

# Batch: B-1 Roll No.: 16010422234 Name: Chandana Ramesh Galgali

# Experiment No.: 03

**Aim:** To implement a database for relational model in experiment no. 2 using DDL statements (Virtual Lab).

**Resources needed:** PostgreSQL PgAdmin4

# Theory:

The Data Definition Language (DDL) is used to create and modify the relational schema. Also it is used to add various constraints to the table like the primary key, foreign key, check constraint, not null constraint and unique constraint.

The DDL statements are: CREATE DROP ALTER

PostgreSQL supports the standard SQL types int, smallint, real, double precision, char(N), varchar(N), date, time, timestamp, and interval for creating tables.

# Procedure:

**Create Database and use it:**

$ createdb mydb

$ psql mydb

**Delete a database:** $ dropdb mydb

# Create table:

CREATE TABLE my\_first\_table ( first\_column text,

second\_column integer

);

CREATE TABLE products ( product\_no integer,

name text, price numeric);

# Drop Table:

DROP TABLE my\_first\_table; DROP TABLE products;

# Default Value:

CREATE TABLE products ( product\_no integer,

name text,

price numeric **DEFAULT 9.99** );

# Constraints:

1. **Primary Key**

CREATE TABLE products ( product\_no integer **PRIMARY KEY**, name text,

price numeric );

Primary keys can also constrain more than one column. CREATE TABLE example (

a integer, b integer, c integer,

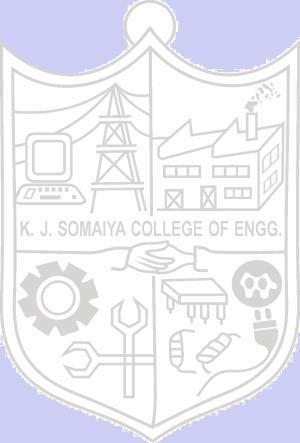
# PRIMARY KEY (a, c)

);

# Check Constraint

CREATE TABLE products ( product\_no integer,

name text,

price numeric **CHECK (price** > **0)** );

# Not Null Constraint

CREATE TABLE products ( product\_no integer **NOT NULL**, name text **NOT NULL**,

price numeric );

1. **Unique Constraint**

CREATE TABLE products ( product\_no integer **UNIQUE**,

name text,

price numeric );

# Foreign Key Constraint

CREATE TABLE products ( product\_no integer PRIMARY KEY, name text,

price numeric );

CREATE TABLE orders ( order\_id integer PRIMARY KEY,

product\_no integer **REFERENCES products (product\_no)**, quantity integer );

Here a foreign key constraint in the order table references the products table.

# Modifying table:

**Adding column**

ALTER TABLE products ADD COLUMN description text;

# Removing column

ALTER TABLE products DROP COLUMN description;

# Adding Constraint

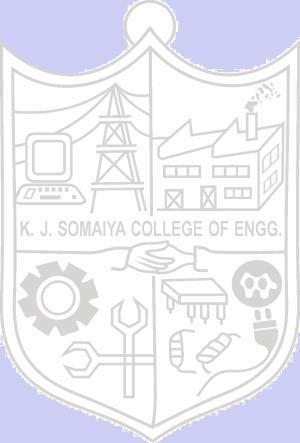
ALTER TABLE products ADD CONSTRAINT some\_name UNIQUE (product\_no); ALTER TABLE products ADD FOREIGN KEY (product\_group\_id) REFERENCES product\_groups;

# Removing Constraint

ALTER TABLE products DROP CONSTRAINT some\_name;

# Adding Not Null Constraint

ALTER TABLE products ALTER COLUMN product\_no SET NOT NULL;



# Removing Not Null Constraint

ALTER TABLE products ALTER COLUMN product\_no DROP NOT NULL;



# Results: (Queries printout with output)

*Query:*

Aviation\_Services(Unique\_Service\_ID,Vegetarian\_Meal,Non\_vegetarian\_Meal,Lounge\_Services,Buggy\_Services,Wheelchair,Medical\_Help)

Airport(Unique\_Service\_ID,Airport\_Name,Airport\_Location,Number\_Of\_Runways)

Airline(Unique\_Service\_ID,Airline\_Name,Country\_Of\_Origin,Fleet\_Size)

Aircraft(Aircraft\_ID,Engine\_Design,Manufacturer)

Operated\_By(Aircraft\_ID,Airline\_Name)

Aircraft\_Model(Aircraft\_ID,Model\_ID,Aircraft\_Model)

Flights(Unique\_Service\_ID,Flight\_ID,Arrival\_Time,Departure\_Time,Arrival\_Location,Destination,Aircraft\_Used)

Passenger(Passenger\_ID,Password,Name,DOB,Email,Address,Account\_Number,Account\_Type,Bank\_Name

Used\_By(Passenger\_ID,Flight\_ID)

Mobile\_Number(Passenger\_ID,Mobile\_Number)

Tickets(Passenger\_ID,Ticket\_ID,Flight\_Number,Seat\_Number,Seat\_Type,Price)

create table Aviation\_Services(

Unique\_Service\_ID int primary key,

Vegetarian\_Meal char(5),

Non\_vegetarian\_Meal char(5),

Lounge\_Services varchar(20),

Buggy\_Services char(5),

Wheelchair char(5),

Medical\_Help varchar(20)

);

select \* from Aviation\_Services

create table Airport(

Unique\_Service\_ID int,

Airport\_Name varchar(20),

Airport\_Location varchar(20) NOT NULL,

Number\_Of\_Runways int check(Number\_Of\_Runways>=1),

primary key(Unique\_Service\_ID,Airport\_Name),

foreign key (Unique\_Service\_ID) references Aviation\_Services(Unique\_Service\_ID)

);

select \* from Airport

create table Airline(

Unique\_Service\_ID int,

Airline\_Name varchar(20),

Country\_Of\_Origin varchar(20),

Fleet\_Size int check(Fleet\_Size>=3),

primary key(Unique\_Service\_ID,Airline\_Name),

foreign key (Unique\_Service\_ID) references Aviation\_Services(Unique\_Service\_ID)

);

select \* from Airline

create table Aircraft(

Aircraft\_ID int primary key,

Engine\_Design varchar(20),

Manufacturer varchar(20)NOT NULL

);

select \* from Aircraft

create table Operated\_By(

Aircraft\_ID int,

Unique\_Service\_ID int,

Airline\_Name varchar(20),

primary key(Aircraft\_ID,Unique\_Service\_ID,Airline\_Name),

foreign key (Aircraft\_ID) references Aircraft(Aircraft\_ID),

foreign key (Unique\_Service\_ID,Airline\_Name) references Airline(Unique\_Service\_ID,Airline\_Name)

);

select \* from Operated\_By

create table Aircraft\_Model(

Aircraft\_ID int,

Model\_ID int,

Aircraft\_Model varchar(20) NOT NULL,

primary key(Aircraft\_ID,Model\_ID),

foreign key (Aircraft\_ID) references Aircraft(Aircraft\_ID)

);

select \* from Aircraft\_Model

create table Flights(

Unique\_Service\_ID int,

Flight\_ID int,

Arrival\_Time time NOT NULL,

Departure\_Time time NOT NULL,

Arrival\_Location varchar(20) NOT NULL,

Destination varchar(20) NOT NULL,

Aircraft\_Used varchar(20)NOT NULL,

primary key(Unique\_Service\_ID,Flight\_ID),

foreign key (Unique\_Service\_ID) references Aviation\_Services(Unique\_Service\_ID)

);

select \* from Flights

create table Passenger(

Passenger\_ID int primary key,

Password varchar(20),

Name varchar(20)NOT NULL,

DOB date,

Email varchar(20) UNIQUE,

Address varchar(20),

Account\_Number int UNIQUE,

Account\_Type varchar(20)NOT NULL,

Bank\_Name varchar(20)NOT NULL,

Flight\_History varchar(20)

);

select \* from Passenger

create table Used\_By(

Passenger\_ID int,

Unique\_Service\_ID int,

Flight\_ID int,

primary key(Passenger\_ID,Unique\_Service\_ID,Flight\_ID),

foreign key (Passenger\_ID) references Passenger(Passenger\_ID),

foreign key (Unique\_Service\_ID,Flight\_ID) references Flights(Unique\_Service\_ID,Flight\_ID)

);

select \* from Used\_By

create table Mobile\_Number(

Passenger\_ID int,

Mobile\_Number int UNIQUE,

primary key(Passenger\_ID,Mobile\_Number),

foreign key (Passenger\_ID) references Passenger(Passenger\_ID)

);

select \* from Mobile\_Number

create table Tickets(

Passenger\_ID int,

Ticket\_ID int,

primary key(Passenger\_ID,Ticket\_ID),

Flight\_Number int NOT NULL,

Seat\_Number int NOT NULL,

Seat\_Type varchar(20)NOT NULL,

Price int NOT NULL,

foreign key (Passenger\_ID) references Passenger(Passenger\_ID)

);

select \* from Tickets

create table Flight\_Crew(

Member\_ID int,

Unique\_Service\_ID int,

Flight\_ID int,

primary key(Member\_ID,Unique\_Service\_ID,Flight\_ID),

Name varchar(20) UNIQUE,

Shift int check(Shift=1 or Shift=2),

foreign key (Unique\_Service\_ID,Flight\_ID) references Flights(Unique\_Service\_ID,Flight\_ID)

);

select \* from Flight\_Crew

ALTER TABLE Flight\_Crew ADD COLUMN Role varchar(20);

ALTER TABLE Flight\_Crew ALTER COLUMN Role SET NOT NULL;

select \* from Flight\_Crew

ALTER TABLE Flight\_Crew DROP CONSTRAINT Flight\_Crew\_Shift\_check;

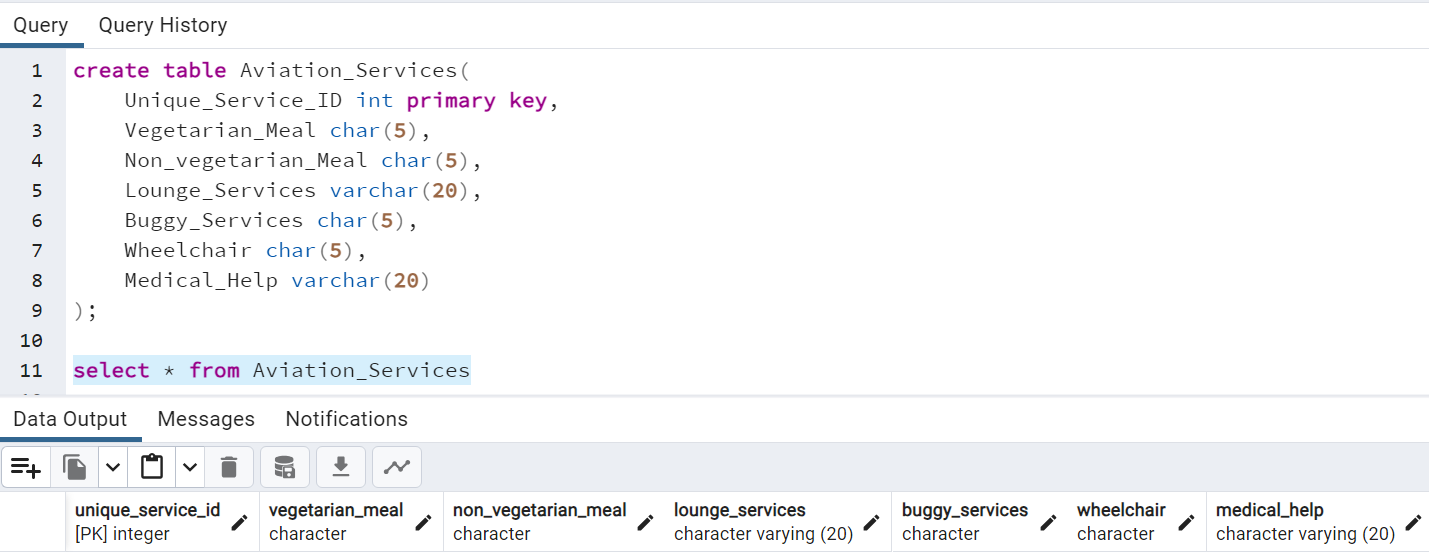
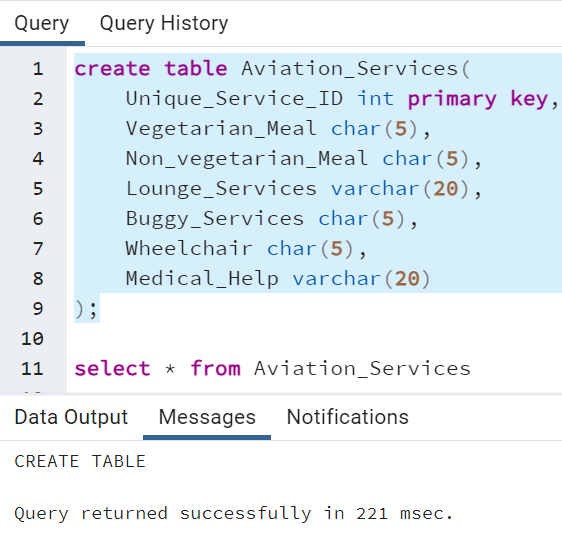
ALTER TABLE Flight\_Crew DROP COLUMN Shift;

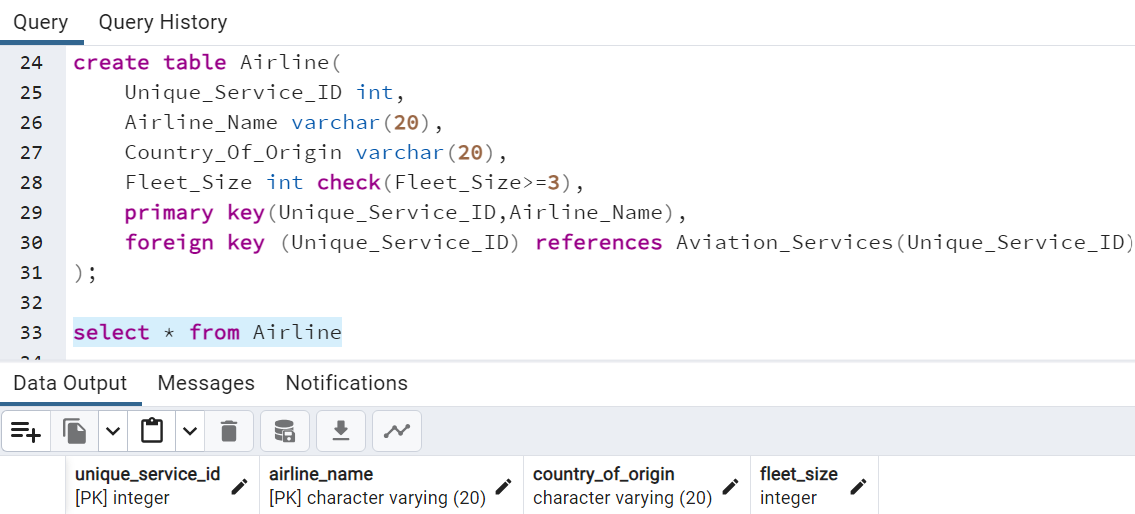
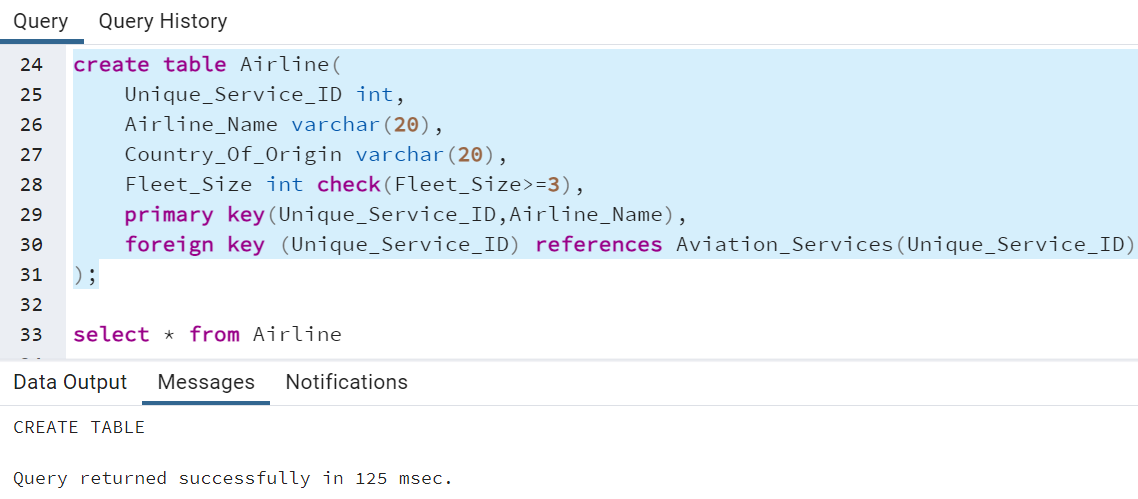
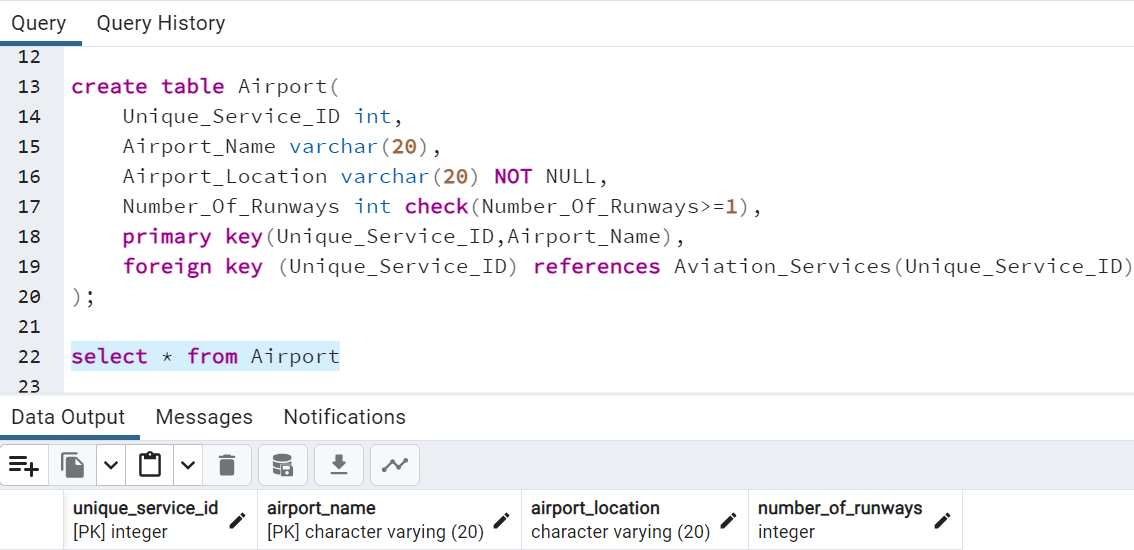
select \* from Flight\_Crew

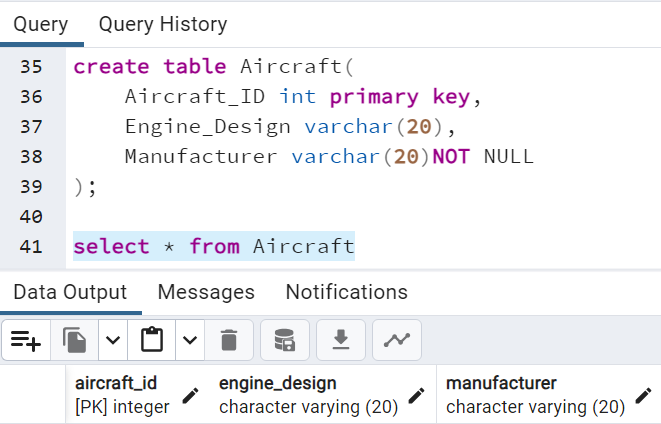
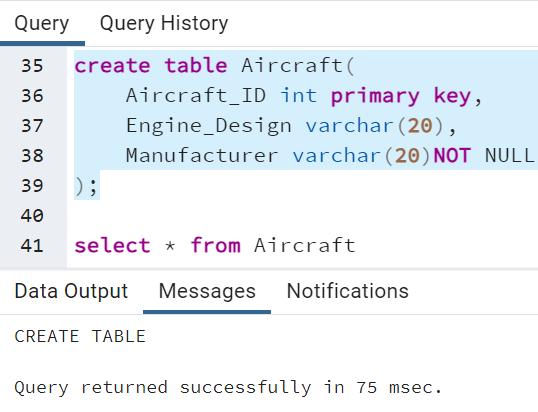
DROP TABLE Flight\_Crew;

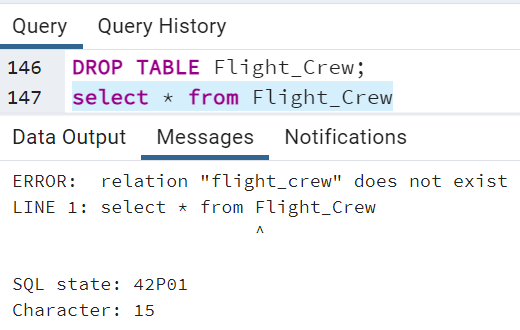
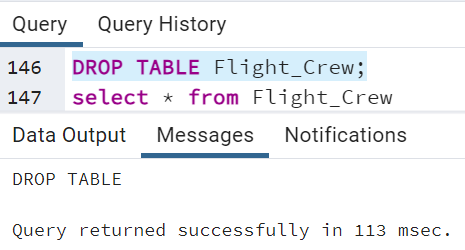
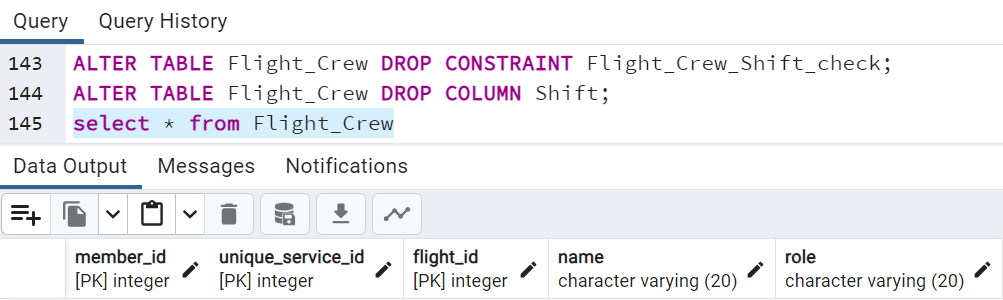
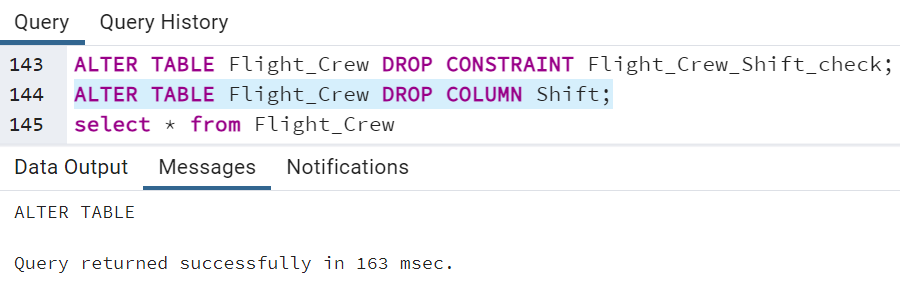
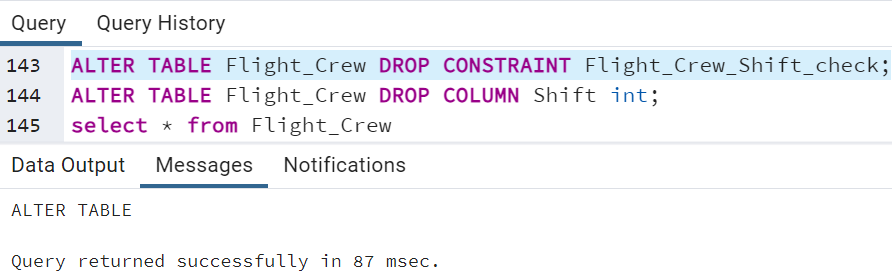
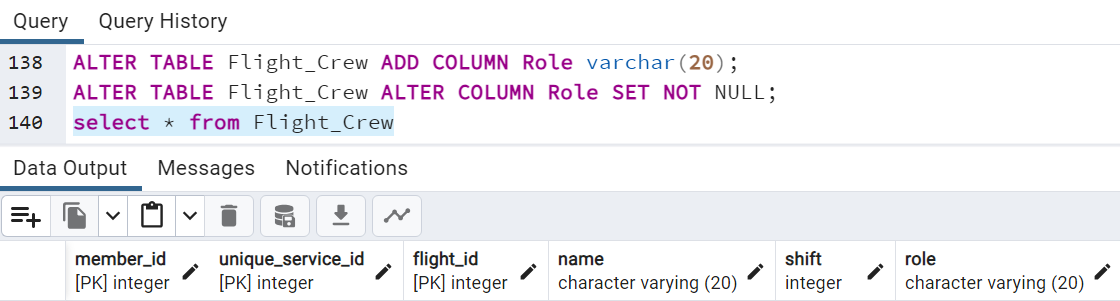
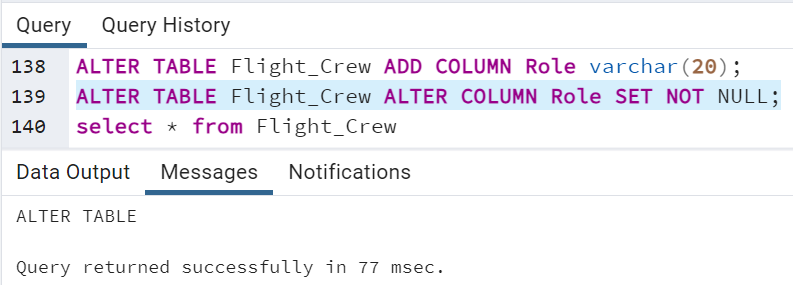
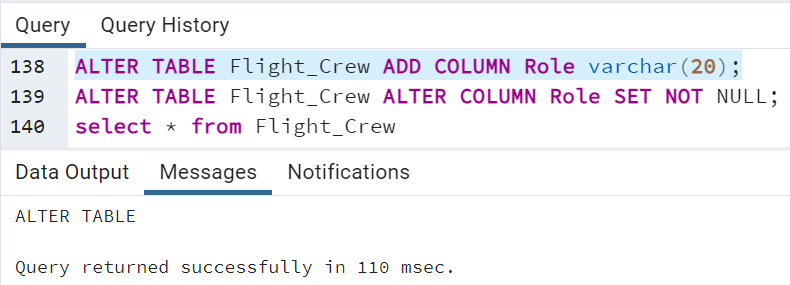
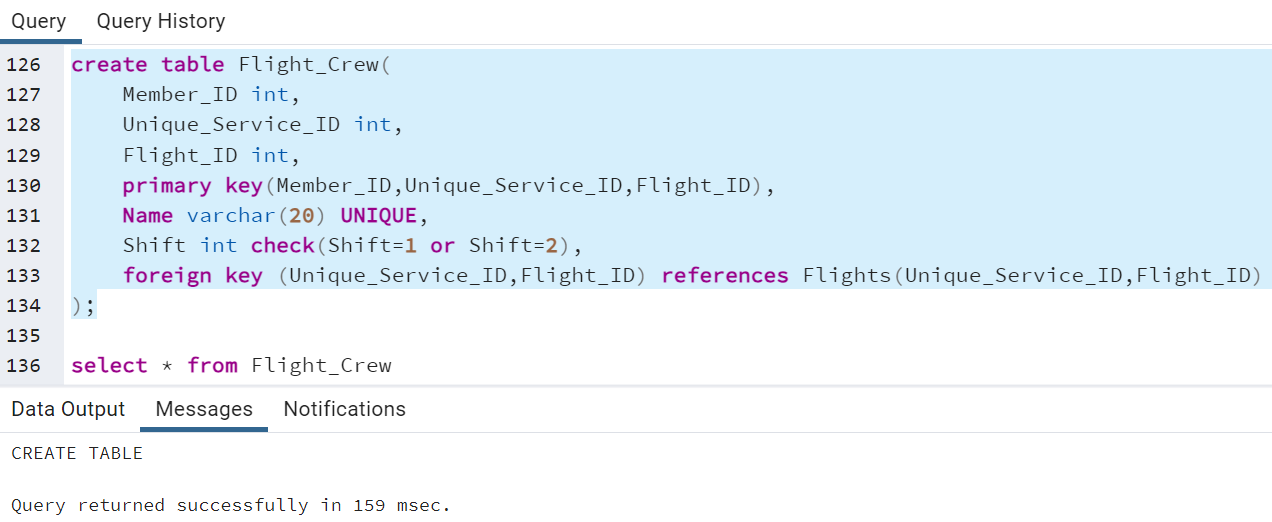
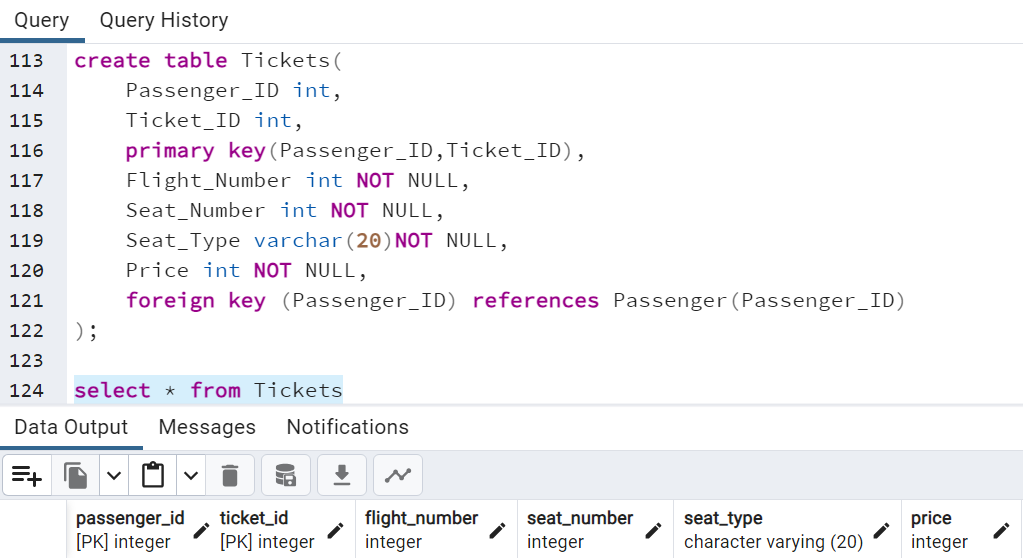
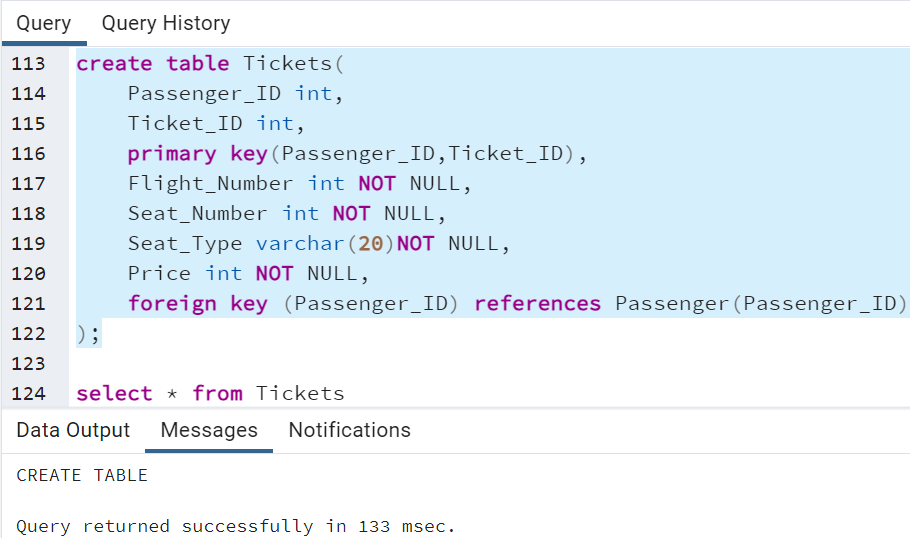
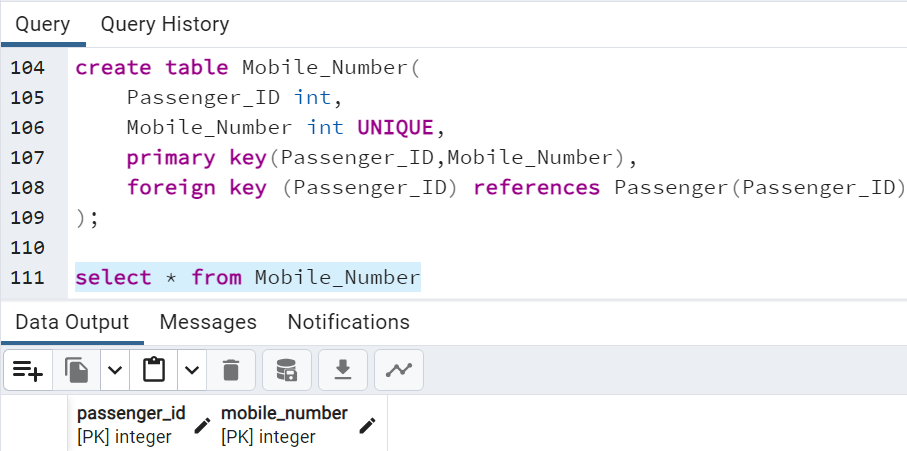
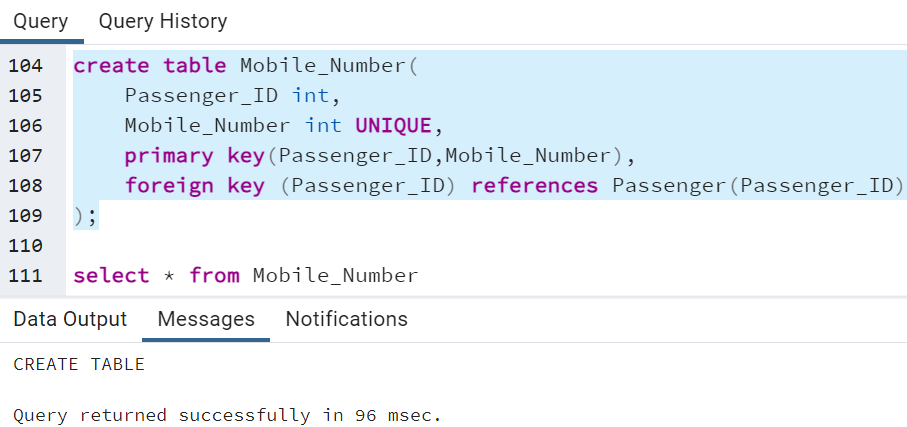
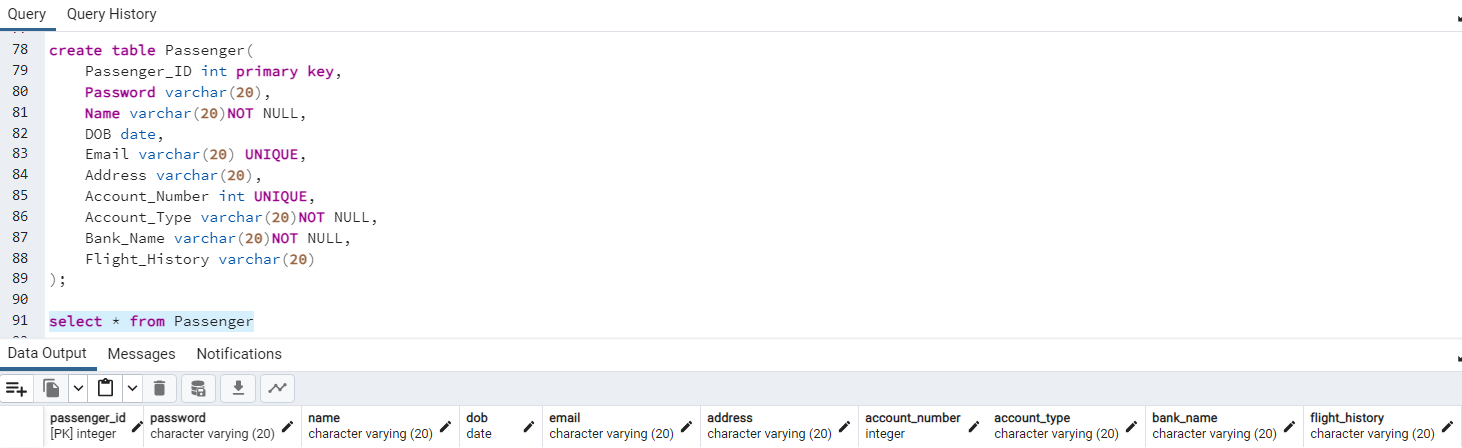
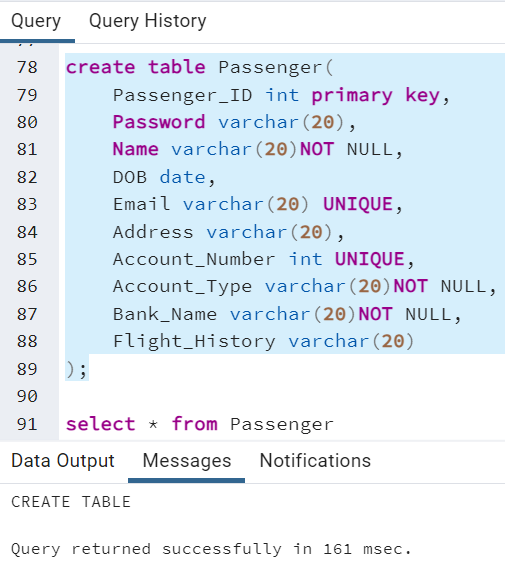
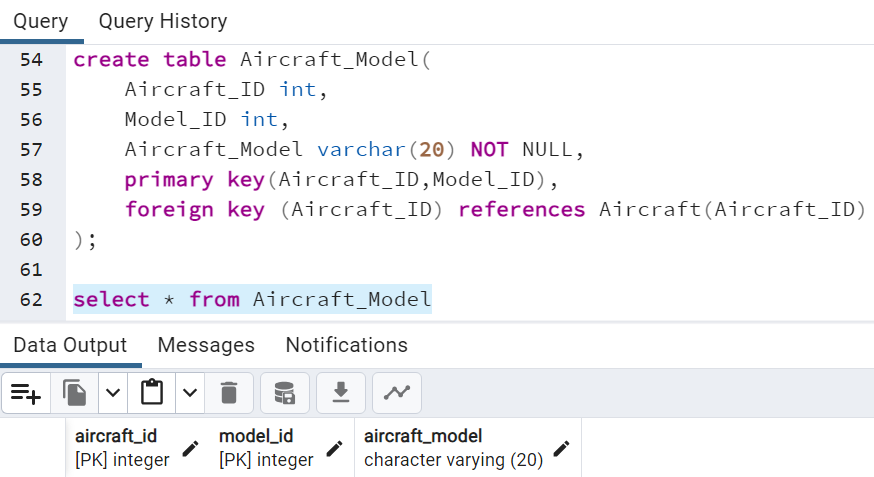
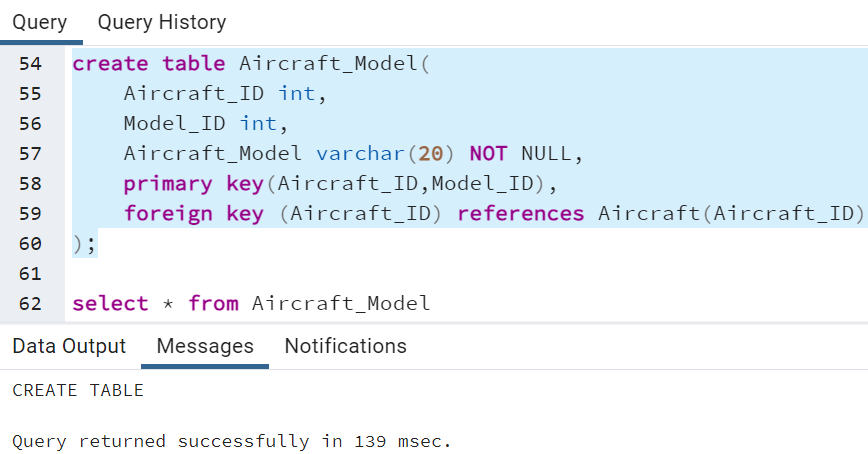
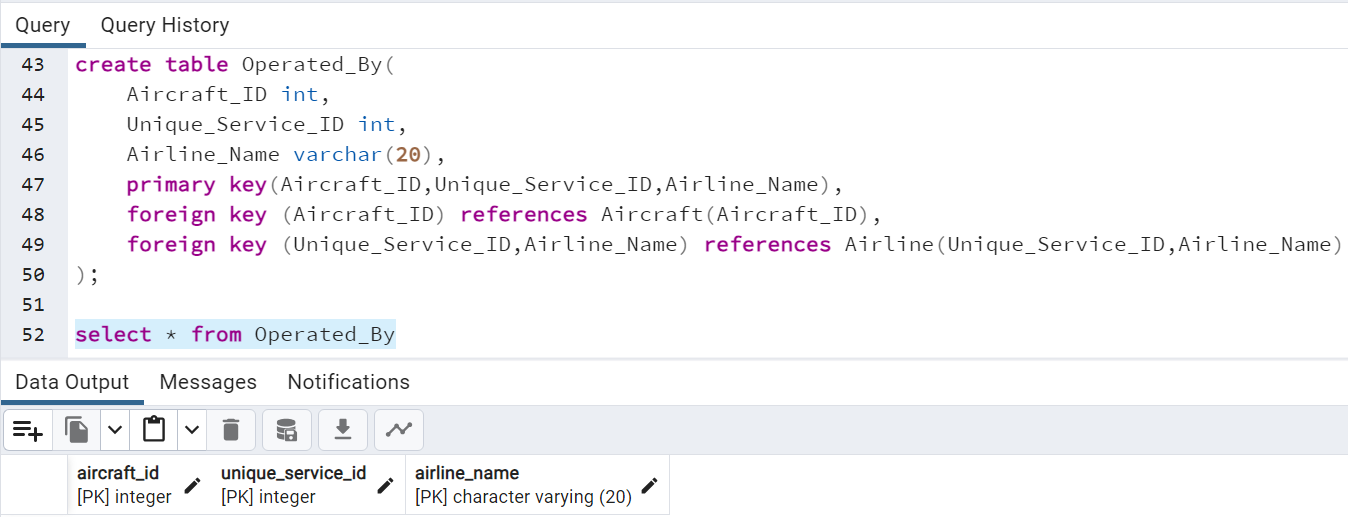
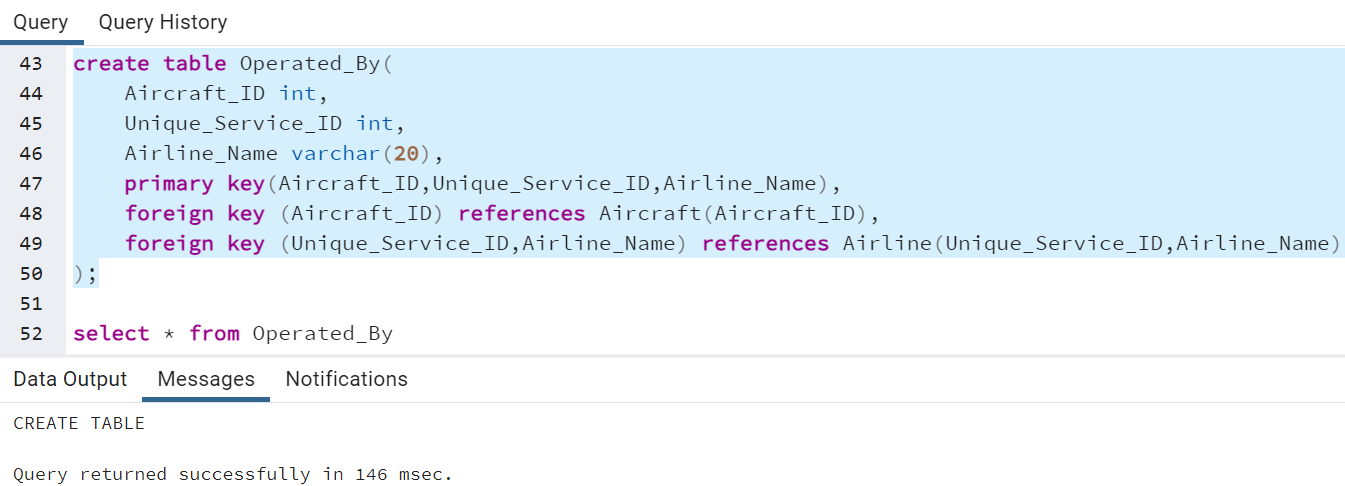
select \* from Flight\_Crew

*Output:*









**Outcomes: Apply data models to real world scenarios.**



**Questions:**

**Q1 What is the difference between Truncate, Drop and Delete? Explain with examples.**

**Ans:** Truncate: The "truncate" operation is used to remove all the data from a table, effectively resetting it to its initial state. It is a fast operation because it deallocates the data pages used by the table, rather than deleting individual rows. Truncate also resets any auto-incrementing identity columns or sequence values. However, it cannot be rolled back, and it requires the user to have the necessary permissions.

Example: Let's say we have a table called "Customers" with multiple rows of data. If we execute the truncate operation on this table, all the rows will be removed, and the table will be empty.

Drop: The "drop" operation is used to remove an entire table from the database schema. It not only removes the data but also permanently deletes the table structure, including any associated indexes, constraints, and triggers. This operation is non-recoverable, and it also requires the necessary permissions.

Example: Suppose we have a table called "Products" that contains information about various products. If we execute the drop operation on this table, the entire table will be deleted, and it will no longer exist in the database.

Delete: The "delete" operation is used to remove specific rows from a table based on specified conditions. It allows for selective removal of data, unlike truncate, which removes all rows. Delete is a slower operation compared to truncate because it logs each row deletion and can be rolled back if necessary. It also triggers any associated delete triggers.

Example: Consider a table called "Employees" with multiple rows of data. If we execute the delete operation with a condition like "WHERE Salary < 5000," it will remove only those rows where the salary is less than 5000, leaving the rest of the data intact.

In summary, truncate removes all data from a table, drop deletes the entire table structure, and delete selectively removes rows based on specified conditions.



**Conclusion:**

We could successfully implement a database for the relational model using Data Definition Language (DDL) statements. The experiment involved creating tables, defining relationships between tables, specifying constraints, and setting up the necessary database structure.

**Grade: AA / AB / BB / BC / CC / CD /DD** 

**Signature of faculty in-charge with data**

**Reference books:**

1. Elmasri and Navathe, “Fundamentals of Database Systems”, 6th Edition, Pearson Education
2. Korth, Slberchatz,Sudarshan, :”Database System Concepts”, 6th Edition, McGraw – Hill.

# WebSite:

1. <http://www.tutorialspoint.com/postgresql/>
2. [http://sage.virtual-labs.ac.in/home/pub/21](http://sage.virtual-labs.ac.in/home/pub/21/)