**Queries:**

**Create Table:**

create table CUSTOMER (

Customer\_Id number,

Email varchar2(50),

First\_name varchar2(50) not null,

Last\_name varchar2(50) not null,

constraint pk\_customer primary key (Customer\_Id)

);

create table Invoice(

Status varchar2(50),

Customer\_Id number,

Bill\_Id number,

FOREIGN KEY (Customer\_Id) REFERENCES Customer(Customer\_Id),

FOREIGN KEY (Bill\_Id) REFERENCES Bill(Bill\_Id)

);

**Alter:**

ALTER TABLE Customer MODIFY First\_name varchar(100);

**Insert:**

insert into CUSTOMER (Customer\_Id, Email, First\_Name, Last\_Name) values

(4,'rajputriya@gmail.com','Riya','Rajput');

insert into CUSTOMER (Customer\_Id, Email, First\_Name, Last\_Name) values

(5,'aakash12@gmail.com','Aakash','Jain');

insert into CUSTOMER (Customer\_Id, Email, First\_Name, Last\_Name) values

(6,'Mahima44@gmail.com','Mahima','Jaisingh');

insert into CUSTOMER (Customer\_Id, Email, First\_Name, Last\_Name) values

(7,'Amarsawla@gmail.com','Amar','Sawla');

**IN:**

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

**AND:**

SELECT \* FROM Customers  
WHERE Country='Germany' AND City='Berlin';

**BETWEEN:**

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

**Update:**

UPDATE Customer\_Phone\_No

SET Phone\_No = 9820735175

WHERE Customer\_Id = 4;

**Joins:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

**Inner Join:** The INNER JOIN keyword selects records that have matching values in both tables.

**Left Join:** The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side if there is no match.

**Right Join**: The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side when there is no match.

**Full Join:** The SQL FULL JOIN combines the results of both left and right outer joins. The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records. FULL OUTER JOIN and FULL JOIN are the same.

**Self Join:** A self JOIN is a regular join, but the table is joined with itself. The SQL SELF JOIN is used to join a table to itself as if the table were two tables; temporarily renaming at least one table in the SQL statement.

**Adv of Views:**

* A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.
* Views can give a user a "personalized" view of the database structure, presenting the database as a set of virtual tables that make sense for that user.
* A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed.

**Serializability:**

When multiple transactions are running concurrently then there is a possibility that the database may be left in an inconsistent state. Serializability is a concept that helps us to check which [schedules](https://beginnersbook.com/2018/12/dbms-schedules/) are serializable. A serializable schedule is the one that always leaves the database in consistent state.

Conflict Serializability: **Conflict Serializability** is one of the types of Serializability, which can be used to check whether a non-serial schedule is conflict serializable or not. A schedule is called conflict serializable if we can convert it into a serial schedule after swapping its non-conflicting operations.

View Serializability: View Serializability is a process to find out that a given [schedule](https://beginnersbook.com/2018/12/dbms-schedules/) is view serializable or not. To check whether a given schedule is view serializable, we need to check whether the given schedule is **View Equivalent** to its serial schedule. (Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner).

**Entity Relationship Diagram (ER Diagram):**

An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design of a database that can later be implemented as a database. The main components of E-R model are entity set and relationship set. An ER diagram shows the relationship among entity sets.

**Enhanced Entity Relationship Model (EER Model):**

EER is a high-level data model that incorporates the extensions to the original ER model.  
  
It is a diagrammatic technique for displaying the following concepts

* Sub Class and Super Class
* Specialization and Generalization
* Union or Category
* Aggregation

**Relational Model:**

Relational Model represents how data is stored in Relational Databases.  A relational database stores data in the form of relations (tables).