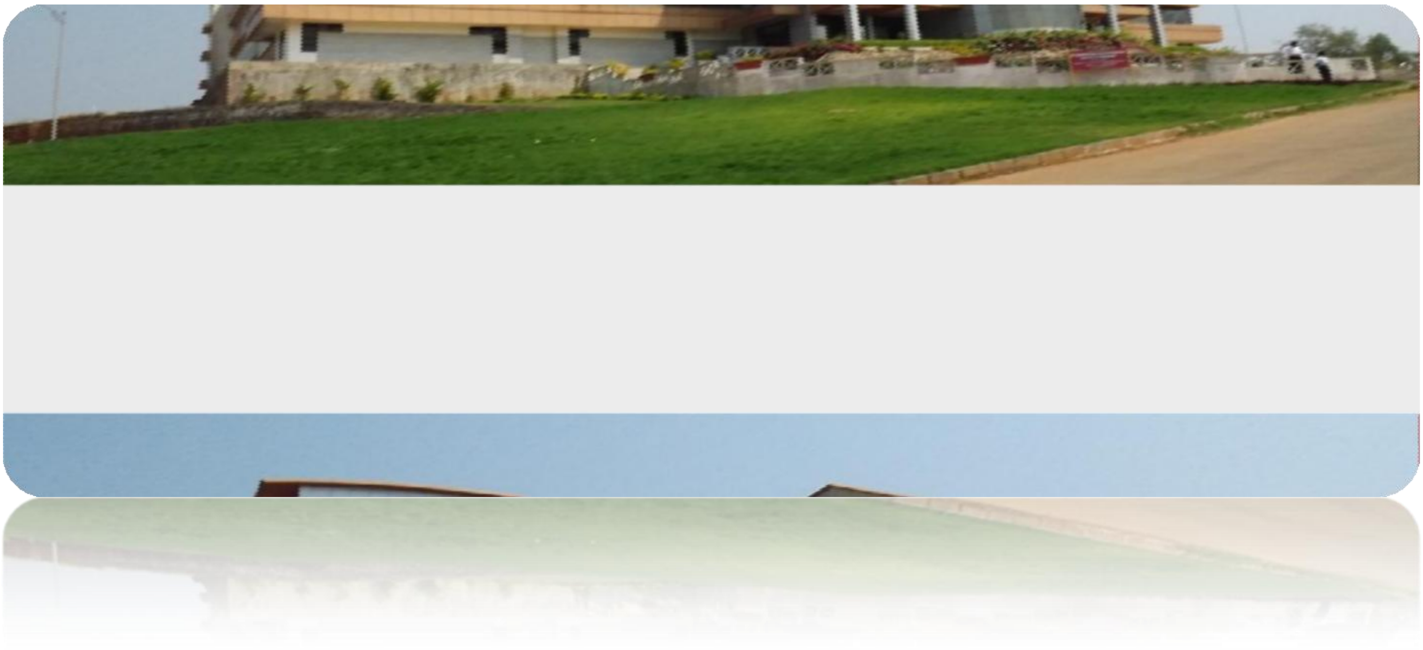
(AFFILIATED TO VISVESVRAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI &APPROVED BY AICTE)

Department of Computer Science & Engineering

DBMS Laboratory with Mini Project

SUBJECT CODE: 18CSL58

ACADEMIC YEAR: 2020-2021



Prepared by,

Saritha.M

Suchetha N V

DBMS LABORATORY WITH MINI PROJECT
(Effective from the academic year 2018 -2019)
SEMESTER – V

Course Code	**18CSL58**	**CIE Marks**	40
Number of Contact Hours/Week	0:2:2	**SEE Marks**	60
Total Number of Lab Contact Hours	36	**Exam Hours**	03

Credits – 2

Course Learning Objectives: This course (18CSL58) 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.)

Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Programs List:

PART A

1. Consider the following schema for a Library Database:
BOOK(Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS(Book_id, Author_Name)
PUBLISHER(Name, Address, Phone)
BOOK_COPIES(Book_id, Programme_id, No-of_Copies)
BOOK_LENDING(Book_id, Programme_id, Card_No, Date_Out, Due_Date)
LIBRARY_PROGRAMME(Programme_id, Programme_Name, Address)
Write SQL queries to
    1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each Programme, etc.
    2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
    3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
    4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
    5. **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) COURSE(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 Courses. 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 controlledby 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
- Make sure that the application should have five or more tables
- Indicative areas include; health care

Laboratory Outcomes: The student 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.

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution *(Courseed to change in accoradance with university regulations)*
  - k) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  - l) For laboratories having PART A and PART B
    - i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
    - ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

1. Introduction to Database

Data is collection of facts about the object of interest. For e.g. data about an employee would include information like name, address, age, educational qualifications etc.

Data is raw, just a set of facts which by itself does not convey anything. We need to understand patterns between factual data and give it a meaning. This is called **information** which helps us with answers to questions like who, when, what, where etc. **Synthesis of data and information** leads us to answer the how question and take business decisions. This is referred to as Knowledge.

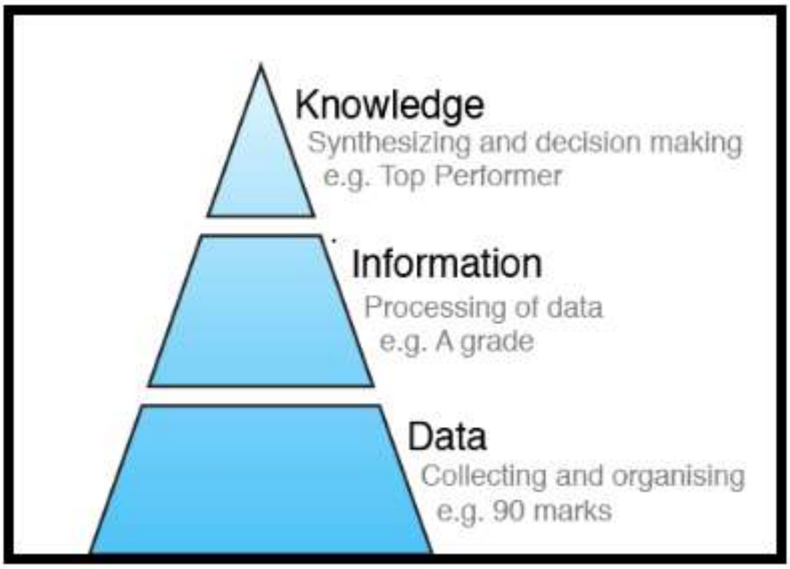


Figure: 1.1 Overview of DBMS

Definition of Database:

A **Database** is a shared collection of logically related data and description of these data, designed to meet the information needs of an organization

Definition of Database Management System:

A **Database Management System** is a software system that enables users to define, create, maintain, and control access to the database. Database Systems typically have high cost and they require high end hardware configurations. An **Application Program** interacts with a database by issuing an appropriate request (typically a SQL statement)

Definition of RDBMS:

RDBMS stands for **R**elational **D**atabase **M**anagement **S**ystem. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

What is a table?

The data in an RDBMS is stored in database objects which are called as **tables**.

This table is basically a collection of related data entries and it consists of numerous columns and rows.

ID	Name	Age	Address	Salary
1	Rama	21	Mangalore	10000
2	Shyam	23	Ujire	20000
3	Krishna	21	Bangalore	25000
4	Ganesh	22	Bangalore	12000
5	Kala	21	Mangalore	24000

What is a field?

Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.

A field is a column in a table that is designed to maintain specific information about every record in the table.

What is a Record or a Row?

A record is also called as a row of data is each individual entry that exists in a table. For example, there are 5 records in the above CUSTOMERS table.

1	Rama	21	Mangalore	10000

What is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table. For Ex: a column in the CUSTOMERS table is Name, which represents customer name description and would be as shown below

Name
Rama
Shyam
Krishna
Ganesh
Kala

SQL Constraints

Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle. Database Systems ensure data integrity through constraints which are used to restrict data that can be entered or modified in the database.

Entity Integrity:Each table must have a column or a set of columns through which we can uniquely identify a row. These column(s) cannot have empty (null) values.   (PRIMARY KEY)
There are no duplicate rows in a table.

Domain Integrity: All attributes in a table must have a defined domain i.e. a finite set of values which have to be used. Enforces valid entries for a given column by restricting the type, the format, or the range of values. When we assign a data type to a column we limit the values that it can contain. e.g. Gender must be M or F.(DATA TYPES)

Referential Integrity: Every value of a column in a table must exist as a value of another column in a different (or the same) table. Rows cannot be deleted, which are used by other records.

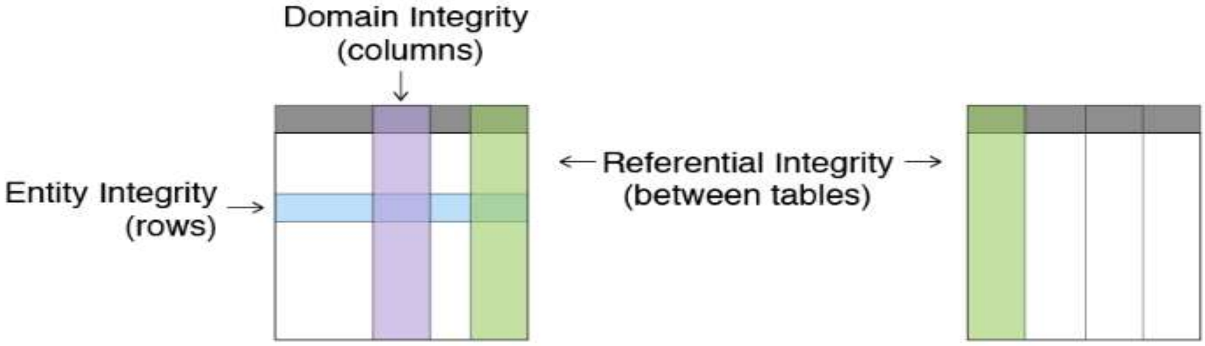


Figure 1.2: SQL Constraints

Candidate key: A Candidate Key is a minimal set of columns/attributes that can be used to uniquely identify a single tuple in a relation. Candidate Keys are determined during database design based on the underlying business rules of the database.

Primary key: Primary key is the candidate key that is selected to uniquely identify a tuple in a relation. Primary key must uniquely identify a tuple,must not allow NULL values**.**

Employee (<u>EmployeeNo,</u> Name, AadharNo, Salary, DateofBirth)

EmployeeNo: Primary key.

When two or more columns together identify the unique row then it's referred to as **Composite Primary Key.** The combination of Name and DateOfBirth if selected as a primary key would be a composite primary key.

Foreign key: A foreign key is a set of one or more columns in the child table whose values are required to match with corresponding columns in the parent table. Foreign key establishes a relationship between these two tables. Foreign key columns on child tables must be primary key or unique on the parent table. The child table can contain NULL values. Let us take Employee and Computer tables as provided below:

Parent/referenced table			Child/referencing table			
Computer table			Employee table			
Computer id	Manufacture	Model	ID	Ename	Dept	Computer id
101	HP	HP51	1	Ram	CSE	101
102	Dell	Dell1	2	Shyam	ECE	102
103	Accer	A-31	3	Sandeep	EE	NULL

SQL Commands:

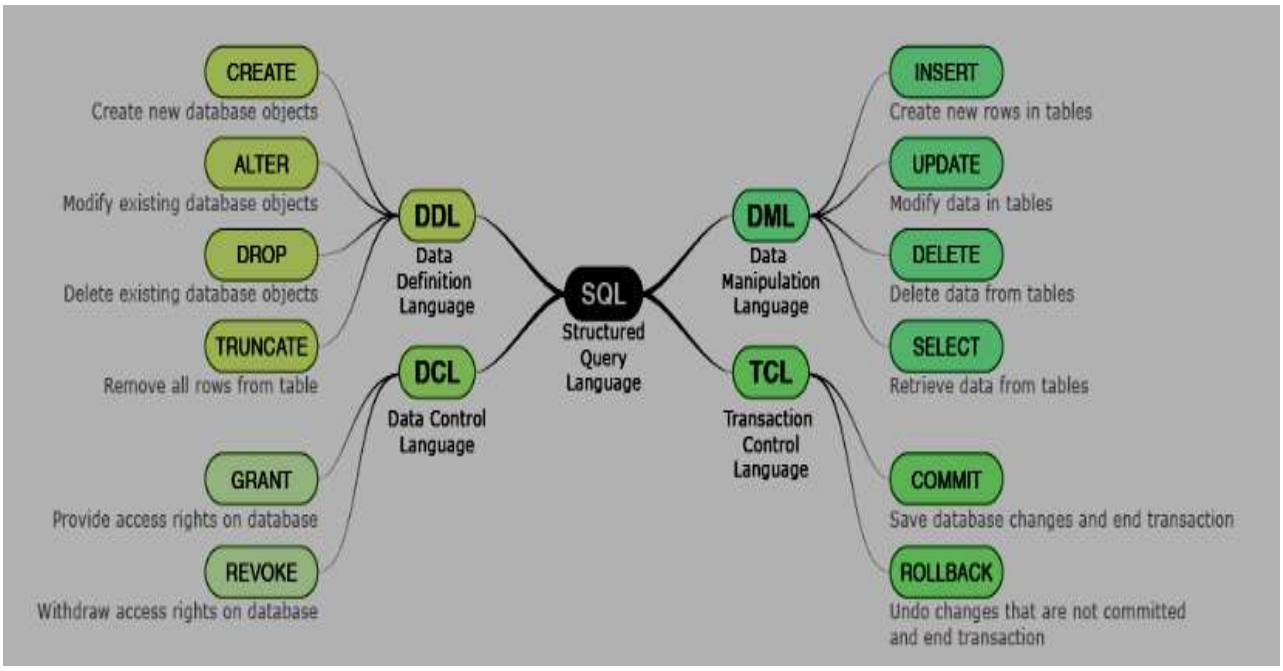

CREATE
Create new database objects
ALTER
Modify existing database objects
DROP
Delete existing database objects
TRUNCATE
Remove all rows from table
GRANT
Provide access rights on database
REVOKE
Withdraw access rights on database
DDL
Data Definition Language
DCL
Data Control Language
SQL
Structured Query Language
DML
Data Manipulation Language
TCL
Transaction Control Language
INSERT
Create new rows in tables
UPDATE
Modify data in tables
DELETE
Delete data from tables
SELECT
Retrieve data from tables
COMMIT
Save database changes and end transaction
ROLLBACK
Undo changes that are not committed and end transaction

Figure 1.3: SQL Commands

DDL - Data Definition Language

Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

DML-Data Manipulation Language: The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

DCL-Data Control Language: It includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

TCL-Transaction Control Language: TCL commands deals with the transaction within the database.

Entity relationship model:

ER model is a graphical representation of entities and their relationships which helps in understanding data independent of the actual database implementation. Let us understand some key terms used in ER Modeling

Entity: Real world objects which have an independent existence and about which we intend to collect data. Example: Employee, Computer

Attribute: A property that describes an entity. An attribute is represented as Oval in an ER diagram

A sample ER Diagram representing the Student entity along with its attributes is presented below

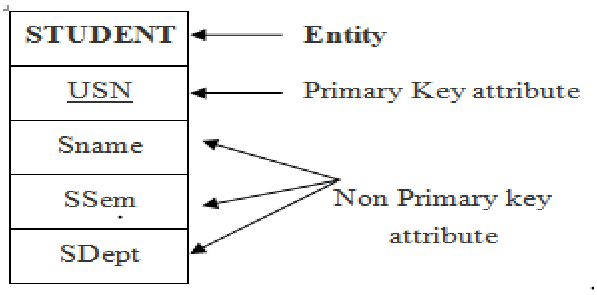


Figure 1.4: Student Entity

Types of Attributes:

1. Key Attributes: A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined in** figure no.**1.4.**

2. Composite attribute: An attribute that is a combination of other attributes is known as composite attribute. For example, in student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country in figure no. 1.5

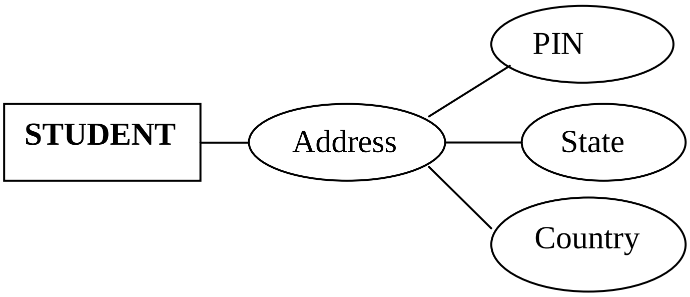


Figure 1.5: Composite attribute

3. Single valued and multi valued attributes: An attribute that has only single value for an entity is known as **single valued attribute.** For example, assume Student is an **entity** and its **attributes** are Name, Age, Address and Phone no. Here the age (attribute) of student (entity) can have only one value. Here, age is **single valued attribute**.

An attribute that can have multiple values for an entity is known as **multi valued attribute.** For example, assume Student is an **entity** and its **attributes** are Name, Age, Address and Phone no. Here the Phone no (attribute) of student (entity) can have multiple value because a student may have many phone numbers. Here, Phone no is **multi valued attribute**.

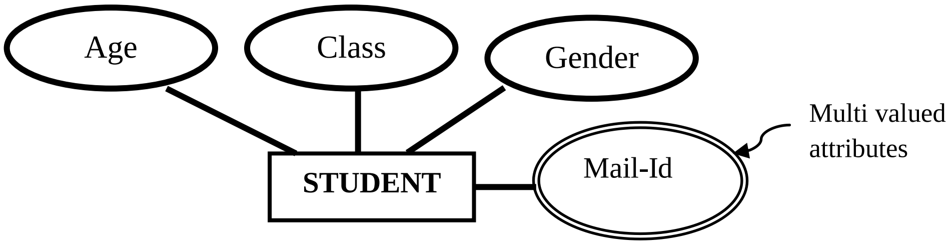


Figure 1.6: Single valued and Multi Valued Attributes

4. **Stored and derived attribute:**

Stored attribute - A simple attribute stored in database is called as stored attribute.

Derived attribute - A value of attribute which can be derived from related stored attribute is called derived attribute.

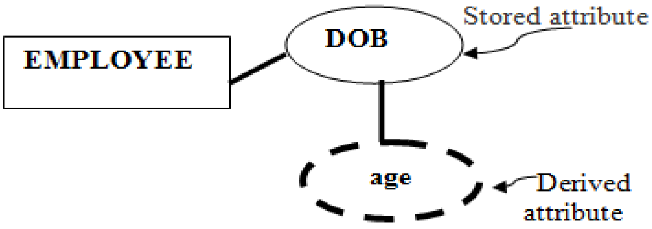


Figure 1.7: Stored and Derived attribute

Relationships:

Relationships are association of one entity with another entity. Each relationship has a name e.g. a Computer **is allocated to** an Employee.

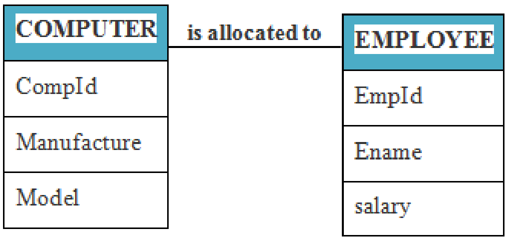


Figure 1.8: Relationship between two Entities

There can be more than one relationship between entities, e.g. an Employee **works in** a Department while the head of department (also an employee) **manages** a Department.

A relationship can also exist between instances of same entity, e.g. an Employee **reports to** another Employee.

One-to-one: When only one instance of an entity is associated with the relationship, it is marked as **'1:1'**. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.

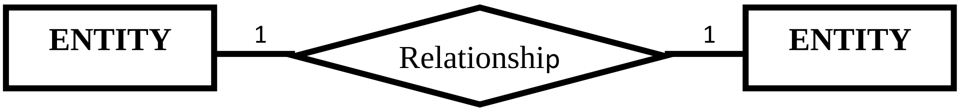


Figure 1.9: One-to-One Relationships

One-to-many: When more than one instance of an entity is associated with a relationship, it is marked as **'1:N'**. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.

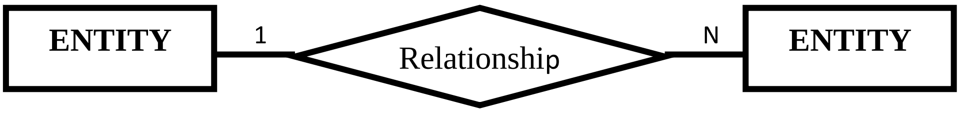


Figure 1.10: One-to-Many Relationships

Many-to-one: When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.

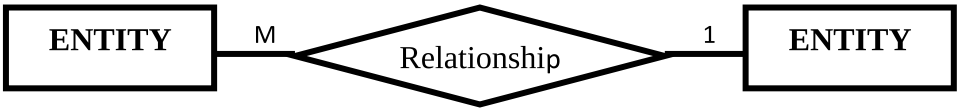


Figure 1.11: Many-to-One Relationships

Many-to-many − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

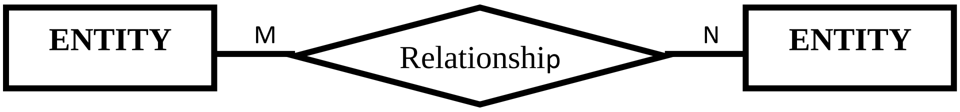


Figure 1.12: Many-to-Many Relationships

Participation Constraints:

- **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.

- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

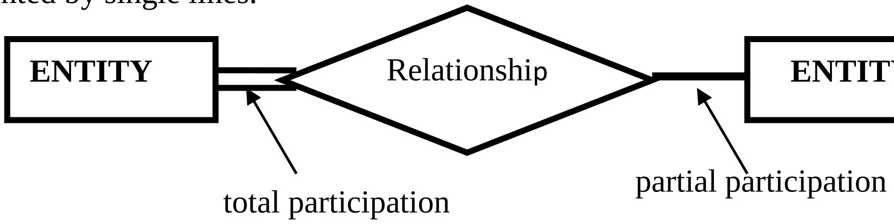


Figure 1.13: Participation constraints

SQL DATATYPES:

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has a related data type in SQL.

SQL data types can be broadly divided into following categories.

1. **Numeric** data types such as int, tinyint, bigint, float, real etc.
2. **Date and Time** data types such as Date, Time, Datetime etc.
3. **Character and String** data types such as char, varchar, text etc.

	CHAR	**VARCHAR2(n)**
Use	Storing characters having pre-determined length	Storing characters whose length vary a lot
Size	size for n characters	size for actual no. of characters + fixed size to store length
Storage	Trailing spaces are applied if data to be stored has smaller length than n.	Trailing spaces are not applied.
Max size	2000 bytes	4000 bytes
Example	A CHAR(10) field will store "Hello" as 10 bytes by appending 5 trailing spaces.	A VARCHAR2(10) field will store "Hello" as 7 bytes (assuming 2 bytes to store length).

4. **Unicode character** string data types, for example nchar, nvarchar, ntext etc.
5. **Binary data** types such as binary, varbinary etc.
6. **Miscellaneous** data types – clob, blob, xml, cursor, table etc.

OPERATORS:

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

Arithmetic Operators:

Operator	Description
+ (Addition)	Adds values on either side of the operator.
- (Subtraction)	Subtracts right hand operand from left hand operand.
* (Multiplication)	Multiplies values on either side of the operator.
/ (Division)	Divides left hand operand by right hand operand.
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.

Comparison operators:

Operator	Description
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
< >	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.

Logical Operators:

Operator	Symbol
And	AND
Or	OR
Not	NOT

CREATE TABLE:

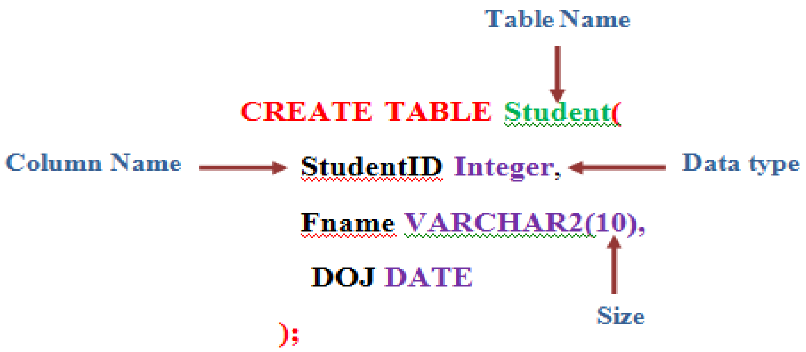
- **Create table** statement is used to create a table in a database. Database tables are organized into rows and columns.
- Each table must have a name and can have any number of columns (minimum 1 column is required).
- Each column must have a data type which determines the type of values that can be stored.
- CREATE TABLE command will fail if a table already exists in the database with same name. All tables must have a unique name.

Syntax:

The basic syntax of the CREATE TABLE statement is as follows −

```
CREATE TABLE table_name(
 column1 datatype,
column2 datatype,
column3 datatype,
.....
columnN datatype,
PRIMARY KEY( one or more columns ));
```

Example:



```
CREATE TABLE Student(
StudentID Integer,
Fname VARCHAR2(10),
 DOJ DATE
);
```

To verify table created or not using **DESC command.**

DESC table_name;

DROP TABLE statement is used to remove an existing table from the database.

Syntax:

The basic syntax of this DROP TABLE statement is as follows −

DROP TABLE table_name;

INSERT QUERY:

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax:

There are two basic syntaxes of the INSERT INTO statement which are shown below.

i) insert into table_name (column1, column2,...columnN) values (value1, value2, ...valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

ii) insert into table_name VALUES (value1,value2,value3,...valueN);

SELECT QUERY:

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows :

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch.

select * from table_name;

SQL - WHERE Clause:

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc.

Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

Ex: SQL> select id, name, salary from customers where salary > 2000;

SQL UPDATE:

UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax:

The basic syntax of the UPDATE queries with a WHERE clause is as follows −

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

SQL DELETE:

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows −

```
DELETE FROM table_name WHERE [condition];
```

SQL ORDER BY QUERY:

Is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

Syntax

The basic syntax of the ORDER BY clause is as follows −

```
SELECT column-list  FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

SQL GROUP BY QUERY:

The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax

The basic syntax of a GROUP BY clause is shown in the following code block. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

```
SELECT column1, column2 FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2;
```

2. LAB EXPERIMENTS

EXPERIMENT 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:

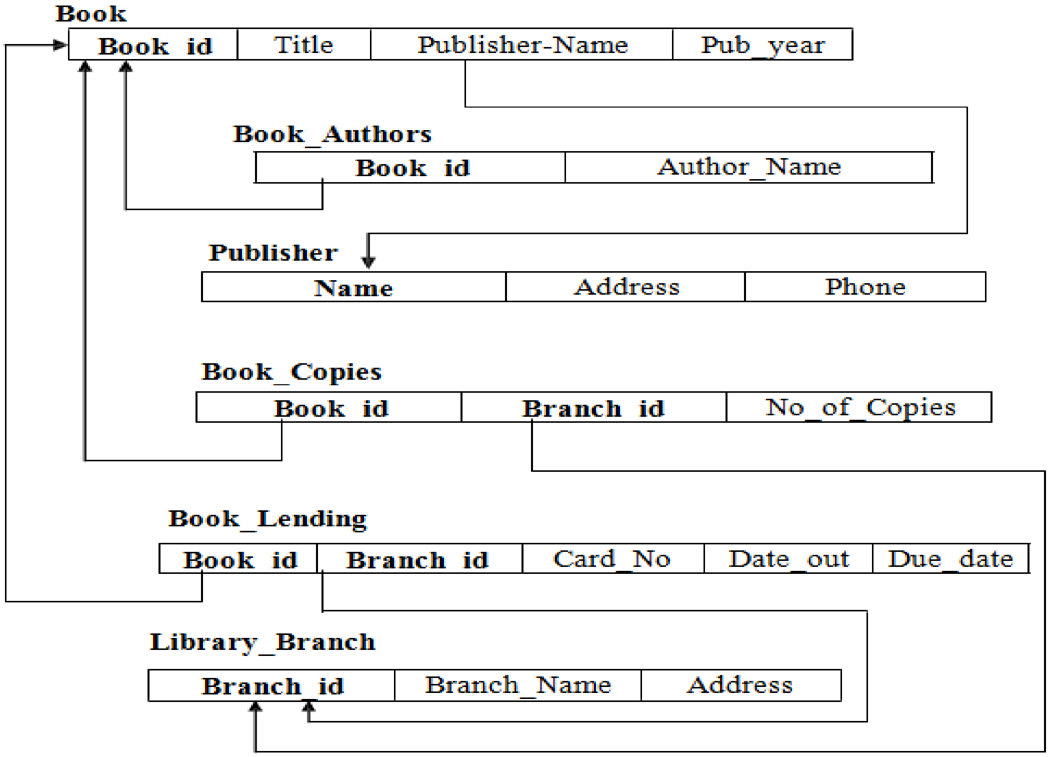


Figure 2.1.1: Schema Diagram for Library Database

Queries for creating tables:

```
create table publisher
          (
          name       varchar(12),
          address    varchar(12),
          phone int,
          primary key(name)
          );

  create table book
          (
          book_id     varchar(5),
          title varchar(15),
          publisher_name varchar(10),
          pub_year int,
          primary key(book_id),
          foreign key(publisher_name) references publisher(name) on delete cascade
          );

  create table book_authors
          (
          book_id            varchar(5),
          author_name     varchar(15),
          primary key(book_id),
          foreign key(book_id) references book(book_id) on delete cascade
          );

  create table library_branch
          (
          branch_id          varchar(5),
          branch_name     varchar(10),
          address varchar(15), primary
          key(branch_id)
          );

  create table book_copies
          (
          book_id       varchar(5),
          branch_id varchar(5),
          no_of_copies int,
          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 book_lending
        (
        book_id       varchar(5),
        branch_id varchar(5),
        card_no varchar(5),
        date_out    date,
        due_date date,
        primary   key(book_id,branch_id,card_no),    foreign
        key(book_id) references book(book_id),
        foreign key(branch_id) references library_branch(branch_id) on delete cascade);
```

Queries for displaying schema of the table:

desc publisher;

```
Name                        Null?                  Type
----------------------      --------              -----------------------
NAME                        NOT NULL              VARCHAR2(12)
ADDRESS                                           VARCHAR2(12)
PHONE                                             NUMBER(38)
```

desc book;

```
Name                        Null?                  Type
---------------------       --------              ---------------------
BOOK_ID                     NOT NULL              VARCHAR2(5)
TITLE                                             VARCHAR2(15)
PUBLISHER_NAME                                    VARCHAR2(10)
PUB_YEAR                                          NUMBER(38)
```

desc book_authors;

```
Name                        Null?                 Type
-----------------------     --------              --------------------
BOOK_ID                     NOT    NULL           VARCHAR2(5)
AUTHOR_NAME                                       VARCHAR2(15)
```

desc library_branch;

```
Name                        Null?                 Type
--------------              ----------            --------------------
BRANCH_ID                   NOT NULL              VARCHAR2(5)
BRANCH_NAME                                       VARCHAR2(10)
ADDRESS                                           VARCHAR2(15)
```

descbook_copies;

```
Name                    Null?           Type
---------------------   --------        -------------------
BOOK_ID                 NOT NULL        VARCHAR2(5)
BRANCH_ID               NOT NULL        VARCHAR2(5)
NO_OF_COPIES                            NUMBER(38)
```

descbook_lending;

```
Name             Null?                  Type
 --------------  --------               -------------------
BOOK_ID          NOT NULL               VARCHAR2(5)
BRANCH_ID        NOT NULL               VARCHAR2(5)
CARD_NO          NOT NULL               VARCHAR2(5)
DATE_OUT                                DATE
DUE_DATE                                DATE
```

<u>Queries for inserting table:</u>

```
insert into publisher values('mcgrawhill','Bangalore','9480506312');
insert into publisher values('pearson','Newdelhi','9785642365');
insert into publisher values('random house','Hydrabad','8796452368');
insert into publisher values('sapna','Chenai','8947589632');
insert into publisher values('oxford','Bangalore','9785642315');

insert into book values('1','DBMS','mcgrawhill','2017');
insert into book values('2','ADBMS','mcgrawhill','2016');
insert into book values('3','CN','pearson','2016');
insert into book values('4','CG','oxford','2015');
insert into book values('5','OS','pearson','2016');

insert into book_authors values('1','navathe');
insert into book_authors values('2','navathe');
insert into book_authors values('3','tenenboum');
insert into book_authors values('4','edward');
insert into book_authors values('5','galvin');

insert into library_branchvalues('10','RR nagar','Bangalore');
insert into library_branch values('11','Manipal','Bangalore');
insert into library_branch values('12','RNSIT','Bangalore');
insert into library_branch values('13','Rajajnagar','Bangalore');
insert into library_branch values('14','Nitte','Mangalore');
insert into book_copiesvalues('1','10','10');
insert into book_copiesvalues('1','11','5');
insert into book_copiesvalues('2','12','2');
insert into book_copiesvalues('2','13','5');
insert into book_copiesvalues('3','14','7');
insert into book_copiesvalues('5','10','1');
insert into book_copiesvalues('4','11','3');

insert into book_lending values('1','10','101','01-jan-17','01-jun-17');
insert into book_lending values('3','14','101','11-jan-17','11-mar-17');
insert into book_lending values('2','13','101','21-feb-17','21-apr-17');
insert into book_lending values('4','11','101','15-mar-17','15-jul-17');
insert into book_lending values('1','11','104','12-apr-17','12-may-17');
```

Queries to display table:

select *from publisher;

```
NAME              ADDRESS           PHONE
------------      ------------      ----------
mcgrawhill        Bangalore         9480506312
pearson           Newdelhi          9785642365
random house      Hydrabad          8796452368
sapna             Chenai            8947589632
oxford            Bangalore         9785642315
```

select *from book;

```
BOOK_ ID          TITLE        PUBLISHER_NAME            PUB_YEAR
-------------     -----------  ------------------------  --------------
1                 DBMS              mcgrawhill                 2017
2                 ADBMS             mcgrawhill                 2016
3                 CN               pearson                     2016
4                 CG               oxford                      2015
5                 OS               pearson                     2016
```

select *from book_authors;

```
BOOK_ID                 AUTHOR_NAME
------------            ---------------------
1                            navathe
2                            navathe
3                            tenenboum
4                            edward
5                            galvin
```

select *from library_branch;

```
BRANCH_ID          BRANCH_NAME             ADDRESS
----------------   ---------------------   -------------
     10                 RR nagar           Bangalore
     11                 Manipal            Bangalore
     12                RNSIT               Bangalore
     13                 Rajajnagar         Bangalore
     14                Nitte               Mangalore
```

select *from book_copies;

```
BOOK_ID            BRANCH_ID               NO_OF_COPIES
--------------     ----------------        ---------------------
     1                  10                      10
     1                  11                      5
     2                  12                      2
     2                  13                      5
     3                  14                      7
     5                  10                      1
     4                  11                      3
```

select *from book_lending;

```
BOOK_ID     BRANCH_ID          CARD_ NO           DATE_OUT           DUE_DATE

-------------   -----------------   ----------------   ----------------   ---------------
1           10                 101                01-JAN-17          01-JUN-17
3           14                 101                11-JAN-17          11-MAR-17
2           13                 101                21-FEB-17          21-APR-17
4           11                 101                15-MAR-17          15-JUL-17
1           11                 104                12-APR-17          12-MAY-17
```

Query 1: select b.book_id,b.title,b.publisher_name,a.author_name,c.no_of_copies,c.branch_id

```
from book b,book_authors a,book_copies c
where b.book_id=a.book_id and b.book_id=c.book_id;
```

<u>OUTPUT</u>

```
BOOK_ID     TITLE     PUBLISHER_NAME   AUTHOR_NAME  NO_OF_COPIES  BRANCH_ID
-----------  --------   ----------------------   ----------------------   -------------------- ---------------
1           DBMS        mcgrawhill        navathe          10                  10
1           DBMS        mcgrawhill        navathe           5                  11
2           ADBMS       mcgrawhill        navathe           2                  12
2           ADBMS       mcgrawhill        navathe           5                  13
3           CN          pearson           tenenboum         7                  14
5           OS          pearson           galvin            1                  10
4           CG          oxford            edward            3                  11

7 rows selected.
```

Query 2:
```
select card_no
        from book_lending
        where date_out between '01-jan-17' and '30-jun-17'
        group by card_no
        having count(*)>3;
```

OUTPUT

```
CARD_NO
--------------
    101
```

Query 3: delete from book where book_id=3;

ID=3 row deleted.

Query 4:
```
create table bookp(
        book_id int,
        title varchar(15),
        pub_name varchar(15),
        pub_year int,
        primary key(book_id))
        partition by range(pub_year) (
        partition p0 values less than(2002),
        partition p1 values less than(2010),
        partition p2 values less than(2015));
```

```
insert into bookp values ('801','dbms','willey','2000');
insert into bookp values ('802','dbms','willey','2009');
insert into bookp values ('803','dbms','willey','2014');
```

OUTPUT

select *from bookp;

```
BOOK_ID          TITLE         PUB_NAME        PUB_YEAR
 ----------      --------       ---------------    --------------
   801           dbms          willey            2000
   802           dbms          willey            2009
   803           dbms          willey            2014
```

select *from bookp partition(p0);

```
BOOK_ID          TITLE         PUB_NAME        PUB_YEAR
----------       -----------   ---------------    --------------
801              dbms            willey           2000
```

select *from bookp partition(p1);

```
BOOK_ID      TITLE        PUB_NAME        PUB_YEAR
------------ ---------    ---------------   --------------
802          dbms         willey            2009
```

Query 5:

```
create view no_of_copies as(
        select book_id,sum(no_of_copies) as
        copies from book_copies
        group by book_id);
```

OUTPUT

select *from no_of_copies;

```
BOOK_ ID     NO_OF_ COPIES
-------------    -----------------------
1                   15
2                   7
3                   7
4                   3
5                   1
```

EXPERIMENT 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:

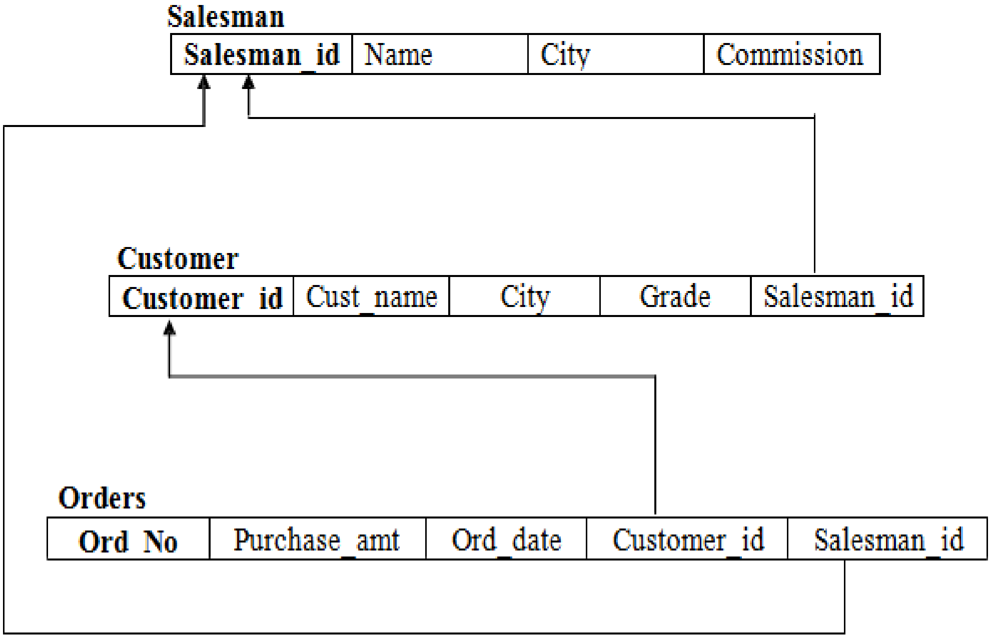


Figure 2.2.1: Schema Diagram for Order database

Queries for creating table:

```
create table salesman(
            sid varchar(5),
            sname varchar(15),
            city varchar(15),
            commission int,
            primary key(sid));
```

```
create table customer(
        cid varchar(5),
        cname varchar(15),
        city varchar(15),
        grade int,
        sid varchar(5),
        primary key(cid),
        foreign key(sid) references salesman(sid) on delete cascade);

create table orders(
        orderno varchar(5),
        purchase_amt int,
        ord_date date,
        cid varchar(5),
        sid varchar(5),
        primary key(orderno),
        foreign key(cid) references customer(cid) on delete cascade,
        foreign key(sid) references salesman(sid) on delete cascad);
```

Queries for displaying schema of the table:

desc salesman;

```
        Name                    Null?                   Type
        -------------------     -------------           ---------------------
        SID                     NOT NULL                VARCHAR2(5)
        SNAME                                           VARCHAR2(15)
        CITY                                            VARCHAR2(15)
        COMMISSION                                      NUMBER(38)
```

desc customer;

```
        Name                    Null?              Type
        --------------          --------           --------------------
        CID                     NOT NULL           VARCHAR2(5)
        CNAME                                      VARCHAR2(15)
        CITY                                       VARCHAR2(15)
        GRADE                                      NUMBER(38)
        SID                                        VARCHAR2(5)
```

desc orders;

```
Name                    Null?            Type
----------------        -------------    ---------------------
ORDERNO                 NOT NULL         VARCHAR2(5)
PURCHASE_AMT                             NUMBER(38)
ORD_DATE                                 DATE
CID                                      VARCHAR2(5)
SID                                      VARCHAR2(5)
```

Queries for inserting table:

```
insert into salesman values('1000','arun','mangalore','15');
insert into salesman values('2000','harsha','bangalore','25');
insert into salesman values('3000','sagar','mysore','20');
insert into salesman values('4000','priya','mangalore','30');
insert into salesman values('5000','divya','delhi','15');

insert into customer values('10','ravi','bangalore','100','1000');
insert into customer values('12','shwetha','mangalore','300','1000');
insert into customer values('14','shalini','chenai','500','2000');
insert into customer values('16','sushmitha','bangalore','700','2000');
insert into customer values('18','pramya','bangalore','500','3000');

insert into orders values('60','5000','04-jan-17','10','1000');
insert into orders values('62','3000','04-jan-17','10','2000');
insert into orders values('64','4000','01-feb-17','12','2000');
insert into orders values('66','450','24-mar-17','14','3000');
insert into orders values('68','2000','01-feb-17','16','1000');
```

Queries to display table:

select *from salesman;

```
SID       SNAME         CITY            COMMISSION
-------   -----------   ------------    -------------------
1000      arun          mangalore              15
2000      harsha        bangalore              25
3000      sagar         mysore                 20
4000      priya         mangalore              30
5000      divya         delhi                  15
```

select *from customer;

```
CID         CNAME           CITY          GRADE        SID
------      -------------   ------------  ------------ ------
10          Ravi            bangalore       100        1000
12          Shwetha         mangalore       300        1000
14          Shalini         chenai          500        2000
16          Sushmitha       bangalore       700        2000
18          Pramya          bangalore       500        3000
```

select *from orders;

```
ORDERNO         PURCHASE_AMT        ORD_DATE        CID      SID
--------------  ------------------- --------------- ------   -----
60                   5000           04-JAN-17       10       1000
62                   3000           04-JAN-17       10       2000
64                   4000           01-FEB-17       12       2000
66                   450            24-MAR-17       14       3000
68                   2000           01-FEB-17       16       1000
```

Query 1: select count(*) as count

```
from customer where grade >(select avg(grade)
                from customer
                where city ='bangalore');
```

<u>OUTPUT</u>

```
COUNT
----------
    3
```

Query 2: select s.sid,sname

```
from salesman s,customer c
where s.sid=c.sid
group by(s.sid,sname)
having count(*)>1;
```

<u>OUTPUT</u>

```
SID          SNAME
-----        -----------
1000         Arun
2000         Harsha
```

Query 3:

```
(select sname,'exists' as same_city from salesman s, customer c
      where s.city=c.city and s.sid=c.sid)
      union
   (select sname,'not exists' as same_city
      from salesman s, customer c
             where s.city!=c.city and s.sid=c.sid);
```

OUTPUT

```
SNAME                   SAME_CITY
-----------              ---------------
arun                    exists
arun                    not exists
harsha                  exists
harsha                  not   exists
sagar                   not exists
```

Query 4:

```
create view sales as
      select s.sid,ord_date,sname
             from salesman s,orders o
                    where s.sid = o.sid and purchase_amt in(select max(purchase_amt)
             from orders
                    group by ord_date);
```

OUTPUT

select *from sales;

```
SID            ORD_DATE              SNAME
------         --------------        ----------
2000           01-FEB-17             harsha
3000           24-MAR-17             sagar
1000           04-JAN-17             arun
```

Query 5: delete from salesman where sid=1000;

OUTPUT

select *from salesman;

```
SID            SNAME              CITY               COMMISSION
-----          -----------        -----------        -------------------
2000           harsha             bangalore                 25
3000           sagar              mysore                    20
4000           priya              mangalore                 30
5000           divya              delhi1                    5
```

EXPERIMENT 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.

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

Schema Diagram:

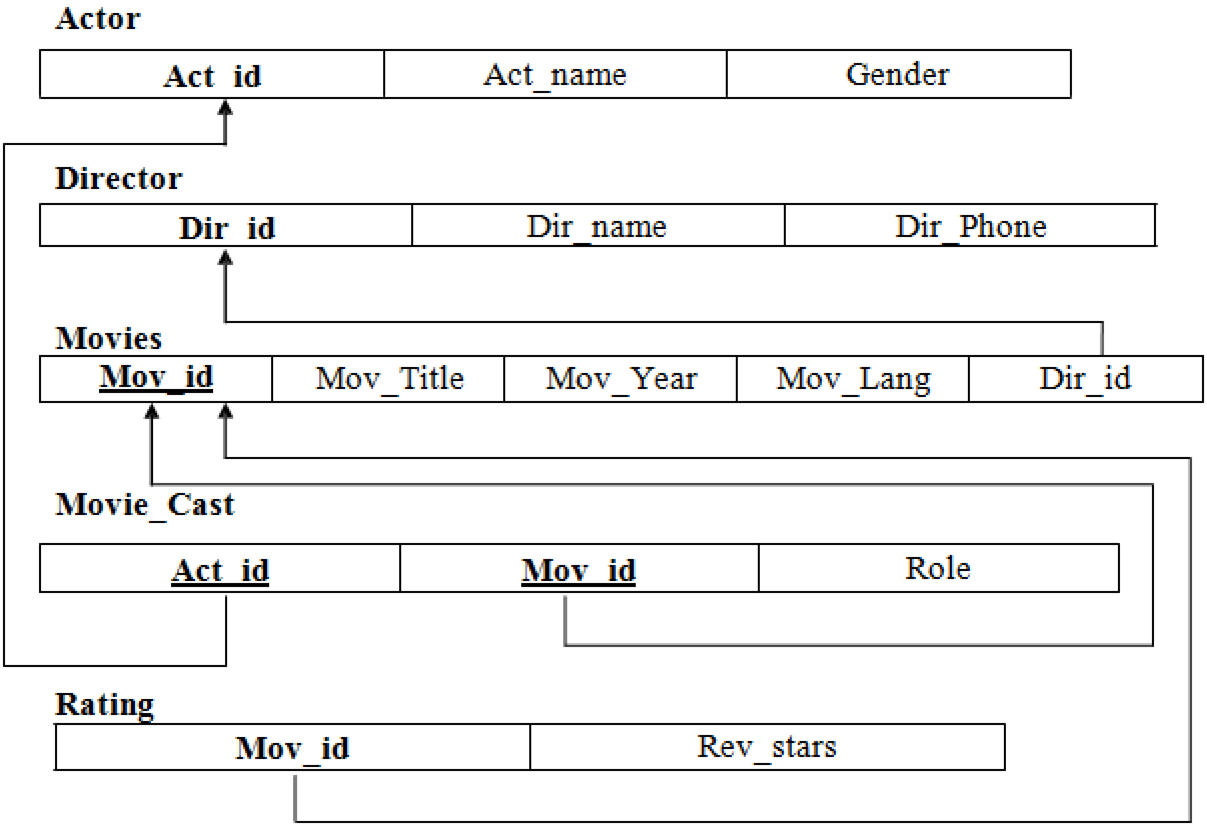


Figure 2.3.1: Schema Diagram for Movie Database

Queries for creating table:

```
create table actor
        (
        act_id int,
        act_name     varchar(10),
        act_gender     varchar(5),
        primary key(act_id)
        );

create table director
        (
        dir_id int,
        dir_name     varchar(20),
        dir_phone int,
        primary key(dir_id)
        );

create table movies(
        mov_id int,
        mov_title     varchar(10),
        mov_year int,
        mov_lang varchar(10),
        dir_id int,
        primary key(mov_id),
        foreign key(dir_id) references director(dir_id)
        );

create table moviecast(
        act_id      int,
        mov_id int,
        role varchar(10),
        primary key(act_id,mov_id),
        foreign key(act_id) references actor(act_id), foreign
        key(mov_id) references movies(mov_id)
        );

create table rating(
        mov_id      int,
        rev_stars int,
        primary key(mov_id),
        foreign key(mov_id) references movies(mov_id)
        );
```

Queries for displaying schema of the table:

desc actor;

```
Name                    Null?           Type
-------------------     -----------     --------------------
ACT_ID                  NOT NULL        NUMBER(38)
ACT_NAME                                VARCHAR2(10)
ACT_GENDER                              VARCHAR2(5)
```

desc director;

```
Name                    Null?           Type
-------------           --------        ------------------
DIR_ID                NOT NULL          NUMBER(38)
DIR_NAME                                VARCHAR2(20)
DIR_PHONE                               NUMBER(38)
```

desc movies;

```
Name                  Null?             Type
-------------           -------- NOT    -------------------
MOV_ID                NULL              NUMBER(38)
MOV_TITLE                               VARCHAR2(10)
MOV_YEAR                                NUMBER(38)
MOV_LANG                                VARCHAR2(10)
DIR_ID                                  NUMBER(38)
```

desc moviecast;

```
Name                    Null?           Type
--------------          --------        ------------------
ACT_ID                  NOT NULL        NUMBER(38)
MOV_ID                  NOT NULL        NUMBER(38)
ROLE                                    VARCHAR2(10)
```

desc rating;

```
Name                    Null?           Type
--------------          --------        ---------------
MOV_ID                  NOT NULL        NUMBER(38)
REV_STARS                               NUMBER(38)
```

Queries for inserting table:

```
insert into actor values('301','Anushka','F');
insert into actor values('302','Prabhas','M');
insert into actor values('303','Punith','M');
insert into actor values('304','Jermy','M');
insert into actor values('305','yash','M');

insert into director values('70','Hitchcock','9487563255');
insert into director values('71','Rajamouli','8948562346');
insert into director values('72','steven spielberg','9823654125');
insert into director values('73','Faran','9786542356');
insert into director values('74','Martin','8974563214');

insert into movies values('1001','Bahubali-2','2017','telugu','70');
insert into movies values('1002','Singham','2008','hindi','71');
insert into movies values('1003','Golmaal-2','2011','telugu','72');
insert into movies values('1004','Allthebest','2015','hindi','73');
insert into movies values('1005','Akash','2009','kannada','74');

insert into moviecast values('301','1001','Heroine');
insert into moviecast values('302','1002','Hero');
insert into moviecast values('303','1003','Guest');
insert into moviecast values('304','1004','Hero');
insert into moviecast values('305','1005','Heroine');

insert into rating values('1001','4');
insert into rating values('1002','2');
insert into rating values('1003','5');
insert into rating values('1004','3');
insert into rating values('1005','4');
```

Queries to display table:

select *from actor;

```
ACT_ID          ACT_NAME        ACT_GENDER
-----------     ---------------  -------------------
     301        Anushka              F
     302        Prabhas              M
     303        Punith               M
     304        Jermy                M
     305        Yash                 M
```

```
select *from director;
 DIR_ID              DIR_NAME                 DIR_PHONE
 ---------           -----------------        -------------------
 70                  Hitchcock                9487563255
 71                  Rajamouli                8948562346
 72                  stevenspielberg          9823654125
 73                  Faran                    9786542356
 74                  Martin                   8974563214
```

```
select *from movies;

 MOV_ID      MOV_TITLE          MOV_YEAR           MOV_LANG          DIR_ID

 ----------  ------------------ ------------------ ------------------ ----------
 1001        Bahubali-2         2017               Telugu              70
 1002        Singham            2008               Hindi               71
 1003        Golmaal-2          2011               Telugu              72
 1005        Akash              2009               Kannada             74
 1004        Allthebest         2015               Hindi               73
```

```
select *from moviecast;

       ACT_ID             MOV_ID             ROLE

       ------------       -------------      ----------
       301                1001               Heroine
       302                1002               Hero
       303                1003               Guest
       304                1004               Hero
       305                1005               Heroine
```

```
select *from rating;
       MOV_ID             REV_STARS
       ------------       -----------------
       1001                     4
       1002                     2
       1003                     5
       1004                     3
       1005                     4
```

Query 1:
```
select mov_title
        from movies m, director d
            where m.dir_id = d.dir_id and dir_name ='Hitchcock';
```

OUTPUT

```
MOV_TITLE
------------------
Bahubali-2
```

Query 2:
```
select distinct mov_title
        from movies m, moviecast mc
            where m.mov_id = mc.mov_id and act_id in(select act_id
                from moviecast
                    group by(act_id) having count(*)>=2)
                    group by(mov_title) having count(*)>=1;
```

OUTPUT

```
MOV_TITLE
----------
Bahubali-2
Allthebest
Bahubali-1
```

Query 3:
```
select act_name
        from (( actor a join moviecast mc on a.act_id=mc.act_id)
        join movies m on m.mov_id=mc.mov_id)
            where mov_year<2000 intersect
    select act_name
        from (( actor a join moviecast mc on a.act_id=mc.act_id)
        join movies m on m.mov_id=mc.mov_id)
            where mov_year>2015;
```

OUTPUT

```
ACT_NAME
----------
Jermy
```

Query 4:
```
select mov_title,max(rev_stars)
        from movies m, rating r
                where m.mov_id= r.mov_id
                group by (mov_title) order by mov_title;
```

OUTPUT

```
MOV_TITLE          MAX(REV_STARS)
---------------    ------------------------
Akash                      4
Allthebest                 3
Bahubali-2                 4
Golmaal-2                  5
Singham                    2
```

Query 5:
```
update rating set rev_stars=5
        where mov_id in(select m.mov_id
                from movies m,director d
                where m.dir_id = d.dir_id and dir_name='stevenspielberg');
```

OUTPUT

```
MOV_ID       REV_STARS
-----------  ---------------
1001                 4
1002                 2
1003                 5
1004                 3
1005                 4
```

EXPERIMENT 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 ‘1BI17CS101’ 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

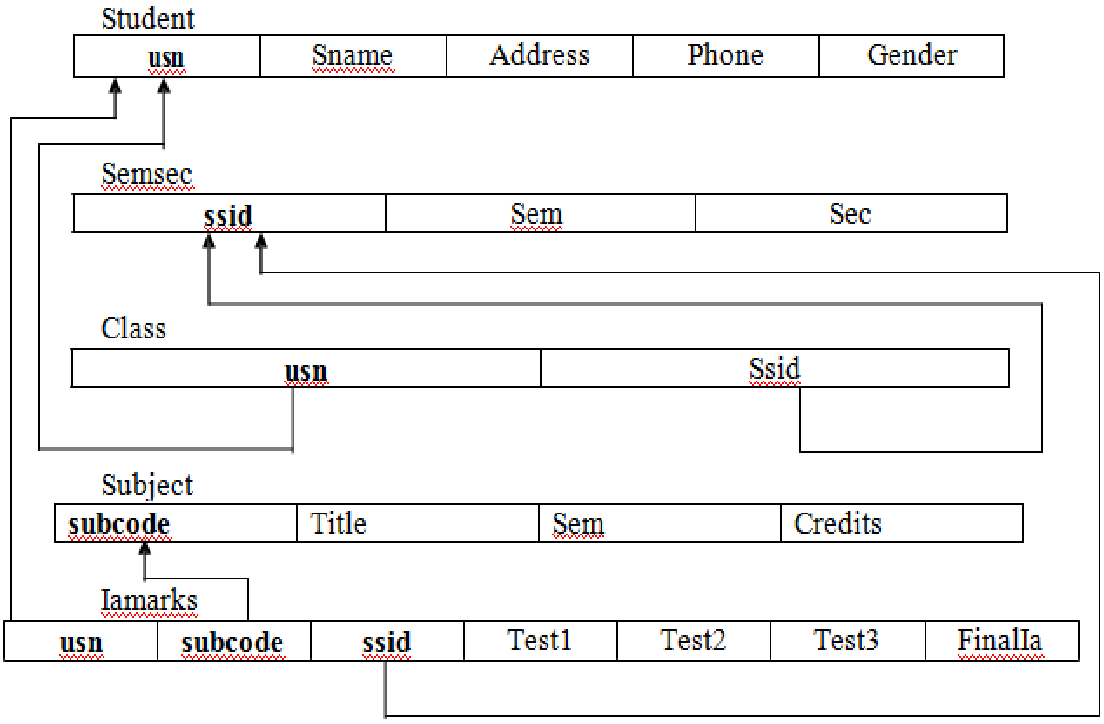


Figure 2.4.1: Schema Diagram for College Database

Queries for creating table:

```
create table student
        (
        usn         varchar(10),
        sname       varchar(15),
        address     varchar(15),
        phone int,
        gender    varchar(6),
        primary key(usn)
        );

create table semsec
        (
        ssid    varchar(5),
        sem int,
        sec        varchar(5),
        primary key(ssid)
        );

create table class
        (
        usn      varchar(10),
        ssid       varchar(5),
        primary key(usn),
        foreign  key(usn)  references  student(usn)  on  delete  cascade,
        foreign key(ssid) references semsec(ssid) on delete cascade
        );

create table subject
        (
        subcode     varchar(8),
        title         varchar(15),
        semint,
        credits int,
        primary key(subcode)
        );
```

```
create table IAmarks
        (
        usn         varchar(10),
        subcode     varchar(8),
        ssid varchar(5),
        test1   int,
        test2 int,
        test3       int,
        finalIA int,
        primary key(usn,subcode,ssid),
        foreign key(usn) references student(usn) on delete cascade,
        foreign key(subcode) references subject(subcode),
        foreign key(ssid) references semsec(ssid)
        );
```

Queries for displaying schema of the table:

desc student;

```
        Name                Null?               Type
        ------------        --------            --------------
        USN                 NOT NULL            VARCHAR2(10)
        SNAME                                   VARCHAR2(15)
        ADDRESS                                 VARCHAR2(15)
        PHONE                                   NUMBER(38)
        GENDER                                  VARCHAR2(6)
```

desc semsec;

```
      Name                Null?               Type
      --------            -------------       ------------------
      SSID                NOT NULL            VARCHAR2(5)
      SEM                                     NUMBER(38)
      SEC                                     VARCHAR2(5)
```

desc class;

```
      Name                Null?               Type
      -------             --------            -----------------
      USN                 NOT NULL            VARCHAR2(10)
      SSID                                    VARCHAR2(5)
```

desc subject;

```
Name               Null?              Type
--------------     --------           --------------
SUBCODE            NOT NULL           VARCHAR2(8)
TITLE                                 VARCHAR2(15)
SEM                                   NUMBER(38)
CREDITS                               NUMBER(38)
```

desc IAmarks;

```
Name               Null?              Type
-----------------  --------           -------------
USN                NOT NULL           VARCHAR2(10)
SUBCODE            NOT NULL           VARCHAR2(8)
SSID                NOT NULL          VARCHAR2(5)
TEST1                                 NUMBER(38)
TEST2                                 NUMBER(38)
TEST3                                 NUMBER(38)
FINALIA                               NUMBER(38)
```

<u>Queries for inserting table:</u>

```
insert into student values('CS101','Arun','ujire','9481235681','Male');
insert into student values('CS102','Pramya','Mangalore','8945689532','Female');
insert into student values('CS103','Ravi','Bangalore','9568742361','Male');
insert into student values('CS104','Vani','Puttur','8945623145','Female');
insert into student values('CS105','Akshatha','Bantwal','9845632147','Female');
insert into student values('CS106','Ranjan','Karwar','9485632158','Male');

insert into semsec values('CS4A','4','A');
insert into semsec values('CS4B','4','B');
insert into semsec values('CS4C','4','C');
insert into semsec values('CS8A','8','A');
insert into semsec values('CS8B','8','B');
insert into semsec values('CS8C','8','C');

insert into class values('CS101','CS8A');
insert into class values('CS102','CS8A');
insert into class values('CS103','CS8B');
insert into class values('CS104','CS8C');
insert into class values('CS105','CS4C');
insert into class values('CS106','CS4C');
insert into subject values('15CS41','ACA','4','5 ');
insert into subject values('15CS42 ','GTE','4','4');
insert into subject values('15CS43','C++','4','3');
insert into subject values('10CS81','JAVA','8','4');
insert into subject values('10CS82','WEB','8','4');
insert into subject values('10CS83','IOT','8','5');

insert into IAmarks values('CS101','10CS81','CS8A','19','20','18',");
insert into IAmarks values('CS102','10CS81','CS8A','16','15','12',");
insert into IAmarks values('CS103','10CS82','CS8B','09','08','12',");
insert into IAmarks values('CS104','10CS83','CS8C','03','05','08',");
insert into IAmarks values('CS105','15CS41','CS4C','10','14','16',");
insert into IAmarks values('CS106','15CS42','CS4C','13','15','20',");
```

Queries to display table:

select *from student;

```
USN          SNAME        ADDRESS          PHONE            GENDER
----------   -----------  ---------------  ----------       ------------
CS101        Arun         Ujire            9481235681       Male
CS102        Pramya       Mangalore        8945689532       Female
CS103        Ravi         Bangalore        9568742361       Male
CS104        Vani         Puttur           8945623145       Female
CS105        Akshatha     Bantwal          9845632147       Female
CS106        Ranjan       Karwar           9485632158       Male
```

select *from semsec;

```
        SSID      SEM          SEC
        -----    -------       -----
        CS4A       4           A
        CS4B       4           B
        CS4C       4           C
        CS8A       8           A
        CS8B       8           B
        CS8C       8           C
```

select *from class;

```
        USN          SSID
        ----------   -------
        CS101        CS8A
        CS102        CS8A
        CS103        CS8B
        CS104        CS8C
        CS105        CS4C
        CS106        CS4C
```

select * from subject;

```
SUBCODE           TITLE       SEM        CREDITS
---------------   ----------   -------   -------------
15CS41            ACA         4       5
15CS42            GTE         4       4
15CS43            C++         4       3
10CS81            JAVA        8       4
10CS82            WEB         8       4
10CS83            IOT         8       5
```

select * from iamarks;

```
USN         SUBCODE     SSID     TEST1     TEST2     TEST3    FINALIA
----------  ------------  ------  ----------  ---------  --------  ----------
CS101       10CS81      CS8A       19       20           18
CS102       10CS81      CS8A       16       15           12
CS103       10CS82      CS8B        9       8            12
CS104       10CS83      CS8C        3        5          8
CS105       15CS41      CS4C       10      14            16
```

Query 1:

```
select s.usn,sname,gender,address,phone
          from student s,semsecsc,class c
                where s.usn=c.usn and c.ssid=sc.ssid and sc.sem=4 and sc.sec=’c’;
```

OUTPUT

```
USN         SNAME        ADDRESS           PHONE            GENDER
----------  -----------  ---------------   ----------       ------------
CS105       Akshatha     Bantwal           9845632147       Female
CS106       Ranjan       Karwar            9485632158       Male
```

Query 2:

```
select sem,sec,gender,count(*) as count
         from student s,semsecsc,class c
               where s.usn=c.usn and sc.ssid=c.ssid group by(sem,sec,gender);
```

OUTPUT

```
 SEM              SEC         GENDER           COUNT
-------           -----       -------------    ----------
4                 C           Male             1
8                 A           Male             1
8                 B           Male             1
8                 C           Female           1
4                 C           Female           1
8                 A           Female           1
```

Query 3: create view test1 as

```
(
     select usn,test1,subcode
               from iamarks
               where usn='CS101'
);
```

OUTPUT

```
USN            TEST1          SUBCODE
----------      --------       ------------
CS101            19           10CS81
```

Query 4: create view average_finder1 as

```
(
      select usn,subcode,greatest(test1,test2,test3) as highest, case
      when  test1<greatest(test1,test2,test3)  and  test1>least(test1,test2,test3)  then test1
      when  test2<greatest(test1,test2,test3)  and  test2>least(test1,test2,test3) then test2
      else test3
      end as second_highest from iamarks
 );
```

OUTPUT

```
USN          SUBCODE      HIGHEST    SECOND_HIGHEST
----------    -------------   -----------   -------------------------
CS101        10CS81       20                19
CS102        10CS81       16                15
CS103        10CS82       12                9
CS104        10CS83        8                5
CS105        15CS41       16                14
```

```
update iamarks a set finalia = (
        select (highest+second_highest)/2
        from average_finder
        where a.usn =usn and a.subcode= subcode);
```

OUTPUT

```
USN          SUBCODE     SSID      TEST1      TEST2      TEST3     FINALIA
-------       -------- ---   -------    --------    --------    ----------   ----------
CS101        10CS81      CS8A        19       20            18      20
CS102        10CS81      CS8A        16       15            12      16
CS103        10CS82      CS8B         9       8             12      11
CS104        10CS83      CS8C         3       5              8      7
CS105        15CS41      CS4C        10      14             16      15
```

Query 5:

```
select usn, subcode, case
        when finalia>=17 and finalia<=20 then 'outstanding'
        when finalia>=12 and finalia<=16 then 'average'
        when finalia<12 then 'weak'
        end as category
        from iamarks where usn in
        (select usn
        from semsecsc,class c
        where sc.ssid=c.ssid and sem=8);
```

<u>OUTPUT</u>

```
USN          SUBCODE     CATEGORY
----------   --------    -----------
CS101        10CS81      Outstanding
CS102        10CS81      Average
CS103        10CS82      Weak
CS104        10CS83      Weak
```

EXPERIMENT 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:

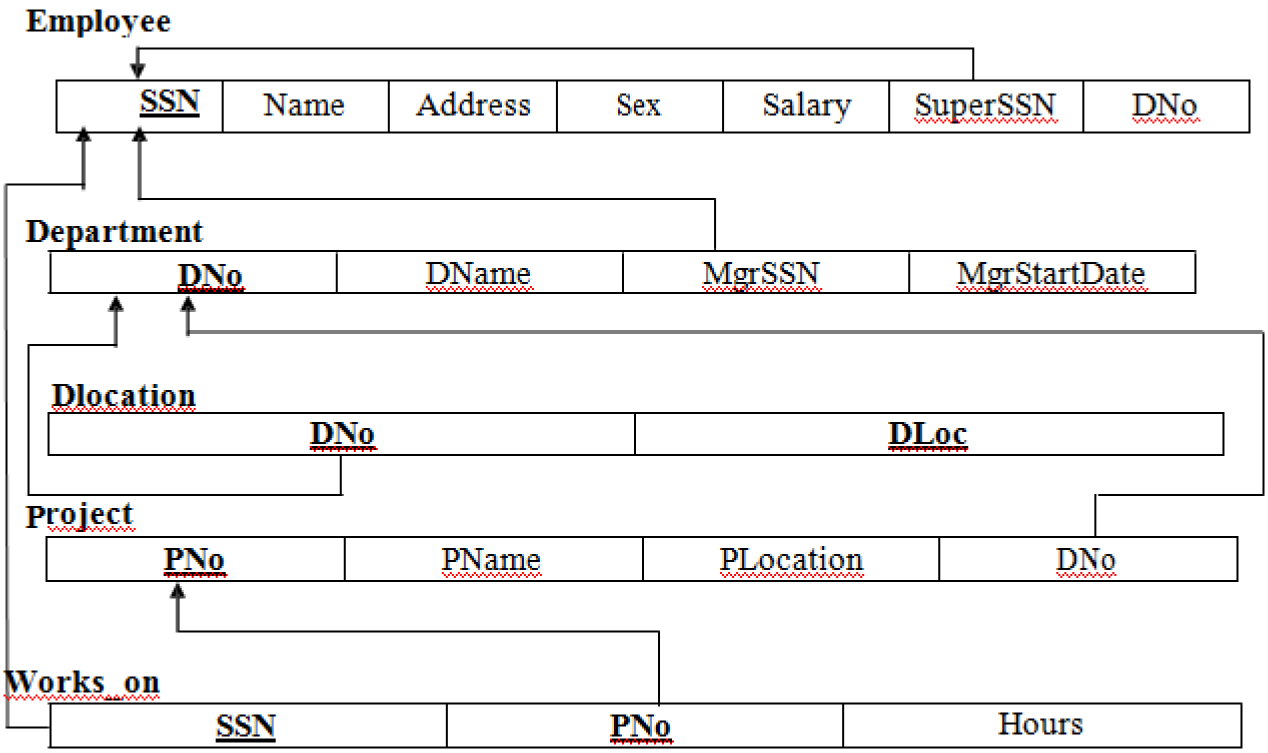


Figure 2.5.1: Schema Diagram for Company Database

Queries for creating table:

```
create table Department
        (
        DNo varchar(5),
        DName varchar(15),
        MgrSSN varchar(15),
        Mgrstartdate date,
        primary key(DNo)
        );

        create table Employee
        (
        SSN varchar(5),
        Name varchar(15), Address
        varchar(15), Sex varchar(6),
        Salary int,
        SuperSSN varchar(5),
        DNo varchar(5),
        primary key(SSN)
        );



        alter table Employee  add  foreign key(DNo) references Department(DNo) on delete cascade;
       alter table Employee  add  foreign key(SuperSSN) references Employee(SSN) on delete
       cascade;
       alter table Department  add  foreign key(MgrSSN) references Employee(SSN) on delete
      cascade;

      create table Dlocation
        (
        Dno varchar(5),
        Dloc varchar(10),
        primary key(Dno,Dloc),
        foreign key(DNo) references Department(DNo) on delete cascade
        );

      create table Project
        (
        PNo varchar(5),
        PName varchar(15), Plocation
        varchar(10), DNO varchar(5),
        primary key(PNo),
        foreign key(DNo) references Department(DNo) on delete cascade
        );
```

```
create table Works_on
        (
        SSN varchar(5),
        PNo varchar(5),
        Hours int,
        primary key(SSN,PNO),
        foreign key(SSN) references Employee(SSN) on delete cascade, foreign
        key(PNo) references Project(PNo) on delete cascade
        );
```

Queries for displaying schema of the table:

desc Department;

```
Name                      Null?              Type
----------------------    --------           ---------------------
DNO                       NOT NULL           VARCHAR2(5)
DNAME                                        VARCHAR2(15)
MGRSSN                                       VARCHAR2(15)
MGRSTARTDATE                                 DATE
```

desc Employee;

```
Name                      Null?              Type
-----------------         --------           -----------------
SSN                       NOT NULL           VARCHAR2(5)
 NAME                                        VARCHAR2(15)
ADDRESS                                      VARCHAR2(15)
SEX                                          VARCHAR2(6)
SALARY                                       NUMBER(38)
SUPERSSN                                     VARCHAR2(5)
DNO                                          VARCHAR2(5)
```

desc Project;

```
--------------------      --------           -----------------
PNO                       NOT NULL           VARCHAR2(5)
PNAME                                        VARCHAR2(15)
PLOCATION                                    VARCHAR2(10)
DNO                                          VARCHAR2(5)
```

desc Works_on;

```
                          Null?              Type
--------------------      ------------       ------------------
SSN                        NOT NULL          VARCHAR2(5)
PNO                         NOT NULL         VARCHAR2(5)
HOURS                                        NUMBER(38)
```

```
  insert into Employee values('11','scott','Bangalore','M','600000',",");
  insert into Employee values('12','john','Mangalore','M','500000',",");
  insert into Employee values('13','James','Hassan','M','400000',",");
  insert into Employee values('14','kavitha','Puttur','F','700000',",");
  insert into Employee values('15','Kavya','Ujire','F','800000',",");
  insert into Employee values('16','Veena','Pune','F','900000', ",");
  insert into Employee values('17','Nagesh','Mysore','M','700000',",");
```

select *from Employee;

```
SSN        NAME       ADDRESS     SEX     SALARY   SUPERSSN  DNO
-----      ---------- ----------- ------  -------  --------- -------
11         scott      Bangalore   M       600000
12         john       Mangalore   M       500000
13         James      Hassan      M       400000
14         kavitha    Puttur      F       700000
15         Kavya      Ujire       F       800000
17         Nagesh     Mysore      M       700000
16         Veena      Pune        F       900000
```

```
insert into Department values('1’,'Datamining','11','16-may-17');
insert into Department values('2','Administration','12','15-may-17');
insert into Department values('3','Networking','13','05-may-17');
insert into Department values('4','Testing','14','12-jun-18');
insert into Department values('5','accounts','15','15-jun-18');
```

select *from Department;

```
     DNO        DNAME            MGRSSN       MGRSTARTDATE
     -----      ------------     ------------  ----------------------
     1          Datamining              11       16-MAY-17
     2          Administration          12       15-MAY-17
     3          Networking              13       05-MAY-17
     4          Testing                 14       12-JUN-18
     5          accounts                15       15-JUN-18
```

```
update Employee set SuperSSN='14',DNo='1' where SSN='11';
update Employee set SuperSSN='14',DNo='5' where SSN='12';
update Employee set SuperSSN='14',DNo='5' where SSN='13';
update Employee set SuperSSN='17',DNo='5' where SSN='14';
update Employee set SuperSSN='17',DNo='5' where SSN='15';
update Employee set SuperSSN='17',DNo='5' where SSN='16';
update Employee set SuperSSN='17',DNo='5' where SSN='17';
```

select *from Employee;

```
SSN   NAME        ADDRESS     SEX    SALARY      SUPERSSN         DNO
----- ----------- ----------- ------ ----------  ---------------  --------
11    scott       Bangalore   M      600000           14          1
12    john        Mangalore   M      500000           14          5
13    James       Hassan      M      400000           14          5
14    kavitha     Puttur      F      700000           17          5
15    Kavya       Ujire       F      800000           17          5
17    Nagesh      Mysore      M      700000           17          5
16    Veena       Pune        F      900000           17          5
```

```
insert into DLocation values('1','Venoor');
insert into DLocation values('2','Karkala');
insert into DLocation values('3','Puttur');
insert into DLocation values('4','Kerala');
insert into DLocation values('5','Pune');
```

select *from DLocation;

```
DNO        DLOC
-----      ----------
1          Venoor
2          Karkala
3          Puttur
4          Kerala
5          Pune
```

```
insert into Project values('100','IOT','Mumbai','1');
insert into Project values('200','Bigdata','Pune','1');
insert into Project values('300','Database','Shirsi','5');
insert into Project values('400','Cloudcomputing','UK','5');
insert into Project values('500','Android','DK','5');
```

select *from Project;

```
PNO        PNAME              PLOCATION          DNO
------     -----------        ------------------ -------
100        IOT                Mumbai             1
200        Bigdata            Pune               1
300        Database           Shirsi             5
500        Android            DK                 5
400        Cloudcomputing     UK                 5
```

```
insert into Works_onvalues('11','100','30');
insert into Works_onvalues('11','300','10');
insert into Works_onvalues('12','300','50');
insert into Works_onvalues('12','400','50');
insert into Works_onvalues('12','500','50');
insert into Works_onvalues('13','200','50');
```

select *from Works_on;

```
SSN         PNO         HOURS
-----       -----       ----------
11          100         30
11          300         10
12          300         50
12          400         50
12          500         50
13          200         50
```

Query 1:
```
(select pno
     from works_onw,employee e
     where name='scott' and w.ssn=e.ssn)
Union
(select pno
     from employee e,departmentd,project p
     where e.ssn=d.mgrssn and d.dno=p.dno and name='scott');
```

<u>OUTPUT</u>

```
PNO
-----
100
200
300
```

Query 2:
```
Select salary*1.1 as salary
     from employee e,projectp,works_on w
     where pname='IOT' and p.pno=w.pno and w.ssn=e.ssn;
```

<u>OUTPUT</u>

```
 SALARY
----------
660000
```

Query 3:
```
select max(salary),min(salary),sum(salary),avg(salary)
     from employee e,department d
     where e.dno=d.dno and dname='accounts';
```

<u>OUTPUT</u>

```
MAX(SALARY)          MIN(SALARY)          SUM(SALARY)          AVG(SALARY)
-------------------  ------------------   -------------------  -------------------
900000               400000               4000000              666666.667
```

Query 4:
```
select name
 from employee e
 where not exists ((select pno
                    from project
                    where dno=5)
                    minus
                    (select pno
                    from works_on w where
                    w.ssn = e.ssn));
```

OUTPUT

```
NAME
---------------
John
```

Query 5:
```
select dno,count(*) as no_of_employees
 from employee
 where salary>600000 and dno in(select dno from employee group by dno having
count(*)>3) group by dno;
```

OUTPUT:

```
DNO          NO_OF_EMPLOYEES
-----        -------------------------
5                   4
```

VIVA QUESTIONS

1. **What Is Database Or Database Management Systems (dbms)?**

Database provides a systematic and organized way of storing, managing and retrieving from collection of logically related information. Secondly the information has to be persistent, that means even after the application is closed the information should be persisted. Finally it should provide an independent way of accessing data and should not be dependent on the application to access the information.

2. **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.

3. **How Many Types Of Database Languages Are?**

There are four types of database languages:

1. Data Definition Language (DDL) e.g. CREATE, ALTER, DROP etc.
2. Data Manipulation Language (DML) e.g. SELECT, UPDATE, INSERT etc.
3. DATA Control Language (DCL) e.g. GRANT and REVOKE.
4. Transaction Control Language (TCL) e.g. COMMIT and ROLLBACK.

4**. What Is An Attribute?**

It is a particular property, which describes the entity.

5. **What Is Stored Procedure?**

A stored procedure is a named group of SQL statements that have been previously created and stored in the server database.

6. **What Is Normalization?**

Normalization is a process of analyzing the given relation schemas according to their functional dependencies. It is used to minimize redundancy and also minimize insertion, deletion and update

7. **What Is The Sql " In " Clause?**

SQL IN operator is used to see if the value exists in a group of values. For instance the below

SQL checks if the Name is either 'David' or 'Craig' SELECT * FROM wbEmployee WHERE name IN ('David','Craig') Also you can specify a not clause with the same. SELECT * FROM wbEmployee WHERE age NOT IN (30,25)

8. **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.

9. **What Is Data Independence?**

Data independence specifies that the application is independent of the storage structure and the access strategy of data. It means the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.
There are two types of data
independence:

1. Physical data independence
2. Logical data independence

10. What Is Sql?

SQL stands for Structured Query Language.SQL is an ANSI (American National StandardsInstitute) standard computer language for accessing and manipulating database systems. SQLstatements are used to retrieve and update data in a database.

11. **Define sub-query.**

A query contained by a query is called Sub-query.

12. **Why is group-clause used?**

Group-clause uses aggregate values to be derived by collecting similar data.

13. **Define Aggregate functions.**

Functions which operate against a collection of values and returning single value is called aggregate functions

14. **What restrictions can you apply when you are creating views?**

Restrictions that are applied are:
Only the current database can have views.

- ➢ You are not liable to change any computed value in any particular view.
- ➢ Integrity constants decide the functionality of INSERT and DELETE.
- ➢ Full-text index definitions cannot be applied.
- ➢ Temporary views cannot be created.
- ➢ Temporary tables cannot contain views.
- ➢ No association with DEFAULT definitions.
- ➢ Triggers such as INSTEAD OF is associated with views.

15. **Define "correlated subqueries".**
A 'correlated subquery' is a sort of sub query but correlated sub query is reliant on another query for a value that is returned. In case of execution, the sub query is executed first and then the correlated query.

16. **Define Join and enlist its types.**

Joins help in explaining the relation between different tables. They also enable you to select data with relation to data in another table.

The various types are:

1. INNER JOINs: Blank rows are left in the middle while more than equal to two tables are joined.
2. OUTER JOINs: Divided into Left Outer Join and Right Outer Join. Blank rows are left at the specified side by joining tables in other side.

17. **Define Entity.**

It can be defined as being a 'thing' with an independent existence in the real world.

18. **What do you mean by Entity type?**

A set of entries having similar attributes are entity types.

19. **Define Entity Set.**

Compilation of all entries of any particular type of entry in the database is called Entity Set.

20. **What is Data Model?**

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

21. **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.

22. **What is DML Compiler?**

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

23. **What is 1 NF (Normal Form)?**

The first normal form or 1NF is the first and the simplest type of normalization that can be implemented in a database. The main aims of 1NF are to:

1. Eliminate duplicative columns from the same table.

2. Create separate tables for each group of related data and identify each row with a unique column (the primary key).

24. **What is Fully Functional dependency?**

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

25. **What is a view?**

A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

26. **What is Trigger?**

A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs.

27. **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.

28. **What do you mean by atomicity and aggregation?**

Atomicity-Atomicity states that database modifications must follow an “all or nothing” rule. Each transaction is said to be “atomic.” If one part of the transaction fails, the entire transaction fails.

Aggregation - A feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.

29. **What is VDL (View Definition Language)?**

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

30. **What is SDL (Storage Definition Language)?**

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

31. **Explain the difference between two and three-tier architectures?**

Three-tier architecture includes a client and two server layers.

The application code is stored on the application server and the database is stored on the database server. A two-tier architecture includes a client and one server layer. The database is stored on the database server.

32. **Briefly describe the three types of SQL commands?**

Data definition language commands are used to create, alter, and drop tables. Data manipulation commands are used to insert, modify, update, and query data in the database. Data control language commands help the DBA to control the database.

33. **List some of the properties of a relation?**

Relations in a database have a unique name and no multi valued attributes exist. Each row is unique and each attribute within a relation has a unique name. The sequence of both columns and rows is irrelevant.

34. **What is a Catalog?**

A catalog is a table that contains the information such as structure of each file, the type and storage format of each data item and various constraints on the data .The information stored in the catalog is called Metadata.

35. **Describe the three levels of data 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.

36. **What is Data Independence?**
Data independence means that the application is independent of the storage structure and access strategy of data.

37. **How many types of relationship exist in database designing?**
There are three major relationship models:- One-to-one.One-to-Many,Many-to-One,Many-to-Many

38. **What is order by clause?**

ORDER BY clause helps to sort the data in either ascending order to descending

39. **What is difference between DELETE & TRUNCATE commands?**

Delete command removes the rows from a table based on the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table and there will be no data in the table after we run the truncate command.

40. **What is a transaction?**
A transaction is a logical unit of database processing that includes one or more database access operations.

41. **Explain the differences between structured data and unstructured data.**

Structured data are facts concerning objects and events. The most important structured data are numeric, character, and dates. Structured data are stored in tabular form. Unstructured data are multimedia data such as documents, photographs, maps, images, sound, and video clips. Unstructured data are most commonly found on Web servers and Web-enabled databases.

42. **Explain minimum and maximum cardinality?**

Minimum cardinality is the minimum number of instances of an entity that can be associated with each instance of another entity. Maximum cardinality is the maximum number of instances of an entity that can be associated with each instance of another entity.

43. **Explain what we mean by an ACID transaction.**

An ACID transaction is one that is atomic, consistent, isolated, and durable. Durable means that database changes are permanent. Consistency can mean either statement level or transaction level consistency. With transaction level consistency, a transaction may not see its own changes. Atomic means it is performed as a unit.

44. **What is Specialization?**
It is the process of defining a set of subclasses of an entity type where each subclass contain all the attributes and relationships of the parent entity and may have additional attributes and relationships which are specific to itself.

45. **What is generalization?**
It is the process of finding common attributes and relations of a number of entities and defining a common super class for them.

46. **What is a foreign key?**

A key of a relation schema is called as a foreign key if it is the primary key of some other relation to which it is related to.

47. **What is schema?**

The description of a data base is called the database schema , which is specified during database design and is not expected to change frequently . A displayed schema is called schema diagram .We call each object in the schema as schema construct.

48. **What is the difference between ORDERBY and GROUPBY?**

A. ORDERBY performs sorting while GROUPBY AGGREGATES Data
B. GROUPBY sorts data while ORDERBY puts data in order
C. Both perform sorting. D. None of the above
Answer: A

The ORDER BY performs a sort operation. So think of a telephone phone directory.
SELECT NAME FROM DIRECTORY ORDER BY NAME
This would ensure that the result set would be sorted in (by default) ascending order.

The GROUP BY operation aggregates data in your result set. Continuing the example of the telephone directory
SELECT CITY, COUNT(CITY) FROM DIRECTORY GROUP BY CITY
This would ensure that the result set would be grouped according to the city where the individual lives. The COUNT and GROUP BY works in conjunction.

50. **Can You Explain Insert, Update And Delete Query In Dbms?**
Insert statement is used to insert new rows in to table. Update to update existing data in the table.
Delete statement to delete a record from the table. Below code snippet for Insert, Update and Delete:-
INSERT INTO wb Employee SET name='maxwell',age='22';
UPDATE wb Employee SET age='22' where name='maxwell';
DELETE FROM wbEmployee WHERE name = 'david';