# DBMS PROJECT

**AIRPORT MANAGEMENT SYSTEM**

**UCS310 -DBMS Project Report**

# End Semester Lab Evaluation

Submitted By:

Abhaijeet Singh - 102217186

Arshdeep Kaur - 102217192

Shreya Agrawal - 102217170

# Submitted To:

**Dr. Chinmaya Panigrahy**



## Computer Science and Engineering Department

# ACKNOWLEDGEMENT

We have taken efforts in this project.

However, it would not have been possible

without the kind support and help of many

individuals and teachers. We would like to

extend our sincere thanks to all of them.

We are highly indebted to Dr. CHINMAYA PANIGRAHY and for his guidance and

constant supervision as well as for providing

necessary information regarding the project

& also for their support in completing the

project.

We would like to express our gratitude towards

our parents for their kind co-operation and

encouragement which helped us in completion of

this project.

Our thanks and appreciations also go to our

colleague in developing the project and people

who have willing helped us out with their abilities.

# PROBLEM STATEMENT

# The Airport Management System (AMS) aims to develop a comprehensive database management solution to streamline and automate various operational aspects of an airport. The AMS will facilitate efficient management of airport resources, passenger services, and employee activities. The system will provide a centralized platform for airport authorities to monitor, control, and optimize airport operations while enhancing the overall passenger experience.

# IDENTIFICATION OF ENTITIES AND RELATIONS

# Entities: Relations:

# City 1) City has Airport

# Airport 2) Airport contains Airline

# Airline 3) Airline has Flight

# Flight 4) Flight carries Ticket

# Passenger 5) Passenger books Ticket

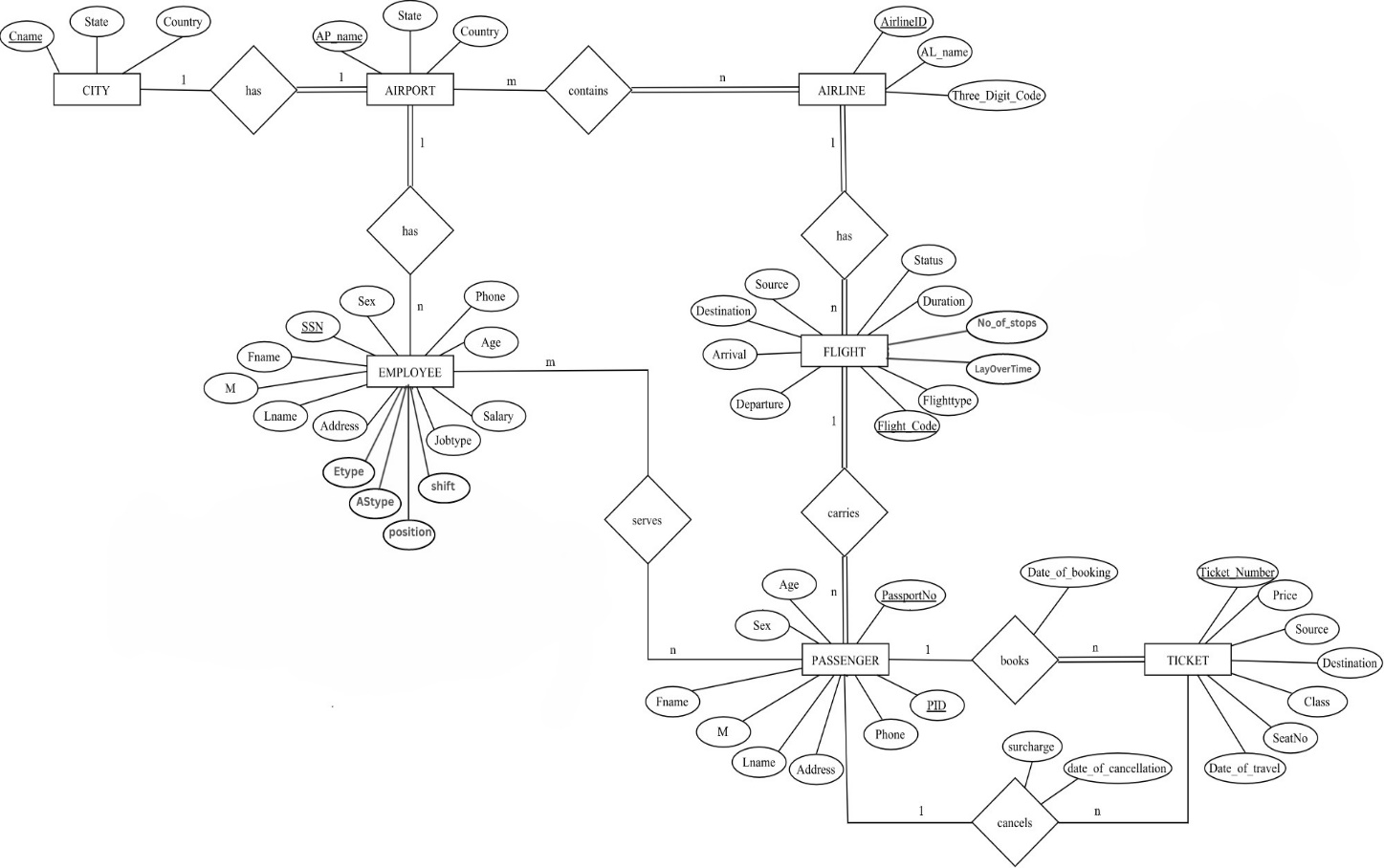
# Ticket 6) Passenger cancels ticket

# Employee 7) Employee serves passenger

# 8) Airport has employee

# ENTITY-RELATIONSHIP

# DIAGRAM



# ER TO TABLE

**NORMALIZATION**

The Process of Reducing data redundancy in a relational database is called **Normalization**. It’s done to achieve the following goals:

1. To avoid insertion, deletiona and updation Anomalies.
2. To minimize the amount of duplicate data in database.
3. To make database more informative to users including different users making different queries.

NORMALIZATION RULES ON DATABASE:

* PASSPORT 🡪 FNAME, M, LNAME, ADDRESS, PHONE, AGE, SEX

Violates 2NF

* PID 🡪 FLIGHT\_CODE

Violates 2NF

* DATE\_OF\_BOOKING, SOURCE, DESTINATION, CLASS 🡪 PRICE

Violates 3NF

* DATE\_OF\_CANCELLATION 🡪 SURCHARGE

Violates 3NF

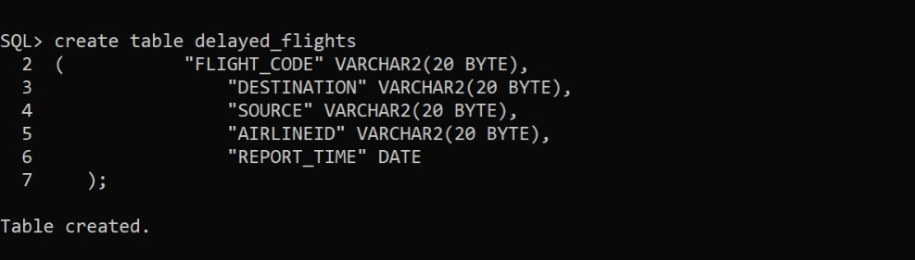
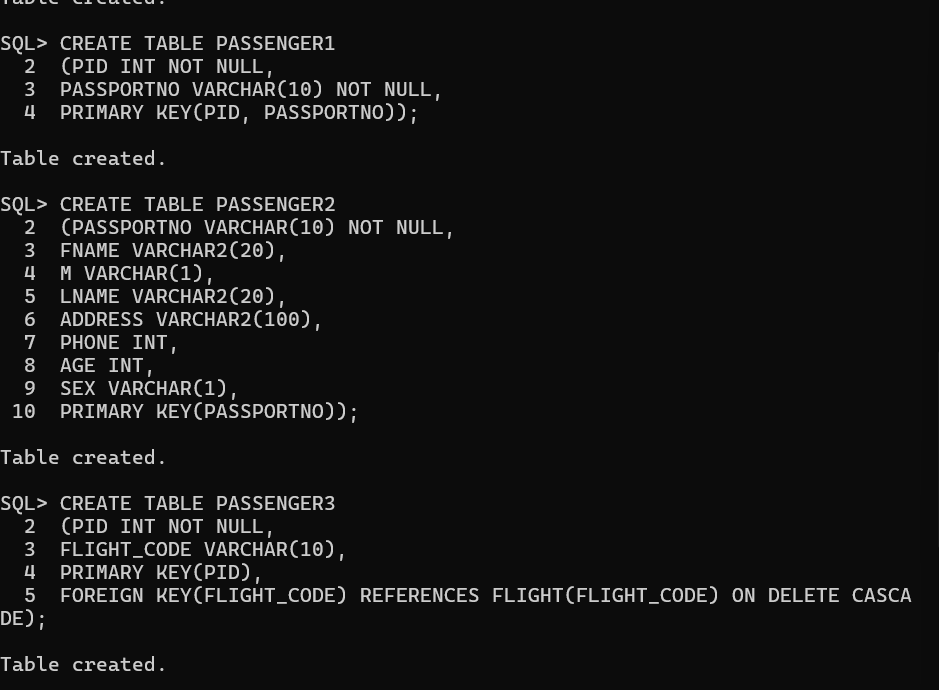
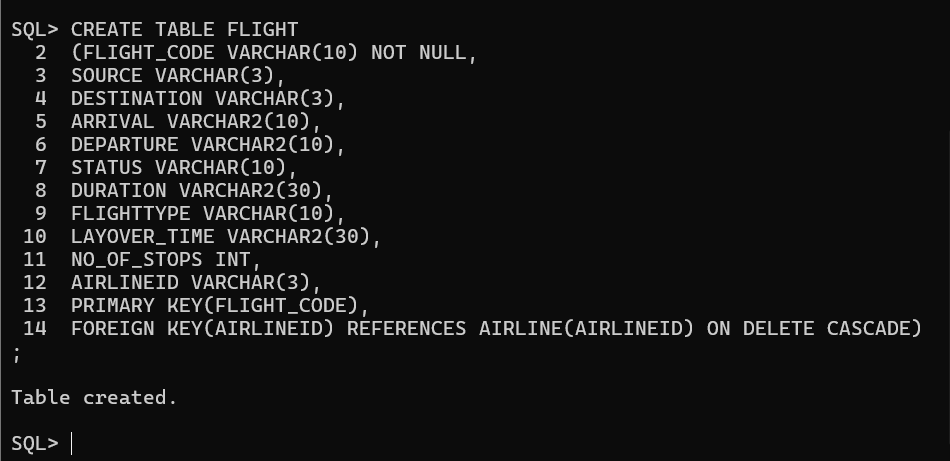
* JOBTYPE 🡪 SALARY

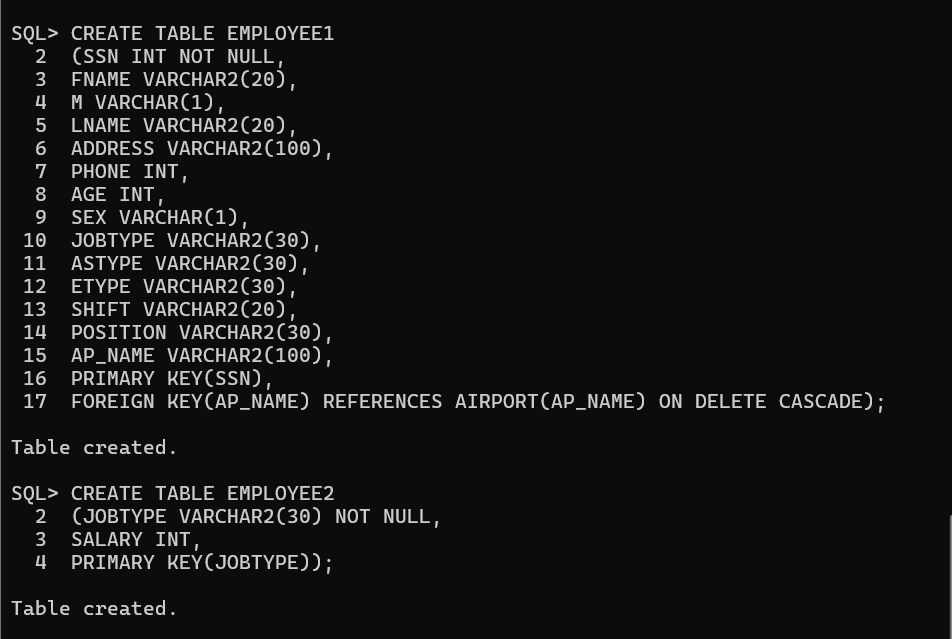
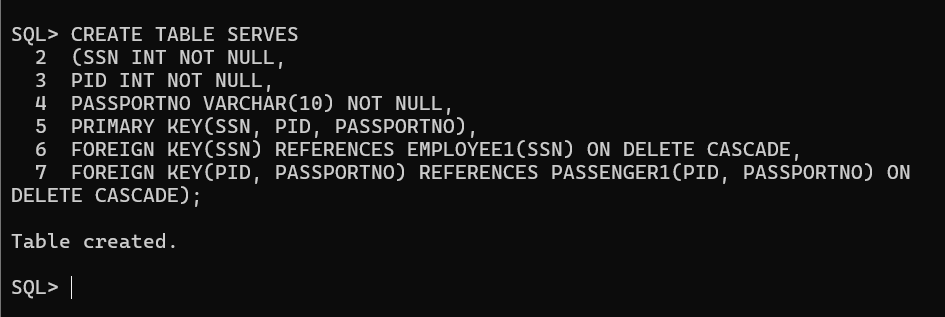
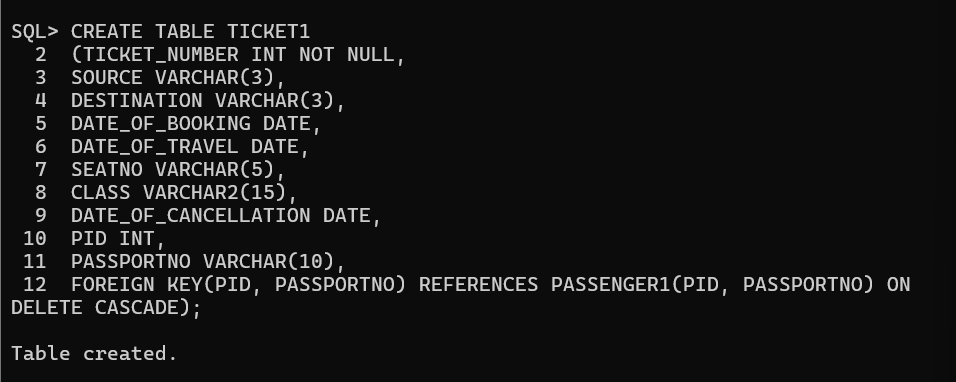
Violates 3NF

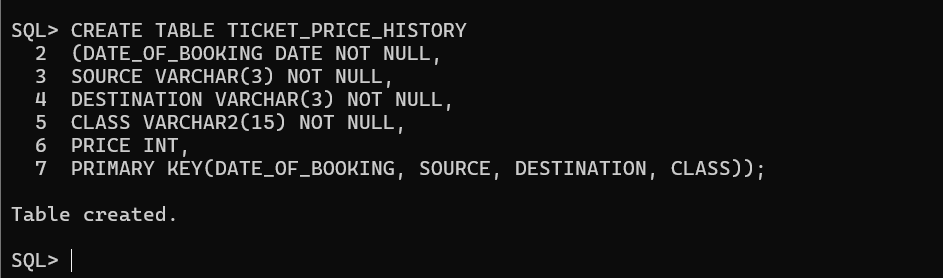
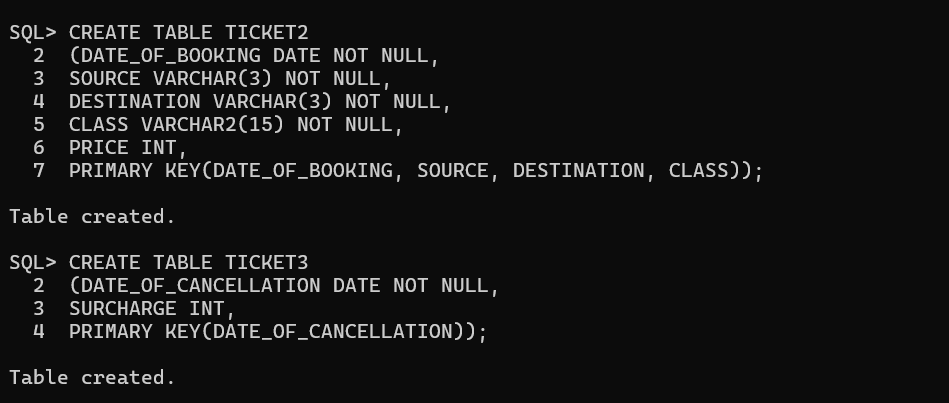
# NORMALIZED RELATIONAL SCHEMA

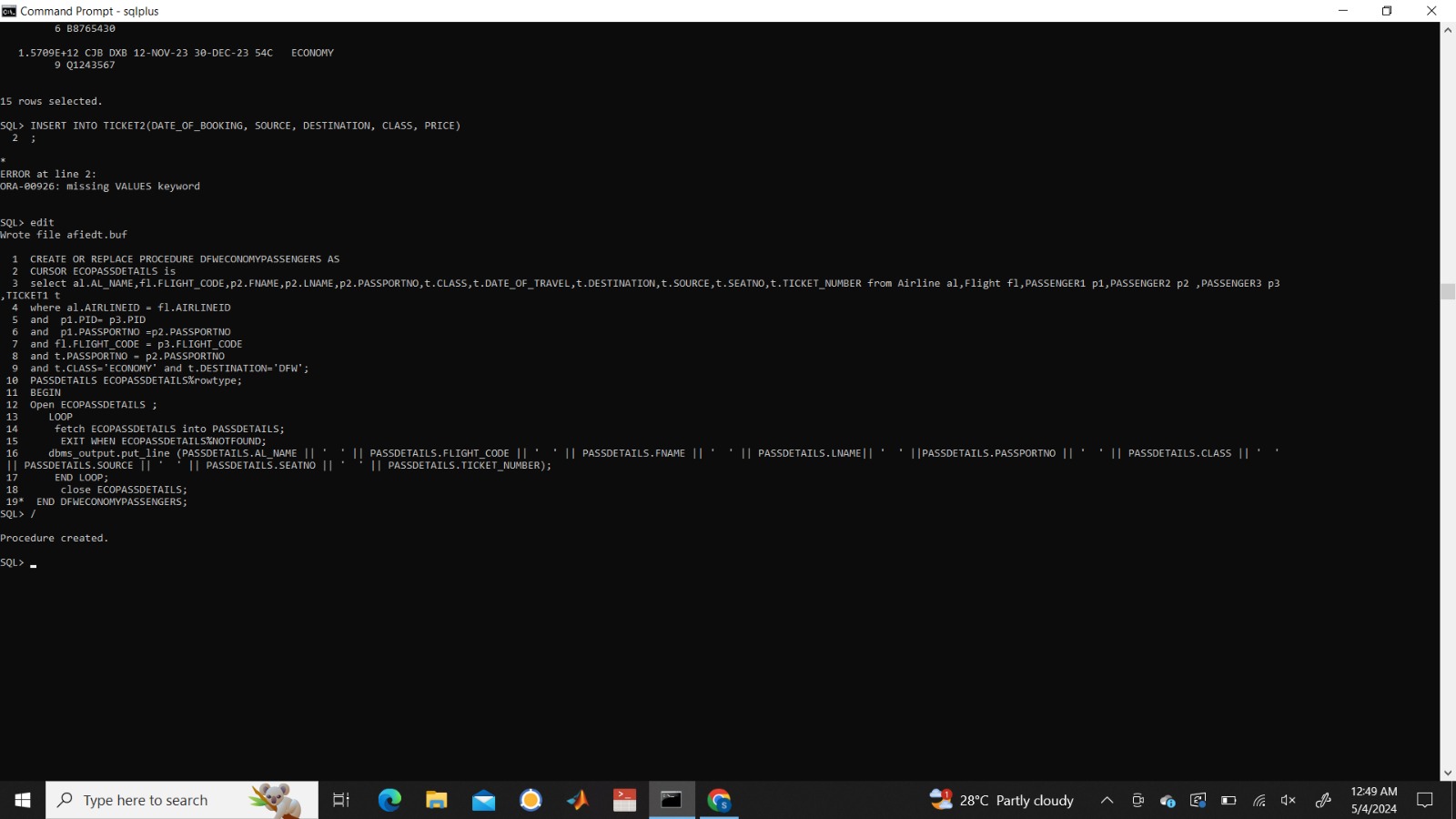
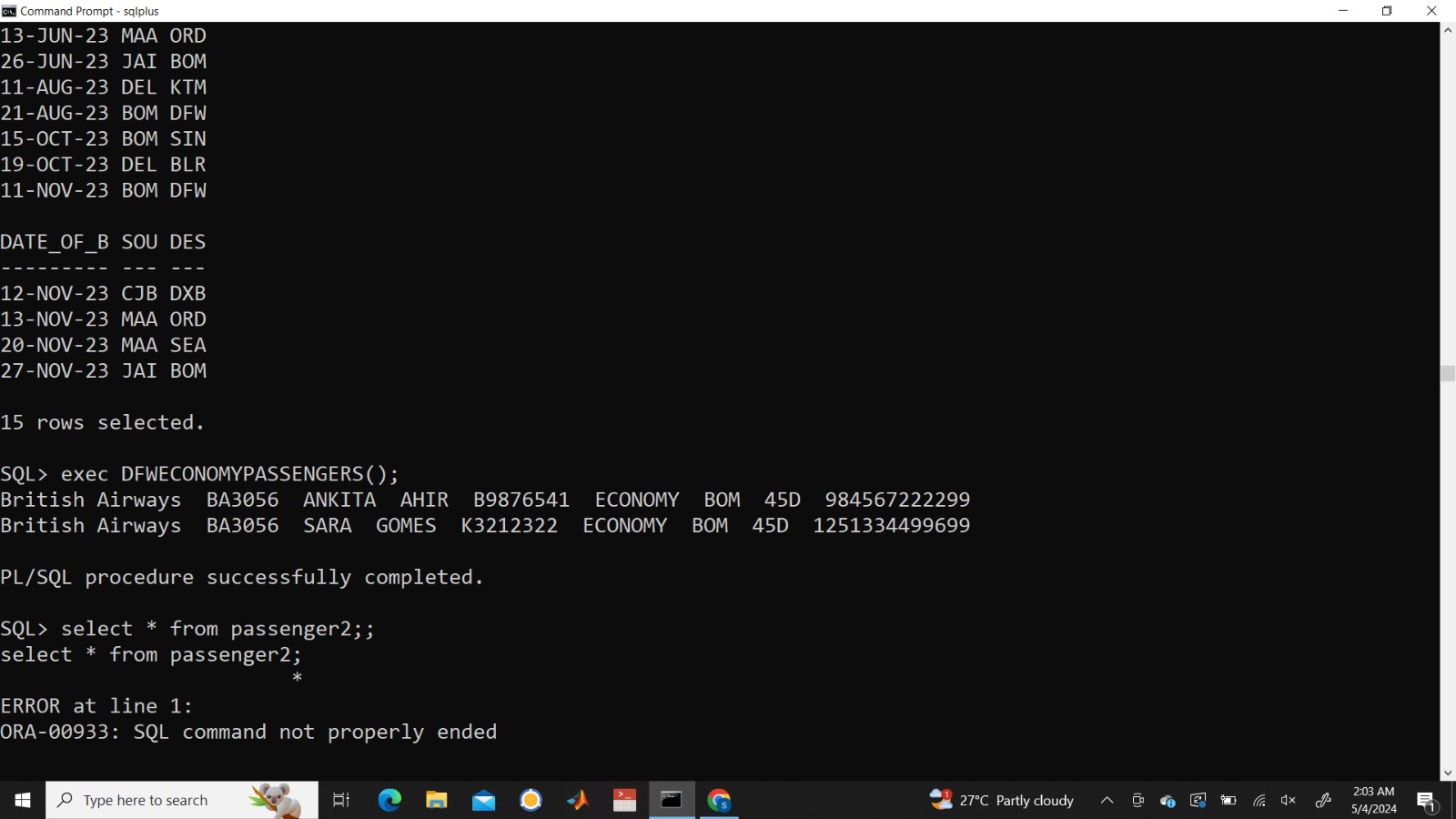
# 

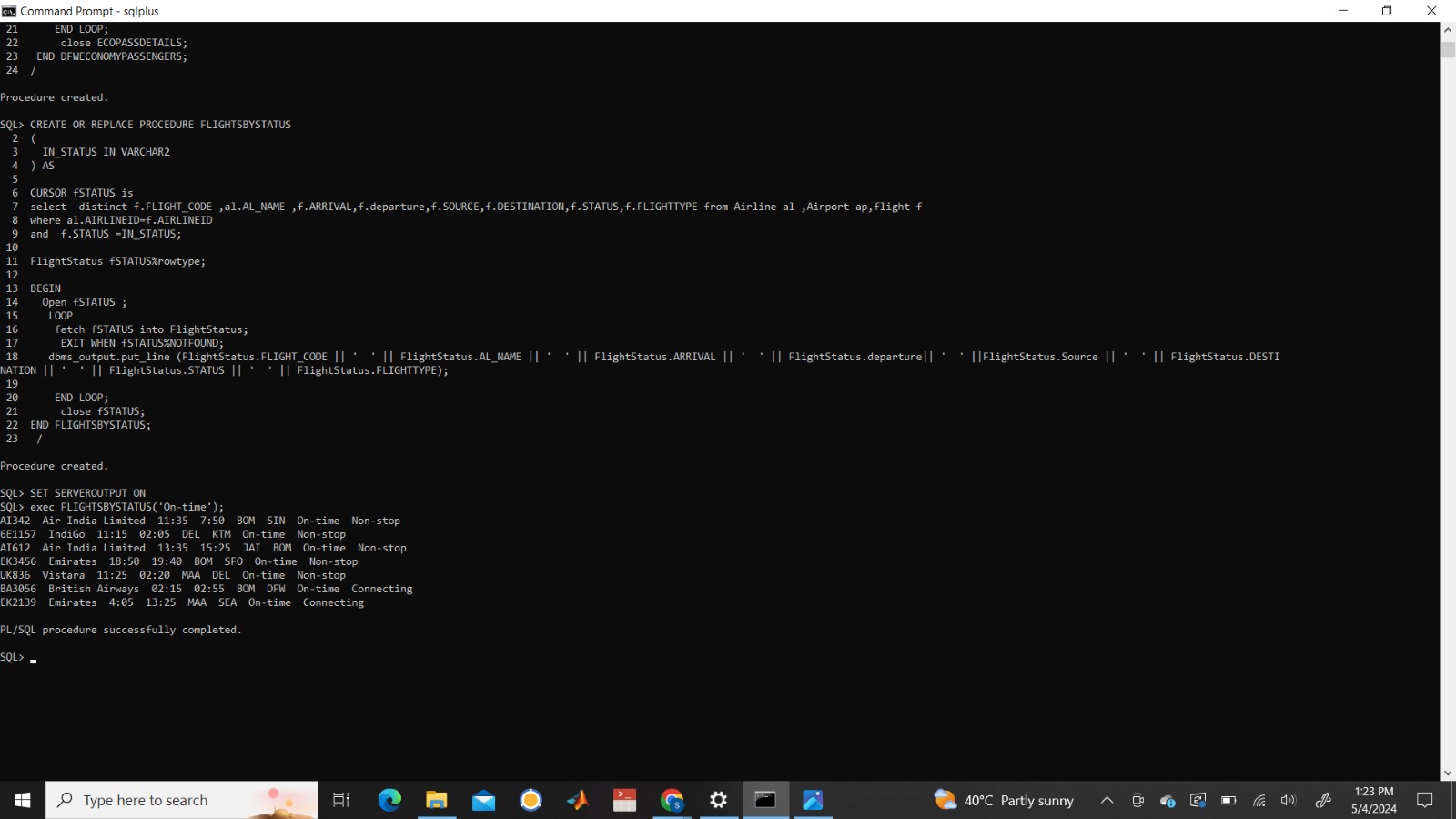
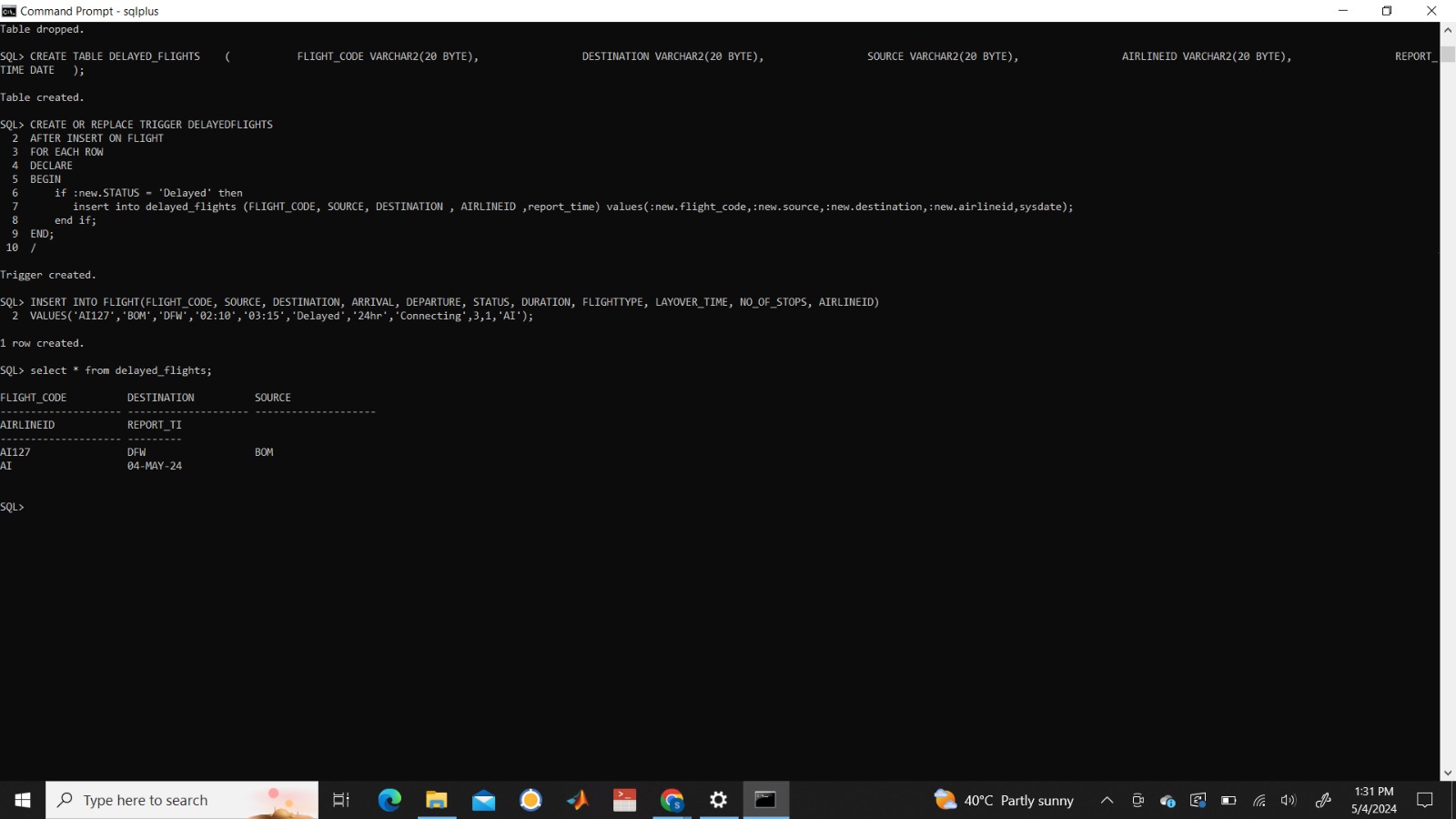
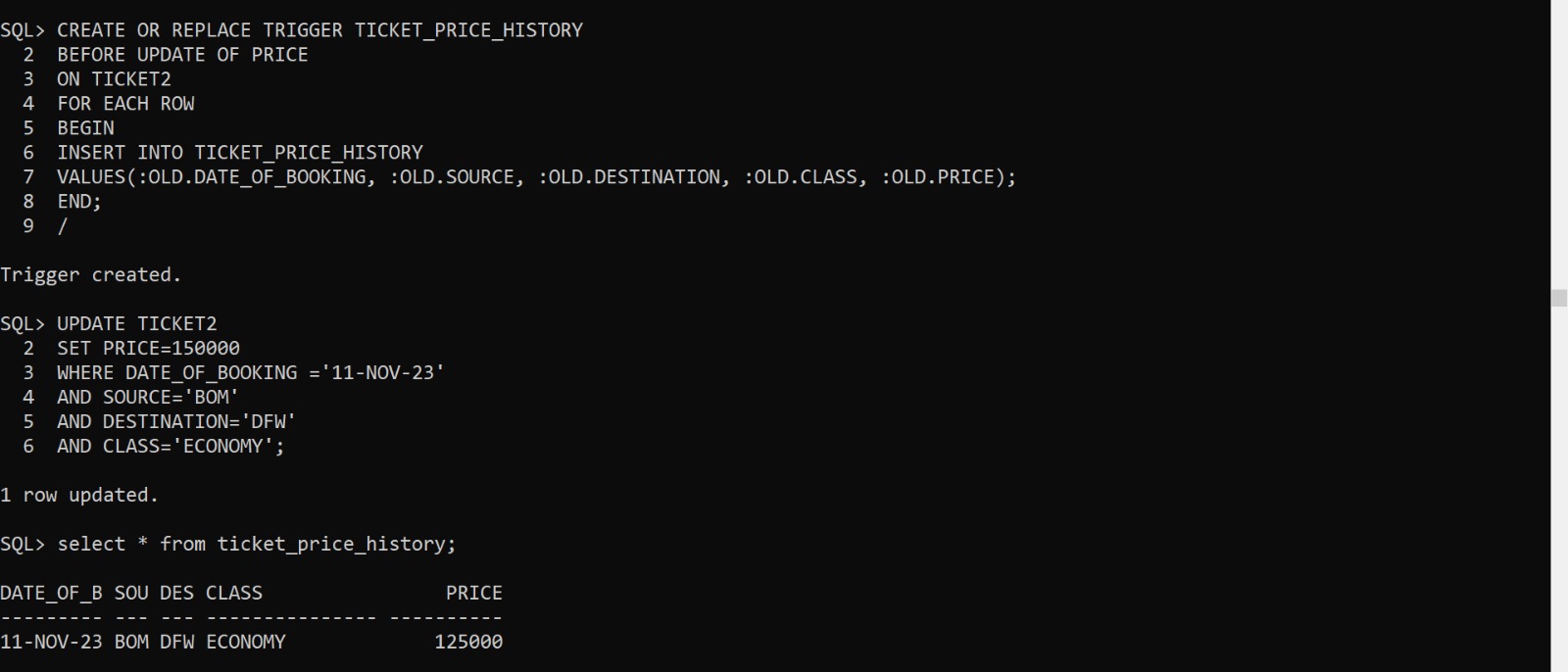
# PLSQL

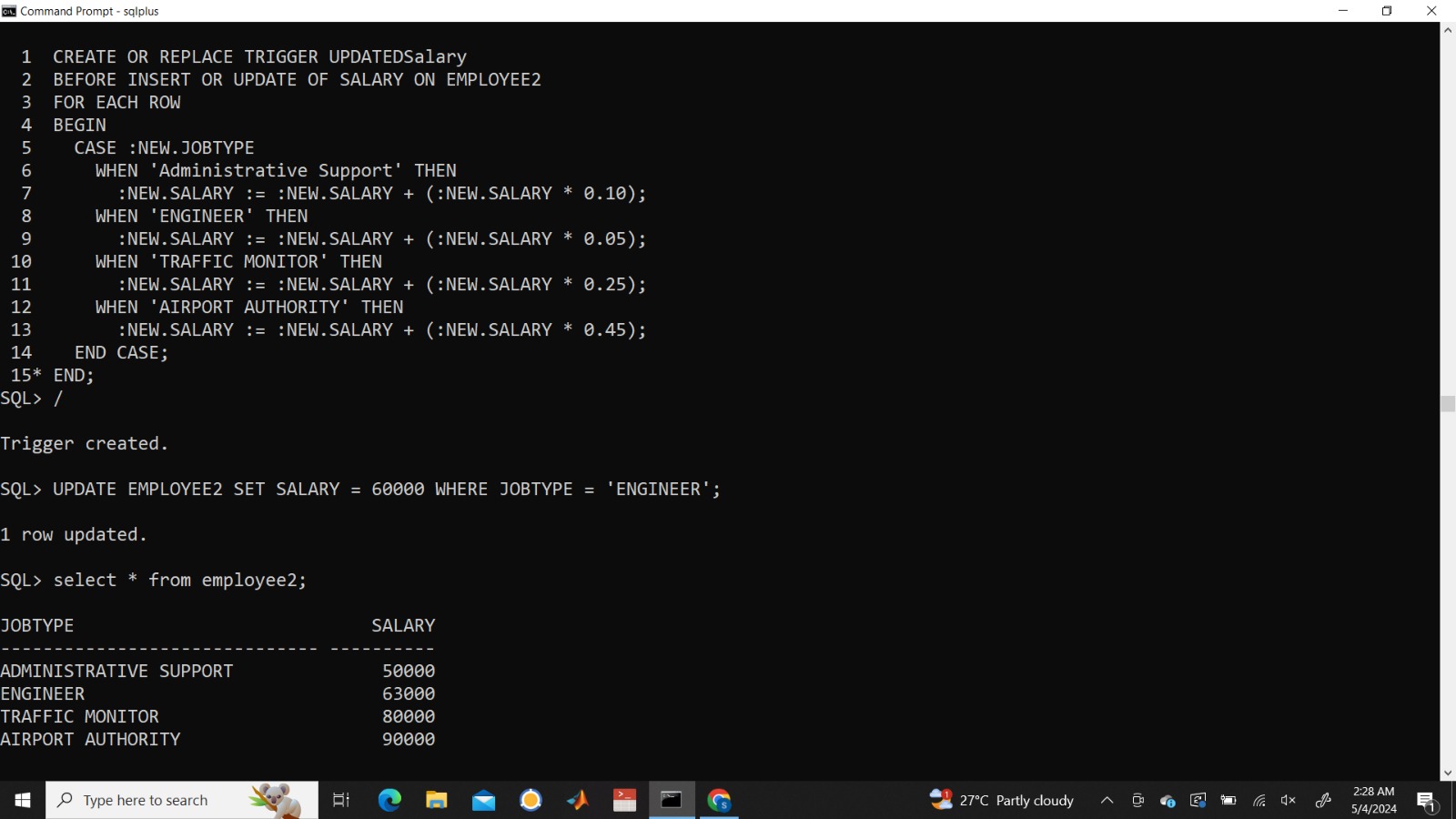




******Procedures/Triggers:**







**TABLES**

CREATE TABLE CITY

(CNAME VARCHAR2(15) NOT NULL,

STATE VARCHAR2(15),

COUNTRY VARCHAR(30),

PRIMARY KEY(CNAME));

CREATE TABLE AIRPORT

(AP\_NAME VARCHAR2(100) NOT NULL,

STATE VARCHAR2(15),

COUNTRY VARCHAR(30),

CNAME VARCHAR2(15),

PRIMARY KEY(AP\_NAME),

FOREIGN KEY(CNAME) REFERENCES CITY(CNAME) ON DELETE CASCADE);

CREATE TABLE AIRLINE

(AIRLINEID VARCHAR(3) NOT NULL,

AL\_NAME VARCHAR2(50),

THREE\_DIGIT\_CODE VARCHAR(3),

PRIMARY KEY(AIRLINEID));

CREATE TABLE CONTAINS

(AIRLINEID VARCHAR(3) NOT NULL,

AP\_NAME VARCHAR2(100) NOT NULL,

PRIMARY KEY(AIRLINEID, AP\_NAME),

FOREIGN KEY(AIRLINEID) REFERENCES AIRLINE(AIRLINEID) ON DELETE CASCADE,

FOREIGN KEY(AP\_NAME) REFERENCES AIRPORT(AP\_NAME) ON DELETE CASCADE);

CREATE TABLE FLIGHT

(FLIGHT\_CODE VARCHAR(10) NOT NULL,

SOURCE VARCHAR(3),

DESTINATION VARCHAR(3),

ARRIVAL VARCHAR2(10),

DEPARTURE VARCHAR2(10),

STATUS VARCHAR(10),

DURATION VARCHAR2(30),

FLIGHTTYPE VARCHAR(10),

LAYOVER\_TIME VARCHAR2(30),

NO\_OF\_STOPS INT,

AIRLINEID VARCHAR(3),

PRIMARY KEY(FLIGHT\_CODE),

FOREIGN KEY(AIRLINEID) REFERENCES AIRLINE(AIRLINEID) ON DELETE CASCADE);

CREATE TABLE PASSENGER1

(PID INT NOT NULL,

PASSPORTNO VARCHAR(10) NOT NULL,

PRIMARY KEY(PID, PASSPORTNO));

CREATE TABLE PASSENGER2

(PASSPORTNO VARCHAR(10) NOT NULL,

FNAME VARCHAR2(20),

M VARCHAR(1),

LNAME VARCHAR2(20),

ADDRESS VARCHAR2(100),

PHONE INT,

AGE INT,

SEX VARCHAR(1),

PRIMARY KEY(PASSPORTNO));

CREATE TABLE PASSENGER3

(PID INT NOT NULL,

FLIGHT\_CODE VARCHAR(10),

PRIMARY KEY(PID),

FOREIGN KEY(FLIGHT\_CODE) REFERENCES FLIGHT(FLIGHT\_CODE) ON DELETE CASCADE);

CREATE TABLE EMPLOYEE1

(SSN INT NOT NULL,

FNAME VARCHAR2(20),

M VARCHAR(1),

LNAME VARCHAR2(20),

ADDRESS VARCHAR2(100),

PHONE INT,

AGE INT,

SEX VARCHAR(1),

JOBTYPE VARCHAR2(30),

ASTYPE VARCHAR2(30),

ETYPE VARCHAR2(30),

SHIFT VARCHAR2(20),

POSITION VARCHAR2(30),

AP\_NAME VARCHAR2(100),

PRIMARY KEY(SSN),

FOREIGN KEY(AP\_NAME) REFERENCES AIRPORT(AP\_NAME) ON DELETE CASCADE);

CREATE TABLE EMPLOYEE2

(JOBTYPE VARCHAR2(30) NOT NULL,

SALARY INT,

PRIMARY KEY(JOBTYPE));

CREATE TABLE SERVES

(SSN INT NOT NULL,

PID INT NOT NULL,

PASSPORTNO VARCHAR(10) NOT NULL,

PRIMARY KEY(SSN, PID, PASSPORTNO),

FOREIGN KEY(SSN) REFERENCES EMPLOYEE1(SSN) ON DELETE CASCADE,

FOREIGN KEY(PID, PASSPORTNO) REFERENCES PASSENGER1(PID, PASSPORTNO) ON DELETE CASCADE);

CREATE TABLE TICKET1

(TICKET\_NUMBER INT NOT NULL,

SOURCE VARCHAR(3),

DESTINATION VARCHAR(3),

DATE\_OF\_BOOKING DATE,

DATE\_OF\_TRAVEL DATE,

SEATNO VARCHAR(5),

CLASS VARCHAR2(15),

DATE\_OF\_CANCELLATION DATE,

PID INT,

PASSPORTNO VARCHAR(10),

FOREIGN KEY(PID, PASSPORTNO) REFERENCES PASSENGER1(PID, PASSPORTNO) ON DELETE CASCADE);

CREATE TABLE TICKET2

(DATE\_OF\_BOOKING DATE NOT NULL,

SOURCE VARCHAR(3) NOT NULL,

DESTINATION VARCHAR(3) NOT NULL,

CLASS VARCHAR2(15) NOT NULL,

PRICE INT,

PRIMARY KEY(DATE\_OF\_BOOKING, SOURCE, DESTINATION, CLASS));

CREATE TABLE TICKET3

(DATE\_OF\_CANCELLATION DATE NOT NULL,

SURCHARGE INT,

PRIMARY KEY(DATE\_OF\_CANCELLATION));

CREATE TABLE TICKET\_PRICE\_HISTORY

(DATE\_OF\_BOOKING DATE NOT NULL,

SOURCE VARCHAR(3) NOT NULL,

DESTINATION VARCHAR(3) NOT NULL,

CLASS VARCHAR2(15) NOT NULL,

PRICE INT,

PRIMARY KEY(DATE\_OF\_BOOKING, SOURCE, DESTINATION, CLASS));

**PROCEDURES**

1)

CREATE OR REPLACE PROCEDURE DFWECONOMYPASSENGERS

AS

CURSOR ECOPASSDETAILS is

select al.AL\_NAME,fl.FLIGHT\_CODE,p2.FNAME,p2.LNAME,p2.PASSPORTNO,t.CLASS,t.DATE\_OF\_TRAVEL,t.DESTINATION,t.SOURCE,t.SEATNO,t.TICKET\_NUMBER from Airline al, Flight fl,PASSENGER1 p1,PASSENGER2 p2 ,PASSENGER3 p3,TICKET1 t

where al.AIRLINEID = fl.AIRLINEID

and p1.PID= p3.PID

and p1.PASSPORTNO =p2.PASSPORTNO

and fl.FLIGHT\_CODE = p3.FLIGHT\_CODE

and t.PASSPORTNO = p2.PASSPORTNO

and t.CLASS='ECONOMY'

and t.DESTINATION='DFW';

PASSDETAILS ECOPASSDETAILS%rowtype;

BEGIN

Open ECOPASSDETAILS ;

LOOP

fetch ECOPASSDETAILS into PASSDETAILS;

EXIT WHEN ECOPASSDETAILS%NOTFOUND;

dbms\_output.put\_line (PASSDETAILS.AL\_NAME || ' ' || PASSDETAILS.FLIGHT\_CODE || ' ' || PASSDETAILS.FNAME || ' ' || PASSDETAILS.LNAME|| ' ' ||PASSDETAILS.PASSPORTNO || ' ' || PASSDETAILS.CLASS || ' ' || PASSDETAILS.SOURCE || ' ' || PASSDETAILS.SEATNO || ' ' || PASSDETAILS.TICKET\_NUMBER);

END LOOP;

close ECOPASSDETAILS;

END DFWECONOMYPASSENGERS;

**Working:**

This procedure is used to retrieve the details of passengers flying to DFW (Dallas Fort International Airport) in the economy class.

The procedure starts with the declaration of a cursor named ECOPASSDETAILS. This cursor selects various details such as airline name, flight code, passenger's first and last name, passport number, class, date of travel, destination, source, seat number, and ticket number from multiple tables (Airline, Flight, PASSENGER1, PASSENGER2, PASSENGER3, TICKET1) based on specific conditions such as class being 'ECONOMY' and destination being 'DFW'

2)

CREATE OR REPLACE PROCEDURE FLIGHTSBYSTATUS

(IN\_STATUS IN VARCHAR2)

AS

CURSOR fSTATUS is

select distinct f.FLIGHT\_CODE, al.AL\_NAME, f.ARRIVAL,

f. departure, f.SOURCE, f.DESTINATION, f.STATUS,f.FLIGHTTYPE from Airline al ,Airport ap,flight f

where al. AIRLINEID=f.AIRLINEID

and f. STATUS =IN\_STATUS;

FlightStatus fSTATUS%rowtype;

BEGIN

Open fSTATUS ;

LOOP

fetch fSTATUS into FlightStatus;

EXIT WHEN fSTATUS%NOTFOUND;

dbms\_output.put\_line (FlightStatus.FLIGHT\_CODE || ‘ ’ || FlightStatus.AL\_NAME || ' ' || FlightStatus.ARRIVAL || ' ' || FlightStatus.departure|| ' ' ||FlightStatus.Source || ' ' || FlightStatus.DESTINATION || ' ' || FlightStatus.STATUS || ' ' || FlightStatus.FLIGHTTYPE);

END LOOP;

close fSTATUS;

END FLIGHTSBYSTATUS;

**Working:**

This procedure is used to retrieve flight details based on a given status.

Cursor ‘fSTATUS’ is declared to retrieve flight details based on the provided status either delayed or on-time.This cursor selects distinct flight details such as flight code, airline name, arrival time, departure time, source airport, destination airport, status, and flight type from multiple tables, that is, AIRLINE, AIRPORT and FLIGHT.

**TRIGGERS**

1)

CREATE OR REPLACE TRIGGER DELAYEDFLIGHTS

AFTER INSERT ON FLIGHT

FOR EACH ROW

DECLARE

BEGIN

if :new.STATUS = 'Delayed' then

insert into delayed\_flights (FLIGHT\_CODE, SOURCE, DESTINATION, AIRLINEID, report\_time) values (new.flight\_code ,:new.source,:new.destination,:new.airlineid,sysdate);

    end if;

END;

**Working:**

This row level trigger is designed to fire after each insert operation on the table. It attempts to insert a row into the ‘delayed\_flights’ table whenever a row is inserted into the ‘FLIGHT’ table but only if the inserted entry has ‘STATUS’ column as delayed.

2)

CREATE OR REPLACE TRIGGER UPDATEDSalary

BEFORE INSERT OR UPDATE OF SALARY ON EMPLOYEE2

FOR EACH ROW

BEGIN

CASE :NEW.JOBTYPE

WHEN 'Administrative Support' THEN

:NEW.SALARY := :NEW.SALARY + (:NEW.SALARY \* 0.10);

WHEN 'ENGINEER' THEN

:NEW.SALARY := :NEW.SALARY + (:NEW.SALARY \* 0.05);

WHEN 'TRAFFIC MONITOR' THEN

:NEW.SALARY := :NEW.SALARY + (:NEW.SALARY \* 0.25);

WHEN ‘AIRPORT AUTHORITY’ THEN

:NEW.SALARY := :NEW.SALARY + (:NEW.SALARY \* 0.45);

END CASE;

END;

**Working:**

This trigger aims at updating the salary of employees in the ‘EMPLOYEE2’ table based on their job types after an insert or update operation on the ‘SALARY’ column.

3)

CREATE OR REPLACE TRIGGER TICKET\_PRICE\_HISTORY

BEFORE UPDATE OF PRICE

ON TICKET2

FOR EACH ROW

BEGIN

INSERT INTO TICKET\_PRICE\_HISTORY

VALUES(:OLD.DATE\_OF\_BOOKING, :OLD.SOURCE, :OLD:DESTINATION, :OLD.CLASS, :OLD.PRICE);

END;

**Working:**

This trigger aims at capturing changes in ticket prices by inserting a record into the ‘TICKET\_PRICE\_HISTORY’ table before an update operation on the ‘PRICE’ column of the ‘TICKET2’ table.