IE2042 – DATABASE MANAGEMENT SYSTEMS FOR SECURITY

Year 02 Semester 01

Cyber Security

Member Details

IT number	Name
IT21162596	Gunasekara M.V.G.R.S.
IT21253058	Kavirathne G.P.R.Y.
IT21173486	Migara H.M.S
IT21201578	Fawsikdeen H.

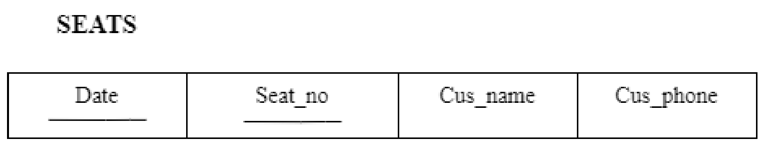
PART 1

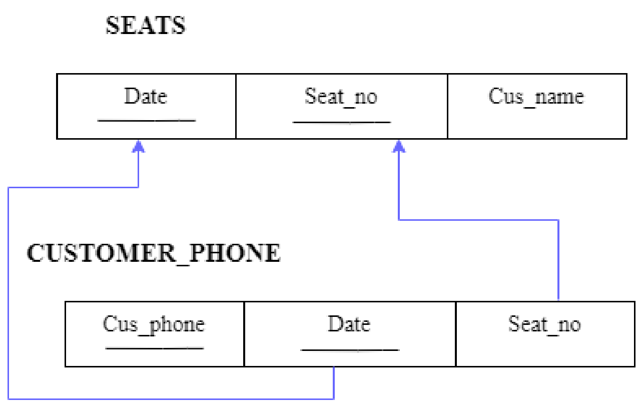
1) Assumptions

- The airplane name **(Airplane_name)** is determined by the airplane type name **(T_name)**.
- The scheduled departure time **(Schedule_dep_time)** and arrival time **(Schedule_arr_time)** depends on the airport code **(Airport_code)**.
- The number of available seats **(No_of_available_seats)** is determined by the airplane ID **(Airplane_ID)**
- It is assumed that the maximum seat number **(Max_seats)** of a flight depends on the company **(Company)** the flight is owned by.
- It is assumed that the customer phone number **(Cus_phone)** attribute in seats entity is a multi-valued attribute.

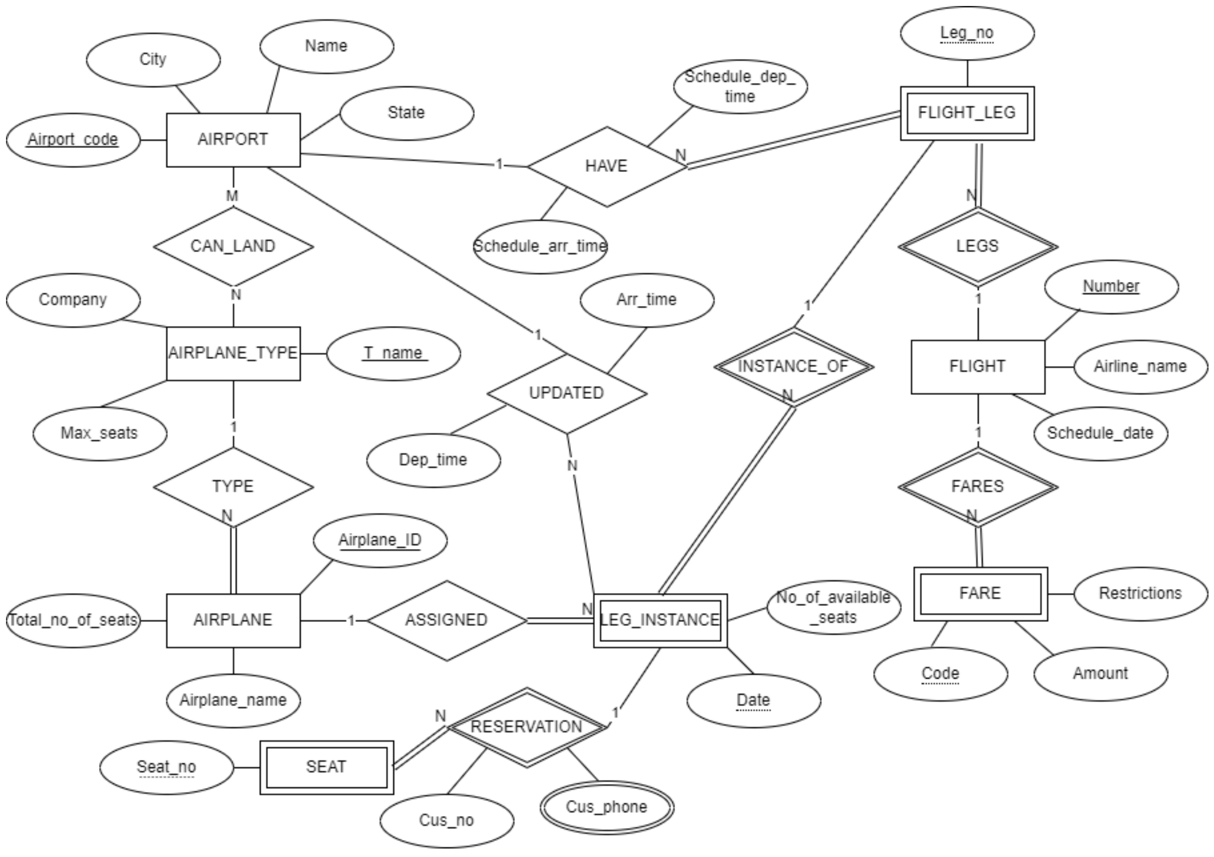
➢ Decomposing Seats Table

❖ customer phone number **(Cus_phone)** attribute in seats entity is a multi-valued attribute.

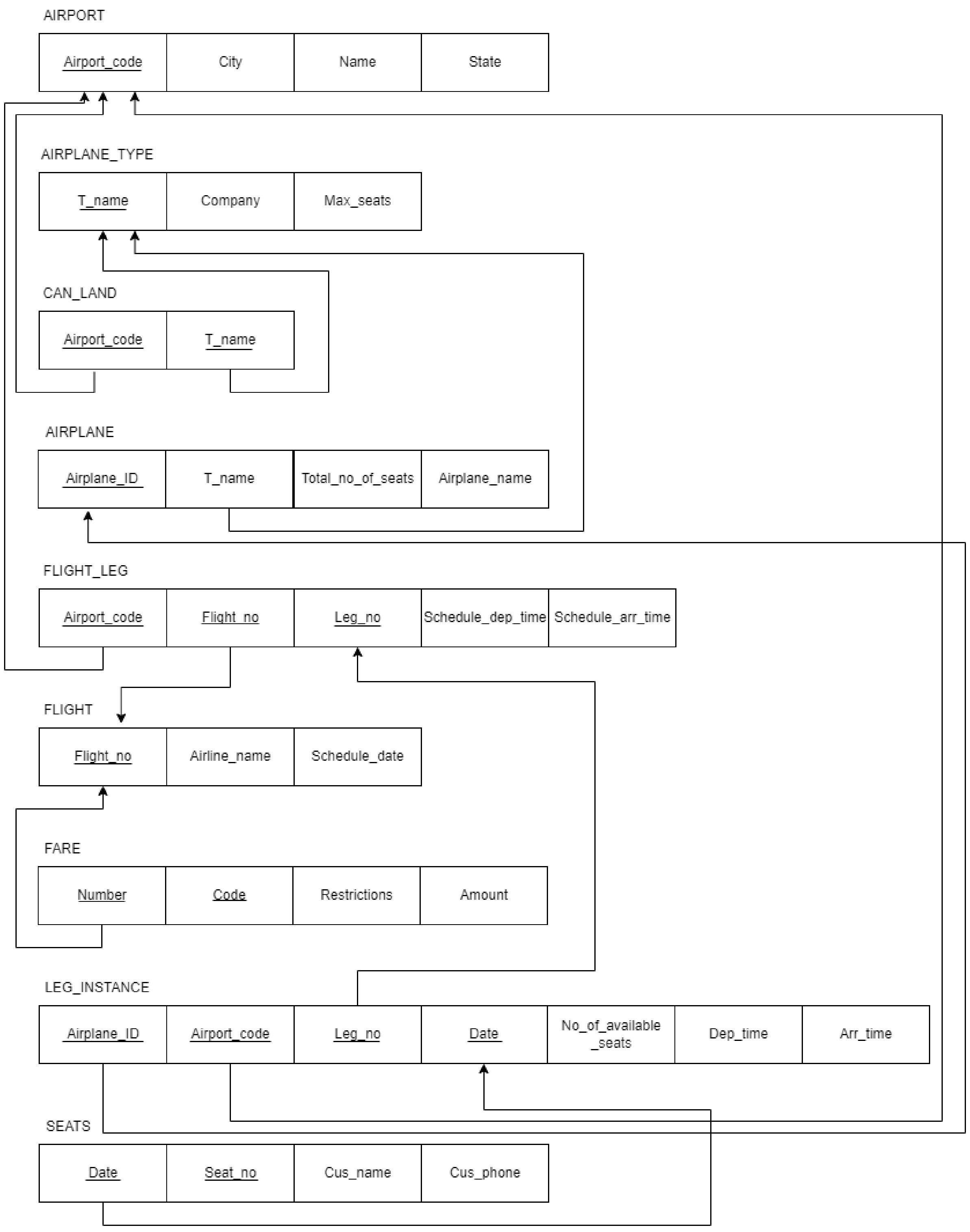




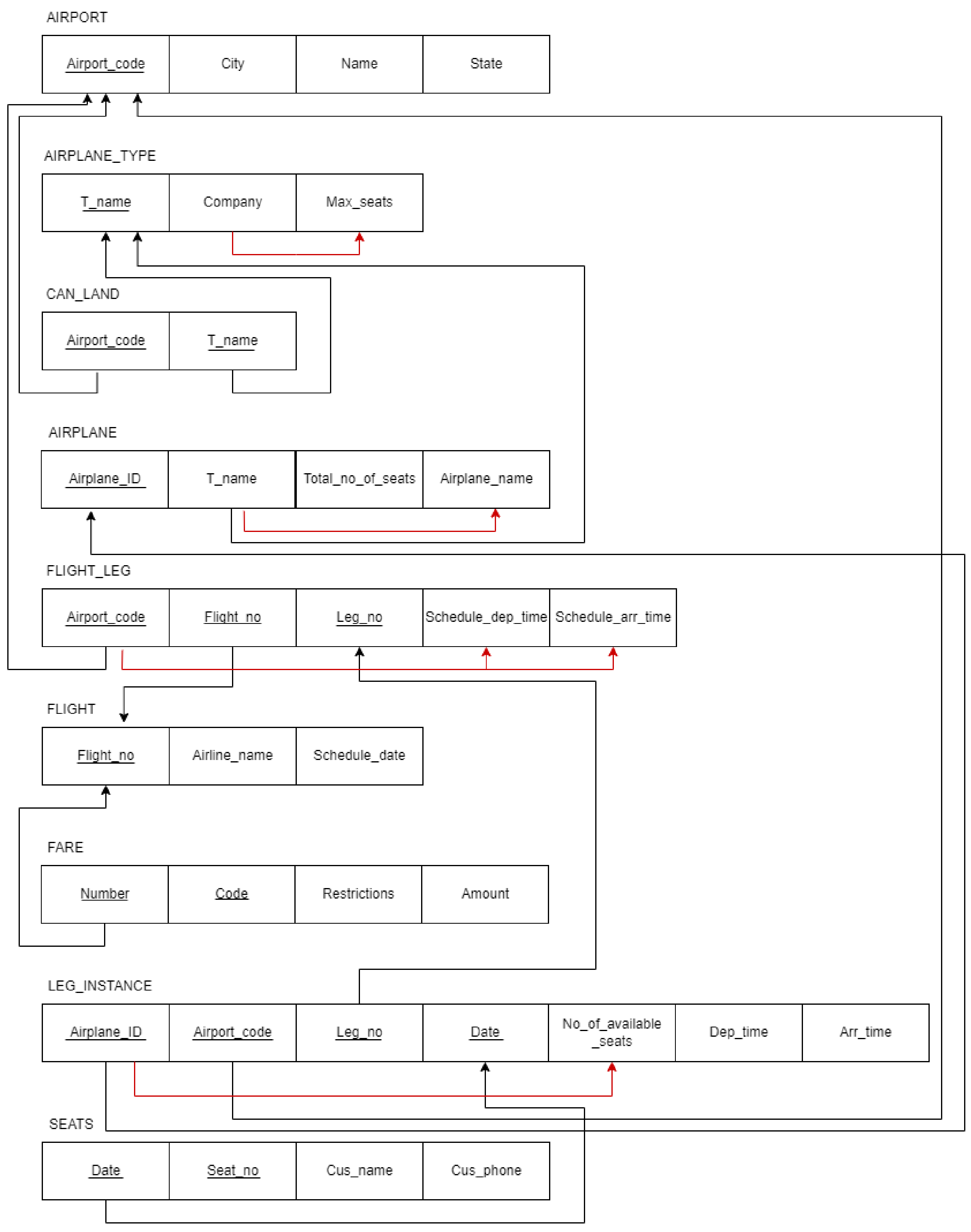
2) Entity Relationship Diagram (ERD)



➢ Logical Model

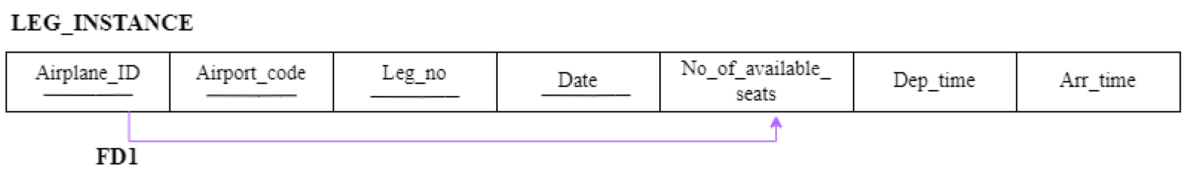


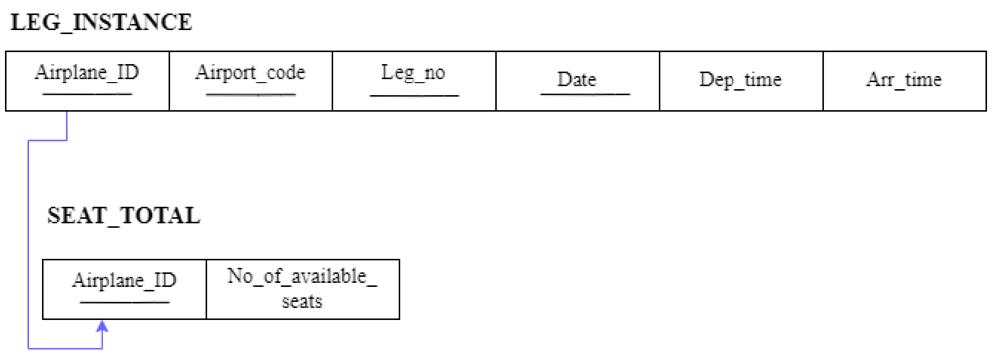
- **Logical Model with the functional dependencies (FD)**



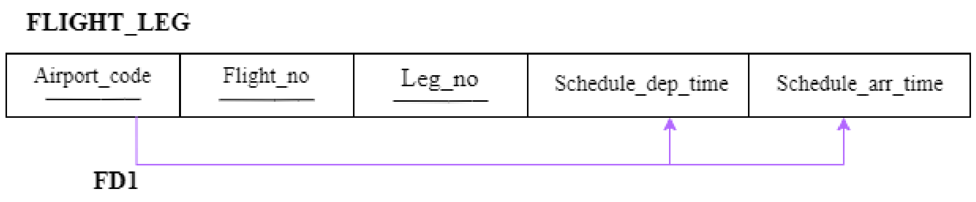
- **Normalizing the logical model to 3NF**

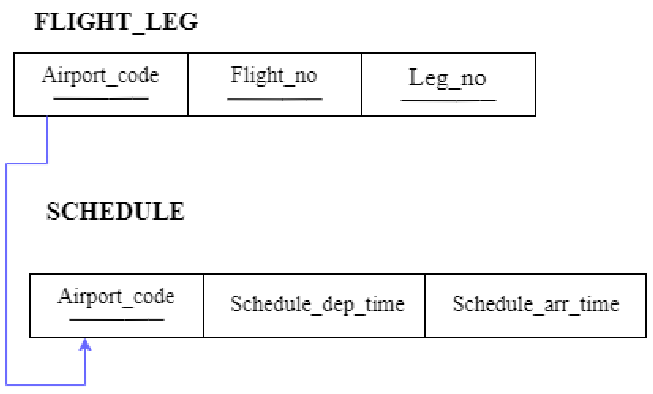
  - **Decomposing Leg Instance Table**



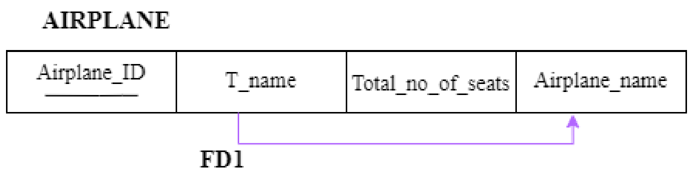


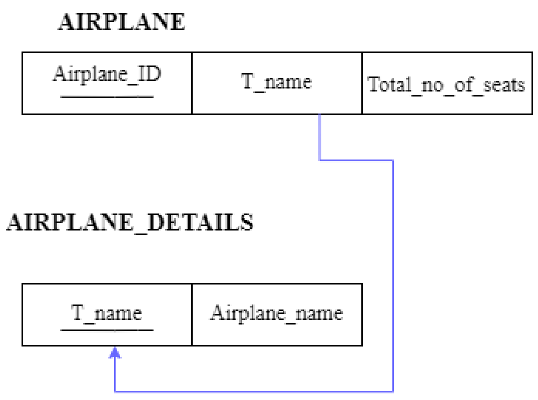
  - **Decomposing Flight Leg Table**



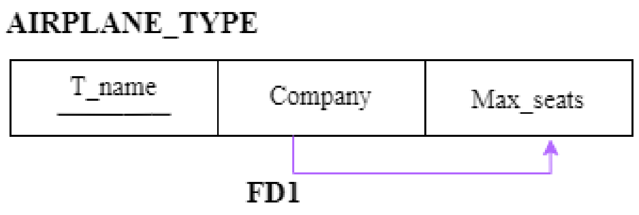


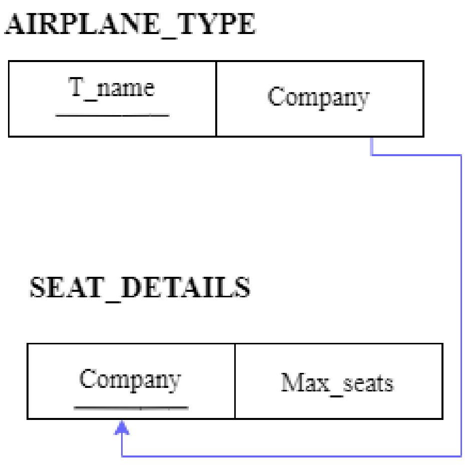
▪ Decomposing Airplane Type Table



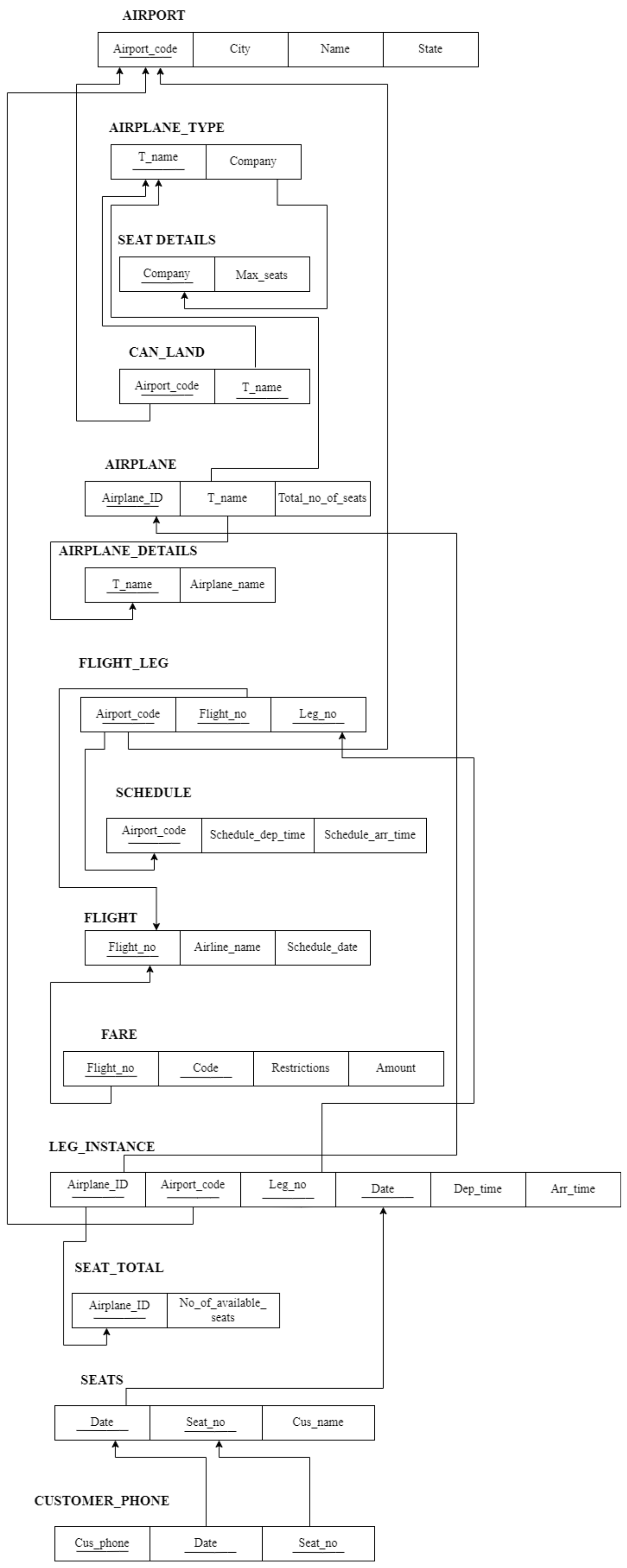


▪ Decomposing Airplane Table





➢ Logical model after normalization



3)

Airport (airport_code: varchar(10), name: varchar(100), city: varchar(30), state: varchar(30))

Seat_details (company: varchar(30), max_seat: integer)

Airplane_type (t_name: varchar(30), company: varchar(30))

Can_land (airport_code: varchar(10), t_name: varchar(30))

Airplane (airplane_ID: integer, t_name: varchar(30), total_no_of_seats: integer)

Airplane_details (airplane_name: varchar(10), t_name: varchar(30))

Schedule (airport_code: varchar(10), schedule_dep_time: time, schedule_arr_time: time)

Flight (flight_no: varchar(12), airplane_name: varchar(30), schedule_date: date)

Flight_leg (leg_no: integer, flight_no: varchar(12), airport_code: varchar(10))

Fare (code: integer, restriction: varchar(30), amount: integer, flight_no: varchar(12))

Leg_instance (leg_date: date, dep_time: time, arr_time: time, airplane_ID: integer, airport_code: varchar(10), leg_no: integer)

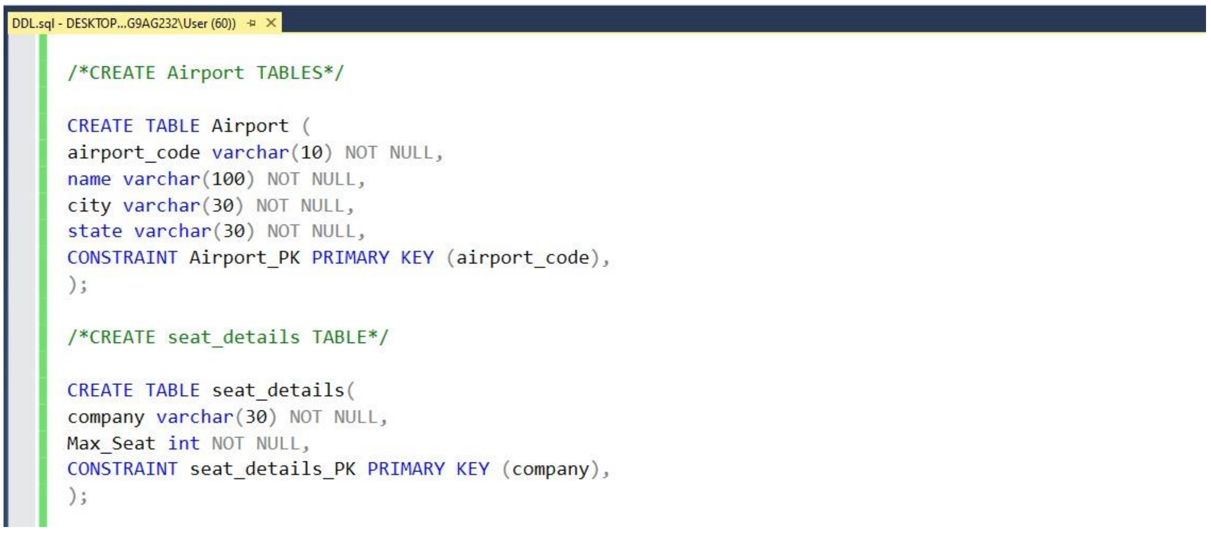
Seats_total (airplane_ID: integer, no_of_available_seat: integer)

Seats (seats_no: integer, cus_no: integer, leg_date: date, cus_name: varchar(50))

Customer_phone (cus_phone: integer, seat_no: integer, leg_date: date)

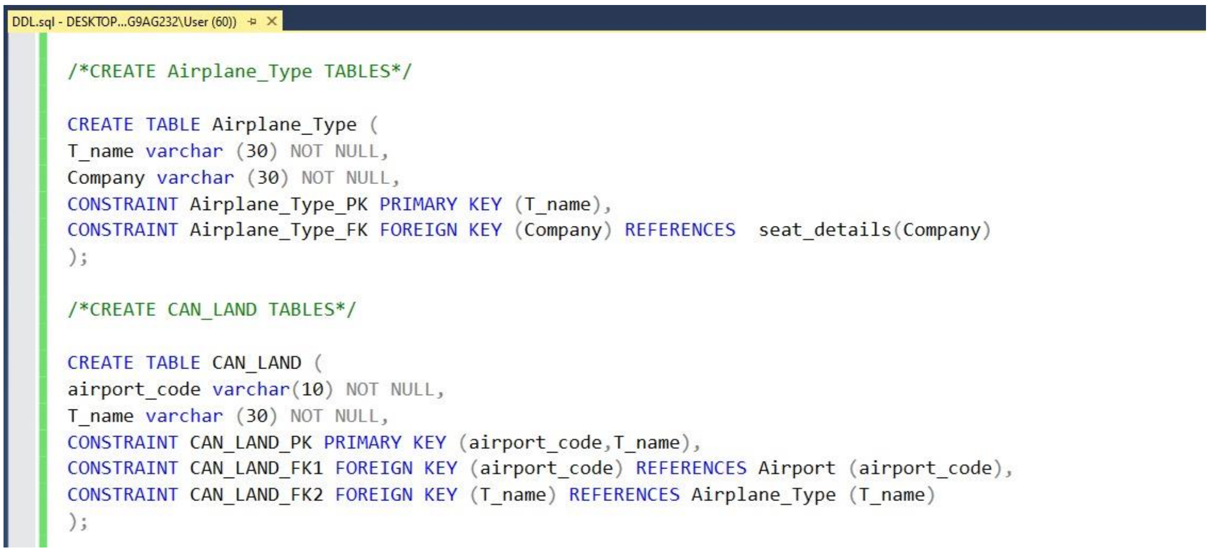
4) SQL Codes (Table creations)

- Airport Table
- Seat_details Table



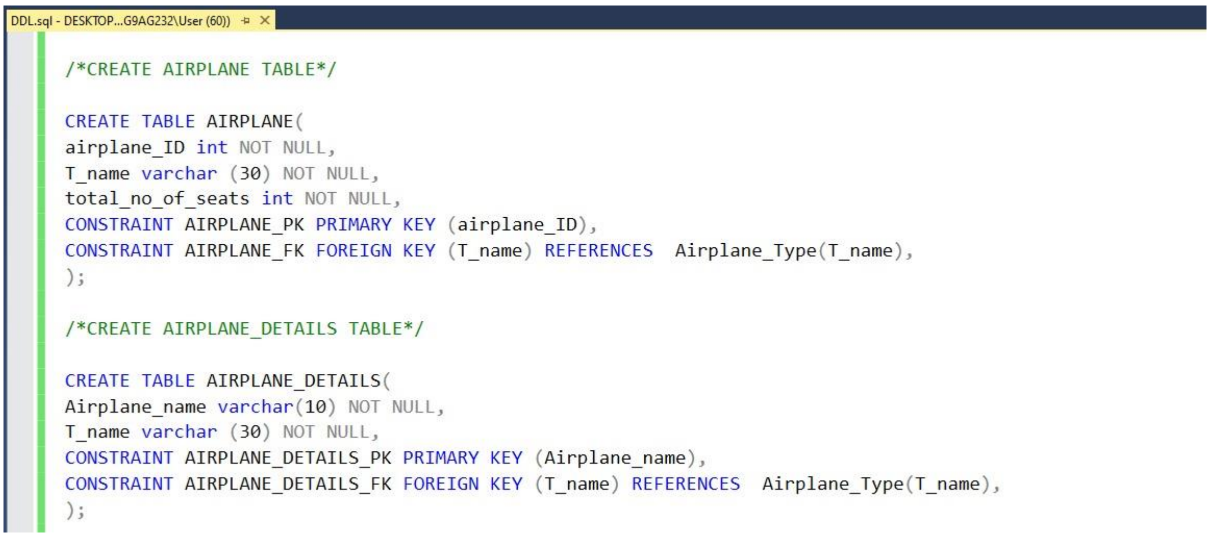
```
/*CREATE Airport TABLES*/

CREATE TABLE Airport (
airport_code varchar(10) NOT NULL,
name varchar(100) NOT NULL,
city varchar(30) NOT NULL,
state varchar(30) NOT NULL,
CONSTRAINT Airport_PK PRIMARY KEY (airport_code),
);

/*CREATE seat_details TABLE*/

CREATE TABLE seat_details(
company varchar(30) NOT NULL,
Max_Seat int NOT NULL,
CONSTRAINT seat_details_PK PRIMARY KEY (company),
);
```

- Airplane_type Table
- Can_land Table



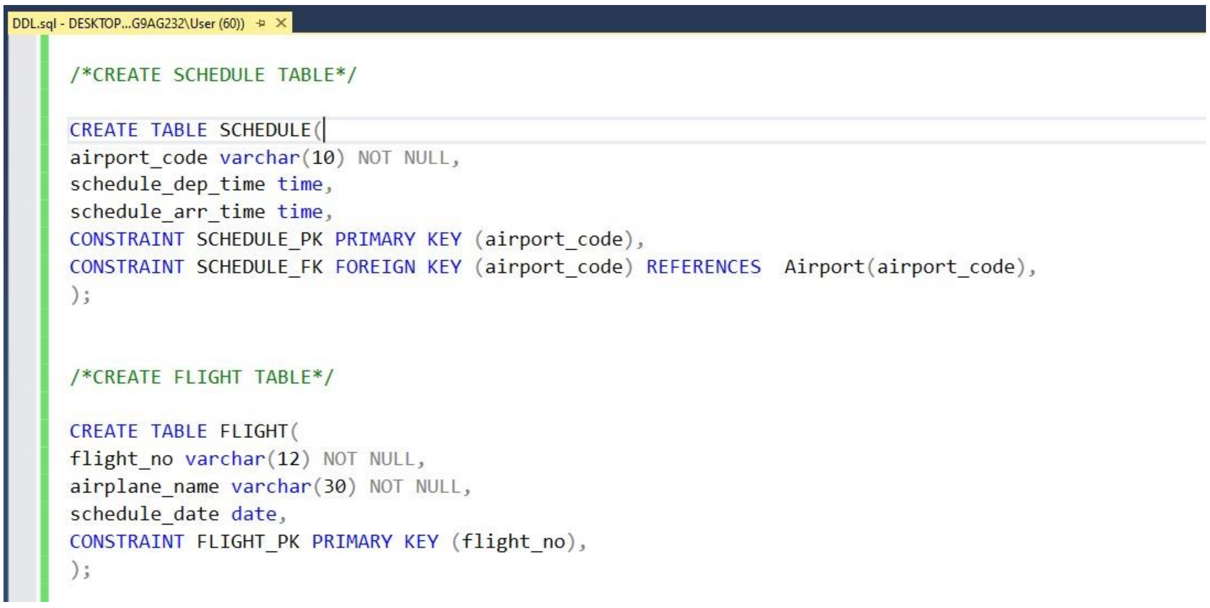
```
/*CREATE Airplane_Type TABLES*/

CREATE TABLE Airplane_Type (
T_name varchar (30) NOT NULL,
Company varchar (30) NOT NULL,
CONSTRAINT Airplane_Type_PK PRIMARY KEY (T_name),
CONSTRAINT Airplane_Type_FK FOREIGN KEY (Company) REFERENCES  seat_details(Company)
);

/*CREATE CAN_LAND TABLES*/

CREATE TABLE CAN_LAND (
airport_code varchar(10) NOT NULL,
T_name varchar (30) NOT NULL,
CONSTRAINT CAN_LAND_PK PRIMARY KEY (airport_code,T_name),
CONSTRAINT CAN_LAND_FK1 FOREIGN KEY (airport_code) REFERENCES Airport (airport_code),
CONSTRAINT CAN_LAND_FK2 FOREIGN KEY (T_name) REFERENCES Airplane_Type (T_name)
);
```

- Airplane Table
- Airplane_details Table



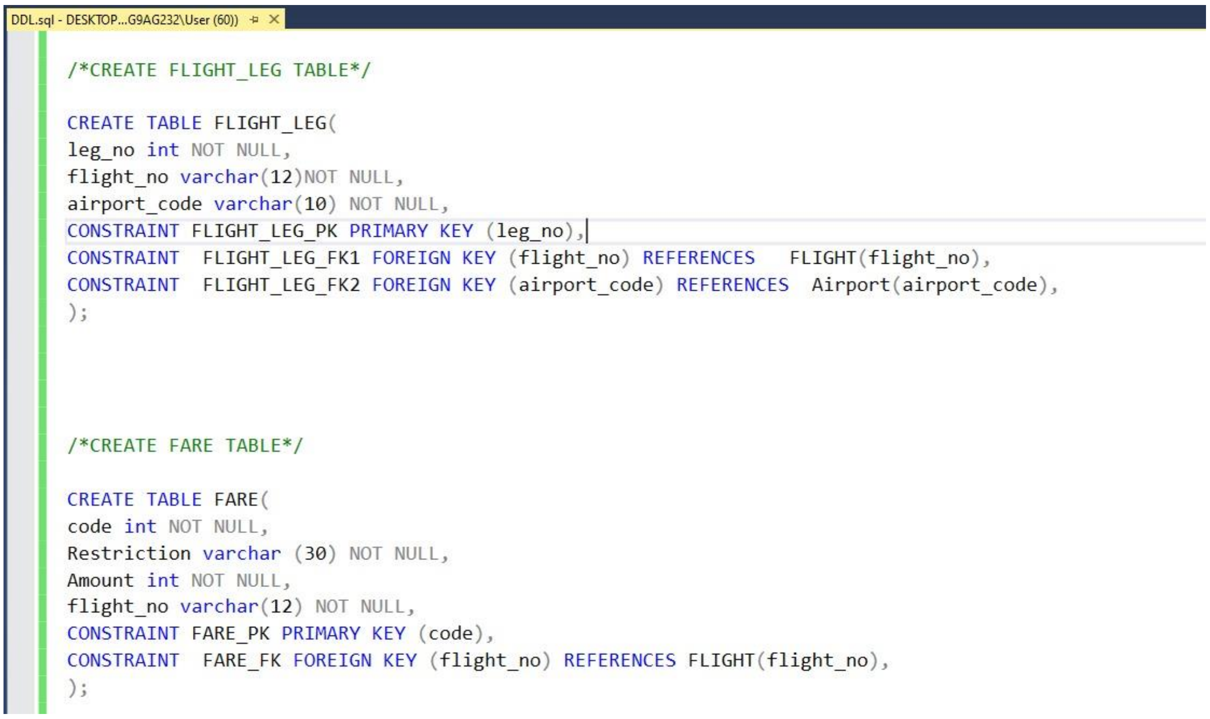
```sql
/*CREATE AIRPLANE TABLE*/

CREATE TABLE AIRPLANE(
airplane_ID int NOT NULL,
T_name varchar (30) NOT NULL,
total_no_of_seats int NOT NULL,
CONSTRAINT AIRPLANE_PK PRIMARY KEY (airplane_ID),
CONSTRAINT AIRPLANE_FK FOREIGN KEY (T_name) REFERENCES  Airplane_Type(T_name),
);

/*CREATE AIRPLANE_DETAILS TABLE*/

CREATE TABLE AIRPLANE_DETAILS(
Airplane_name varchar(10) NOT NULL,
T_name varchar (30) NOT NULL,
CONSTRAINT AIRPLANE_DETAILS_PK PRIMARY KEY (Airplane_name),
CONSTRAINT AIRPLANE_DETAILS_FK FOREIGN KEY (T_name) REFERENCES  Airplane_Type(T_name),
);
```

- Schedule Table
- Flight Table



```sql
/*CREATE SCHEDULE TABLE*/

CREATE TABLE SCHEDULE(
airport_code varchar(10) NOT NULL,
schedule_dep_time time,
schedule_arr_time time,
CONSTRAINT SCHEDULE_PK PRIMARY KEY (airport_code),
CONSTRAINT SCHEDULE_FK FOREIGN KEY (airport_code) REFERENCES  Airport(airport_code),
);


/*CREATE FLIGHT TABLE*/

CREATE TABLE FLIGHT(
flight_no varchar(12) NOT NULL,
airplane_name varchar(30) NOT NULL,
schedule_date date,
CONSTRAINT FLIGHT_PK PRIMARY KEY (flight_no),
);
```

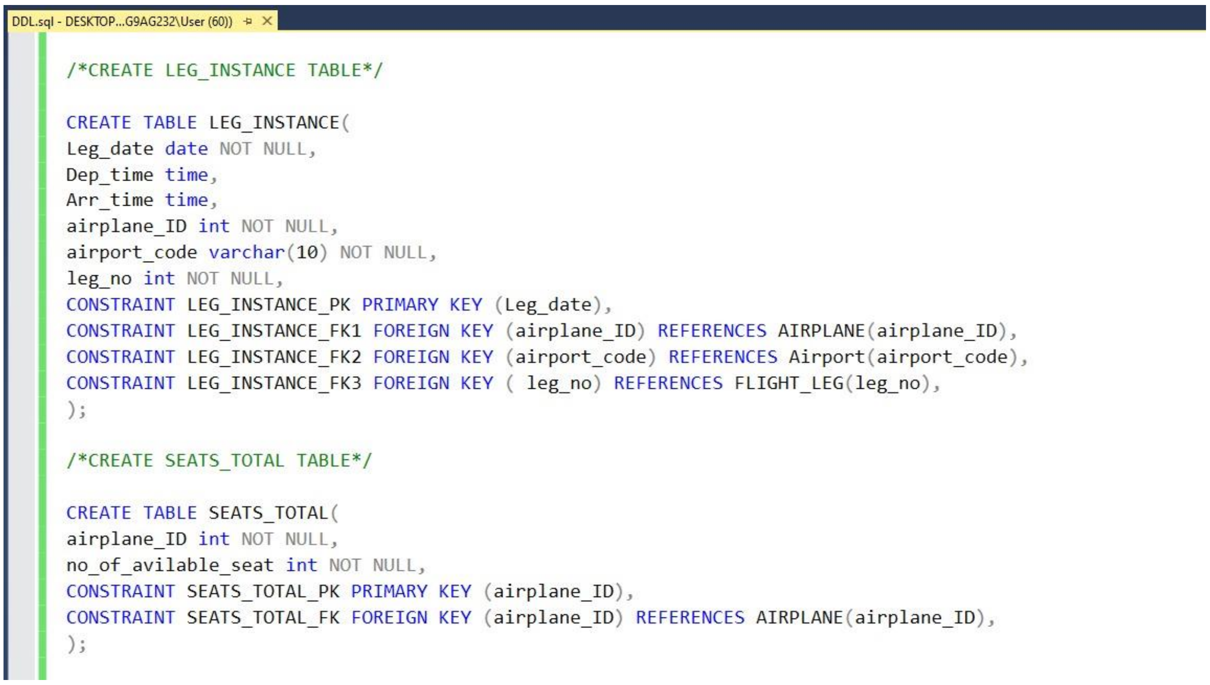
- Flight_leg Table
- Fare Table



DDL.sql - DESKTOP...G9AG232\User (60))

```sql
/*CREATE FLIGHT_LEG TABLE*/

CREATE TABLE FLIGHT_LEG(
leg_no int NOT NULL,
flight_no varchar(12)NOT NULL,
airport_code varchar(10) NOT NULL,
CONSTRAINT FLIGHT_LEG_PK PRIMARY KEY (leg_no),
CONSTRAINT  FLIGHT_LEG_FK1 FOREIGN KEY (flight_no) REFERENCES   FLIGHT(flight_no),
CONSTRAINT  FLIGHT_LEG_FK2 FOREIGN KEY (airport_code) REFERENCES  Airport(airport_code),
);



/*CREATE FARE TABLE*/

CREATE TABLE FARE(
code int NOT NULL,
Restriction varchar (30) NOT NULL,
Amount int NOT NULL,
flight_no varchar(12) NOT NULL,
CONSTRAINT FARE_PK PRIMARY KEY (code),
CONSTRAINT  FARE_FK FOREIGN KEY (flight_no) REFERENCES FLIGHT(flight_no),
);
```

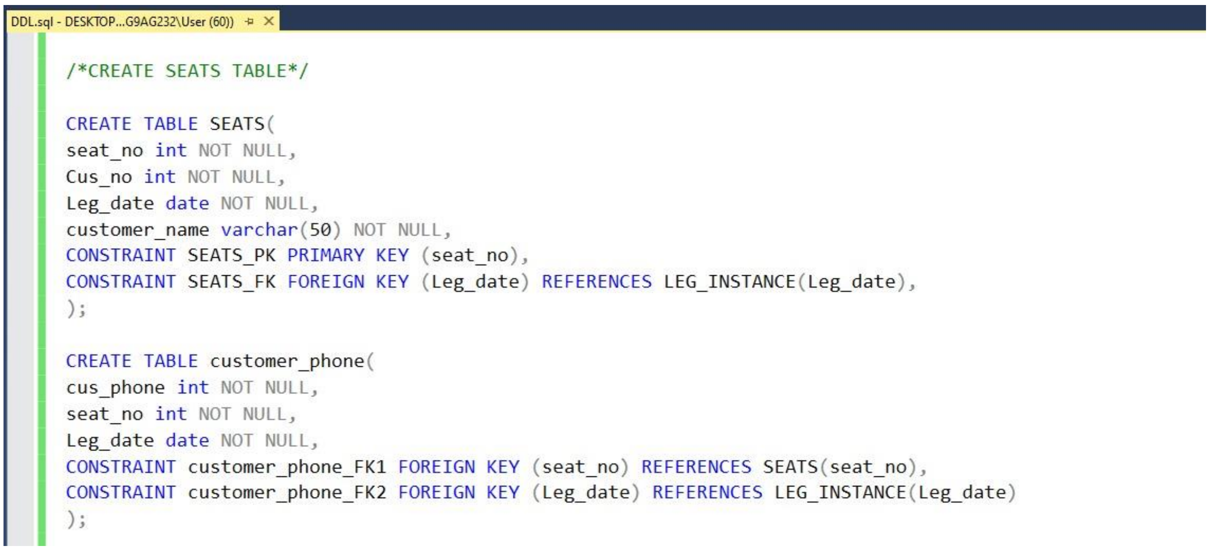
- Leg_instance Table
- Seats_total Table



DDL.sql - DESKTOP...G9AG232\User (60))

```sql
/*CREATE LEG_INSTANCE TABLE*/

CREATE TABLE LEG_INSTANCE(
Leg_date date NOT NULL,
Dep_time time,
Arr_time time,
airplane_ID int NOT NULL,
airport_code varchar(10) NOT NULL,
leg_no int NOT NULL,
CONSTRAINT LEG_INSTANCE_PK PRIMARY KEY (Leg_date),
CONSTRAINT LEG_INSTANCE_FK1 FOREIGN KEY (airplane_ID) REFERENCES AIRPLANE(airplane_ID),
CONSTRAINT LEG_INSTANCE_FK2 FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
CONSTRAINT LEG_INSTANCE_FK3 FOREIGN KEY ( leg_no) REFERENCES FLIGHT_LEG(leg_no),
);

/*CREATE SEATS_TOTAL TABLE*/

CREATE TABLE SEATS_TOTAL(
airplane_ID int NOT NULL,
no_of_avilable_seat int NOT NULL,
CONSTRAINT SEATS_TOTAL_PK PRIMARY KEY (airplane_ID),
CONSTRAINT SEATS_TOTAL_FK FOREIGN KEY (airplane_ID) REFERENCES AIRPLANE(airplane_ID),
);
```

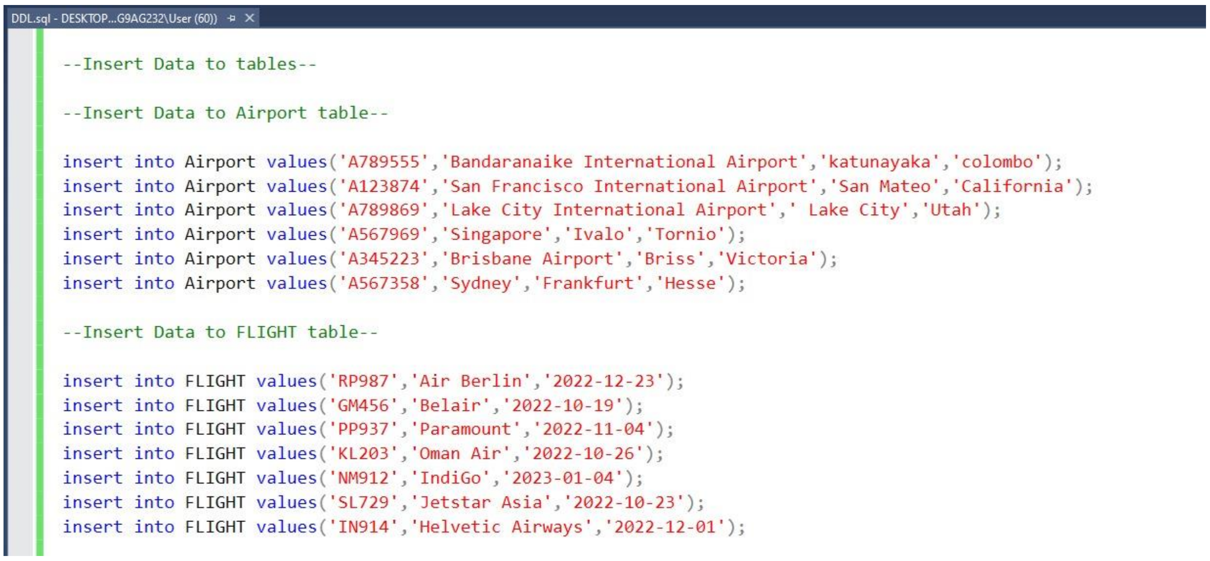
- Seats Table
- Customer_phone Table



```
/*CREATE SEATS TABLE*/

CREATE TABLE SEATS(
seat_no int NOT NULL,
Cus_no int NOT NULL,
Leg_date date NOT NULL,
customer_name varchar(50) NOT NULL,
CONSTRAINT SEATS_PK PRIMARY KEY (seat_no),
CONSTRAINT SEATS_FK FOREIGN KEY (Leg_date) REFERENCES LEG_INSTANCE(Leg_date),
);

CREATE TABLE customer_phone(
cus_phone int NOT NULL,
seat_no int NOT NULL,
Leg_date date NOT NULL,
CONSTRAINT customer_phone_FK1 FOREIGN KEY (seat_no) REFERENCES SEATS(seat_no),
CONSTRAINT customer_phone_FK2 FOREIGN KEY (Leg_date) REFERENCES LEG_INSTANCE(Leg_date)
);
```

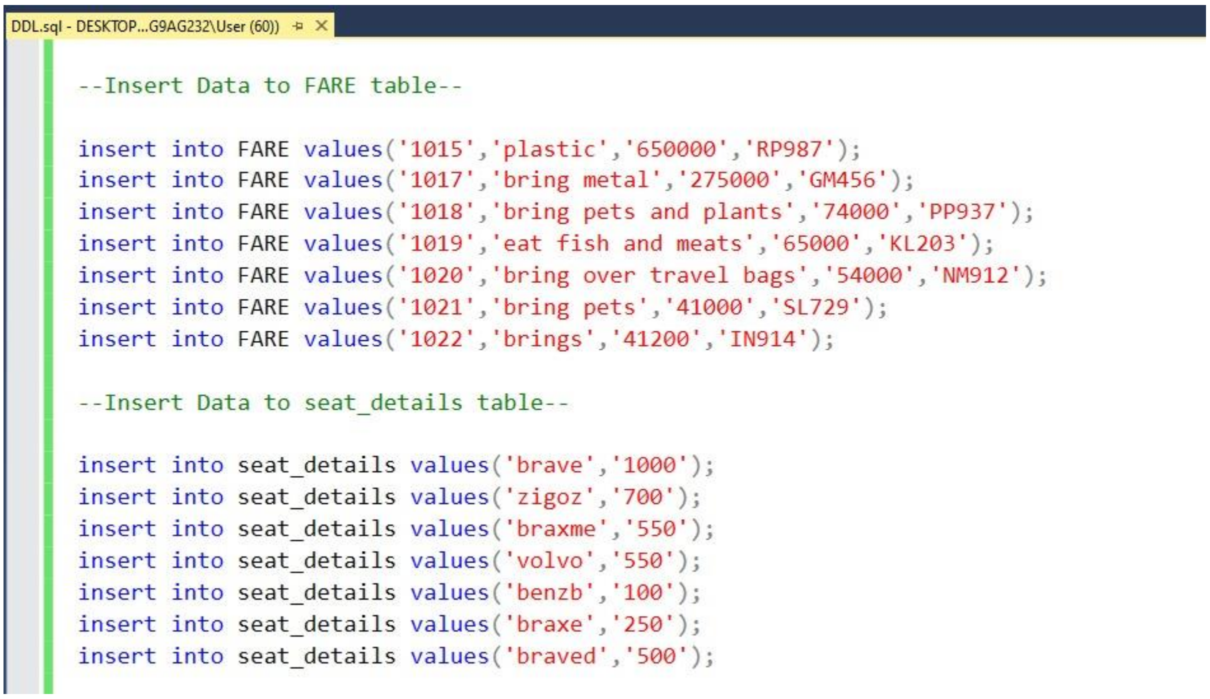
Data insertion

- Airport Table
- Flight Table



```
--Insert Data to tables--

--Insert Data to Airport table--

insert into Airport values('A789555','Bandaranaike International Airport','katunayaka','colombo');
insert into Airport values('A123874','San Francisco International Airport','San Mateo','California');
insert into Airport values('A789869','Lake City International Airport',' Lake City','Utah');
insert into Airport values('A567969','Singapore','Ivalo','Tornio');
insert into Airport values('A345223','Brisbane Airport','Briss','Victoria');
insert into Airport values('A567358','Sydney','Frankfurt','Hesse');

--Insert Data to FLIGHT table--

insert into FLIGHT values('RP987','Air Berlin','2022-12-23');
insert into FLIGHT values('GM456','Belair','2022-10-19');
insert into FLIGHT values('PP937','Paramount','2022-11-04');
insert into FLIGHT values('KL203','Oman Air','2022-10-26');
insert into FLIGHT values('NM912','IndiGo','2023-01-04');
insert into FLIGHT values('SL729','Jetstar Asia','2022-10-23');
insert into FLIGHT values('IN914','Helvetic Airways','2022-12-01');
```

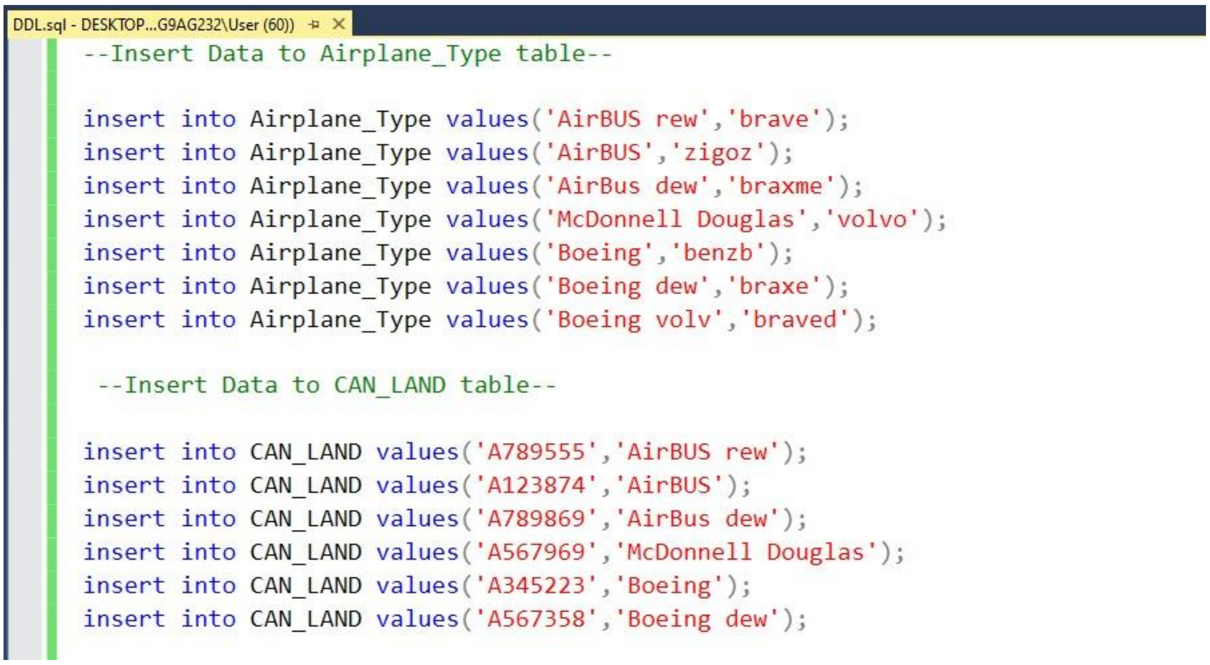
- Fare Table
- Seat_details Table



DDL.sql - DESKTOP...G9AG232\User (60))

```
--Insert Data to FARE table--

insert into FARE values('1015','plastic','650000','RP987');
insert into FARE values('1017','bring metal','275000','GM456');
insert into FARE values('1018','bring pets and plants','74000','PP937');
insert into FARE values('1019','eat fish and meats','65000','KL203');
insert into FARE values('1020','bring over travel bags','54000','NM912');
insert into FARE values('1021','bring pets','41000','SL729');
insert into FARE values('1022','brings','41200','IN914');

--Insert Data to seat_details table--

insert into seat_details values('brave','1000');
insert into seat_details values('zigoz','700');
insert into seat_details values('braxme','550');
insert into seat_details values('volvo','550');
insert into seat_details values('benzb','100');
insert into seat_details values('braxe','250');
insert into seat_details values('braved','500');
```

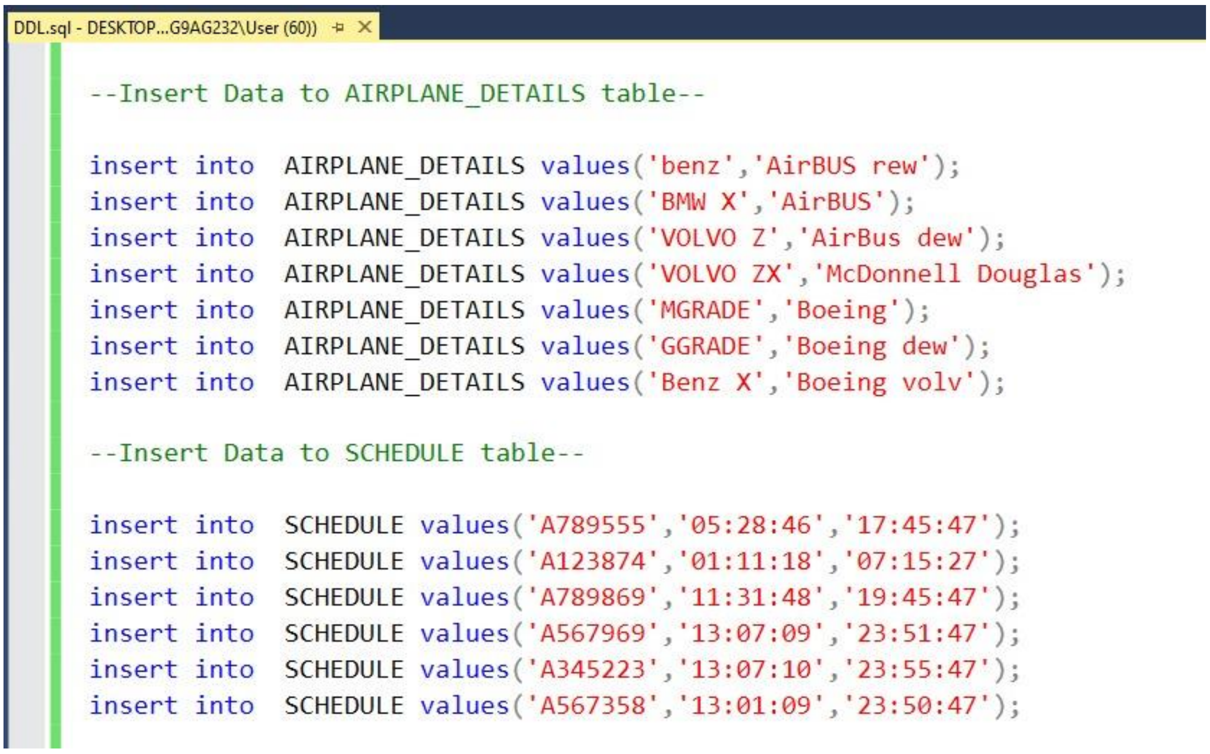
- Airplane_type Table
- Can_land Table



DDL.sql - DESKTOP...G9AG232\User (60))

```
--Insert Data to Airplane_Type table--

insert into Airplane_Type values('AirBUS rew','brave');
insert into Airplane_Type values('AirBUS','zigoz');
insert into Airplane_Type values('AirBus dew','braxme');
insert into Airplane_Type values('McDonnell Douglas','volvo');
insert into Airplane_Type values('Boeing','benzb');
insert into Airplane_Type values('Boeing dew','braxe');
insert into Airplane_Type values('Boeing volv','braved');

 --Insert Data to CAN_LAND table--

insert into CAN_LAND values('A789555','AirBUS rew');
insert into CAN_LAND values('A123874','AirBUS');
insert into CAN_LAND values('A789869','AirBus dew');
insert into CAN_LAND values('A567969','McDonnell Douglas');
insert into CAN_LAND values('A345223','Boeing');
insert into CAN_LAND values('A567358','Boeing dew');
```

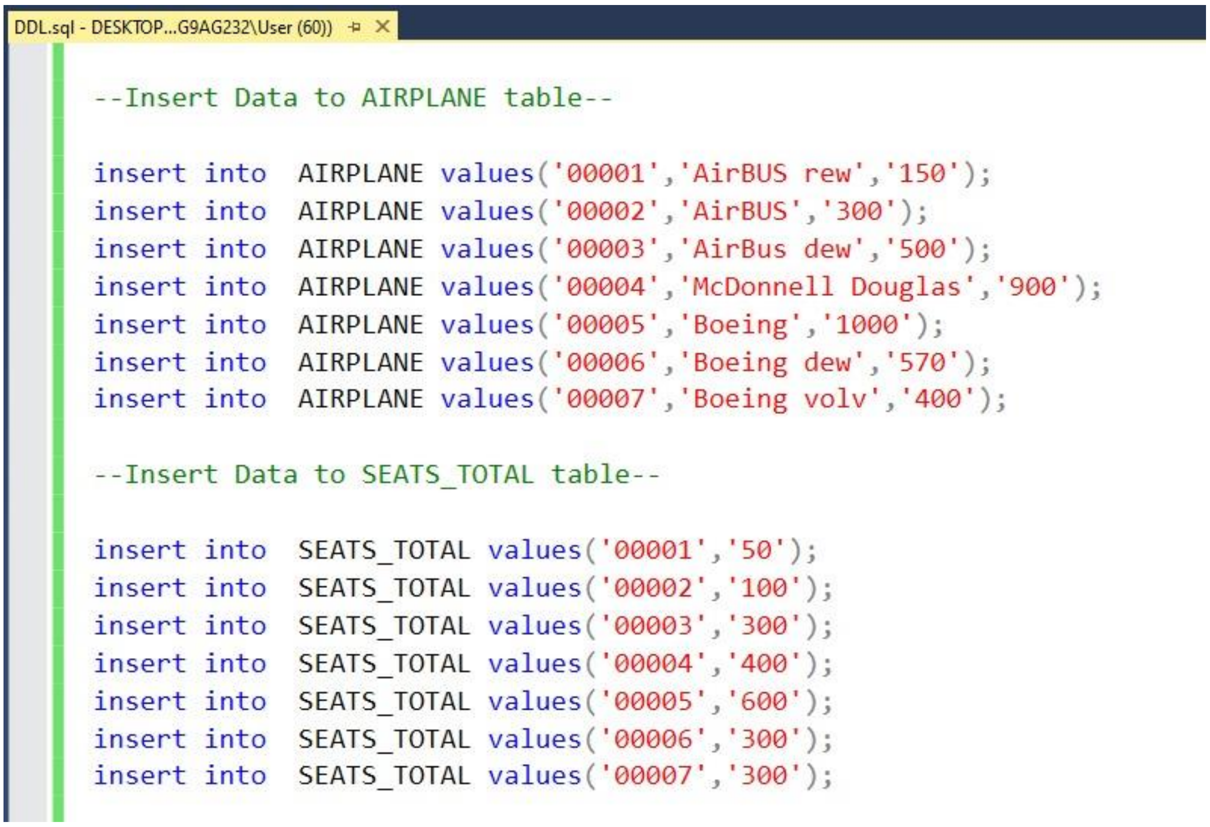
- Airplane_details Table
- Schedule Table



DDL.sql - DESKTOP...G9AG232\User (60))

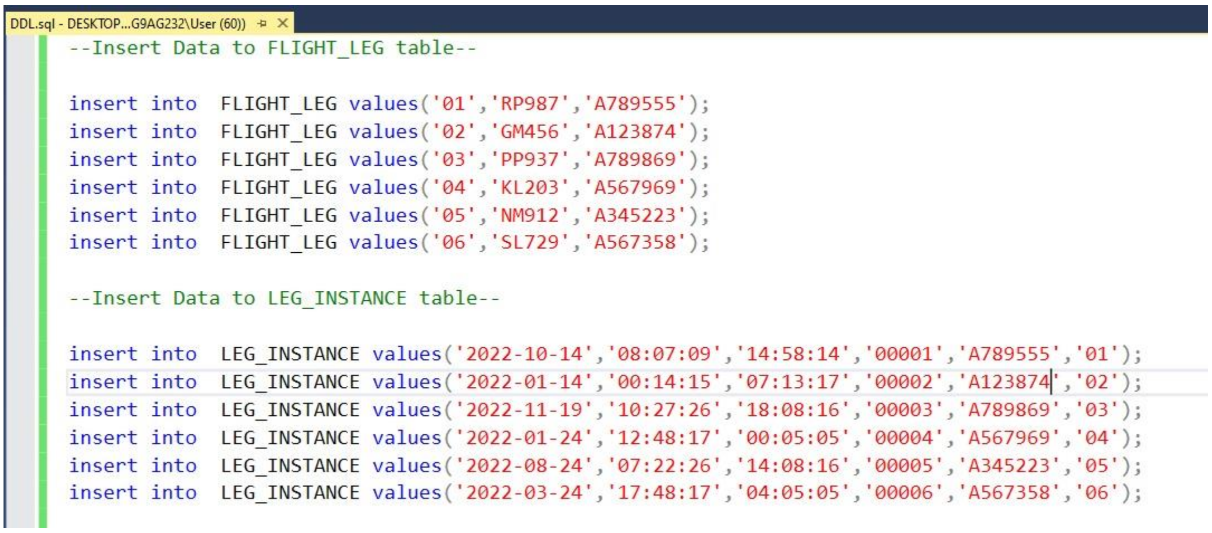
```
--Insert Data to AIRPLANE_DETAILS table--

insert into  AIRPLANE_DETAILS values('benz','AirBUS rew');
insert into  AIRPLANE_DETAILS values('BMW X','AirBUS');
insert into  AIRPLANE_DETAILS values('VOLVO Z','AirBus dew');
insert into  AIRPLANE_DETAILS values('VOLVO ZX','McDonnell Douglas');
insert into  AIRPLANE_DETAILS values('MGRADE','Boeing');
insert into  AIRPLANE_DETAILS values('GGRADE','Boeing dew');
insert into  AIRPLANE_DETAILS values('Benz X','Boeing volv');

--Insert Data to SCHEDULE table--

insert into  SCHEDULE values('A789555','05:28:46','17:45:47');
insert into  SCHEDULE values('A123874','01:11:18','07:15:27');
insert into  SCHEDULE values('A789869','11:31:48','19:45:47');
insert into  SCHEDULE values('A567969','13:07:09','23:51:47');
insert into  SCHEDULE values('A345223','13:07:10','23:55:47');
insert into  SCHEDULE values('A567358','13:01:09','23:50:47');
```

- Airplane Table
- Seats_total Table



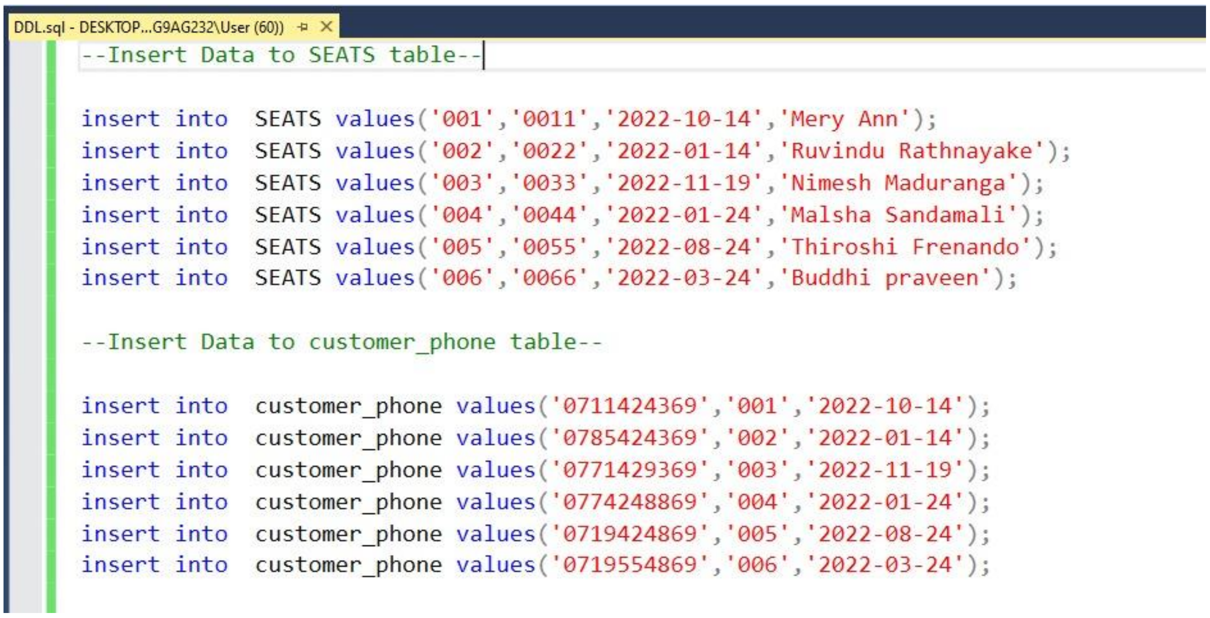
```sql
--Insert Data to AIRPLANE table--

insert into  AIRPLANE values('00001','AirBUS rew','150');
insert into  AIRPLANE values('00002','AirBUS','300');
insert into  AIRPLANE values('00003','AirBus dew','500');
insert into  AIRPLANE values('00004','McDonnell Douglas','900');
insert into  AIRPLANE values('00005','Boeing','1000');
insert into  AIRPLANE values('00006','Boeing dew','570');
insert into  AIRPLANE values('00007','Boeing volv','400');

--Insert Data to SEATS_TOTAL table--

insert into  SEATS_TOTAL values('00001','50');
insert into  SEATS_TOTAL values('00002','100');
insert into  SEATS_TOTAL values('00003','300');
insert into  SEATS_TOTAL values('00004','400');
insert into  SEATS_TOTAL values('00005','600');
insert into  SEATS_TOTAL values('00006','300');
insert into  SEATS_TOTAL values('00007','300');
```

- Flight_leg Table
- Leg_instance Table



```sql
--Insert Data to FLIGHT_LEG table--

insert into  FLIGHT_LEG values('01','RP987','A789555');
insert into  FLIGHT_LEG values('02','GM456','A123874');
insert into  FLIGHT_LEG values('03','PP937','A789869');
insert into  FLIGHT_LEG values('04','KL203','A567969');
insert into  FLIGHT_LEG values('05','NM912','A345223');
insert into  FLIGHT_LEG values('06','SL729','A567358');

--Insert Data to LEG_INSTANCE table--

insert into  LEG_INSTANCE values('2022-10-14','08:07:09','14:58:14','00001','A789555','01');
insert into  LEG_INSTANCE values('2022-01-14','00:14:15','07:13:17','00002','A123874','02');
insert into  LEG_INSTANCE values('2022-11-19','10:27:26','18:08:16','00003','A789869','03');
insert into  LEG_INSTANCE values('2022-01-24','12:48:17','00:05:05','00004','A567969','04');
insert into  LEG_INSTANCE values('2022-08-24','07:22:26','14:08:16','00005','A345223','05');
insert into  LEG_INSTANCE values('2022-03-24','17:48:17','04:05:05','00006','A567358','06');
```

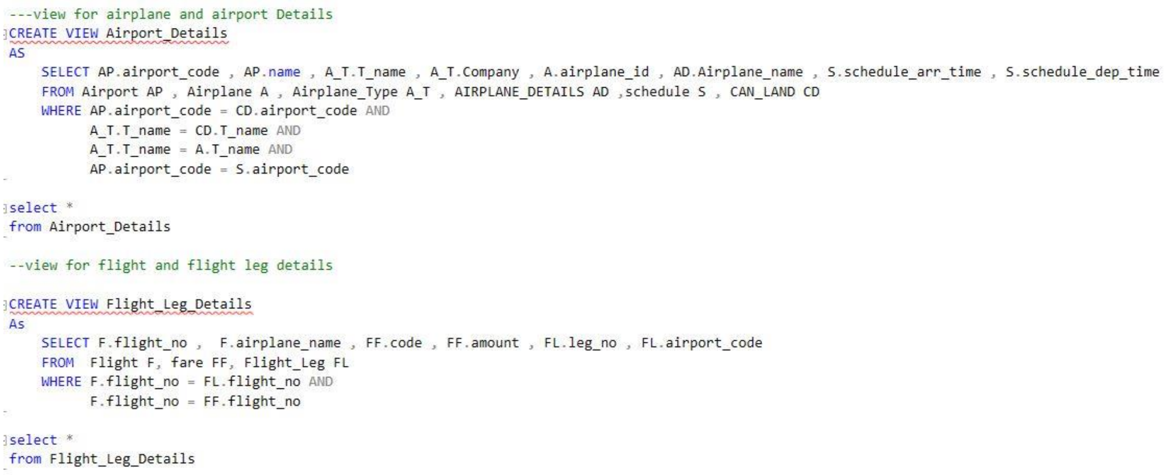
- Seats Table
- Customer_phone Table



DDL.sql - DESKTOP...G9AG232\User (60))

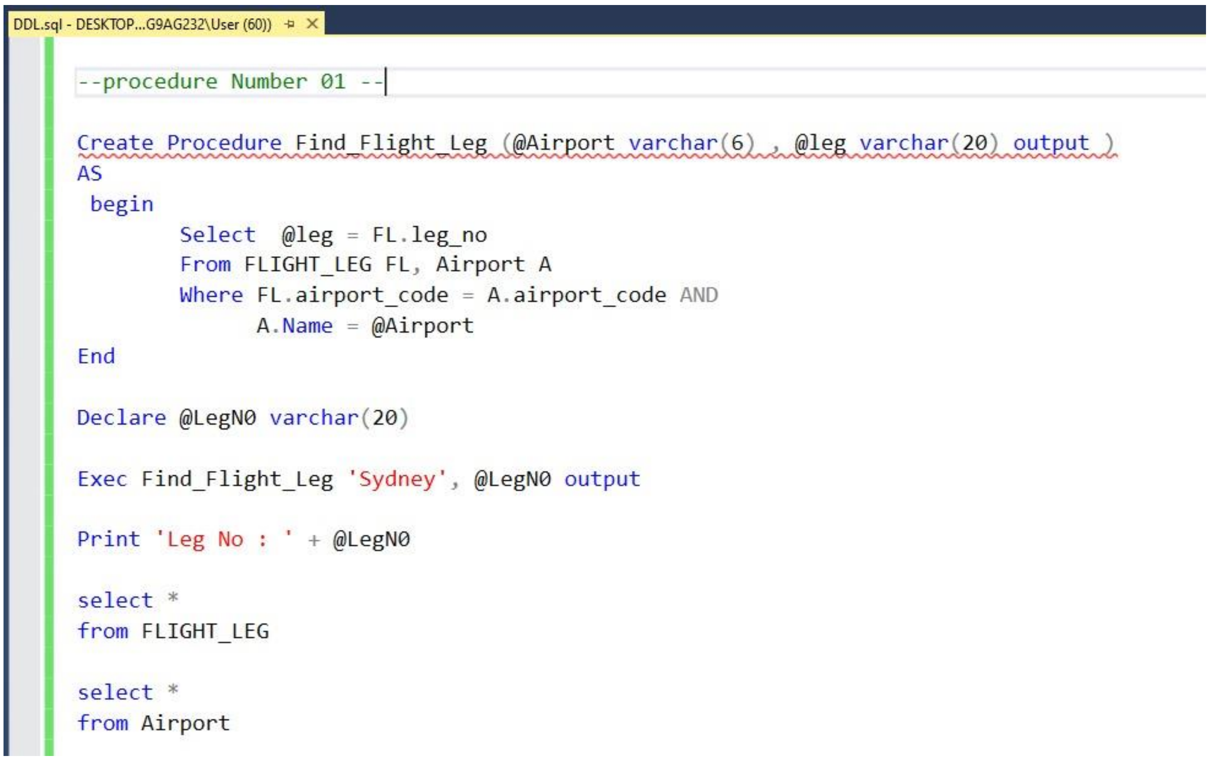
```sql
--Insert Data to SEATS table--

insert into  SEATS values('001','0011','2022-10-14','Mery Ann');
insert into  SEATS values('002','0022','2022-01-14','Ruvindu Rathnayake');
insert into  SEATS values('003','0033','2022-11-19','Nimesh Maduranga');
insert into  SEATS values('004','0044','2022-01-24','Malsha Sandamali');
insert into  SEATS values('005','0055','2022-08-24','Thiroshi Frenando');
insert into  SEATS values('006','0066','2022-03-24','Buddhi praveen');

--Insert Data to customer_phone table--

insert into  customer_phone values('0711424369','001','2022-10-14');
insert into  customer_phone values('0785424369','002','2022-01-14');
insert into  customer_phone values('0771429369','003','2022-11-19');
insert into  customer_phone values('0774248869','004','2022-01-24');
insert into  customer_phone values('0719424869','005','2022-08-24');
insert into  customer_phone values('0719554869','006','2022-03-24');
```

Views



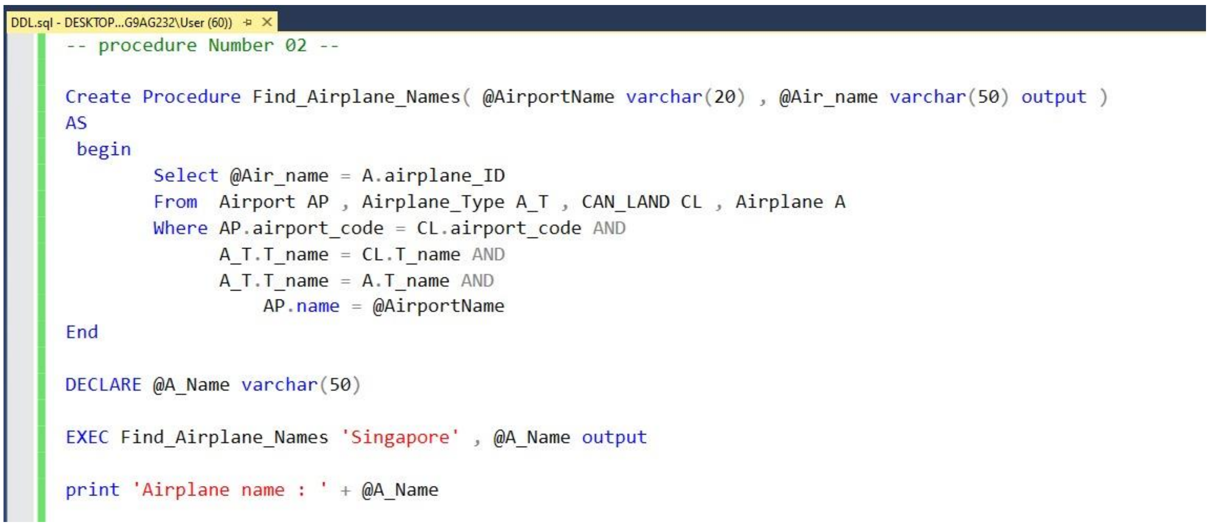
```sql
---view for airplane and airport Details
CREATE VIEW Airport_Details
AS
    SELECT AP.airport_code , AP.name , A_T.T_name , A_T.Company , A.airplane_id , AD.Airplane_name , S.schedule_arr_time , S.schedule_dep_time
    FROM Airport AP , Airplane A , Airplane_Type A_T , AIRPLANE_DETAILS AD ,schedule S , CAN_LAND CD
    WHERE AP.airport_code = CD.airport_code AND
          A_T.T_name = CD.T_name AND
          A_T.T_name = A.T_name AND
          AP.airport_code = S.airport_code

select *
from Airport_Details

--view for flight and flight leg details

CREATE VIEW Flight_Leg_Details
As
    SELECT F.flight_no ,  F.airplane_name , FF.code , FF.amount , FL.leg_no , FL.airport_code
    FROM  Flight F, fare FF, Flight_Leg FL
    WHERE F.flight_no = FL.flight_no AND
          F.flight_no = FF.flight_no

select *
from Flight_Leg_Details
```

Stored procedures



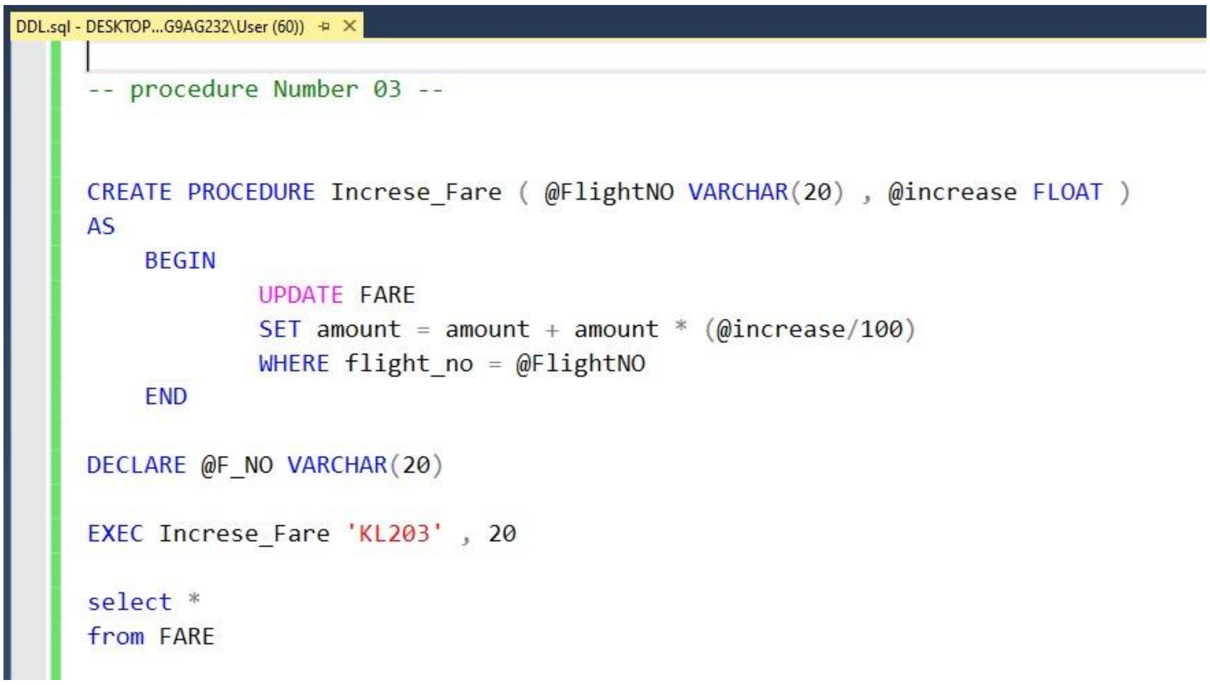
DDL.sql - DESKTOP...G9AG232\User (60))

```
--procedure Number 01 --

Create Procedure Find_Flight_Leg (@Airport varchar(6) , @leg varchar(20) output )
AS
 begin
        Select  @leg = FL.leg_no
        From FLIGHT_LEG FL, Airport A
        Where FL.airport_code = A.airport_code AND
              A.Name = @Airport
End

Declare @LegN0 varchar(20)

Exec Find_Flight_Leg 'Sydney', @LegN0 output

Print 'Leg No : ' + @LegN0

select *
from FLIGHT_LEG

select *
from Airport
```

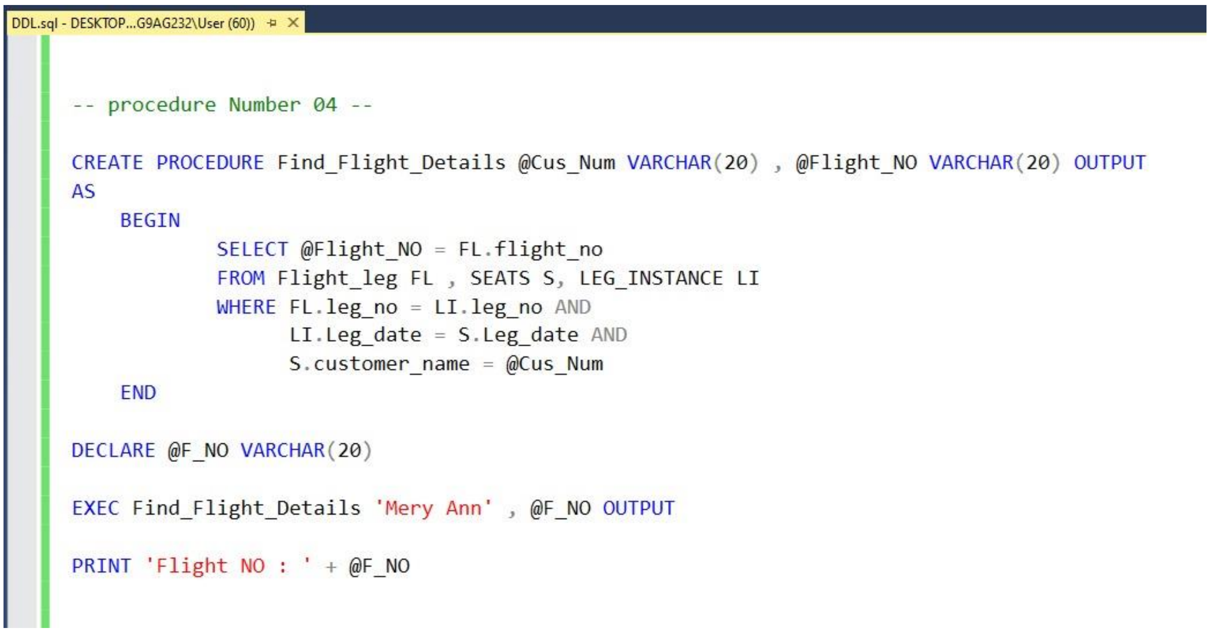


DDL.sql - DESKTOP...G9AG232\User (60))

```
-- procedure Number 02 --

Create Procedure Find_Airplane_Names( @AirportName varchar(20) , @Air_name varchar(50) output )
AS
 begin
        Select @Air_name = A.airplane_ID
        From  Airport AP , Airplane_Type A_T , CAN_LAND CL , Airplane A
        Where AP.airport_code = CL.airport_code AND
              A_T.T_name = CL.T_name AND
              A_T.T_name = A.T_name AND
                  AP.name = @AirportName
End

DECLARE @A_Name varchar(50)

EXEC Find_Airplane_Names 'Singapore' , @A_Name output

print 'Airplane name : ' + @A_Name
```

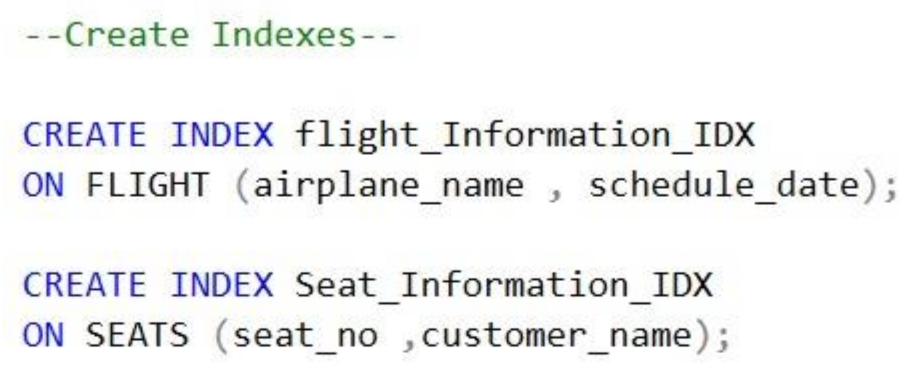


```
-- procedure Number 03 --


CREATE PROCEDURE Increse_Fare ( @FlightNO VARCHAR(20) , @increase FLOAT )
AS
    BEGIN
            UPDATE FARE
            SET amount = amount + amount * (@increase/100)
            WHERE flight_no = @FlightNO
    END

DECLARE @F_NO VARCHAR(20)

EXEC Increse_Fare 'KL203' , 20

select *
from FARE
```



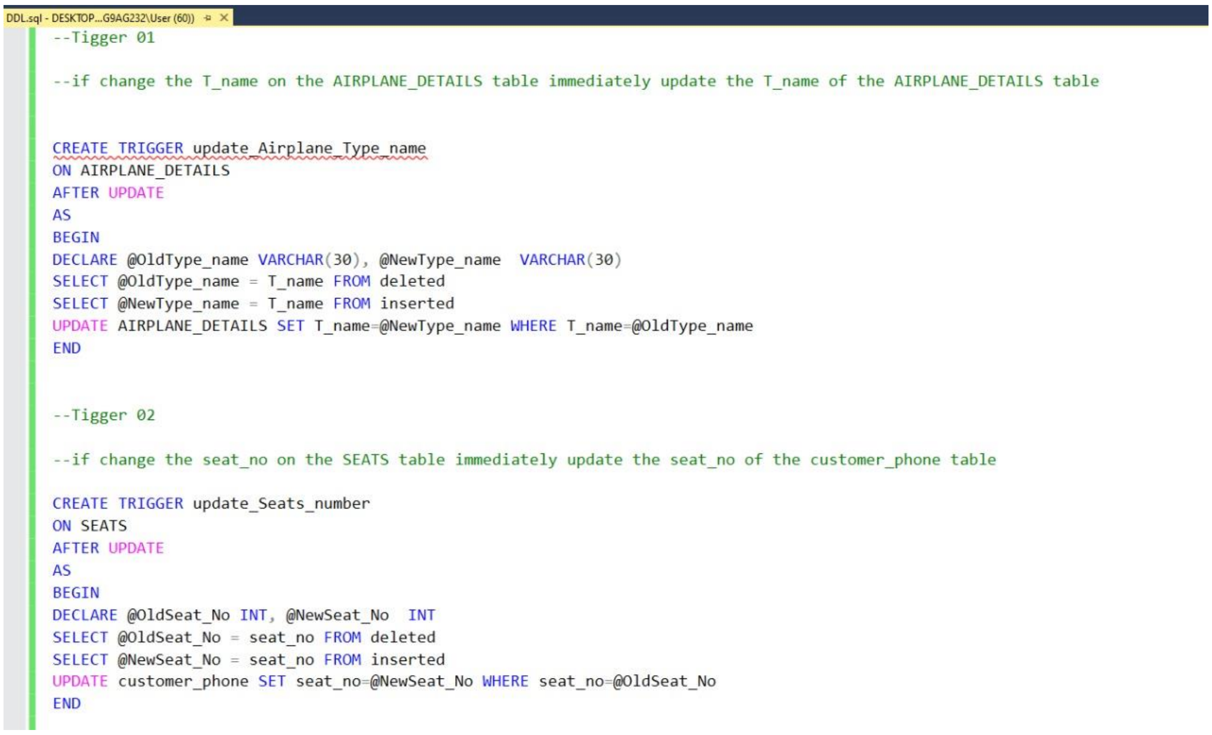
```
-- procedure Number 04 --

CREATE PROCEDURE Find_Flight_Details @Cus_Num VARCHAR(20) , @Flight_NO VARCHAR(20) OUTPUT
AS
    BEGIN
            SELECT @Flight_NO = FL.flight_no
            FROM Flight_leg FL , SEATS S, LEG_INSTANCE LI
            WHERE FL.leg_no = LI.leg_no AND
                  LI.Leg_date = S.Leg_date AND
                  S.customer_name = @Cus_Num
    END

DECLARE @F_NO VARCHAR(20)

EXEC Find_Flight_Details 'Mery Ann' , @F_NO OUTPUT

PRINT 'Flight NO : ' + @F_NO
```

Indexes



```
--Create Indexes--

CREATE INDEX flight_Information_IDX
ON FLIGHT (airplane_name , schedule_date);

CREATE INDEX Seat_Information_IDX
ON SEATS (seat_no ,customer_name);
```

Triggers



DDL.sql - DESKTOP...G9AG232\User (60))

```
--Tigger 01

--if change the T_name on the AIRPLANE_DETAILS table immediately update the T_name of the AIRPLANE_DETAILS table


CREATE TRIGGER update_Airplane_Type_name
ON AIRPLANE_DETAILS
AFTER UPDATE
AS
BEGIN
DECLARE @OldType_name VARCHAR(30), @NewType_name  VARCHAR(30)
SELECT @OldType_name = T_name FROM deleted
SELECT @NewType_name = T_name FROM inserted
UPDATE AIRPLANE_DETAILS SET T_name=@NewType_name WHERE T_name=@OldType_name
END


--Tigger 02

--if change the seat_no on the SEATS table immediately update the seat_no of the customer_phone table

CREATE TRIGGER update_Seats_number
ON SEATS
AFTER UPDATE
AS
BEGIN
DECLARE @OldSeat_No INT, @NewSeat_No  INT
SELECT @OldSeat_No = seat_no FROM deleted
SELECT @NewSeat_No = seat_no FROM inserted
UPDATE customer_phone SET seat_no=@NewSeat_No WHERE seat_no=@OldSeat_No
END
```

Script of SQL

/*CREATE Airport TABLES*/

CREATE TABLE Airport (

airport_code varchar(10) NOT NULL,

```
name varchar(100) NOT NULL,
city varchar(30) NOT NULL,
state varchar(30) NOT NULL,
CONSTRAINT Airport_PK PRIMARY KEY (airport_code),
);

/*CREATE seat_details TABLE*/

CREATE TABLE seat_details(
company varchar(30) NOT NULL,
Max_Seat int NOT NULL,
CONSTRAINT seat_details_PK PRIMARY KEY (company),
);

/*CREATE Airplane_Type TABLES*/

CREATE TABLE Airplane_Type (
T_name varchar (30) NOT NULL,
Company varchar (30) NOT NULL,
CONSTRAINT Airplane_Type_PK PRIMARY KEY (T_name),
CONSTRAINT Airplane_Type_FK FOREIGN KEY (Company) REFERENCES
seat_details(Company)
);

/*CREATE CAN_LAND TABLES*/
```

```
CREATE TABLE CAN_LAND (

airport_code varchar(10) NOT NULL,

T_name varchar (30) NOT NULL,

CONSTRAINT CAN_LAND_PK PRIMARY KEY (airport_code,T_name),

CONSTRAINT CAN_LAND_FK1 FOREIGN KEY (airport_code) REFERENCES Airport
(airport_code),

CONSTRAINT CAN_LAND_FK2 FOREIGN KEY (T_name) REFERENCES
Airplane_Type (T_name)

);


/*CREATE AIRPLANE TABLE*/


CREATE TABLE AIRPLANE(

airplane_ID int NOT NULL,

T_name varchar (30) NOT NULL,

total_no_of_seats int NOT NULL,

CONSTRAINT AIRPLANE_PK PRIMARY KEY (airplane_ID),

CONSTRAINT AIRPLANE_FK FOREIGN KEY (T_name) REFERENCES
Airplane_Type(T_name),

);


/*CREATE AIRPLANE_DETAILS TABLE*/


CREATE TABLE AIRPLANE_DETAILS(
Airplane_name varchar(10) NOT NULL,

T_name varchar (30) NOT NULL,

CONSTRAINT AIRPLANE_DETAILS_PK PRIMARY KEY (Airplane_name),

CONSTRAINT AIRPLANE_DETAILS_FK FOREIGN KEY (T_name) REFERENCES Airplane_Type(T_name),

);


/*CREATE SCHEDULE TABLE*/


CREATE TABLE SCHEDULE(

airport_code varchar(10) NOT NULL,

schedule_dep_time time,

schedule_arr_time time,

CONSTRAINT SCHEDULE_PK PRIMARY KEY (airport_code),

CONSTRAINT SCHEDULE_FK FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),

);



/*CREATE FLIGHT TABLE*/


CREATE TABLE FLIGHT(

flight_no varchar(12) NOT NULL,

airplane_name varchar(30) NOT NULL,

schedule_date date,
CONSTRAINT FLIGHT_PK PRIMARY KEY (flight_no),

);


/*CREATE FLIGHT_LEG TABLE*/


CREATE TABLE FLIGHT_LEG(

leg_no int NOT NULL,

flight_no varchar(12)NOT NULL,

airport_code varchar(10) NOT NULL,

CONSTRAINT FLIGHT_LEG_PK PRIMARY KEY (leg_no),

CONSTRAINT FLIGHT_LEG_FK1 FOREIGN KEY (flight_no) REFERENCES
FLIGHT(flight_no),

CONSTRAINT FLIGHT_LEG_FK2 FOREIGN KEY (airport_code) REFERENCES
Airport(airport_code),

);




/*CREATE FARE TABLE*/


CREATE TABLE FARE(

code int NOT NULL,

Restriction varchar (30) NOT NULL,

Amount int NOT NULL,
flight_no varchar(12) NOT NULL,
CONSTRAINT FARE_PK PRIMARY KEY (code),
CONSTRAINT  FARE_FK FOREIGN KEY (flight_no) REFERENCES FLIGHT(flight_no),
);

/*CREATE LEG_INSTANCE TABLE*/

CREATE TABLE LEG_INSTANCE(
Leg_date date NOT NULL,
Dep_time time,
Arr_time time,
airplane_ID int NOT NULL,
airport_code varchar(10) NOT NULL,
leg_no int NOT NULL,
CONSTRAINT LEG_INSTANCE_PK PRIMARY KEY (Leg_date),
CONSTRAINT LEG_INSTANCE_FK1 FOREIGN KEY (airplane_ID) REFERENCES AIRPLANE(airplane_ID),
CONSTRAINT LEG_INSTANCE_FK2 FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
CONSTRAINT LEG_INSTANCE_FK3 FOREIGN KEY ( leg_no) REFERENCES FLIGHT_LEG(leg_no),
);

/*CREATE SEATS_TOTAL TABLE*/
```

```
CREATE TABLE SEATS_TOTAL(

airplane_ID int NOT NULL,

no_of_avilable_seat int NOT NULL,

CONSTRAINT SEATS_TOTAL_PK PRIMARY KEY (airplane_ID),

CONSTRAINT SEATS_TOTAL_FK FOREIGN KEY (airplane_ID) REFERENCES
AIRPLANE(airplane_ID),

);



/*CREATE SEATS TABLE*/


CREATE TABLE SEATS(

seat_no int NOT NULL,

Cus_no int NOT NULL,

Leg_date date NOT NULL,

customer_name varchar(50) NOT NULL,

CONSTRAINT SEATS_PK PRIMARY KEY (seat_no),

CONSTRAINT SEATS_FK FOREIGN KEY (Leg_date) REFERENCES
LEG_INSTANCE(Leg_date),

);


CREATE TABLE customer_phone(

cus_phone int NOT NULL,

seat_no int NOT NULL,

Leg_date date NOT NULL,
CONSTRAINT customer_phone_FK1 FOREIGN KEY (seat_no) REFERENCES SEATS(seat_no),

CONSTRAINT customer_phone_FK2 FOREIGN KEY (Leg_date) REFERENCES LEG_INSTANCE(Leg_date)

);


--Insert Data to tables--


--Insert Data to Airport table--


insert into Airport values('A789555','Bandaranaike International Airport','katunayaka','colombo');

insert into Airport values('A123874','San Francisco International Airport','San Mateo','California');

insert into Airport values('A789869','Lake City International Airport',' Lake City','Utah');

insert into Airport values('A567969','Singapore','Ivalo','Tornio');

insert into Airport values('A345223','Brisbane Airport','Briss','Victoria');

insert into Airport values('A567358','Sydney','Frankfurt','Hesse');


--Insert Data to FLIGHT table--


insert into FLIGHT values('RP987','Air Berlin','2022-12-23');

insert into FLIGHT values('GM456','Belair','2022-10-19');

insert into FLIGHT values('PP937','Paramount','2022-11-04');

insert into FLIGHT values('KL203','Oman Air','2022-10-26');
insert into FLIGHT values('NM912','IndiGo','2023-01-04');

insert into FLIGHT values('SL729','Jetstar Asia','2022-10-23');

insert into FLIGHT values('IN914','Helvetic Airways','2022-12-01');


--Insert Data to FARE table--


insert into FARE values('1015','plastic','650000','RP987');

insert into FARE values('1017','bring metal','275000','GM456');

insert into FARE values('1018','bring pets and plants','74000','PP937');

insert into FARE values('1019','eat fish and meats','65000','KL203');

insert into FARE values('1020','bring over travel bags','54000','NM912');

insert into FARE values('1021','bring pets','41000','SL729');

insert into FARE values('1022','brings','41200','IN914');


--Insert Data to seat_details table--


insert into seat_details values('brave','1000');

insert into seat_details values('zigoz','700');

insert into seat_details values('braxme','550');

insert into seat_details values('volvo','550');

insert into seat_details values('benzb','100');

insert into seat_details values('braxe','250');

insert into seat_details values('braved','500');
```

```
--Insert Data to Airplane_Type table--


insert into Airplane_Type values('AirBUS rew','brave');

insert into Airplane_Type values('AirBUS','zigoz');

insert into Airplane_Type values('AirBus dew','braxme');

insert into Airplane_Type values('McDonnell Douglas','volvo');

insert into Airplane_Type values('Boeing','benzb');

insert into Airplane_Type values('Boeing dew','braxe');

insert into Airplane_Type values('Boeing volv','braved');


 --Insert Data to CAN_LAND table--


insert into CAN_LAND values('A789555','AirBUS rew');

insert into CAN_LAND values('A123874','AirBUS');

insert into CAN_LAND values('A789869','AirBus dew');

insert into CAN_LAND values('A567969','McDonnell Douglas');

insert into CAN_LAND values('A345223','Boeing');

insert into CAN_LAND values('A567358','Boeing dew');


--Insert Data to AIRPLANE_DETAILS table--


insert into  AIRPLANE_DETAILS values('benz','AirBUS rew');

insert into  AIRPLANE_DETAILS values('BMW X','AirBUS');

insert into  AIRPLANE_DETAILS values('VOLVO Z','AirBus dew');
insert into  AIRPLANE_DETAILS values('VOLVO ZX','McDonnell Douglas');

insert into  AIRPLANE_DETAILS values('MGRADE','Boeing');

insert into  AIRPLANE_DETAILS values('GGRADE','Boeing dew');

insert into  AIRPLANE_DETAILS values('Benz X','Boeing volv');


--Insert Data to SCHEDULE table--


insert into  SCHEDULE values('A789555','05:28:46','17:45:47');

insert into  SCHEDULE values('A123874','01:11:18','07:15:27');

insert into  SCHEDULE values('A789869','11:31:48','19:45:47');

insert into  SCHEDULE values('A567969','13:07:09','23:51:47');

insert into  SCHEDULE values('A345223','13:07:10','23:55:47');

insert into  SCHEDULE values('A567358','13:01:09','23:50:47');


--Insert Data to AIRPLANE table--


insert into  AIRPLANE values('00001','AirBUS rew','150');

insert into  AIRPLANE values('00002','AirBUS','300');

insert into  AIRPLANE values('00003','AirBus dew','500');

insert into  AIRPLANE values('00004','McDonnell Douglas','900');

insert into  AIRPLANE values('00005','Boeing','1000');

insert into  AIRPLANE values('00006','Boeing dew','570');

insert into  AIRPLANE values('00007','Boeing volv','400');
```

```
--Insert Data to SEATS_TOTAL table--

insert into  SEATS_TOTAL values('00001','50');
insert into  SEATS_TOTAL values('00002','100');
insert into  SEATS_TOTAL values('00003','300');
insert into  SEATS_TOTAL values('00004','400');
insert into  SEATS_TOTAL values('00005','600');
insert into  SEATS_TOTAL values('00006','300');
insert into  SEATS_TOTAL values('00007','300');

--Insert Data to FLIGHT_LEG table--

insert into  FLIGHT_LEG values('01','RP987','A789555');
insert into  FLIGHT_LEG values('02','GM456','A123874');
insert into  FLIGHT_LEG values('03','PP937','A789869');
insert into  FLIGHT_LEG values('04','KL203','A567969');
insert into  FLIGHT_LEG values('05','NM912','A345223');
insert into  FLIGHT_LEG values('06','SL729','A567358');

--Insert Data to LEG_INSTANCE table--

insert into LEG_INSTANCE values('2022-10-
14','08:07:09','14:58:14','00001','A789555','01');
insert into LEG_INSTANCE values('2022-01-
14','00:14:15','07:13:17','00002','A123874','02');
```

```
insert into LEG_INSTANCE values('2022-11-19','10:27:26','18:08:16','00003','A789869','03');

insert into LEG_INSTANCE values('2022-01-24','12:48:17','00:05:05','00004','A567969','04');

insert into LEG_INSTANCE values('2022-08-24','07:22:26','14:08:16','00005','A345223','05');

insert into LEG_INSTANCE values('2022-03-24','17:48:17','04:05:05','00006','A567358','06');


--Insert Data to SEATS table--


insert into  SEATS values('001','0011','2022-10-14','Mery Ann');

insert into  SEATS values('002','0022','2022-01-14','Ruvindu Rathnayake');

insert into  SEATS values('003','0033','2022-11-19','Nimesh Maduranga');

insert into  SEATS values('004','0044','2022-01-24','Malsha Sandamali');

insert into  SEATS values('005','0055','2022-08-24','Thiroshi Frenando');

insert into  SEATS values('006','0066','2022-03-24','Buddhi praveen');


--Insert Data to customer_phone table--


insert into  customer_phone values('0711424369','001','2022-10-14');

insert into  customer_phone values('0785424369','002','2022-01-14');

insert into  customer_phone values('0771429369','003','2022-11-19');

insert into  customer_phone values('0774248869','004','2022-01-24');

insert into  customer_phone values('0719424869','005','2022-08-24');
insert into  customer_phone values('0719554869','006','2022-03-24');



--procedure Number 01 --


Create Procedure Find_Flight_Leg (@Airport varchar(6) , @leg varchar(20) output )

AS

 begin

                Select  @leg = FL.leg_no

                From FLIGHT_LEG FL, Airport A

                Where FL.airport_code = A.airport_code AND

                        A.Name = @Airport

End


Declare @LegN0 varchar(20)


Exec Find_Flight_Leg 'Sydney', @LegN0 output


Print 'Leg No : ' + @LegN0


select *

from FLIGHT_LEG


select *
```

```
from Airport


-- procedure Number 02 --


Create Procedure Find_Airplane_Names( @AirportName varchar(20) , @Air_name varchar(50) output )
AS
 begin
                Select @Air_name = A.airplane_ID
                From  Airport AP , Airplane_Type A_T , CAN_LAND CL , Airplane A
                Where AP.airport_code = CL.airport_code AND
                    A_T.T_name = CL.T_name AND
                    A_T.T_name = A.T_name AND
                  AP.name = @AirportName
End


DECLARE @A_Name varchar(50)


EXEC Find_Airplane_Names 'Singapore' , @A_Name output


print 'Airplane name : ' + @A_Name


-- procedure Number 03 --
```

```
CREATE PROCEDURE Increse_Fare ( @FlightNO VARCHAR(20) , @increase FLOAT )

AS

        BEGIN

                        UPDATE FARE

                        SET amount = amount + amount * (@increase/100)

                        WHERE flight_no = @FlightNO

   END


DECLARE @F_NO VARCHAR(20)


EXEC Increse_Fare 'KL203' , 20


select *

from FARE



-- procedure Number 04 --


CREATE PROCEDURE Find_Flight_Details @Cus_Num VARCHAR(20) , @Flight_NO
VARCHAR(20) OUTPUT

AS

        BEGIN

                        SELECT @Flight_NO = FL.flight_no

                        FROM Flight_leg FL , SEATS S, LEG_INSTANCE LI

                        WHERE FL.leg_no = LI.leg_no AND
```

```
                                                    LI.Leg_date = S.Leg_date AND

                                                    S.customer_name = @Cus_Num

    END


DECLARE @F_NO VARCHAR(20)


EXEC Find_Flight_Details 'Mery Ann' , @F_NO OUTPUT


PRINT 'Flight NO : ' + @F_NO




--Create Indexes--


CREATE INDEX flight_Information_IDX

ON FLIGHT (airplane_name , schedule_date);


CREATE INDEX Seat_Information_IDX

ON SEATS (seat_no ,customer_name);


---view for airplane and airport Details

CREATE VIEW Airport_Details

AS

            SELECT AP.airport_code , AP.name , A_T.T_name , A_T.Company , A.airplane_id ,
AD.Airplane_name , S.schedule_arr_time , S.schedule_dep_time
```

```
        FROM Airport AP , Airplane A , Airplane_Type A_T , AIRPLANE_DETAILS AD
,schedule S , CAN_LAND CD

        WHERE AP.airport_code = CD.airport_code AND

                A_T.T_name = CD.T_name AND

                A_T.T_name = A.T_name AND

                AP.airport_code = S.airport_code


select *

from Airport_Details


--view for flight and flight leg details


CREATE VIEW Flight_Leg_Details

As

        SELECT F.flight_no ,  F.airplane_name , FF.code , FF.amount , FL.leg_no ,
FL.airport_code

        FROM  Flight F, fare FF, Flight_Leg FL

        WHERE F.flight_no = FL.flight_no AND

                F.flight_no = FF.flight_no


select *

from Flight_Leg_Details


--Tigger 01
```

--if change the T_name on the AIRPLANE_DETAILS table immediately update the T_name of the AIRPLANE_DETAILS table

```
CREATE TRIGGER update_Airplane_Type_name
ON AIRPLANE_DETAILS
AFTER UPDATE
AS
BEGIN
DECLARE @OldType_name VARCHAR(30), @NewType_name  VARCHAR(30)
SELECT @OldType_name = T_name FROM deleted
SELECT @NewType_name = T_name FROM inserted
UPDATE AIRPLANE_DETAILS SET T_name=@NewType_name WHERE T_name=@OldType_name
END
```

--Tigger 02

--if change the seat_no on the SEATS table immediately update the seat_no of the customer_phone table

```
CREATE TRIGGER update_Seats_number
ON SEATS
AFTER UPDATE
```

```
AS

BEGIN

DECLARE @OldSeat_No INT, @NewSeat_No  INT

SELECT @OldSeat_No = seat_no FROM deleted

SELECT @NewSeat_No = seat_no FROM inserted

UPDATE customer_phone SET seat_no=@NewSeat_No WHERE seat_no=@OldSeat_No

END
```

PART 2

One of the most valuable resources for any individual or an organization is their data. Maintaining the data and properly structuring it is a crucial responsibility. Datasets are maintained inside of databases to make the process productive and efficient to access, manage data and automate a broad range of tasks both inside and outside of the organization. Digital technology is expanding and becoming more sophisticated at a rapid phase. Cybercrime is consistently impacting a vast range of industries. There are approximately around 4000 cyberattacks every day. While database heavily depend on the internet, thus leading it to being extremely vulnerable. According to the 2015 Verizon Data Breach Investigations Report, databases are one of the most frequently breached assets [1]. Breaches might be caused by a variety of software flaws, configuration errors, patterns of exploitation, negligence and many other. The most common database vulnerabilities are SQL Injection and Buffer Overflow.

1. SQL Injections

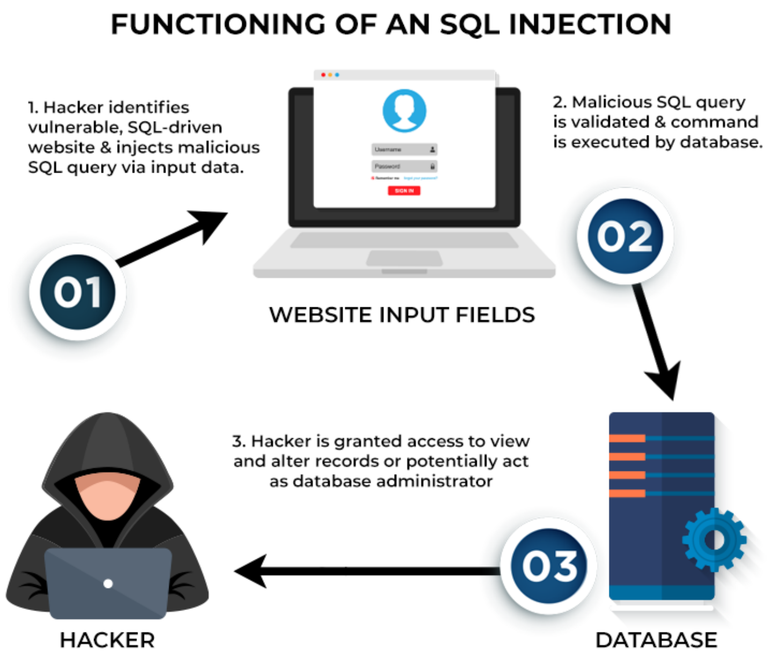
SQL injection, generally referred to as SQLI, is a popular attack method that use malicious SQL code to manipulate backend databases and access data that was not intended to be revealed. Several varieties of elements, such as private customer information, user lists, or sensitive corporate data, may be included in this data. A security vulnerability in an application's database layer is exploited by SQL injection.

Any website or online application that makes use of a SQL database, such as MySQL, Oracle, SQL Server, or another one, may be impacted by a SQL Injection. Attackers can get through application security safeguards by using SQL Injection. They have the ability to bypass the authentication and authorization processes of a webpage or web application and obtain the entire content from the SQL database. Additionally, they are able to add, edit, and delete records from the database utilizing SQL Injection.

There are several types of SQL Injection attacks namely; in-band SQLi (using database errors or UNION commands), blind SQLi, and out-of-band SQLi.

How SQL Injections work?

An SQL injection attack involves the insertion or injection of a structured SQL query that triggers the desired response using the application's input data from the client. The attacker needs the response in order to comprehend the database design and get access to the application's secured data. Once the attacker identifies the vulnerabilities, they attempt to find more information such as the database type, backend languages used, the structure or the syntax of the query. The execution of predefined SQL commands is impacted by these malicious SQL commands that are inserted into data-plane input.


FUNCTIONING OF AN SQL INJECTION
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
2. Malicious SQL query is validated & command is executed by database.
01
02
WEBSITE INPUT FIELDS
3. Hacker is granted access to view and alter records or potentially act as database administrator
03
HACKER
DATABASE

The Impact

A successful SQL injection would allow the attacker to discover other user’s login credential in the database and impersonate as them. In this way, they are able to carry out database administrative operations who would have the privilege to the entire database system which could cause several consequences to the organization. They could also retrieve the content of a file that is contained in the database management system. As the attacker has access to the database, it gives the ability to alter database data, for instance, an attacker may utilize SQL Injection in a banking application to change balances, cancel transactions, or move money to their own account.

Moreover, by utilizing SQL Injection the attacker is able to delete records from the database. Even if the database administrator creates backups, data deletion may reduce the availability of an application until the database is restored. Additionally, backups may not include the latest data. As a result, due to the loss of data it would cause disruptions to the organization’s system.

Furthermore, in some database servers it allows to use the database server to access the operating system. In certain cases, attackers may utilize the SQL Injection to issue commands to the operating system and then attack the internal network behind a firewall. If SQL injection is not prevented, the application might be severely compromised, endangering the security and integrity of data as well as the authentication and authorization functions of the application.

2. <u>Buffer Overflow Attack</u>

Buffer Overflow

Memory storage spaces termed buffers are used to keep data while it is being transmitted from one place to another. A buffer flow, also commonly referred to as buffer overrun, occurs when a process tries to write a large volume of data to the buffer and that overruns the storage capacity of the memory buffer. As a result, the application that is attempting to write to the buffer overwrites neighboring memory locations. Buffer overflow frequently occurs as a result of incorrect inputs or inadequate buffer space allocation. If the transaction overwrites executable code, the application may perform erratically, provide inaccurate results, make memory access issues, or program crashes. All sorts of software can be impacted by buffer overflows.

What is a Buffer Overflow Attack?

By overwriting an application's memory, attackers take advantage of buffer overflow vulnerabilities. By altering the program's execution path, this might trigger a reaction that corrupt files or reveal sensitive information. For instance, to access IT systems, a hacker can add more code and modify the application’s instructions. Attackers who are aware of a program's memory structure can maliciously enter data that the buffer is not intended to retain and replace with their own code by overwriting the regions containing executable code. Due to the lack of built-in protections against overwriting or accessing memory data, C and C++ are two languages that are particularly vulnerable to buffer overflow attacks.

In general, there are two types of buffer overflow attacks namely; stack-based buffer overflow and heap-based overflow.

- **Stack-based buffer overflow**

A software can run into a stack buffer overflow problem if it writes more data to a stack buffer than was permitted for it. Comparatively stack-based buffer overflow is the common attack that occurs.

- **Heap-based attacks**

A particular kind of buffer overflow attack that targets the heap is known as a heap overflow attack. When a buffer overflow occurs on a heap, the attacker attempts to modify certain aspects to suit their purposes by corrupting information in the heap. Heap attacks are frequently more difficult to execute than Stack-based attacks since the success of the attack frequently depends on more than merely the presence of an overflow; in many cases, the information on the heap must also be overwritten.

The Impact

An attacker has the ability to take control of, crash, or alter a process by exploiting a buffer overflow. When attackers exploit buffer overflow by overwriting the memory of a database and triggers a response, it could damage files or expose sensitive information. This would violate the confidentiality, integrity, and the availability of the organization's database.

Mitigations for the vulnerabilities

In order to avoid these database vulnerabilities there are certain precautionary methods that could be followed.

There are several effective ways to prevent SQLI attacks from taking place, as well as protecting against them.

1. **Validate User Inputs**

Verifying user input is the most typical initial step in avoiding SQL injection attacks. Determine the important SQL statements first, and then make a list of all legitimate SQL statements that are permitted, leaving the unauthorized ones in question. Sanitization and validation of user input on the client side should only be seen as a convenience that enhances the user experience. It could be helpful, for instance, to comment on a suggested username and say whether or not it complies with the requirements of the application.

Input validation and server-side sanitization ensure that user-supplied data is free of characters like single or double quotes that might alter a SQL query and cause it to return information that was not intended.

2. **Program Analysis and Techniques and Proxies**

Although writing safe SQL code ought to be your first concern, there are tools that can make the process easier. A database and its queries can be analyzed against a set of criteria by SQL-specific static analysis tools like SQL Code Guard to identify vulnerabilities. Other tools, such as SQLProb, function as a proxy between an application and its database, intercepting, examining, and discarding potentially harmful queries before they reach the database.

3. **Utilizing Firewalls**

To help filter out harmful data, organizations should consider about using a web application firewall 'WAF', which may be either software or appliance based. The best ones will have a thorough set of default rules and make it simple to create new ones as when required. Before a solution is identified, a WAF can be very helpful in providing some security protection against a specific new vulnerability.

Buffer overflows can be mitigated and prevented using a variety of techniques. They include providing secure coding training to software developers, enforcing secure coding standards, implementing safe buffer handling functions, conducting code reviews, statically analyzing source code, identifying buffer overflows in real time, and putting an end to exploits through the operating system. Furthermore, avoiding computer languages that are vulnerable to buffer overflow vulnerabilities is the simplest strategy to prevent them.

Additionally, organizations could follow the precautions mentioned below to avoid buffer overflow attacks

1. **Bounds Checking**

Buffer overflows can be prevented by the bounds checking included in libraries of abstract data types. Organizations could avoid utilizing common library methods like gets(), strcpy(), and strcat() as much as possible since they can lead to buffer overflows.

1. **Executable Space Protection**

Memory locations can be marked or designated as non-executable to prevent machine code from running there.

2. **Use Modern Operating Systems**

The vast majority of modern operating systems come with built-in runtime protection features, such as protection of the non-executable area against attacks and random address space location reordering of the primary data sections of a process. These built-in runtime defenses lessen the impact of buffer overflow attacks.

It could be challenging to identify buffer overflow vulnerabilities, particularly when the application is vast and complex. However, a robust buffer overflow defense may be constructed by using secure coding techniques, safe buffer handling methods, and suitable security features of the compiler and operating system. The regular scanning and discovery of these defects is a crucial step in preventing an exploit in addition to these preventive measures.

References

Roy Maurer. (2015). Top Database Security Threats and How to Mitigate Them. *SHRM*. Retrieved from https://www.shrm.org/resourcesandtools/hr-topics/risk-management/pages/top-database-security-threats.aspx