Sri Lanka Institute of Information Technology

BSc Honors in Information Technology Specializing in Cyber Security

IE2042- Database Management Systems for Security

June 2022

Group Assignment

Database Design, Implementation and Security

Group Members

DE ZOYSA A.S.	**IT21167096**
NILUPUL S.A.	**IT21167478**
MADURANGA D.B.W.N.	**IT21170270**
BANDARA K.M.N.M	**IT21163418**

Contents

Document any assumptions made

1. Flight_leg entity is a weak entity of the Flight entity
2. Leg_intance entity is a weak entity of the Flight_leg entity
3. Seat entity is a weak entity of the Leg_instance entity
4. Flight_fare entity is a weak entity of the Flight entity

5. flight_number can determine the schedule_arr_time and schedule_dep_time

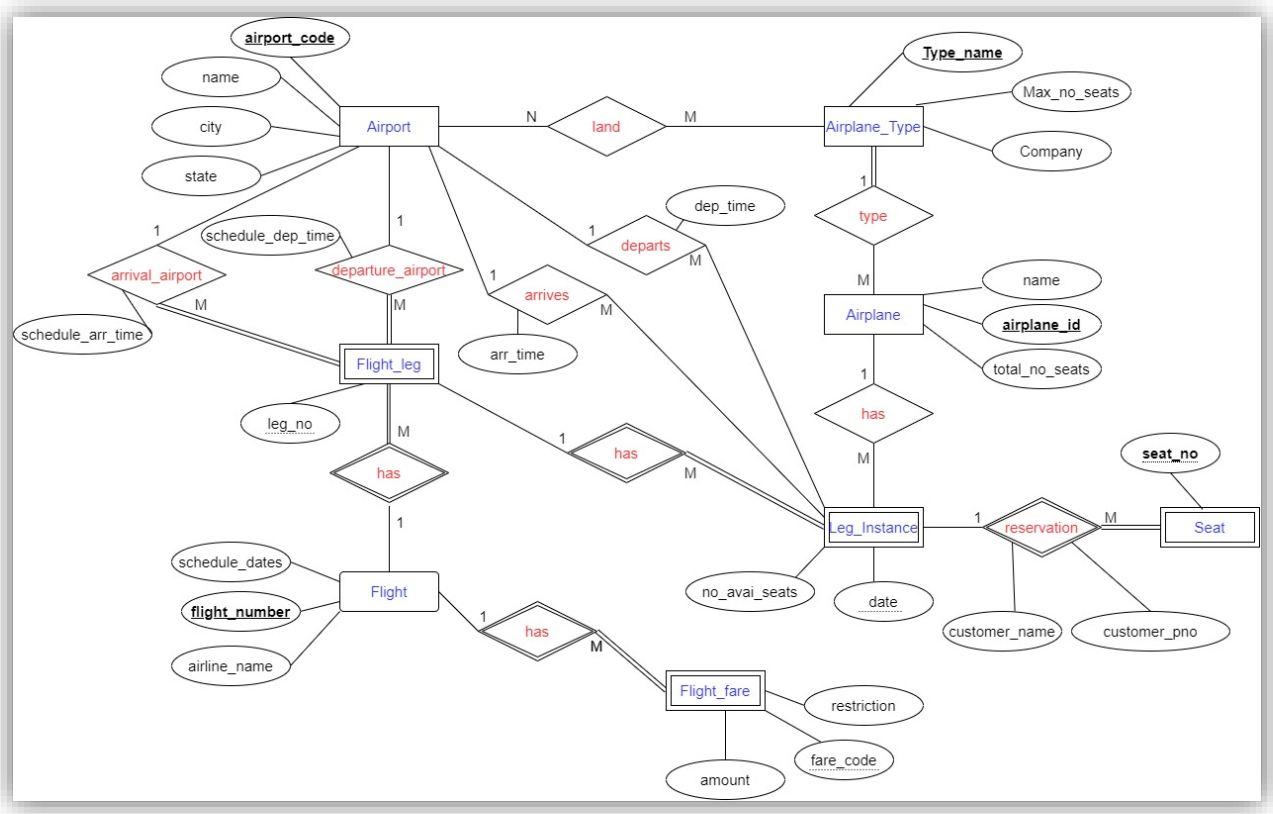
   Flight-number → schedule_arr_time , schedule_dep_time

6. airplane_id can determine the arr_time and dep_time

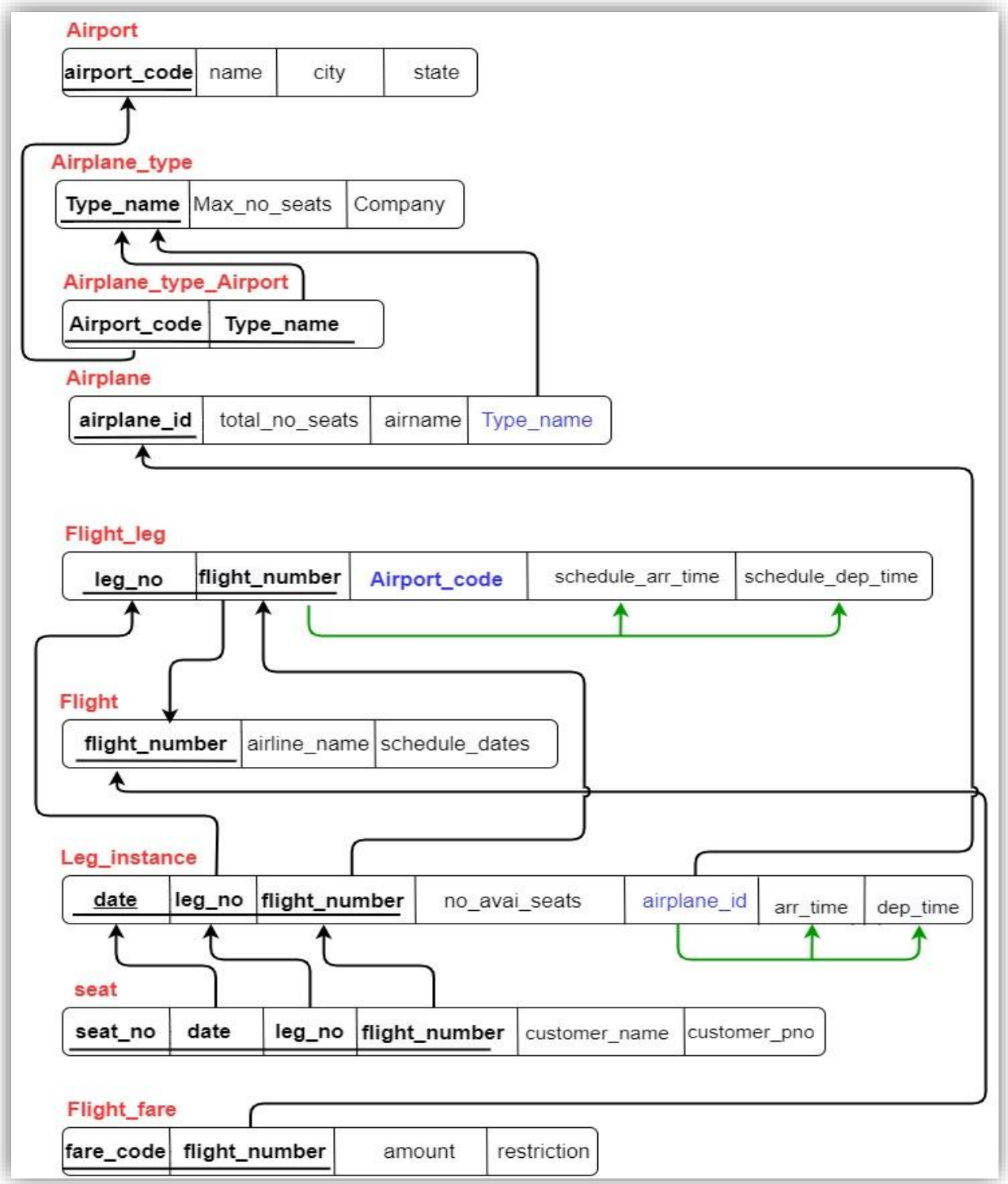
   airplane_id → arr_time , dep_time

Develop the ERD and logical model

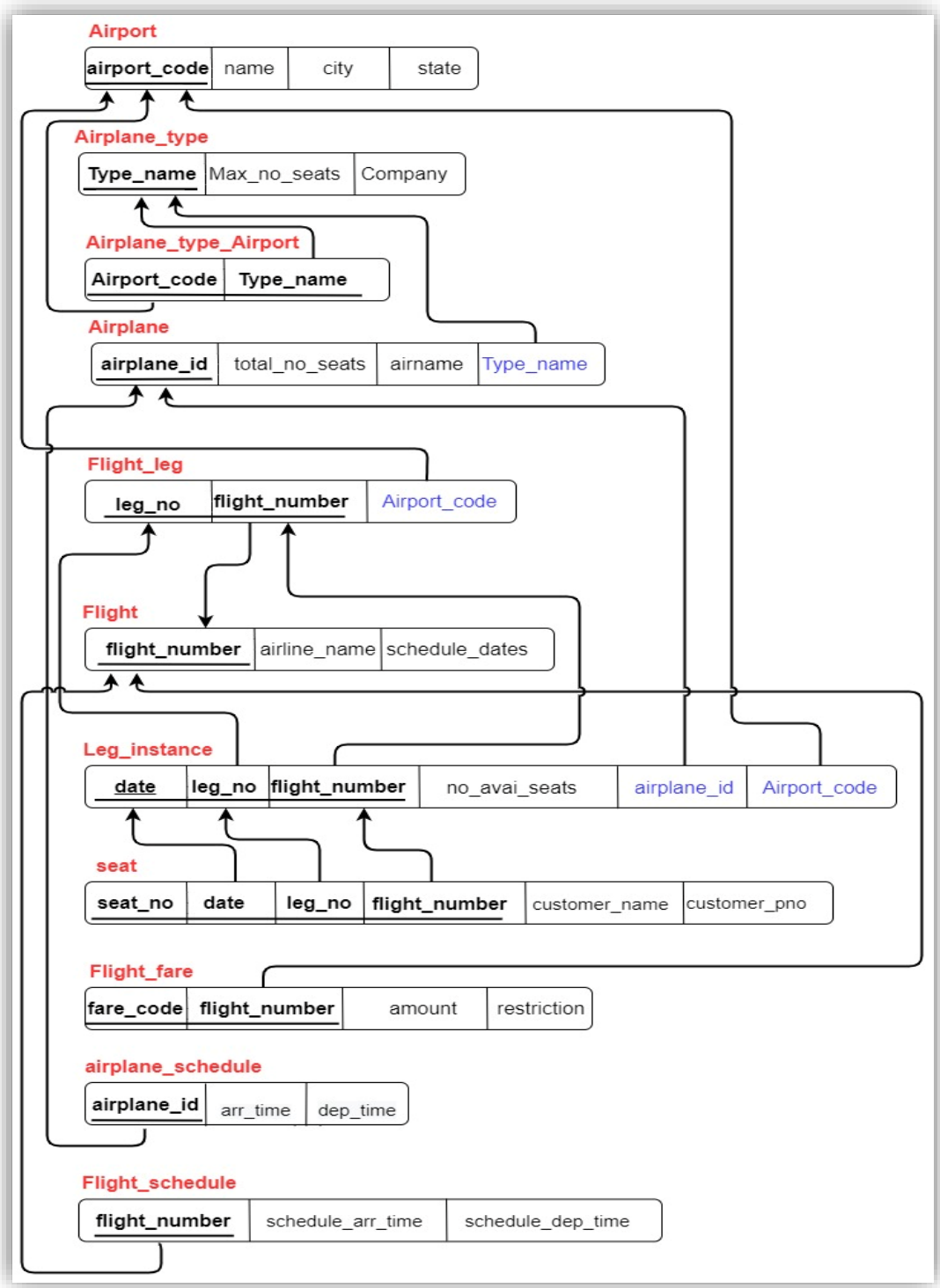
ER DIAGRAM



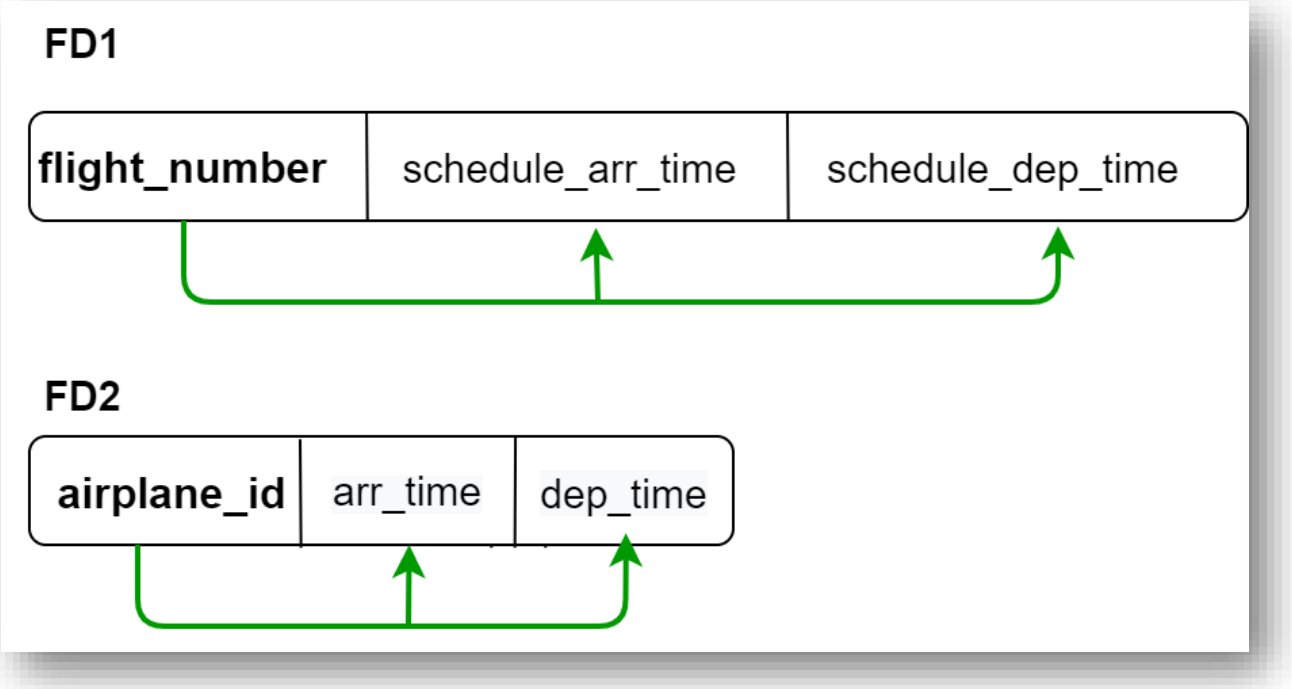
UNNORMALIZE RELATION SCHEMA



NORMALIZED RELATIONAL SCHEMA



FUNCTIONAL DEPENDENCIES



SQL Codes

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
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),
CONSTRAINT checkAirport_airport_code check (airport_code LIKE '[a/A][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
);


CREATE TABLE Airplane_Type (
Type_name varchar (30) NOT NULL,
max_no_seats int NOT NULL,
Company varchar (50) NOT NULL,
CONSTRAINT Airplane_Type_PK PRIMARY KEY (Type_name),
);

CREATE TABLE Airplane_Type_Airport (
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:39:57.6993583+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE Airplane_Type (
Type_name varchar (30) NOT NULL,
max_no_seats int NOT NULL,
Company varchar (50) NOT NULL,
CONSTRAINT Airplane_Type_PK PRIMARY KEY (Type_name),
);

CREATE TABLE Airplane_Type_Airport (
airport_code varchar(10) NOT NULL,
Type_name varchar (30) NOT NULL,
CONSTRAINT Airplane_Type_Airport_PK PRIMARY KEY (airport_code , Type_name ),
CONSTRAINT Airplane_Type_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),
CONSTRAINT Airport_FK FOREIGN KEY (airport_code) REFERENCES Airport(airport_code)
);


CREATE TABLE Airplane (
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:41:21.4644017+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE Airplane_Type_Airport (
airport_code varchar(10) NOT NULL,
Type_name varchar (30) NOT NULL,
CONSTRAINT Airplane_Type_Airport_PK PRIMARY KEY (airport_code , Type_name ),
CONSTRAINT Airplane_Type_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),
CONSTRAINT Airport_FK FOREIGN KEY (airport_code) REFERENCES Airport(airport_code)
);


CREATE TABLE Airplane (
airplane_id varchar (10) NOT NULL,
total_no_seats int NOT NULL,
airname varchar (50) NOT NULL,
Type_name varchar (30) NOT NULL,
CONSTRAINT Airplane_PK PRIMARY KEY (airplane_id),
CONSTRAINT Airplane_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),
CONSTRAINT checkAirplane_airplane_id check (airplane_id LIKE '[A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:42:29.2263954+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE Airplane (
airplane_id varchar (10) NOT NULL,
total_no_seats int NOT NULL,
airname varchar (50) NOT NULL,
Type_name varchar (30) NOT NULL,
CONSTRAINT Airplane_PK PRIMARY KEY (airplane_id),
CONSTRAINT Airplane_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),
CONSTRAINT checkAirplane_airplane_id check (airplane_id LIKE '[A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
);


CREATE TABLE Flight (
flight_number varchar (5) NOT NULL,
airline_name varchar (50) NOT NULL,
schedule_date date NOT NULL,
CONSTRAINT Flight_PK PRIMARY KEY (flight_number),
CONSTRAINT checkFlight_flight_number check (flight_number LIKE '[A-Z][A-Z][0-9][0-9][0-9]')
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:43:05.1916986+05:30
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql


CREATE TABLE Flight (
flight_number varchar (5) NOT NULL,
airline_name varchar (50) NOT NULL,
schedule_date date NOT NULL,
CONSTRAINT Flight_PK PRIMARY KEY (flight_number),
CONSTRAINT checkFlight_flight_number check (flight_number LIKE '[A-Z][A-Z][0-9][0-9][0-9]')
);

```

119 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-24T23:44:26.7054461+05:30
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql
CREATE TABLE Flight_leg (
leg_no int NOT NULL,
flight_number varchar (5) NOT NULL,
airport_code varchar (10) NOT NULL,
CONSTRAINT Flight_leg_PK PRIMARY KEY(leg_no,flight_number),
CONSTRAINT Flight_leg_FK FOREIGN KEY (flight_number) REFERENCES Flight(flight_number),
CONSTRAINT Flight_leg_FK1 FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
);
```

119 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-24T23:44:26.7054461+05:30
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql
CREATE TABLE Leg_instance (
date date NOT NULL,
leg_no int NOT NULL,
flight_number varchar (5) NOT NULL,
airport_code varchar (10) NOT NULL,
no_avil_seats int NOT NULL,
airplane_id varchar (10),
CONSTRAINT Leg_instance_PK PRIMARY KEY (date,leg_no,flight_number),
CONSTRAINT Leg_instance_FK FOREIGN KEY (leg_no,flight_number) REFERENCES Flight_leg(leg_no,flight_number),
CONSTRAINT Leg_instance_FK1 FOREIGN KEY (airplane_id) REFERENCES Airplane(airplane_id),
CONSTRAINT Leg_instance_FK2 FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
);
```

119 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-24T23:44:26.7054461+05:30
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql
CREATE TABLE Seat (
seat_no VARCHAR(10) DEFAULT 'Seat' NOT NULL,
date date NOT NULL,
leg_no int NOT NULL,
flight_number varchar (5) NOT NULL,
customer_name varchar (50) ,
customer_pno varchar (10) ,
CONSTRAINT Seat_PK PRIMARY KEY (seat_no,date,leg_no,flight_number),
CONSTRAINT Seat_FK FOREIGN KEY (date,leg_no,flight_number) REFERENCES Leg_instance(date,leg_no,flight_number),
CONSTRAINT check_Seat_customer_pno check (customer_pno LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
);
```

119 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-24T23:44:26.7054461+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE Flight_fare (
fare_code varchar  (10) NOT NULL,
flight_number varchar (5) NOT NULL,
amount  money NOT NULL,
restriction varchar (100) NOT NULL,
CONSTRAINT Flight_fare_PK PRIMARY KEY (fare_code,flight_number),
CONSTRAINT Flight_fare_FK FOREIGN KEY (flight_number) REFERENCES Flight(flight_number),
);
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:50:11.8529015+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE airplane_schedule (
airplane_id varchar (10),
arr_time time(0) NOT NULL,
dep_time time(0) NOT NULL,
CONSTRAINT airplane_schedule_PK PRIMARY KEY (airplane_id),
CONSTRAINT airplane_schedule_FK FOREIGN KEY (airplane_id) REFERENCES Airplane(airplane_id),
);

CREATE TABLE Flight_schedule (
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:50:49.7434981+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
CREATE TABLE Flight_schedule (
flight_number varchar (5) NOT NULL,
schedule_arr_time time(0) NOT NULL,
schedule_dep_time time(0) NOT NULL,
CONSTRAINT Flight_schedule_PK PRIMARY KEY (flight_number),
CONSTRAINT Flight_schedule_FK FOREIGN KEY (flight_number) REFERENCES Flight(flight_number),
);

select *
from Airport
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T21:52:14.4211050+05:30
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
select *
from Airport

/*insert airport table*/
insert into Airport values('A020171223','San Francisco International Airport','San Mateo','California');
insert into Airport values('A020221223','Sydney','Frankfurt','Hesse');
insert into Airport values('A020271126','Salt Lake City International Airport','Salt Lake City','Utah');
insert into Airport values('a020251231','Singapore','Ivalo','Kemi-Tornio');
insert into Airport values('a020230215','Brisbane International Airport','Brisbane','Victoria');
insert into Airport values('a171223200','Bandaranaike International Airport','katunayaka','Colombo');


select*
from Airplane_Type

/*insert Airplane_Type*/
insert into Airplane_Type values('Boeing 787','330',' Rolls Royce');
insert into Airplane_Type values('Airbus A330','440','Pratt & Whitney PW4000');
insert into Airplane_Type values('Boeing 747','467','Pratt & Whitney');
insert into Airplane_Type values('Boeing 757','295','Miami-based Eastern Airlines');
insert into Airplane_Type values('McDonnell Douglas MD-80 Series','135','Pratt & Whitney JT8-D');
insert into Airplane Type values('Antonov AN28','174','Czechoslovakia LET Aircraft Industries');
```

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
select*
from Airplane_Type_Airport

/*insert Airplane_Type_Airplan*/
insert into Airplane_Type_Airport values('A020171223','Boeing 787');
insert into Airplane_Type_Airport values('a020251231','Airbus A330');
insert into Airplane_Type_Airport values('a171223200','Boeing 747');
insert into Airplane_Type_Airport values('A020221223','Boeing 757');
insert into Airplane_Type_Airport values('A020271126','McDonnell Douglas MD-80 Series');
insert into Airplane Type Airport values('a171223200','Antonov AN28');


select*
from Airplane

/*insert Airplane*/
insert into Airplane values('R200171223','295','Douglas DC-3','Boeing 757');
insert into Airplane values('N987654321','467','Airbus A321XLR ','Boeing 747');
insert into Airplane values('Q224466889','440','COMAC C919','Airbus A330');
insert into Airplane values('D779955331','174','Universal Hydrogen ATR 72','Antonov AN28');
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql

select*
from Flight

/*insert Flight*/
insert into Flight values('AL987','Air Berlin','2022-12-23');
insert into Flight values('TY456','Belair','2022-10-19');
insert into Flight values('RE937','Paramount','2022-11-04');
insert into Flight values('KL203','Oman Air','2022-10-26');
insert into Flight values('BH912','IndiGo','2023-01-02');
insert into Flight values('LK729','Jetstar Asia','2022-10-23');
insert into Flight values('PR914','Helvetic Airways','2022-12-01');


select*
from Flight_leg

/*insert Flight_leg*/
insert into Flight_leg values(2,'AL987','A020171223');
insert into Flight_leg values(3,'PR914','a171223200');
insert into Flight_leg values(4,'AL987','A020271126');
insert into Flight_leg values(1,'KL203','a020230215');
insert into Flight_leg values(2,'LK729','A020221223');
insert into Flight_leg values(3,'BH912','a020251231');
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql

select*
from airplane_schedule

/*insert airplane_schedule*/
insert into airplane_schedule values('D779955331','09:28:48','15:45:47');
insert into airplane_schedule values('N987654321','01:11:18','08:15:27');
insert into airplane_schedule values('R200171223','11:31:48','19:45:47');
insert into airplane_schedule values('Q224466889','13:07:09','23:55:47');

select*
from Flight_schedule

/*insert Flight_shedule*/
insert into Flight_schedule values('AL987','08:07:09','14:58:14');
insert into Flight_schedule values('TY456','00:14:15','07:13:17');
insert into Flight_schedule values('LK729','10:27:26','18:08:16');
insert into Flight_schedule values('KL203','12:48:17','00:05:05');
insert into Flight_schedule values('BH912','07:22:26','14:08:16');
insert into Flight_schedule values('PR914','17:48:17','04:05:05');
```

SQLQuery1.sql - D...CTN4K\Nimesh (52))*

```sql

/*insert Flight_fare*/
insert into Flight_fare values('AF1015','AL987','650000','plastic');
insert into Flight_fare values('CDIJ17','TY456','275000','bring metal');
insert into Flight_fare values('GNQS03','RE937','74000','bring pets and plants');
insert into Flight_fare values('PW4589','KL203','65000','eat fish and meats');
insert into Flight_fare values('YBHKL1','PR914','54000','bring over 50KG travel bags');
insert into Flight_fare values('NQS144','BH912','41000','bring pets');
insert into Flight_fare values('NOQS51','LK729','41265','bring gold,spicies');


/*insert Leg_instance*/
insert into Leg_instance values('2022-11-14','2','AL987','A020171223','74','R200171223');
insert into Leg_instance values('2022-12-23','3','BH912','a171223200','52','N987654321');
insert into Leg_instance values('2023-01-01','1','KL203','A020271126','158','D779955331');
insert into Leg_instance values('2022-12-14','4','AL987','a020251231','74','Q224466889');
```

```sql
/*insert seat*/
insert into seat values('11F','2022-11-14','2','AL987','Mary Ann','0617651449');
insert into seat values('55JH','2022-12-23','3','BH912','Kelli Joana','0358895642');
insert into seat values('14LA','2023-01-01','1','KL203','Lulifer Konandoil','0662066887');
insert into seat values('12HBA','2022-12-14','4','AL987','Shivangi Munasinghe','9476966515');
```

Views

1. View for Airplane and Airport Details

```sql
---view for airplane and airport Details
CREATE VIEW Airplane_Details
AS
    SELECT AP.airport_code , AP.name , A_T.Type_name , A_T.Company , A.airplane_id , A.airname , A_S.arr_time , A_S.dep_time
    FROM Airport AP , Airplane A , Airplane_Type A_T , airplane_schedule A_S , Airplane_Type_Airport A_T_A
    WHERE AP.airport_code = A_T_A.airport_code AND
          A_T.Type_name = A_T_A.Type_name AND
          A_T.Type_name = A.Type_name AND
          A.airplane_id = A_S.airplane_id;


select *
from Airplane_Details

--view for flight and flight leg details

CREATE VIEW Flight_Details
```

```
Commands completed successfully.

Completion time: 2022-10-23T22:29:41.9031515+05:30
```

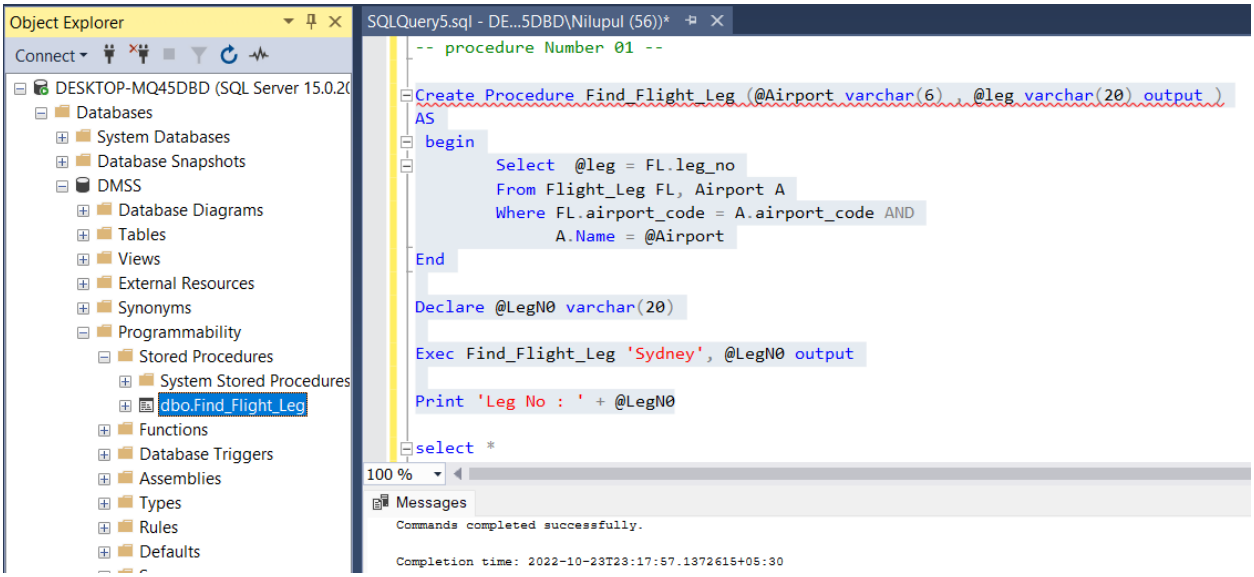
2. View for the Flight and Flight leg details

```sql
--view for flight and flight leg details

CREATE VIEW Flight_Details
As
    SELECT F.flight_number ,  F.airline_name , FF.fare_code , FF.amount , FL.leg_no , FL.airport_code
    FROM  Flight F, Flight_fare FF, Flight_Leg FL
    WHERE F.flight_number = FL.flight_number AND
          F.flight_number = FF.flight_number;

select *
from Flight_Details
select *
from Airplane_Details



--- Procedures --
```

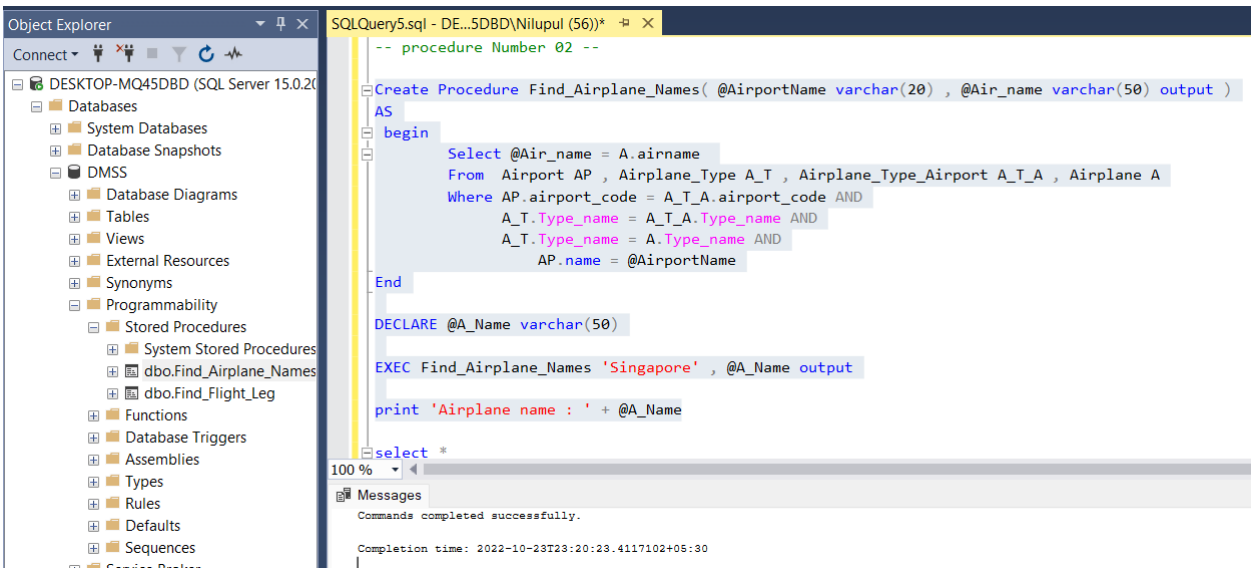
```
Commands completed successfully.

Completion time: 2022-10-23T23:12:49.3306885+05:30
```

Procedures

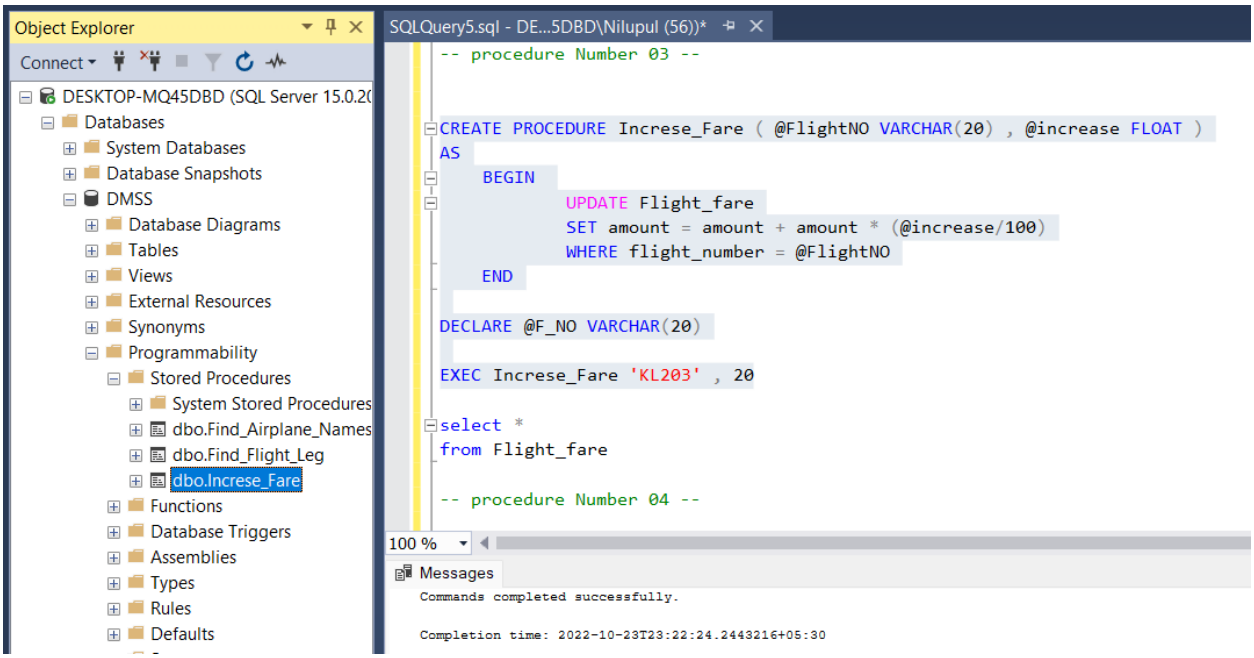
1. Procedure Number 01



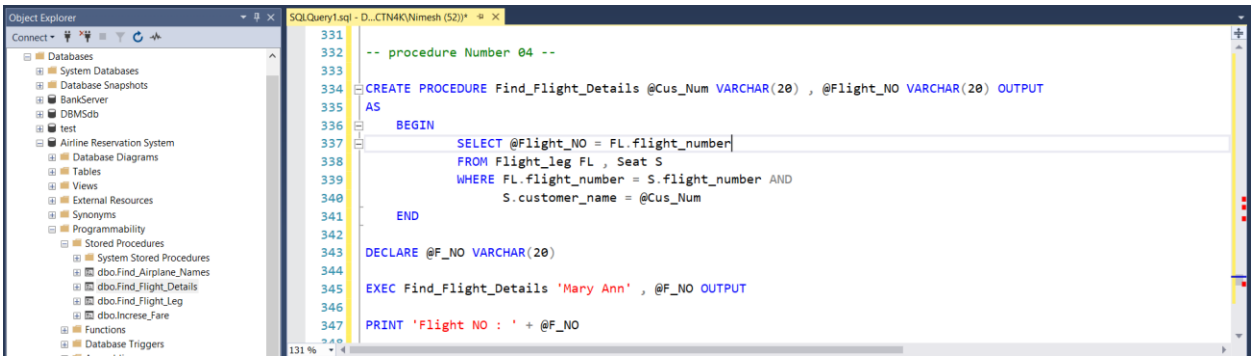
2. Procedure Number 02



3. Procedure Number 03



4. Procedure Number 04



Indexes

1. Index 01

Create index in seat table for customer information

```
SQLQuery5.sql - DE...5DBD\Nilupul (56))*

--create index in seat table for customer information
--index 01

CREATE INDEX Cus_Information_IDX
ON Seat (customer_name , customer_pno);

--create index in Flight_schedule table for Schedule_details
--index 02

100 %

Messages
Commands completed successfully.

Completion time: 2022-10-23T23:26:37.8875285+05:30
```

2. Index 02

Create index in Flight_schedule table for Schedule_details

```
SQLQuery5.sql - DE...5DBD\Nilupul (56))*

--create index in Flight_schedule table for Schedule_details
--index 02

CREATE INDEX Flight_details_IDX
ON Flight_schedule (schedule_arr_time , schedule_dep_time);

--Tigger 01

100 %

Messages
Commands completed successfully.

Completion time: 2022-10-23T23:29:30.7707989+05:30
```

Triggers

1. Trigger 01

   If change the flight_number on the flight table immediately update the flight_number of the Flight_schedule table

SQLQuery5.sql - DE...5DBD\Nilupul (56))*

```sql
--Tigger 01

--if change the flight flight_number on the flight table immediately update the flight_number of the Flight_schedule table

CREATE TRIGGER update_Flight_Number
ON Flight
AFTER UPDATE
AS
BEGIN
DECLARE @OldFlight_Number INT, @NewFlight_Number  INT
SELECT @OldFlight_Number = flight_number FROM deleted
SELECT @NewFlight_Number = flight_number FROM inserted
UPDATE Flight_schedule SET flight_number=@NewFlight_Number WHERE flight_number=@OldFlight_Number
END

--Tigger 02
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-10-23T23:32:38.7446236+05:30
```

2. Trigger 02

   When you enter date, leg no and flight_number values for the leg_instance table, will be automatically inserted into the seat table column as well

Object Explorer

Connect

DESKTOP-OKCTN4K\SQLEXPRESS (SQL Server 15.0.20

Databases
System Databases
Database Snapshots
Airline Reservation System
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Airplane
dbo.airplane_schedule
dbo.Airplane_Type
dbo.Airplane_Type_Airport
dbo.Airport
dbo.Flight
dbo.Flight_fare
dbo.Flight_leg
dbo.Flight_schedule
dbo.Leg_instance
Columns
Keys
Constraints
Triggers
Update_Seat_Details
Indexes

SQLQuery5.sql - D...CTN4K\Nimesh (67))* | SQLQuery4.sql - D...CTN4K\Nimesh (66)) | SQLQuery3.sql - D...CTN4K\Nimesh (56)) | SQLQuery1.sql - D...CTN4K\Nimesh (58))*

```sql
--Tigger 02

--when you enter  date, leg no and flight_number code  values for the leg_instance table, will be automatically ins

CREATE TRIGGER Update_Seat_Details
ON Leg_instance
FOR INSERT
AS
BEGIN
DECLARE @S_date DATE , @L_No INT , @A_Code VARCHAR(10)

SELECT @S_date = date , @L_No = leg_no , @A_Code = flight_number FROM inserted

INSERT INTO Seat (date,leg_no,flight_number) values (@S_date,@L_No,@A_Code)

END
```

SQL Code

```
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),
CONSTRAINT checkAirport_airport_code check (airport_code LIKE '[a/A][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
);


CREATE TABLE Airplane_Type (
Type_name varchar (30) NOT NULL,
max_no_seats int NOT NULL,
Company varchar (50) NOT NULL,
CONSTRAINT Airplane_Type_PK PRIMARY KEY (Type_name),

);

CREATE TABLE Airplane_Type_Airport (
airport_code varchar(10) NOT NULL,
Type_name varchar (30) NOT NULL,
CONSTRAINT Airplane_Type_Airport_PK PRIMARY KEY (airport_code , Type_name ),
```

```
CONSTRAINT Airplane_Type_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),

CONSTRAINT Airport_FK FOREIGN KEY (airport_code) REFERENCES Airport(airport_code)

);



CREATE TABLE Airplane (

airplane_id varchar (10) NOT NULL,

total_no_seats int NOT NULL,

airname varchar (50) NOT NULL,

Type_name varchar (30) NOT NULL,

CONSTRAINT Airplane_PK PRIMARY KEY (airplane_id),

CONSTRAINT Airplane_FK FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),

CONSTRAINT checkAirplane_airplane_id check (airplane_id LIKE '[A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

);



CREATE TABLE Flight (

flight_number varchar (5) NOT NULL,

airline_name varchar (50) NOT NULL,

schedule_date date NOT NULL,

CONSTRAINT Flight_PK PRIMARY KEY (flight_number),

CONSTRAINT checkFlight_flight_number check (flight_number LIKE '[A-Z][A-Z][0-9][0-9][0-9]')
```

```
);

CREATE TABLE Flight_leg (
leg_no int NOT NULL,
flight_number varchar (5) NOT NULL,
airport_code varchar (10) NOT NULL,
CONSTRAINT Flight_leg_PK PRIMARY KEY(leg_no,flight_number),
CONSTRAINT Flight_leg_FK FOREIGN KEY (flight_number) REFERENCES
Flight(flight_number),
CONSTRAINT Flight_leg_FK1 FOREIGN KEY (airport_code) REFERENCES
Airport(airport_code),
);

CREATE TABLE Leg_instance (
date date NOT NULL,
leg_no int NOT NULL,
flight_number varchar (5) NOT NULL,
airport_code varchar (10) NOT NULL,
no_avil_seats int NOT NULL,
airplane_id varchar (10),
CONSTRAINT Leg_instance_PK PRIMARY KEY (date,leg_no,flight_number),
CONSTRAINT Leg_instance_FK FOREIGN KEY (leg_no,flight_number)
REFERENCES Flight_leg(leg_no,flight_number),
```

```
CONSTRAINT Leg_instance_FK1 FOREIGN KEY (airplane_id) REFERENCES Airplane(airplane_id),

CONSTRAINT Leg_instance_FK2 FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),

);



CREATE TABLE Seat (

seat_no VARCHAR(10) DEFAULT 'Seat' NOT NULL,

date date NOT NULL,

leg_no int NOT NULL,

flight_number varchar (5) NOT NULL,

customer_name varchar (50) ,

customer_pno varchar (10) ,

CONSTRAINT Seat_PK PRIMARY KEY (seat_no,date,leg_no,flight_number),

CONSTRAINT Seat_FK FOREIGN KEY (date,leg_no,flight_number) REFERENCES Leg_instance(date,leg_no,flight_number),

CONSTRAINT check_Seat_customer_pno check (customer_pno LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

);



CREATE TABLE Flight_fare (

fare_code varchar  (10) NOT NULL,

flight_number varchar (5) NOT NULL,

amount  money NOT NULL,
```

```
restriction varchar (100) NOT NULL,

CONSTRAINT Flight_fare_PK PRIMARY KEY (fare_code,flight_number),

CONSTRAINT Flight_fare_FK FOREIGN KEY (flight_number) REFERENCES
Flight(flight_number),

);



CREATE TABLE airplane_schedule (

airplane_id varchar (10),

arr_time time(0) NOT NULL,

dep_time time(0) NOT NULL,

CONSTRAINT airplane_schedule_PK PRIMARY KEY (airplane_id),

CONSTRAINT airplane_schedule_FK FOREIGN KEY (airplane_id)
REFERENCES Airplane(airplane_id),

);



CREATE TABLE Flight_schedule (

flight_number varchar (5) NOT NULL,

schedule_arr_time time(0) NOT NULL,

schedule_dep_time time(0) NOT NULL,

CONSTRAINT Flight_schedule_PK PRIMARY KEY (flight_number),

CONSTRAINT Flight_schedule_FK FOREIGN KEY (flight_number)
REFERENCES Flight(flight_number),

);
```

```
/*insert airport table*/

insert into Airport values('A020171223','San Francisco International Airport','San Mateo','California');

insert into Airport values('A020221223','Sydney','Frankfurt','Hesse');

insert into Airport values('A020271126','Salt Lake City International Airport','Salt Lake City','Utah');

insert into Airport values('a020251231','Singapore','Ivalo','Kemi-Tornio');

insert into Airport values('a020230215','Brisbane International Airport','Brisbane','Victoria');

insert into Airport values('a171223200','Bandaranaike International Airport','katunayaka','Colombo');


/*insert Airplane_Type*/

insert into Airplane_Type values('Boeing 787','330',' Rolls Royce');

insert into Airplane_Type values('Airbus A330','440','Pratt & Whitney PW4000');

insert into Airplane_Type values('Boeing 747','467','Pratt & Whitney');

insert into Airplane_Type values('Boeing 757','295','Miami-based Eastern Airlines');

insert into Airplane_Type values('McDonnell Douglas MD-80 Series','135','Pratt & Whitney JT8-D');

insert into Airplane_Type values('Antonov AN28','174','Czechoslovakia LET Aircraft Industries');


/*insert Airplane_Type_Airplan*/

insert into Airplane_Type_Airport values('A020171223','Boeing 787');

insert into Airplane_Type_Airport values('a020251231','Airbus A330');

insert into Airplane_Type_Airport values('a171223200','Boeing 747');
```

```
insert into Airplane_Type_Airport values('A020221223','Boeing 757');
insert into Airplane_Type_Airport values('A020271126','McDonnell Douglas MD-80 Series');
insert into Airplane_Type_Airport values('a171223200','Antonov AN28');

/*insert Airplane*/
insert into Airplane values('R200171223','295','Douglas DC-3','Boeing 757');
insert into Airplane values('N987654321','467','Airbus A321XLR ','Boeing 747');
insert into Airplane values('Q224466889','440','COMAC C919','Airbus A330');
insert into Airplane values('D779955331','174','Universal Hydrogen ATR 72','Antonov AN28');

/*insert Flight*/
insert into Flight values('AL987','Air Berlin','2022-12-23');
insert into Flight values('TY456','Belair','2022-10-19');
insert into Flight values('RE937','Paramount','2022-11-04');
insert into Flight values('KL203','Oman Air','2022-10-26');
insert into Flight values('BH912','IndiGo','2023-01-02');
insert into Flight values('LK729','Jetstar Asia','2022-10-23');
insert into Flight values('PR914','Helvetic Airways','2022-12-01');

/*insert Flight_leg*/
insert into Flight_leg values(2,'AL987','A020171223');
insert into Flight_leg values(3,'PR914','a171223200');
insert into Flight_leg values(4,'AL987','A020271126');
insert into Flight_leg values(1,'KL203','a020230215');
```

```
insert into Flight_leg values(2,'LK729','A020221223');
insert into Flight_leg values(3,'BH912','a020251231');

/*insert airplane_schedule*/
insert into airplane_schedule values('D779955331','09:28:48','15:45:47');
insert into airplane_schedule values('N987654321','01:11:18','08:15:27');
insert into airplane_schedule values('R200171223','11:31:48','19:45:47');
insert into airplane_schedule values('Q224466889','13:07:09','23:55:47');

/*insert Flight_shedule*/
insert into Flight_schedule values('AL987','08:07:09','14:58:14');
insert into Flight_schedule values('TY456','00:14:15','07:13:17');
insert into Flight_schedule values('LK729','10:27:26','18:08:16');
insert into Flight_schedule values('KL203','12:48:17','00:05:05');
insert into Flight_schedule values('BH912','07:22:26','14:08:16');
insert into Flight_schedule values('PR914','17:48:17','04:05:05');

/*insert Flight_fare*/
insert into Flight_fare values('AF1015','AL987','650000','plastic');
insert into Flight_fare values('CDIJ17','TY456','275000','bring metal');
insert into Flight_fare values('GNQS03','RE937','74000','bring pets and plants');
insert into Flight_fare values('PW4589','KL203','65000','eat fish and meats');
insert into Flight_fare values('YBHKL1','PR914','54000','bring over 50KG travel
bags');
insert into Flight_fare values('NQS144','BH912','41000','bring pets');
insert into Flight_fare values('NOQS51','LK729','41265','bring gold,spicies');
```

```
/*insert Leg_instance*/

insert into Leg_instance values('2022-11-14','2','AL987','A020171223','74','R200171223');

insert into Leg_instance values('2022-12-23','3','BH912','a171223200','52','N987654321');

insert into Leg_instance values('2023-01-01','1','KL203','A020271126','158','D779955331');

insert into Leg_instance values('2022-12-14','4','AL987','a020251231','74','Q224466889');


/*insert seat*/

insert into seat values('11F','2022-11-14','2','AL987','Mary Ann','0617651449');

insert into seat values('55JH','2022-12-23','3','BH912','Kelli Joana','0358895642');

insert into seat values('14LA','2023-01-01','1','KL203','Lulifer Konandoil','0662066887');

insert into seat values('12HBA','2022-12-14','4','AL987','Shivangi Munasinghe','9476966515');
```

Part 2

Database Vulnerability Report

SQL Injection

SQL injection is an injection attack that is performed to execute malicious SQL statements. These malicious claims are able to control the data of a web application. SQL injection is a cyber security attack that targets these databases using specially crafted SQL statements to trick systems into doing unexpected and unwanted things. SQL injection attacks are a widespread Internet vulnerability. SQL injection can damage any website or web application that uses SQL. MySQL, Oracle, and SQL servers are SQL databases. SQL injection allows cybercriminals i.e. hackers to access unrestricted databases. There are two types of computer attacks. That is SQL and NoSQL injection. SQL injection targets a traditional database while NoSQL injection targets large databases. SQL injection attacks allow attackers to remove, modify, or delete personal data on a web application or website. And these SQL injections are easy to steal confidential data and even insert new data. A similar example is SQL injection, where hackers can steal credit card numbers, passwords, and personal information of bank customers.

Types of SQL Injection (SQLi)

SQL Injection may be exploited to cause major difficulties in a variety of ways. An attacker might use SQL Injection to overcome authentication, access, alter, and remove data in a database. SQL Injection can also be used to execute instructions on the operating system in some situations, possibly allowing an attacker to escalate to more severe assaults inside a network protected by a firewall. SQL Injection can be classified into three major categories – In-band SQLi, Inferential SQLi and Out-of-band SQLi.

In-band SQLI

The most frequent and straightforward type of SQL Injection attack is in-band SQL Injection. When a hacker has access to the same communication channel for both the attack launch and the data collection, this is known as in-band SQL injection. Union-based and error-based SQL injection are the two most popular varieties.

- Error-based SQLi

An in-band SQL Injection approach called error-based SQLi makes use of error messages thrown by the database server to gather details about the database's structure. An attacker may sometimes

enumerate a database's complete contents using just error-based SQL injection. Errors are tremendously helpful during the development phase of a web application, but they should be deactivated on a live site or logged to a file with restricted access in their place.

- Union-based SQLi

Union-based SQL injection uses the UNION SQL operator to combine the results of two or more SELECT queries into a single result, which is subsequently returned as part of the HTTP response. Union-based SQL injection is a type of in-band SQL injection.

Inferential SQLi (Blind SQLi)

Although it could take an attacker longer to exploit inferential SQL injection than in-band SQL injection, it is just as harmful. Because no data is actually passed through the web application during an inferential SQLi assault, the attacker cannot observe the attack's outcome in real time (thus the term "blind SQL Injection attacks" for these types of attacks). Instead, by sending payloads, tracking how the web application responds, and seeing how the database server behaves as a result, an attacker may reassemble the database structure. Blind-boolean-based and Blind-time-based inferential SQL Injection are the two forms of inferential SQL Injection.

- Boolean-based (content-based) Blind SQLi

Boolean-based SQL Injection is an inferential SQL Injection approach that depends on submitting a SQL query to the database that compels the application to provide a different response dependent on whether the query produces a TRUE or FALSE result. Depending on the outcome, the HTTP response's content will either change or stay the same. Due to this, even though no database data is returned, an attacker may determine if the payload used returned true or false. Because an attacker would have to go through a database character by character, this approach is often slow (particularly on big databases).

- Time-based Blind SQLi

A speculative SQL injection technique called time-based SQL injection works by giving the database a SQL query that makes it wait for a predetermined period of time (in seconds) before answering. The attacker can determine if the query's result is real or false based on the response time. Depending on the outcome, either a delayed or instantaneous HTTP response is given. Even if no data is returned in the database, an attacker can infer whether the payload utilized is true or false thanks to this. Due to the attacker's requirement to calculate the database character by character, this attack is typically slow (particularly on big databases).

Out-of-band SQLi

Because it depends on the functionality enabled on the database server that the web application is utilizing, out-of-region SQL injection is not very prevalent. When an attacker is unable to use that channel to launch the attack and gather results, out-of-band SQL Injection happens. In certain cases where server answers are not particularly consistent, out-of-band techniques offer an attacker an option to guess time-based tactics (making a guess time-based attack unreliable).Out-of-zone In order to give information to an attacker, SQLi attacks rely on the database server's capacity to issue DNS or HTTP queries. Such is the case with the xp darter command of Microsoft SQL Server, which can be used to send DNS requests to a server under the control of the attacker; and the UTL HTTP package of Oracle Database, which can be used to send HTTP requests from SQL and PL/SQL to a server under the control of the attacker.

An effort to bring down a network or system via a denial-of-service (DoS) attack prevents intended users from using the system. DoS attacks do this by sending data to the targeted website, resulting in its failure or overcrowding by users. Every time a DoS attack occurs, it deprives genuine users (staff, members, or account holders) of the service or resource they were expecting.

SQL Injection Prevention Techniques (SQLi)

Protecting against SQL injection problems is difficult. Depending on the subtype of the SQLi vulnerability, the SQL database engine, and the programming language, different prevention methods should be implemented. There are some basic strategic ideas to follow to secure your internet usage.

- Creating and maintaining awareness

First, develop and maintain awareness. If you want to secure your online application, everyone on the development team needs to be aware of the threats of SQL injections. Your developers, QA staff, DevOps, and System Admins should all receive security training. Start by visiting this webpage.

- Do not take any user comments seriously.

The second point is to never depend on user input.

Always be cautious of any user input. Any user input used in an SQL query introduces a SQL injection vulnerability. The same restrictions that apply to Public Input also apply to Authorized and Internal User Input.

- Use a white list instead of a black list.

Step 3: Use a whitelist instead of a blacklist.

User input should not be filtered using a blacklist. Often, a clever attacker will find a way to bypass your blacklist. Use only strict whitelisting to authenticate and limit user input whenever possible.

- Use advanced technologies.

Step 4: Use advanced technologies.

Older web development tools do not have SQLi security built in. Use the most up-to-date versions of the programming language, development environment, and related technologies. For example, use PDO instead of MySQLi in PHP.

- Use reliable approaches.

In step five, use reliable procedures.

Do not attempt to create SQLi security from scratch. Many modern programming tools may include SQLi protections. Instead of reinventing the wheel, use methods like these. For example, use stored procedures or parameterized queries.

- Regularly (with Acunetix).

Step 6: Scan frequently (with Acunetix)

SQL injections or other libraries, modules, or applications may be added by your developers. Use a web vulnerability scanner like Acunetix to regularly check your online applications. If you use Jenkins, enable automatic build scanning using the Acunetix plugin.

Denial-of-service attacks

DoS assaults often target well-known companies' web servers, including banks, businesses, media websites, and governmental and corporate bodies. Denial-of-service attacks seldom cause criminal activity, the loss of sensitive data, or the loss of other assets, but they do cost the victim a lot of time and money to cope with.DoS attacks may be service flooding or service burning.

Flood assaults happen when the server is overloaded with abnormally high amounts of traffic, causing the system to slow and finally crash. Flood attacks often take the following forms

- Buffer overflows are the most frequent kind of DoS attack. The design's goal is to send a network address more data than it can manage. Along with those that target specific flaws in networks or systems, it encompasses the following attacks.By sending malicious signals to all laptops on the targeted network rather than just one, ICMP flood takes advantage of network hardware that has been poorly configured. As a consequence of the network being engaged after that, traffic volume will increase. Additionally known as the smurf assault and the ping of death, this blow.

- SYN flood: Attempts to connect to a server but fails up to the point at which all available ports are occupied, at which point there are no more open ports for real users to connect to.A system or service that is the target of another denial-of-service (DoS) assault crashes. These attacks consist on supplying input that takes advantage of weaknesses present in the target, bringing to a system crash or a large amount of instability, and then preventing use or access to it.

Denial-of-service attack Prevention Techniques

- Check your network for problems.

You must first identify the weaknesses in your network in order to appropriately protect it. Check each device that is linked to your network. This approach involves outlining the network's function, gathering system information, and identifying any existing vulnerabilities. With this degree of knowledge, you might identify network vulnerabilities, evaluate their significance, and plug any openings to prevent intrusion. Even though they require time, audits are valuable. No matter how severe, it is always preferable for a team member to find a security hole than an outsider.

- Keep your current infrastructure safe.

You must make sure that the whole perimeter of your fortress is defended if you want to fend off a DoS assault successfully. Multi-level security approaches, such as intrusion prevention and threat management systems, are thus required. To identify and stop assaults before they overwhelm your network, these systems may make use of anti-spam, content filtering, VPNs, firewalls, load balancing, and security layers. Hardware is necessary, but software cannot complete the work on its own. One of the best methods for defending your network against DoS attacks is edge micro-segmentation

- Cut back on the assault zone.

Reducing the size of the likely assault zone is one of the best DoS prevention strategies. The smaller the attack surface, the easier it is to defend. Although there are several ways to implement this method, micro segmentation is a cutting-edge alternative that is gaining in popularity. A network is divided into teeny, secure segments by micro-segmentation. As a consequence, the security profile as a whole is enhanced. By protecting endpoints on tiny microsegments with hardware-enforced isolation, Bios has developed an effective edge micro-segmentation solution that enhances the network's overall defensive capabilities.

- Make a DoS response plan.

The same is true for DoS assaults, as Benjamin Franklin is quoted as saying, "If you don't plan, you plan to fail." The goal of the plan is to make sure that your present setup is safe, that you can identify an attack as soon as possible, that each member of your team is aware of their responsibilities in the event of an attack, and that everyone is familiar with the escalation and resolution processes.

This implies that the strategy has to include the whole response procedure, including the creation of a response team and a systems checklist. A denial-of-service attack response strategy guarantees that everyone is prepared for it when it happens. It is simple to become distracted and make mistakes during an attack.

- Be aware of the warning signs.

Your chances of successfully defending against a DoS assault increase with the speed at which you discover it. An attack's beginnings are often signaled by a bad connection, a sluggish network, recurrent site failures, or any other persistent performance disturbance.

It is crucial to keep in mind that symptoms caused by high-volume and low-volume DoS assaults may be similar. It is crucial to have team members with the ability or instinct to follow up on the subtle warning indications that may predict a more significant breach since low-volume assaults are challenging to detect due to their similarity to less significant security breaches.

References

- https://www.w3schools.com/sql/
- https://www.w3schools.in/sql
- https://en.m.wikipedia.org/wiki/SQL_injection#:~:text=SQL%20injection%20is%20a%20code%20injection%20technique%2C%20used,to%20dump%20the%20database%20contents%20to%20the%20attacker%29.
- https://brightsec.com/blog/sql-injection-attack/
- https://www.youtube.com/watch?v=NrBJmtD0kEw