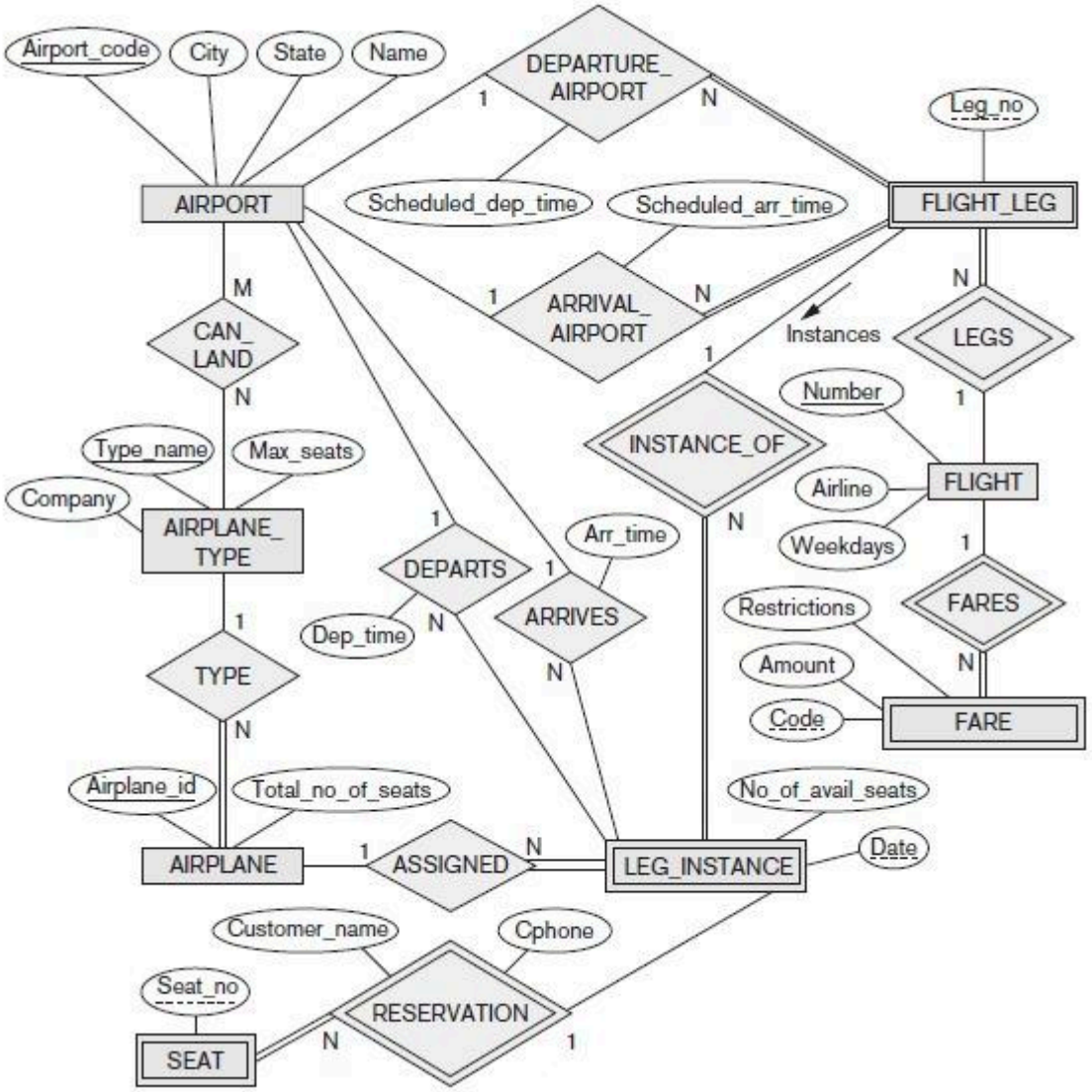
The National University of Computer and Emerging Sciences

Assignment 01:
Database Systems

Sr no.	Name:	Roll no:	Section:
1.	Hasan Yahya	22L-7971	BSE-4B

Question: 01



Firstly Creating the Database **Airline_Reservation_System** using these SQL Queries:

```
-- Create the Database
use master
go
create database Airline_Reservation_System

-- Use the Database
use [Airline_Reservation_System]
```

After creating the Database in SQL Server, Creating the Table **Airport:**

```sql
-- Create Table Airport
CREATE TABLE Airport (
    airport_code VARCHAR(10) PRIMARY KEY,
    city VARCHAR(100) NOT NULL,
    state VARCHAR(100) NOT NULL,
    name VARCHAR(255) NOT NULL
);
```

This will Create the Table with Columns:

airport_code	city	state	name

Here **airport_code** is a **Primary Key.** Now Creating the Table **Flight**:

```sql
-- Create Table Flight
CREATE TABLE Flight (
number INT PRIMARY KEY,
airline VARCHAR(100) NOT NULL,
weekdays VARCHAR(50) NOT NULL
);
```

This will create **Flight** table with columns:

number	airline	weekdays

Here, **number** is a **Primary Key**. Creating the table for Weak Entity **Flight_Leg** and partial key (discriminator) **Leg_no:**

```sql
-- Create Table Flight_Leg
CREATE TABLE Flight_Leg (
Leg_no INT NOT NULL,
number INT NOT NULL, -- This is connected to Flight Number from Flight Table
FOREIGN KEY (number) REFERENCES Flight(number),
PRIMARY KEY (Leg_no, number)
);
```

This will create **Flight_Leg** table:

number	Leg_no

Here number is **Foreign key** from **Flight** table and **Leg_no** is **partial key** that depends upon the **number (primary key)** from **Flight** table. This creates a final composite key.

Creating a table for **Departure_Airport** relation:

```
-- Create Table Departure_Airport
CREATE TABLE Departure_Airport (
scheduled_dep_time DATETIME NOT NULL,
airport_code VARCHAR(10) NOT NULL, -- This is connected to Airport Code from
Airport Table
Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
number INT NOT NULL, -- This is connected to Flight Number from Flight Table
FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
FOREIGN KEY (Leg_no, number) REFERENCES Flight_Leg(Leg_no, number), --
Referencing both columns in Flight_Leg
PRIMARY KEY (airport_code, Leg_no)
);
```

It isn't possible to only get **Leg_no** from **Flight_Leg** table because it is making a composite key with Flight **number** from **Flight** table. So, you have to Reference them both.

scheduled_dep_time	airport_code	Leg_no	number

Now, making a table for **Arrival_Airport** relation.

```
-- Create Table Departure_Airport
CREATE TABLE Arival_Airport (
    scheduled_arr_time DATETIME NOT NULL,
    airport_code VARCHAR(10) NOT NULL, -- This is connected to Airport Code
from Airport Table
    Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
    number INT NOT NULL, -- This is connected to Flight Number from Flight
Table
    FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
    FOREIGN KEY (Leg_no, number) REFERENCES Flight_Leg(Leg_no, number), --
Referencing both columns in Flight_Leg
    PRIMARY KEY (airport_code, Leg_no)
);
```

This will give us the table:

scheduled_arr_time	airport_code	Leg_no	number

Creating the table **Airplane_Type:**

```sql
-- Create Table Airplane_Type
CREATE TABLE Airplane_Type (
    Type_name VARCHAR(100) PRIMARY KEY,
    Company VARCHAR(100) NOT NULL,
    Max_Seats INT NOT NULL
);
```

Type_name	Company	Max_Seats

Here **Type_name** is a **Primary Key**. Now for table **Airplane:**

```sql
-- Create Table Airplane
CREATE TABLE Airplane (
    Airplane_ID INT PRIMARY KEY,
    Total_Seats INT NOT NULL,
);
```

Airplane_ID	Total_Seats

Here, **Airplane_ID** is a **Primary Key.** For the Weak Entity **Fare** with the relation **Fares:**

```sql
-- Create Table Flight_Leg
CREATE TABLE Fare (
    Code INT NOT NULL,
    Restrictions VARCHAR(100) NOT NULL,
    Amount INT NOT NULL,
    number INT NOT NULL, -- This is connected to the Flight Number from the
Flight Table
    FOREIGN KEY (number) REFERENCES Flight(number),
    PRIMARY KEY (Code, number)
);
```

Code	Restrictions	Amount	number

Here, **Code** and **number** are a **Composite Primary Key**. Creating table **Leg_Instance:**

```sql
-- Create Table Leg_Instance
CREATE TABLE Leg_Instance (
    No_of_seats INT NOT NULL,
    Date DATE NOT NULL,
    Leg_no INT NOT NULL,
    number INT NOT NULL, -- This is connected to the Flight Number from the
```

```
Flight Table
    FOREIGN KEY (number) REFERENCES Flight(number),
    FOREIGN KEY (Leg_no, number) REFERENCES Flight_Leg(Leg_no, number), --
Referencing both columns in Flight_Leg
    PRIMARY KEY (Leg_no, number, Date)
);
```

This creates table:

No_of_seats	Date	Leg_no	number

To show the **Can_Land** attribute, we create the table:

```
-- Create Can_Land Table
CREATE TABLE Can_Land (
    Type_name VARCHAR(100) NOT NULL, -- This is connected to Type Name from
Airplane Type Table
    airport_code VARCHAR(10) NOT NULL, -- This is connected to Airport Code
from Airport Table
    FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name),
    FOREIGN KEY (airport_code) REFERENCES Airport(airport_code),
    PRIMARY KEY (Type_name, airport_code)
);
```

Type_name	airport_code

After this table, we edit the **Airplane** table to put the **Type_name** key from **Airplane Type** table as a foreign key inside the **Airplane** table**.**

```
-- Alter Table Airplane to add Type_name as Foreign Key
ALTER TABLE Airplane
ADD Type_name VARCHAR(100) NOT NULL,
FOREIGN KEY (Type_name) REFERENCES Airplane_Type(Type_name);
```

Now the Airplane Table schema has changed to:

Airplane_ID	Total_Seats	Type_Name

Here **Airplane_ID** is **Primary key** and **Type_Name** is **Foreign Key.**

```
-- Alter Table Leg_Instance to add Airplane_ID as Foreign Key
```

```
ALTER TABLE Leg_Instance
ADD Airplane_ID INT NOT NULL,
FOREIGN KEY (Airplane_ID) REFERENCES Airplane(Airplane_ID);
```

Now the **Leg_Instance** table is:

No of Seats	Date	Leg_no	number	Airplane_ID

Here, **Leg_no, number, Date** are **composite primary keys** and **Airplane_ID** is **Foreign Key.** For the Departs Relation:

```
-- Create Table Departs
CREATE TABLE Departs (
    Dep_time TIME NOT NULL,
    Airport_code VARCHAR(10) NOT NULL, -- This is connected to Airport Code
from Airport Table
    Date DATE NOT NULL, -- This is connected to Date from Leg_Instance Table
    Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
    number INT NOT NULL, -- This is connected to Flight Number from Flight
Table
    FOREIGN KEY (Airport_code) REFERENCES Airport(airport_code),
    FOREIGN KEY (Leg_no, number, Date) REFERENCES Leg_Instance(Leg_no,
number, Date), -- Referencing all columns in Leg_Instance
    PRIMARY KEY (Airport_code, Date, Leg_no, number)
);
```

Dep_time	Airport_Code	Date	Leg_no	number

Here, all underlined column names are **composite primary keys.** Only, Dep_time is a Normal key. Similarly, for **Arrives:**

```
-- Create Table Arrives
CREATE TABLE Arrives (
    Arr_time TIME NOT NULL,
    Airport_code VARCHAR(10) NOT NULL, -- This is connected to Airport Code
from Airport Table
    Date DATE NOT NULL, -- This is connected to Date from Leg_Instance Table
    Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
    number INT NOT NULL, -- This is connected to Flight Number from Flight
Table
    FOREIGN KEY (Airport_code) REFERENCES Airport(airport_code),
    FOREIGN KEY (Leg_no, number, Date) REFERENCES Leg_Instance(Leg_no,
```

```
number, Date), -- Referencing all columns in Leg_Instance
    PRIMARY KEY (Airport_code, Date, Leg_no, number)
);
```

Arr_time	Airport_Code	Date	Leg_no	number

Here, all underlined column names are **composite primary keys.** Only, Arr_time is a Normal key. For the weak entity, **Seat:**

```
-- Create table Seat
CREATE TABLE Seat (
    Seat_no INT NOT NULL,
    Customer_name VARCHAR(100) NOT NULL,
    Cphone INT NOT NULL,
    DATE DATE NOT NULL, -- This is connected to Date from Leg_Instance Table
    Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
    number INT NOT NULL, -- This is connected to Flight Number from Flight
Table
    FOREIGN KEY (Leg_no, number, Date) REFERENCES Leg_Instance(Leg_no,
number, Date), -- Referencing all columns in Leg_Instance
    PRIMARY KEY (Seat_no, Date, Leg_no, number)
);
```

Seat_no	name	Cphone	Date	Leg_no	number

Here all underlined ones are **composite** primary keys. Now, for the **Reservation** table:

```
-- Create table Reservation
CREATE TABLE Reservation (
    Seat_no INT NOT NULL,
    Customer_name VARCHAR(100) NOT NULL,
    Cphone INT NOT NULL,
    DATE DATE NOT NULL, -- This is connected to Date from Leg_Instance Table
    Leg_no INT NOT NULL, -- This is connected to Leg Number from Flight_Leg
Table
    number INT NOT NULL, -- This is connected to Flight Number from Flight
Table
    FOREIGN KEY (Leg_no, number, Date) REFERENCES Leg_Instance(Leg_no,
number, Date), -- Referencing all columns in Leg_Instance
    PRIMARY KEY (Seat_no, Date, Leg_no, number)
);
```

Seat_no	name	Cphone	Date	Leg_no	number

Here underlined, keys are composite primary keys.

Assumptions (Extra):

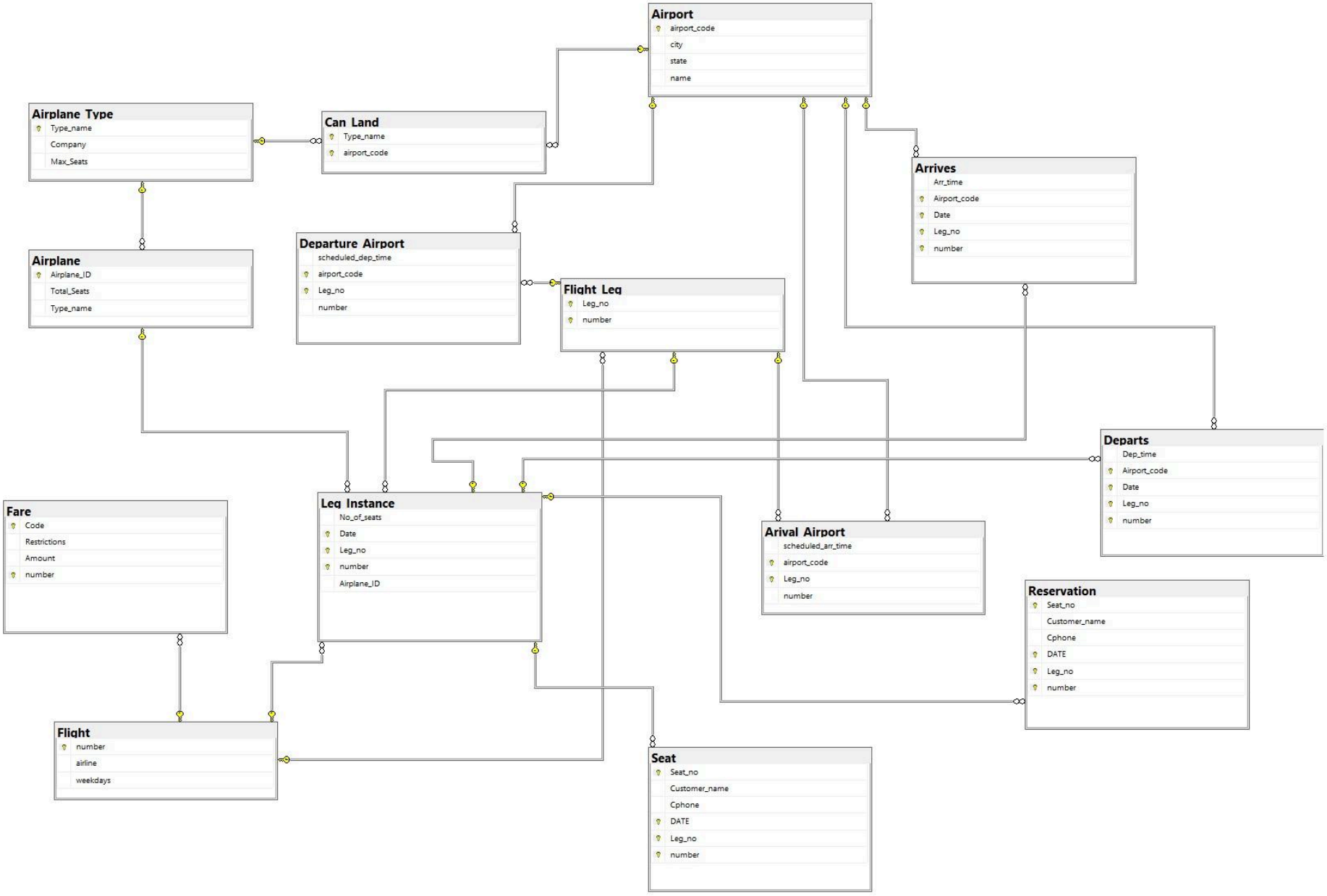
- Whenever, a composite key is used. I used it because there is an 1 to N relation between a strong and an (N) relation weak entity.
- In a normal 1 to N relation, you make a primary key of 1 Entity as a foreign key in N entity. However, when we met the Course Instructor in her office she said we can make a separate table for both 1 to N and N to M relations, so I did that instead.
- I carried the Composite keys after already made tables as primary keys to other tables. Because, there can't be 2 primary keys in a table (without composition).

The SQL Code to Drop the Tables and Database is given below:

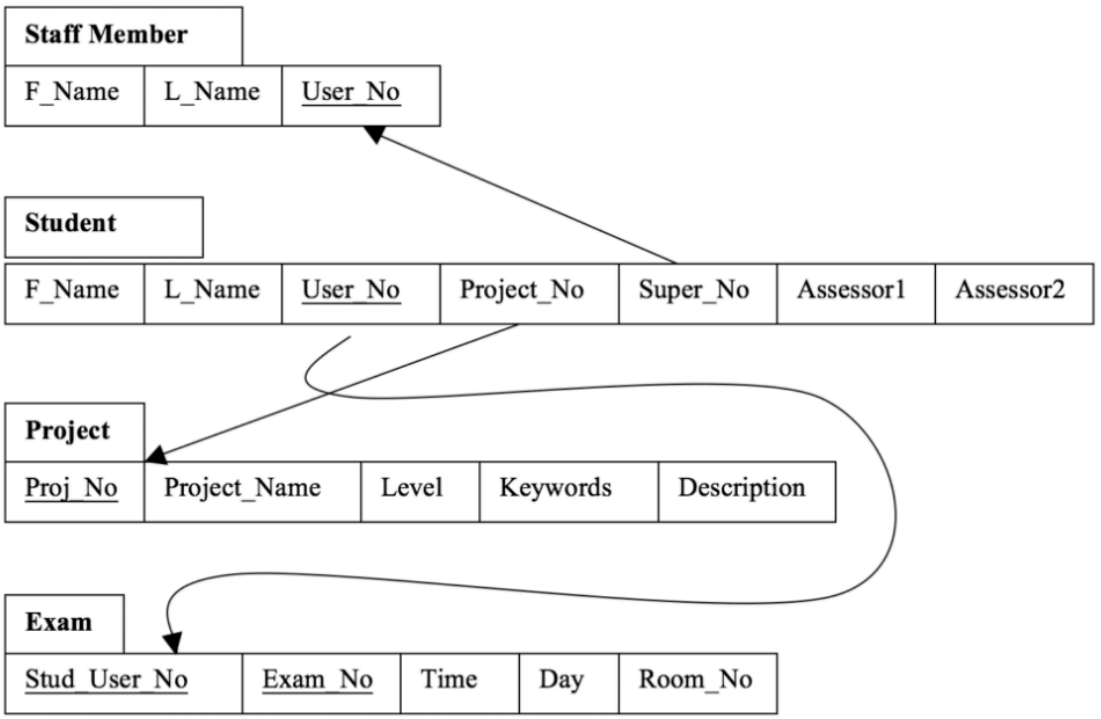
```
-- Drop the Tables
DROP TABLE Airport;
DROP TABLE Flight;
DROP TABLE Flight_Leg;
DROP TABLE Departure_Airport;
DROP TABLE Arival_Airport;
DROP TABLE Airplane_Type;
DROP TABLE Airplane;
DROP TABLE Fare;
DROP TABLE Leg_Instance;
DROP TABLE Can_Land;
DROP TABLE Departs;
DROP TABLE Arrives;
DROP TABLE Seat;
DROP TABLE Reservation;

-- Drop the Database
DROP DATABASE Airline_Reservation_System;
```

I uploaded the original SQL File to Google Classroom as `Airline_Reservation_System.sql` and you can view all the Code i used (from creating Database to Final Schema) in one place there. Moreover, the Schema Diagram for this Database is given on the Next Page:


Airport
airport_code
city
state
name
Airplane Type
Type_name
Company
Max_Seats
Can Land
Type_name
airport_code
Arrives
Arr_time
Airport_code
Date
Leg_no
number
Airplane
Airplane_ID
Total_Seats
Type_name
Departure Airport
scheduled_dep_time
airport_code
Leg_no
number
Flight Leg
Leg_no
number
Departs
Dep_time
Airport_code
Date
Leg_no
number
Fare
Code
Restrictions
Amount
number
Leg Instance
No_of_seats
Date
Leg_no
number
Airplane_ID
Arival Airport
scheduled_arr_time
airport_code
Leg_no
number
Reservation
Seat_no
Customer_name
Cphone
DATE
Leg_no
number
Flight
number
airline
weekdays
Seat
Seat_no
Customer_name
Cphone
DATE
Leg_no
number

Question: 02



There is a problem in the above table: the Arrow direction between Student table and Exam table is supposed to be opposite. The Stud_User_No is a Foreign Key in the exam table. I used the fixed Arrow Direction for this Schema (ie, this Solution is correct).

1. **Populate sensible data in the following database. Using Insert statement to insert 3 rows per table.**

Writing SQL Code to CREATE and INSERT values into tables:

```
-- Create the Database
use master
go
create database Exam_Project_System;

-- Use the Database
use [Exam_Project_System];

-- Drop all tables if exist
DROP TABLE Staff_Member;
DROP TABLE Project;
DROP TABLE Student;
DROP TABLE Exam;
```

```sql
-- Select all values in tables
SELECT * FROM Staff_Member;
SELECT * FROM Project;
SELECT * FROM Student;
SELECT * FROM Exam;

-- Create Table Staff_Member
create table Staff_Member
(
    F_Name varchar(100) NOT NULL,
    L_Name varchar(100) NOT NULL,
    User_No int PRIMARY KEY NOT NULL,
);

-- Create table Project
create table Project
(
    Proj_No INT NOT NULL PRIMARY KEY,
    Project_Name VARCHAR(100) NOT NULL,
    Level INT NOT NULL,
    Keywords VARCHAR(100) NOT NULL,
    Description VARCHAR(100) NOT NULL
);

-- Create Table Student
create table Student
(
    F_Name VARCHAR(100) NOT NULL,
    L_Name VARCHAR(100) NOT NULL,
    User_No int PRIMARY KEY NOT NULL,
    Project_No int NOT NULL,
    Super_No int NOT NULL,
    FOREIGN KEY (Super_No) REFERENCES Staff_Member(User_No),
    Assessor1 VARCHAR NOT NULL,
    Assessor2 VARCHAR NOT NULL,
    FOREIGN KEY (Project_No) REFERENCES Project(Proj_No)
);

-- Create Table Exam
create table Exam
(
    Stud_User_No INT NOT NULL,
    Exam_No INT NOT NULL,
    Time Time NOT NULL,
    Day INT NOT NULL,
    Room_No INT NOT NULL,
    -- stud_user_no refrences user no in student
```

```
    FOREIGN KEY (Stud_User_No) REFERENCES Student(User_No),
    PRIMARY KEY (Stud_User_No, Exam_No)
);

-- Insert values into Staff_Member table
INSERT INTO Staff_Member (F_Name, L_Name, User_No)
VALUES ('Hasan', 'Yahya', 1001),
       ('Wasee', 'Rehman', 1002),
       ('Zayed', 'Abdullah', 1003);

-- Insert values into Project table
INSERT INTO Project (Proj_No, Project_Name, Level, Keywords, Description)
VALUES (1, 'Weather', 1, 'Weather, App', 'Made using Javascript'),
       (2, 'Pacman', 2, 'Pacman, SFML', 'Made using C++ and SFML Library'),
       (3, 'Sudoku', 3, 'Sudoku, SFML', 'Made using C++ and SFML Library');

-- Alter Table Student
ALTER TABLE Student
ALTER COLUMN Assessor1 VARCHAR(100) NOT NULL;
ALTER TABLE Student
ALTER COLUMN Assessor2 VARCHAR(100) NOT NULL;

-- Insert values into Student table
INSERT INTO Student (F_Name, L_Name, User_No, Project_No, Super_No,
Assessor1, Assessor2)
VALUES ('Ali', 'Sahab', 1010, 1, 1001, 'Assessor1A', 'Assessor2A'),
       ('Michael', 'Johnson', 1011, 2, 1002, 'Assessor1B', 'Assessor2B'),
       ('Mani', 'Hasan', 1012, 2, 1002, 'Assessor1C', 'Assessor2C');

-- Insert values into Exam table
INSERT INTO Exam (Stud_User_No, Exam_No, Time, Day, Room_No)
VALUES (1010, 1, '09:00:00', 1, 005),
       (1011, 1, '10:30:00', 1, 009),
       (1011, 4, '10:30:00', 1, 007);

-- Now all Values have been entered into Tables
```

After entering the Values, we get the tables:

- **Staff Member (Table):**

F_Name	L_Name	User_No
Hasan	Yahya	1001
Wasee	Rehman	1002

Zayed	Abdullah	1003

- **Project (Table):**

Proj_No	**Project_Name**	**Level**	**Keywords**	**Description**
1	Weather	1	Weather, App	Made using Javascript
2	Pacman	2	Pacman, SFML	Made using C++ and SFML Library
3	Sudoku	3	Sudoku, SFML	Made using C++ and SFML Library

- **Student (Table):**

F_Name	**L_Name**	**User_No**	**Project_No**	**Super_No**	**Assessor1**	**Assessor2**
Ali	Sahab	1010	1	1001	Assessor1A	Assessor2A
Michel	Johnson	1011	2	1002	Assessor1B	Assessor2B
Mani	Hasan	1012	3	1003	Assessor1C	Assessor2C

- **Exam (Table):**

Stud_User_No	**Exam_No**	**Time**	**Day**	**Room_No**
1010	1	09:00:00	1	5
1011	1	10:30:00	1	9
1011	4	10:30:00	1	7

2. **Delete the records of all students who are working on project no 10.**

```
-- 2) Delete the records of all students who are working on project no 10.
DELETE FROM Student
WHERE Project_No = 10;
```

Since, the tables have no **Project_No: 10**. No, change will happen in the tables.

3. **Change the level of projects from 4 to 5 which have ‘Android’ in their keywords.**

```
-- 3) Change the level of projects from 4 to 5 which have 'Android' in their
keywords.
UPDATE Project
SET Level = 5
WHERE Keywords LIKE '%Android%' AND Level = 4;
```

Since, there are no **"Android"** keywords in the given tables. This query won't cause any change in the tables.

4. **Write query for the following problems:**

- **Find exam time and date of student user no: 1010 and Room_no = 005:**

```
-- 1. Find exam time and date of student user no: 1010 and Room_no = 005
SELECT Time, Day
FROM Exam
WHERE Stud_User_No = 1010 AND Room_No = 005;
```

This will return the following table:

Time	Day
09:00:00	1

- **Show all project names in descending order:**

```
-- 2. Show all project names in descending order.
SELECT Project_Name
FROM Project
ORDER BY Project_Name DESC;
```

This returns the following table:

Project_Name
Weather
Sudoku
Pacman

- **Show F_Name of staff member with L_Name starting with A and ending with A:**

```
-- 3. Show fname of staff member with L_Name starting with A and ending with
A.
SELECT F_Name
FROM Staff_Member
WHERE L_Name LIKE 'A%A';
```

This returns the following table (It has a column but no tuples/rows):

F_Name

- **Show name of student who has not been assigned any supervisor as yet:**

```
-- 4. Show name of student who has not been assigned any supervisor as yet.
SELECT F_Name, L_Name
FROM Student
WHERE Super_No IS NULL;
```

This returns a table with two columns (but no tuples):

F_Name	L_Name

- **Show user no who are sitting between room 005 and 009 during exam:**

```
-- 5. Show user no who are sitting between room 005 and 009 during exam.
SELECT Stud_User_No
FROM Exam
WHERE Room_No BETWEEN 005 AND 009;
```

This returns the table:

Stud_User_No
1010
1011
1011

- **Show students name with its supervisor name and project number:**

```sql
-- 6. Show students name with its supervisor name and project number.
SELECT S.F_Name, S.L_Name, SM.F_Name, SM.L_Name, S.Project_No
FROM Student S
JOIN Staff_Member SM
ON S.Super_No = SM.User_No;
```

This returns the table:

F_Name	L_Name	F_Name	L_Name	Project_No
Ali	Sahab	Hasan	Yahya	1
Michael	Johnson	Wasee	Rehman	2
Mani	Hasan	Wasee	Rehman	2

Note: I submitted the original SQL Files in Google Classroom as well for both Question 01 & 02. You can check the `Exam_Project_System.sql` and `Airline_Reservation_System.sql` for both of them. Moreover, ***Microsoft SQL Server*** was used for the Database (which is reflected in the SQL Syntax). In Question 01, I created extra tables for Attributes (1 to N) and used composition keys because when I visited Course Instructor, **Hina Iqbal** in her office, she allowed me to do so.
