

University of Texas at Arlington
CSE-3330-002
May 4, 2020

Flight Reservation Project

Team 02

Ashwitha Kassetty
Katherine Baumann
Ariella Amanuel

HONOR CODE

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence. I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code.

SIGNATURES: Ashwitha Venkata Kassetty, Ariella Amanuel, Katherine Baumann

Table of Contents

Mini-World description	2
EER Diagram	4
ERR Diagram Description	5
Relational Database Schema	6
Phase 2: Database Creation and Querying	8
Phase 3: Querying In MySQL	32

Mini-World description

Every airline company owns a number of aircrafts and it is associated with a number of airports from where it operates. Each airline has a two-letter ID from which it is being identified uniquely. For example, the ID for American Airlines is AA, and the ID for United Airlines is UA. Similarly, each airport has a three-letter ID. For example, EWR, LGA, and JFK are well known local airport codes.

A flight is operated by an airline and a specific aircraft and operates on a given set of days of the week (e.g. every Monday, Wednesday). Flights can either be domestic or international. For every flight, it must record its flight number (unique only within that airline), the departure and destination airports, as well as the departure and arrival time. Customers should be able to make reservations.

Customers should first be able to search for specific flights by providing information about the departure and arrival airport as well as the date they wish to fly. The flight ticket can either be one way, or round-trip and they should be able to set if they are flexible about flight dates (+- 3 days).

A flight ticket has a unique number and is for just a single passenger. Each ticket is associated with a sequence of flights. For example, a ticket might be associated with just one flight if it is one-way or with 2 flights if it is round-trip. Each ticket must include all the associated flights and include information for the departure and arrival airport, flight numbers (along with its airline), departure date and time, and class (economy/business/first). It also has the following attributes: total fare, and date and time when ticket was purchased.

In case the class of the ticket is economy, the customer should not be able to Page 2 of 3 change/cancel their ticket unless a fee is paid. For business/first class, customers should be able to change their ticket with no fee. A customer may partake in any number of flight transactions and she/he is associated with one member account which includes a reservation portfolio, indicating all the flight history held in this account (past flights and upcoming).

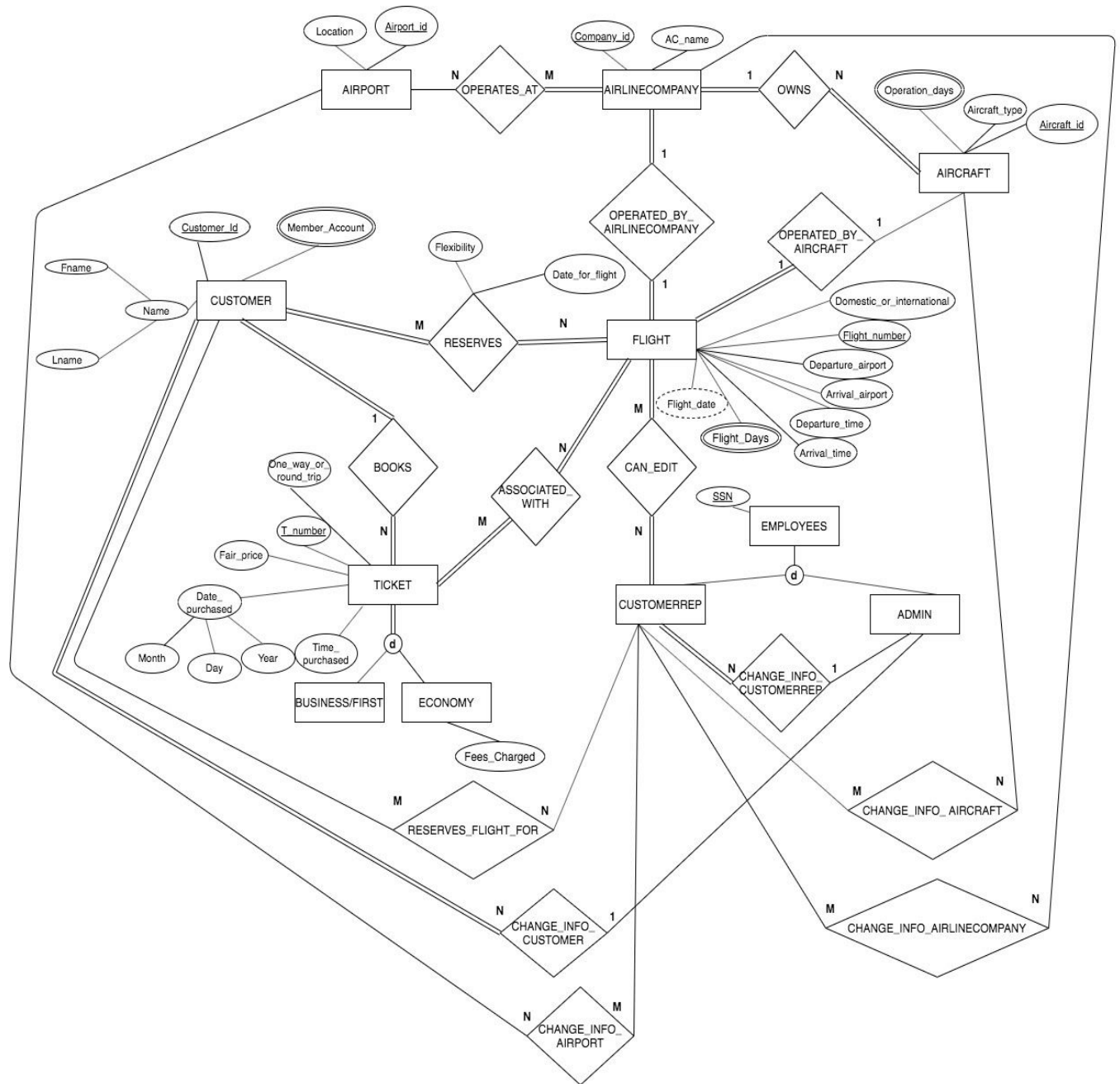
Brief Introduction

Our goal is to create a database system that allows customers to manually search for flights based on a variety of criteria, create flight reservations, and finally book flights. Administrators and customer representatives can also search for and edit reservations on behalf of customers. We are assuming our users will have limited knowledge on databases, so our goal is to make this system user friendly and easy to operate.

We used the information in the mini-world description from above in order to derive our depicted entities, attributes, and relationships.

For phase one of this project we have completed our documentation including an EER diagram and a relational database schema. Both of these diagrams were created to help the designers of the database understand the relationships between the determined entities and the entities respective attributes.

EER Diagram



ERR Diagram Description

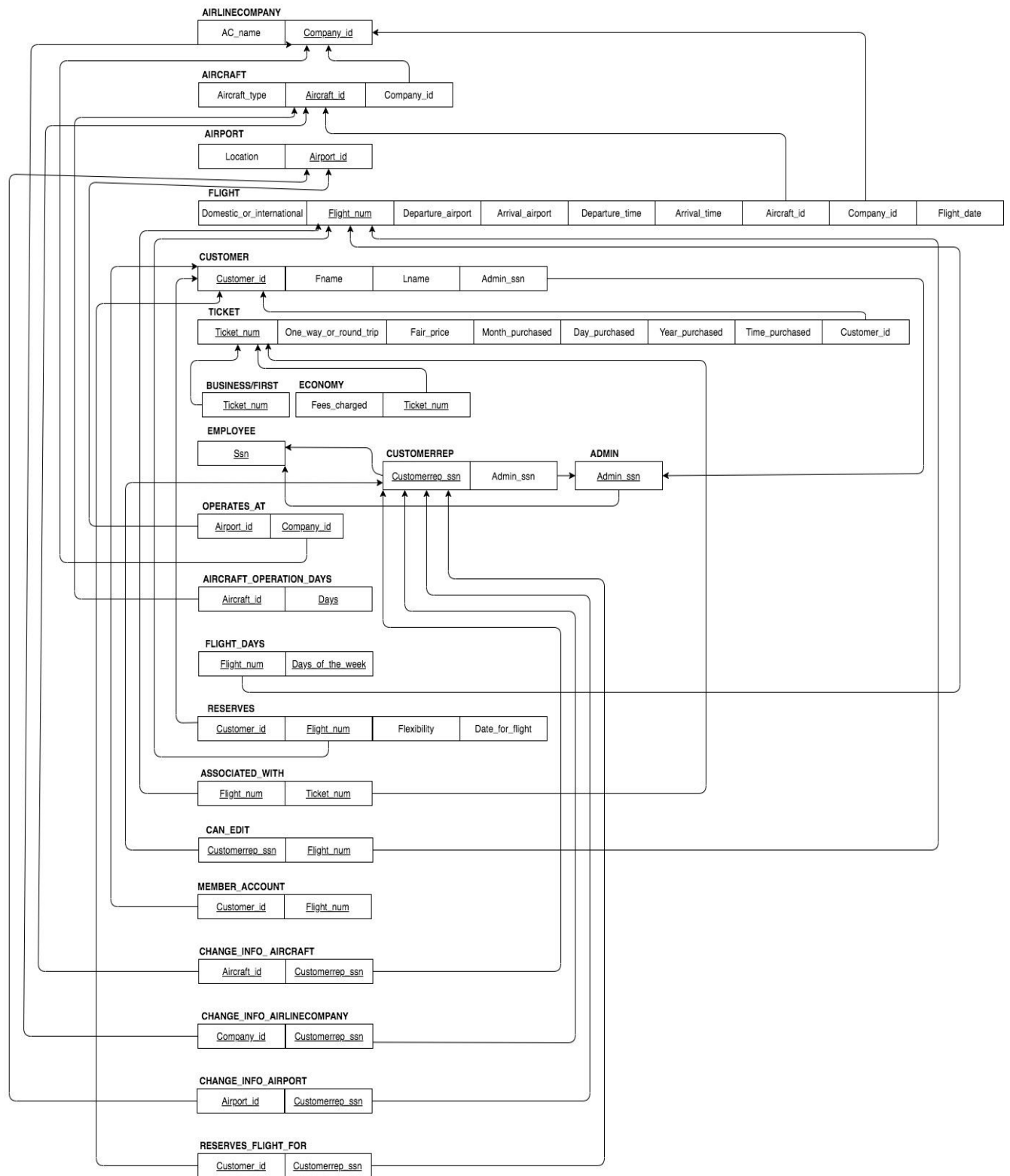
When designing our EER diagram we decided on creating 9 entities with an average of 4 to 5 attributes for each of them. We had to make a few assumptions for the relationships between our entities. Those assumptions include:

- There can be more employees at the company that just hold the title administrator or customer representative.
- Aircrafts that are owned by a specific airline company can be shared by other airline companies.
- A customer's reservation becomes a ticket once they have paid for it.

We have also accounted for some missing info. This includes SSN numbers that are used as key attributes for employees, customer ids which are used as key attributes for customers, and aircraft id numbers which are used as key attributes for aircrafts. Even though these attributes are not mentioned in the document, we used them because they are real world identifiers and attributes that are available.

We have also determined that we will not describe a few of the functionalities on either of the diagrams. We made this choice because functionalities such as “produce a list of most active flights” or “produce a list of all flights for a given airport” can be represented as queries to the database. We have represented functional requirements such as “customer representatives can make flight reservations on behalf of users” and “customer representatives can edit flight reservations for a customer”.

Relational Database Schema



Relational Diagram Description

For the relational diagram schema we specifically used option 8A as our design choice. We did this because we have incorporated specializations between our different entities. By using the mapping style 8A we were able to apply and convey disjoint, total, and partial relations. For the mapping of the relationships between entities we used primary keys of the entities so they can serve as foreign keys in the relationships.

Phase 2: Database Creation and Querying In MySQL

Table **USERS**:

```
CREATE TABLE USERS (
  USRID int AUTO_INCREMENT,
  Name varchar(255) NOT NULL,
  email varchar(255),
  Phone varchar(255),
  Type int NOT NULL,
  PRIMARY KEY (USRID),
  CONSTRAINT Type_values CHECK ((Type=1 OR Type=2 OR Type=3))
);
```

I decided to use an incrementing, artificial key for the USRID field so the DBMS will provide a field for USRID. Also, I chose to not use the NOT NULL constraint for the email and Phone attributes in case a user does not have a phone number or email.

Table **AIRCRAFT**:

```
CREATE TABLE AIRCRAFT (
  ARCID int AUTO_INCREMENT,
  Name varchar(255) NOT NULL,
  ARLID varchar(2) NOT NULL,
  PRIMARY KEY (ARCID)
);
```

I decided to use an incrementing, artificial key for the ARCID field so the DBMS will provide a unique field ARCID.

Table **AIRLINE**:

```
CREATE TABLE AIRLINE (
  ARLID varchar(2),
  Name varchar(255) NOT NULL,
  PRIMARY KEY (ARLID)
);
```

Table **Airport**:

```
CREATE TABLE AIRPORT (
  ARPID varchar(3),
  Name varchar(255) NOT NULL,
  PRIMARY KEY (ARPID)
);
```

Table ARCCCLASS:

```
CREATE TABLE ARCCCLASS (
  ARCID int NOT NULL,
  CLASS varchar(255) NOT NULL,
  ChangeFee DECIMAL(8,2) NOT NULL,
  NumofSeats int NOT NULL,
  FOREIGN KEY (ARCID) REFERENCES AIRCRAFT (ARCID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  PRIMARY KEY (ARCID, CLASS)
);
```

For ChangeFee, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

Table ARLARP:

```
CREATE TABLE ARLARP (
  ARLID varchar(2),
  ARPID varchar(3),
  FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY (ARPID) REFERENCES AIRPORT (ARPID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  PRIMARY KEY (ARLID, ARPID)
);
```

Table FLIGHT:

```
CREATE TABLE FLIGHT (
  ARLID varchar(2),
  FLGID int AUTO_INCREMENT,
  Type int NOT NULL,
  DepARPID varchar(3) NOT NULL,
  DepTime time NOT NULL,
  DesARPID varchar(3) NOT NULL,
  DesTime time NOT NULL,
  ARCID int NOT NULL,
  FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  PRIMARY KEY (ARLID, FLGID),
  CONSTRAINT Type_flvalues CHECK ((Type=1 OR Type=0))
);
```

I decided to use an incrementing, artificial key for the FLGID field so the DBMS will provide a unique value for FLGID.

Table FLGCLASS:

```
CREATE TABLE FLGCLASS (
  ARLID varchar(2),
  FLGID int,
  CLASS varchar(255),
  Fare decimal(8,2) NOT NULL,
  FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  PRIMARY KEY(ARLID, FLGID, CLASS)
);
```

For Fare, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

Table FLGDAYS:

```
CREATE TABLE FLGDAYS (
  ARLID varchar(2),
  FLGID int,
  DAY varchar(3),
  FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)
  ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT DAY_values CHECK ((DAY='Mon' OR DAY='Tue' OR DAY='Wed' OR
  DAY='Thu' OR DAY='Fri' OR DAY='Sat' OR DAY='Sun')),
  PRIMARY KEY(ARLID, FLGID, DAY)
);
```

Table TICKET:

```
CREATE TABLE TICKET (
  TCKID int AUTO_INCREMENT,
  USRID int NOT NULL,
  Type int NOT NULL,
  TotalFare decimal(8,2) NOT NULL,
  PurchaseDateTime timestamp NOT NULL,
  PRIMARY KEY (TCKID),
  CONSTRAINT Type_tckvalues CHECK ((Type=1 OR Type=2))
);
```

For TotalFare, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

Table SEQUENCE:

```
CREATE TABLE SEQUENCE (  
  TCKID int,  
  ARLID varchar(2),  
  FLGID int,  
  CLASS varchar(255) NOT NULL,  
  TravelDate date NOT NULL,  
  FOREIGN KEY (TCKID) REFERENCES TICKET (TCKID)  
  ON DELETE RESTRICT ON UPDATE CASCADE,  
  FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)  
  ON DELETE RESTRICT ON UPDATE CASCADE,  
  FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)  
  ON DELETE RESTRICT ON UPDATE CASCADE,  
  PRIMARY KEY(TCKID,ARLID,FLGID)  
);
```

Task 2

USERS:

Command:

```
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('Lula Wiley','wiley@gmail.com','+1-202-555-0177','1');
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('Isabell Horn','horn@gmail.com','+1-202-555-0136','1');
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('Usman Hook','hook@hotmail.com','+1-219-555-0126','1');
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('Abdullah Singleton','singleton@outlook.com','+1-404-555-0199','1');
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('Sumaiya Dean','dean@yahoo.com','+1-512-555-0161','1');
INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`) VALUES('August Philips','phillips@gmail.com','+1-213-555-0128','1');
```

Table:

USRID	Name	email	Phone	Type
1	Lula Wiley	wiley@gmail.com	+1-202-555-0177	1
2	Isabell Horn	horn@gmail.com	+1-202-555-0136	1
3	Usman Hook	hook@hotmail.com	+1-219-555-0126	1
4	Abdullah Singleton	singleton@outlook.com	+1-404-555-0199	1
5	Sumaiya Dean	dean@yahoo.com	+1-512-555-0161	1
6	August Philips	phillips@gmail.com	+1-213-555-0128	1

Action Output:

✓ 5	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.0050 sec
✓ 6	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.0023 sec
✓ 7	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.0012 sec
✓ 8	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.0041 sec
✓ 9	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.00090 sec
✓ 10	19:39:31	INSERT INTO `Phase_two`.`USERS` (`Name`,`email`,`phone`,`Type`...	1 row(s) affected	0.00093 sec

AIRCRAFT:

Command:

```
SELECT * FROM Phase_two.AIRCRAFT;
INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('1','Airbus A321','AA');
INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('2','Boeing 737','AA');
INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('3','Airbus A320','UA');
```

Table:

► 1	Airbus A321	AA
2	Boeing 737	AA
3	Airbus A320	UA

Action Output:

✓ 31	14:34:35	SELECT * FROM Phase_two.AIRCRAFT LIMIT 0, 1000	0 row(s) returned	0.00040 sec / 0.000...
✓ 32	14:34:35	INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('1','Airbus...	1 row(s) affected	0.010 sec
✓ 33	14:34:35	INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('2','Boeing...	1 row(s) affected	0.0075 sec
✓ 34	14:34:35	INSERT INTO `Phase_two`.`AIRCRAFT` (`ARCID`,`Name`,`ARLID`) VALUES ('3','Airbus...	1 row(s) affected	0.0043 sec

AIRLINE:

Command:

```
SELECT * FROM Phase_two.AIRLINE;
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('AA', 'American Airlines');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('AC', 'Air Canada');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('DL', 'Delta Air Lines');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('EK', 'Emirates');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('LX', 'SWISS International Air Lines');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('OA', 'Olympic Air');
INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('UA', 'United Airlines');
```

Table:

AA	American Airlines
AC	Air Canada
DL	Delta Air Lines
EK	Emirates
LX	SWISS International Air Lines
OA	Olympic Air
UA	United Airlines

Action Output:

38	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('AA', 'American Airlines')	1 row(s) affected	0.0049 sec
39	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('AC', 'Air Canada')	1 row(s) affected	0.0018 sec
40	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('DL', 'Delta Air Lines')	1 row(s) affected	0.0011 sec
41	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('EK', 'Emirates')	1 row(s) affected	0.0040 sec
42	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('LX', 'SWISS International Air Lines')	1 row(s) affected	0.0041 sec
43	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('OA', 'Olympic Air')	1 row(s) affected	0.0032 sec
44	14:44:28	INSERT INTO `Phase_two`.`AIRLINE` (`ARLID`, `Name`) VALUES ('UA', 'United Airlines')	1 row(s) affected	0.00094 sec

AIRPORT:

Command:

```
SELECT * FROM Phase_two.AIRPORT;
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('AIA', 'Athens International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('ATL', 'Hartsfield-Jackson Atlanta International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('DFW', 'Dallas/Fort Worth International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('LAX', 'Los Angeles International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('ORD', 'O'Hare International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('SFO', 'San Francisco International Airport');
INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('YYZ', 'Toronto Pearson International Airport');
```

Table:

ARPID	Name
AIA	Athens International Airport
ATL	Hartsfield-Jackson Atlanta International Airport
DFW	Dallas/Fort Worth International Airport
LAX	Los Angeles International Airport
ORD	O'Hare International Airport
SFO	San Francisco International Airport
▶ YYZ	Toronto Pearson International Airport

Action Output:

48	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('AIA', 'Athens International Airport')	1 row(s) affected	0.0042
49	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('ATL', 'Hartsfield-Jackson Atlanta Inter...	1 row(s) affected	0.0018
50	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('DFW', 'Dallas/Fort Worth International...	1 row(s) affected	0.0017
51	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('LAX', 'Los Angeles International Airpo...	1 row(s) affected	0.0008
52	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('ORD', 'O'Hare International Airport')	1 row(s) affected	0.0008
53	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('SFO', 'San Francisco International Air...	1 row(s) affected	0.0028
54	15:01:08	INSERT INTO `Phase_two`.`AIRPORT` (`ARPID`, `Name`) VALUES ('YYZ', 'Toronto Pearson International A...	1 row(s) affected	0.0034

ARCCLASS:

Command:

```

INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(1, 'Economy', 50.00, 171);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(1, 'First', 0.00, 16);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'Economy', 50.00, 141);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'Extra', 50.00, 30);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'First', 0.00, 16);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'Economy', 50.00, 96);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'First', 0.00, 12);
INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'PLUS', 50.00, 42);

```

Table:

1	Economy	50.00	171
1	First	0.00	16
2	Economy	50.00	141
2	Extra	50.00	30
2	First	0.00	16
3	Economy	50.00	96
3	First	0.00	12
3	PLUS	50.00	42

Action Output:

6	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(1, 'Econ...	1 row(s) affected	0.010 sec
7	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(1, 'First',...	1 row(s) affected	0.0055 sec
8	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'Econ...	1 row(s) affected	0.0051 sec
9	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'Extra...	1 row(s) affected	0.0083 sec
10	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(2, 'First...	1 row(s) affected	0.00092 sec
11	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'Econ...	1 row(s) affected	0.00080 sec
12	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'First'...	1 row(s) affected	0.00084 sec
13	15:50:18	INSERT INTO `Phase_two`.`ARCCLASS`(`ARCID`,`CLASS`,`ChangeFee`,`NumofSeats`)VALUES(3, 'PLU...	1 row(s) affected	0.00084 sec

ARLARP:

Command:

```

2 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (AA, AIA);
3 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (OA, AIA);
4 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (AA, ATL);
5 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (AA, DFW);
6 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (AA, LAX);
7 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (DL, ORD);
8 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (EK, ORD);
9 • INSERT INTO Phase_two.ARLARP (ARLID, ARPID) VALUES (AC, YYZ);

```

Table:

ARLID	ARPID
AA	AIA
OA	AIA
AA	ATL
AA	DFW
AA	LAX
DL	ORD
EK	ORD
AC	YYZ

Action Output:

18	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('AA','AIA')	1 row(s) affected	0.014 sec
19	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('OA','AIA')	1 row(s) affected	0.0054 sec
20	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('AA','ATL')	1 row(s) affected	0.0012 sec
21	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('AA','DFW')	1 row(s) affected	0.0070 sec
22	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('AA','LAX')	1 row(s) affected	0.00095 sec
23	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('DL','ORD')	1 row(s) affected	0.00094 sec
24	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('EK','ORD')	1 row(s) affected	0.00098 sec
25	16:10:06	INSERT INTO 'Phase_two'.ARLARP ('ARLID', 'ARPID')VALUES('AC','YYZ')	1 row(s) affected	0.00092 sec

FLIGHT:

Command:

```

INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (AA, 242, 1, ATL, 18:21:00, DFW, 20:50:00, 2);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (AA, 2403, 1, DFW, 14:30:00, ATL, 17:37:00, 2);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (AA, 2459, 1, DFW, 14:35:00, LAX, 16:02:00, 1);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (AA, 2733, 1, LAX, 16:40:00, DFW, 21:43:00, 1);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (AA, 2737, 1, DFW, 7:09:00, ATL, 10:19:00, 2);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (UA, 589, 1, DFW, 17:56:00, ORD, 20:23:00, 3);
INSERT INTO Phase_two.FLIGHT (ARLID, FLGID, Type, DepARPID, DepTime, DesARPID, DesTime, ARCID) VALUES (UA, 775, 1, ORD, 12:50:00, DFW, 15:28:00, 3);

```

Table:

ARLID	FLGID	Type	DepARPID	DepTime	DesARPID	DesTime	ARCID
AA	242	1	ATL	18:21:00	DFW	20:50:00	2
UA	589	1	DFW	17:56:00	ORD	20:23:00	3
UA	775	1	ORD	12:50:00	DFW	15:28:00	3
AA	2403	1	DFW	14:30:00	ATL	17:37:00	2
AA	2459	1	DFW	14:35:00	LAX	16:02:00	1
AA	2733	1	LAX	16:40:00	DFW	21:43:00	1
AA	2737	1	DFW	07:09:00	ATL	10:19:00	2

Action Output:

28	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0077 se
29	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0014 se
30	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0041 se
31	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0037 se
32	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0017 se
33	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0011 se
34	16:25:42	INSERT INTO 'Phase_two'.FLIGHT ('ARLID', 'FLGID', 'Type', 'DepARPID', 'DepTime', 'DesARPID', '...	1 row(s) affected	0.0025 se

FLGCLASS:

Command:

```

INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',242,'Economy',63.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',242,'Extra',75.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',242,'First',500.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2403,'Economy',63.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2403,'Extra',80.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2403,'First',450.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2459,'Economy',60.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2459,'First',300.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2733,'Economy',57.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2733,'First',300.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2737,'Economy',60.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2737,'Extra',75.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('AA',2737,'First',300.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',589,'Economy',90.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',589,'First',600.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',589,'Plus',110.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',775,'Economy',45.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',775,'First',250.00);
INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGID`,`CLASS`,`Fare`)VALUES('UA',775,'Plus',50.00);

```

Table:

ARLID	FLGID	CLASS	Fare
AA	242	Economy	63.00
AA	242	Extra	75.00
AA	242	First	500.00
AA	2403	Economy	63.00
AA	2403	Extra	80.00
AA	2403	First	450.00
AA	2459	Economy	60.00
AA	2459	First	300.00
AA	2733	Economy	57.00
AA	2733	First	300.00
AA	2737	Economy	60.00
AA	2737	Extra	75.00
AA	2737	First	300.00
UA	589	Economy	90.00
UA	589	First	600.00
UA	589	Plus	110.00
UA	775	Economy	45.00
UA	775	First	250.00
UA	775	Plus	50.00

Action Output:

40	16:29:20	SELECT * FROM Phase_two.FLGCLASS LIMIT 0, 1000	0 row(s) returned	0.00039 sec / 0.0
41	16:29:42	SELECT * FROM Phase_two.FLGCLASS LIMIT 0, 1000	0 row(s) returned	0.00042 sec / 0.0
42	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.011 sec
43	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0044 sec
44	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0013 sec
45	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0015 sec
46	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0042 sec
47	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0024 sec
48	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0052 sec
49	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0023 sec
50	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0042 sec
51	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0040 sec
52	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0041 sec
53	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0042 sec
54	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0012 sec
55	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0018 sec
56	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0018 sec
57	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.0040 sec
58	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.00098 sec
59	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.00079 sec
60	16:41:17	INSERT INTO `Phase_two`.`FLGCLASS`(`ARLID`,`FLGI...	1 row(s) affected	0.00083 sec

FLGDAYS:

Command:

```

SELECT FROM Phase_two.FLGDAY;
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',242,'Sat');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',242,'Sun');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2403,'Mon');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2403,'Tue');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2459,'Fri');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2459,'Thu');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2733,'Fri');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2733,'Thu');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('AA',2737,'Wed');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('UA',589,'Fri');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('UA',775,'Fri');
INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`,`DAY`)VALUES('UA',775,'Sat');

```

Table:

	ARLID	FLGID	DAY
►	AA	242	Sat
	AA	242	Sun
	AA	2403	Mon
	AA	2403	Tue
	AA	2459	Fri
	AA	2459	Thu
	AA	2733	Fri
	AA	2733	Thu
	AA	2737	Wed
	UA	589	Fri
	UA	775	Fri
	UA	775	Sat

Action Output:

63	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.012 sec
64	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0051 sec
65	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0012 sec
66	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0064 sec
67	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0083 sec
68	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0046 sec
69	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0021 sec
70	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.00100 sec
71	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0041 sec
72	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0052 sec
73	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0043 sec
74	16:55:04	INSERT INTO `Phase_two`.`FLGDAYS`(`ARLID`,`FLGID`...	1 row(s) affected	0.0049 sec

TICKET:

Command:

```
INSERT INTO `Phase_two`.`TICKET` (`TCKID`,`USRID`,`Type`,`TotalFare`,`PurchaseDateTime`)VALUES(1,1,1,9.99, '2020-04-02 01:39:43');
INSERT INTO `Phase_two`.`TICKET` (`TCKID`,`USRID`,`Type`,`TotalFare`,`PurchaseDateTime`)VALUES(2,2,1,9.99, '2020-01-23 11:18:16');
```

Table:

TCKID	USRID	Type	TotalFare	PurchaseDateTime
1	1	1	9.99	2020-04-02 01:39:43
2	2	1	9.99	2020-01-23 11:18:16

Action Output:

77	17:05:28	INSERT INTO `Phase_two`.`TICKET` (`TCKID`,`USRID`,`Type`,`TotalFare`,`PurchaseDateTime`)VAL...	1 row(s) affected	0.0079 sec
78	17:05:28	INSERT INTO `Phase_two`.`TICKET` (`TCKID`,`USRID`,`Type`,`TotalFare`,`PurchaseDateTime`)VAL...	1 row(s) affected	0.0046 sec

SEQUENCE:

Command:

```
INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,`CLASS`,`TravelDate`)VALUES(1,'UA',589,'Economy', '2020-04-23');
INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,`CLASS`,`TravelDate`)VALUES(1,'UA',775,'Plus', '2020-05-02');
INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,`CLASS`,`TravelDate`)VALUES(2,'AA',242,'Economy', '2020-03-15');
INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,`CLASS`,`TravelDate`)VALUES(2,'AA',2403,'Extra', '2020-03-13');
```

Table:

TCKID	ARLID	FLGID	CLASS	TravelDate
1	UA	589	Economy	2020-04-23
1	UA	775	Plus	2020-05-02
2	AA	242	Economy	2020-03-15
2	AA	2403	Extra	2020-03-13

Action Output:

92	17:24:40	INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,...	1 row(s) affected	0.0072 sec
93	17:24:40	INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,...	1 row(s) affected	0.0050 sec
94	17:24:40	INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,...	1 row(s) affected	0.0041 sec
95	17:24:40	INSERT INTO `Phase_two`.`SEQUENCE` (`TCKID`,`ARLID`,`FLGID`,...	1 row(s) affected	0.0061 sec

Task 3

1) Insert yourself as a New Customer. Do not provide the customer id in your query.

I used my UTA email and a fake phone number for the email and phone values.

Query Text:

```
INSERT INTO USERS(Name,email,Phone,Type)
VALUES('Katie Baumann','katherine.baumann@mavs.uta.edu','+1-123-456-7890',1);
```

Query Screenshot:

```
INSERT INTO USERS(Name,email,Phone,Type)
VALUES('Katie Baumann','katherine.baumann@mavs.uta.edu','+1-123-456-7890',1);
```

Query Output:

8	Katie Baumann	katherine.baumann@mavs.uta.edu	+1-123-456-7890	1
---	---------------	--------------------------------	-----------------	---

Action Output:

✓ 54 11:47:05 INSERT INTO USERS(Name,email,Phone,Type)VALU... 1 row(s) affected 0.125 sec

2) Update your phone number to +1-837-721-8965**Query Text:**

```
UPDATE USERS  
SET Phone='+1-837-721-8965'  
WHERE Name='Katie Baumann';
```

Query Screenshot:A screenshot of a SQL query editor showing the following text: UPDATE USERS, SET Phone='+1-837-721-8965', WHERE Name='Katie Baumann';. The text is displayed in a monospaced font with syntax highlighting: 'UPDATE' and 'WHERE' are in blue, 'USERS' is in grey, 'SET' is in blue, 'Phone' is in blue, and the phone number and name are in orange.

```
UPDATE USERS  
SET Phone='+1-837-721-8965'  
WHERE Name='Katie Baumann';
```

Query Output:

N/A

Action Output:

✓ 3 16:27:25 UPDATE USERS SET Phone='+1-837-721-8965' WHERE Name=... 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.156 sec

3) The value of TICKET.TotalFare for both records is wrong, and you need to write commands to update it.

a) Write an SQL query that returns for every line of the table SEQUENCE the flight fare.

Query Text:

```
SELECT FLGCLASS.Fare
FROM SEQUENCE
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
WHERE SEQUENCE.CLASS=FLGCLASS.CLASS;
```

Query Screenshot:

```
SELECT FLGCLASS.Fare
FROM SEQUENCE
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
WHERE SEQUENCE.CLASS=FLGCLASS.CLASS;
```

Query Output:

	Fare
▶	50.00
	63.00
	80.00
	90.00

Action Output:

✓ 63 20:14:42 SELECT FLGCLASS.Fare FROM SEQUENCE INNER J... 4 row(s) returned

0.000 sec / 0.000 sec

b) Then, write another query that will return the total fare per 'SEQUENCE' record.

Query Text:

```
SELECT TICKET.TotalFare
FROM SEQUENCE
INNER JOIN TICKET ON SEQUENCE.TCKID=TICKET.TCKID
GROUP BY TICKET.TCKID;
```

Query Screenshot:

```
SELECT TICKET.TotalFare
FROM SEQUENCE
INNER JOIN TICKET ON SEQUENCE.TCKID=TICKET.TCKID
GROUP BY TICKET.TCKID;
```

Query Output:

	TotalFare
▶	9.99
	9.99

Action Output:

✓	80	20:47:08	SELECT TICKET.TotalFare FROM SEQUENCE INNER...	2 row(s) returned	0.000 sec / 0.000 sec
---	----	----------	--	-------------------	-----------------------

c) Finally, run an update command to correct the wrong values.**Query Text:**

```

UPDATE TICKET
INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
SET TICKET.TotalFare=FLGCLASS.Fare
WHERE TICKET.TCKID=SEQUENCE.TCKID
AND SEQUENCE.FLGID=FLGCLASS.FLGID
AND SEQUENCE.CLASS=FLGCLASS.CLASS;

```

Query Screenshot:

```

UPDATE TICKET
INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
SET TICKET.TotalFare=FLGCLASS.Fare
WHERE TICKET.TCKID=SEQUENCE.TCKID
AND SEQUENCE.FLGID=FLGCLASS.FLGID
AND SEQUENCE.CLASS=FLGCLASS.CLASS;

```

Query Output:

N/A

Action Output:

✓	1	21:09:23	UPDATE TICKET INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID INNER JOIN FLGCLASS ON ...	2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0	0.265 sec
---	---	----------	---	--	-----------

d) Select all records from table 'TICKET'.

Query Text:

```
SELECT *
FROM TICKET;
```

Query Screenshot:



```
SELECT *
FROM TICKET;
```

Query Output:

	TCKID	USRID	Type	TotalFare	PurchaseDateTime
▶	1	1	1	50.00	2020-04-02 01:39:43
	2	2	1	63.00	2020-01-23 11:18:16
✱	NULL	NULL	NULL	NULL	NULL

Action Output:

<div>  </div>	<div>3 21:14:48 SELECT * FROM TICKET LIMIT 0, 1000</div>	<div>2 row(s) returned</div>	<div>0.000 sec / 0.000 sec</div>
---	--	------------------------------	----------------------------------

4) Insert a new 'TICKET' record for you, as a customer, based on the information below:

Flight: AA2459,

Departure from: Dallas/Fort Worth International Airport,

Destination: Los Angeles International Airport,

Travel date: 05/29/2020

Class: Economy, One-way

Make sure that you will update with the correct information for both tables ('TICKET' and 'SEQUENCE'). Show your work in steps.

First, I inserted a record into the TICKET table using my USRID and the information provided from the question. I used the current_timestamp() function to use the time I ran the command as the time I purchased the "ticket". I also just used 9.99 for TotalFare as a placeholder until I updated said column with correct information.

Query Text:

```
INSERT INTO TICKET(TCKID,USRID,Type,TotalFare,PurchaseDateTime)
VALUES(3,8,1,9.99,current_timestamp());
```

Query Screenshot:

```
INSERT INTO TICKET(TCKID,USRID,Type,TotalFare,PurchaseDateTime)
VALUES(3,8,1,9.99,current_timestamp());
```

Query Output:

3	8	1	9.99	2020-04-16 11:55:23
---	---	---	------	---------------------

Action Output:

60	11:55:23	INSERT INTO TICKET(TCKID,USRID,Type,TotalFare,...	1 row(s) affected	0.078 sec
----	----------	---	-------------------	-----------

Next, I inserted the rest of the information provided from the question into the SEQUENCE table.

Query Text:

```
INSERT INTO SEQUENCE(TCKID,ARLID,FLGID,CLASS,TravelDate)
VALUES(3,'AA',2459,'Economy','2020-05-29');
```

Query Screenshot:

```
INSERT INTO SEQUENCE(TCKID,ARLID,FLGID,CLASS,TravelDate)
VALUES(3,'AA',2459,'Economy','2020-05-29');
```

Query Output:

3	AA	2459	Economy	2020-05-29
---	----	------	---------	------------

Action Output:

61	12:12:12	INSERT INTO SEQUENCE(TCKID,ARLID,FLGID,CLAS...	1 row(s) affected	0.156 sec
----	----------	--	-------------------	-----------

Finally, I updated the price in TICKET.TotalFare with the correct information.

Query Text:

```
UPDATE TICKET
INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
SET TICKET.TotalFare=FLGCLASS.Fare
WHERE TICKET.TCKID=SEQUENCE.TCKID
AND SEQUENCE.FLGID=FLGCLASS.FLGID
AND SEQUENCE.CLASS=FLGCLASS.CLASS
AND TICKET.TCKID=3;
```

Query Screenshot:

```
UPDATE TICKET
INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID
SET TICKET.TotalFare=FLGCLASS.Fare
WHERE TICKET.TCKID=SEQUENCE.TCKID
AND SEQUENCE.FLGID=FLGCLASS.FLGID
AND SEQUENCE.CLASS=FLGCLASS.CLASS
AND TICKET.TCKID=3;
```

Query Output:

3	8	1	60.00	2020-04-16 11:55:23
---	---	---	-------	---------------------

Action Output:

64 12:17:02 UPDATE TICKET INNER JOIN SEQUENCE ON TICKE... 1 row(s) affected Rows matched: 1 Changed: 1 Wami... 0.141 sec

5) Return for every flight, the airline id, the flight id, the departure and destination airport description, and the aircraft description. Furthermore, on the same query, return the type, but instead of 1, return 'Domestic'. Order your results by the departure airport id.

I created a nested JOIN on tables FLIGHT, AIRPORT (2x) and AIRCRAFT to achieve a combined table with all the required fields. I used CASE to ensure that "Domestic" is printed for type = 1.

Query Text:

```
SELECT F.ARLID, F.FLGID, ARP.Name AS 'Departure Airport', AP.Name AS 'Destination
Airport', ARC.Name AS 'Aircraft Name',
       CASE WHEN F.Type = 1 THEN 'Domestic'
       ELSE 'International'
       END AS "Type"
FROM (((FLIGHT AS F JOIN AIRPORT AS ARP ON F.DepARPID = ARP.ARPID)
      JOIN AIRPORT AS AP ON F.DesARPID = AP.ARPID)
      JOIN AIRCRAFT AS ARC ON F.ARCID = ARC.ARCID)
ORDER BY F.DepARPID;
```

Query Screenshot:

```
SELECT F.ARLID, F.FLGID, ARP.Name AS "Departure Airport", AP.Name AS "Destination Airport", ARC.Name AS "Aircraft Name",
       CASE WHEN F.Type = 1 THEN "Domestic"
       ELSE "International"
       END AS "Type"
FROM (((FLIGHT AS F JOIN AIRPORT AS ARP ON F.DepARPID = ARP.ARPID)
      JOIN AIRPORT AS AP ON F.DesARPID = AP.ARPID)
      JOIN AIRCRAFT AS ARC ON F.ARCID = ARC.ARCID)
ORDER BY F.DepARPID;
```

Query Output:

	ARLID	FLGID	Departure Airport	Destination Airport	Aircraft Name	Type
▶	AA	242	Hartsfield-Jackson Atlanta International Airport	Dallas/Fort Worth Internatioal Airport	Boeing 737	Domestic
	UA	589	Dallas/Fort Worth Internatioal Airport	O'Hare International Airport	Airbus A320	Domestic
	AA	2403	Dallas/Fort Worth Internatioal Airport	Hartsfield-Jackson Atlanta International Airport	Boeing 737	Domestic
	AA	2459	Dallas/Fort Worth Internatioal Airport	Los Angeles International Airport	Airbus A321	Domestic
	AA	2737	Dallas/Fort Worth Internatioal Airport	Hartsfield-Jackson Atlanta International Airport	Boeing 737	Domestic
	AA	2733	Los Angeles International Airport	Dallas/Fort Worth Internatioal Airport	Airbus A321	Domestic
	UA	775	O'Hare International Airport	Dallas/Fort Worth Internatioal Airport	Airbus A320	Domestic

Action Output:

16 10:56:21 SELECT F.ARLID, F.FLGID, ARP.Name AS "Departure Airport", AP.Name AS "Destination Airport", ARC.Nam... 7 row(s) returned

6) Present only for the economy class a prediction of an increased fare by 5%. In your results, you need to include all classes, not only the predicted economy fare. Note that this query will only present a prediction of 5% for the economy fare. It will not update any record in your database.

I used the UNION command to combine the two cases mentioned in the question.

Query Text:

```
(SELECT CLASS, (Fare + (Fare*0.05)) AS 'Predicted Fare'
FROM FLGCLASS
WHERE CLASS = 'Economy')
UNION
(SELECT CLASS, Fare
FROM FLGCLASS
WHERE CLASS <> 'Economy');
```

Query Screenshot:

```
(SELECT CLASS, (Fare+(Fare*0.05)) AS 'Predicted Fare'
FROM FLGCLASS
WHERE CLASS = 'Economy')
UNION
(SELECT CLASS, Fare
FROM FLGCLASS
WHERE CLASS <> 'Economy');
```

Query Output:

	CLASS	Predicted Fare
▶	Economy	66.1500
	Economy	63.0000
	Economy	59.8500
	Economy	94.5000
	Economy	47.2500
	Extra	75.0000
	First	500.0000
	Extra	80.0000
	First	450.0000
	First	300.0000
	First	600.0000
	Plus	110.0000
	First	250.0000
	Plus	50.0000

Action Output:

15 14:25:16 (SELECT CLASS, (Fare+(Fare*0.05)) AS 'Predicted Fare' FROM FLGCLASS WHERE CLASS = 'Economy') UNI... 14 row(s) returned

7) Return the airline ids and the airline description for those airlines that are not associated with any airport.

I used nested queries with a NOT IN clause. I found the airline ids that were associated with an airport, and then used a NOT IN clause in the main query to get the expected output.

Query Text:

```
SELECT AL.ARLID, AL.Name
FROM AIRLINE AS AL, ARLARP AS AR
WHERE AL.ARLID NOT IN
(SELECT AR.ARLID
FROM AIRLINE AS AL, ARLARP AS AR
WHERE AL.ARLID = AR.ARLID);
```

Query Screenshot:

```
SELECT DISTINCT AL.ARLID, AL.Name
FROM AIRLINE AS AL, ARLARP AS AR
WHERE AL.ARLID NOT IN
(SELECT AR.ARLID
FROM AIRLINE AS AL, ARLARP AS AR
WHERE AL.ARLID = AR.ARLID);
```

Query Output:

	ARLID	Name
▶	LX	SWISS International Air Lines
	UA	United Airlines

Action Output:

5 09:02:26 SELECT DISTINCT AL.ARLID, AL.Name FROM AIRLINE AS AL, ARLARP AS AR WHERE AL.ARLID NOT IN ... 2 row(s) returned

8) Return the airline id, airline description, and the associated airports' ids and descriptions. For the airlines that are not associated with any airport, instead of having blank cells, type 'N/A'. (Hint: There is a DBMS function that replaces empty values).

I created a nested LEFT OUTER JOIN on tables AIRPORT, ARLARP, and AIRPORT to produce a table that was combined by the airline ids and airport ids. I used the IFNULL() function to replace any empty values in airport id and airport name. I also renamed the column names of airline id, airline description, airport id, and airport description to Airline ID, Airline Name, Airport ID, Airport Name respectively.

Query Text:

```
SELECT AR.ARLID AS 'Airport ID', AR.Name AS 'Airport Name', IFNULL(AP.ARPID, 'N/A') AS 'Airport ID', IFNULL(AP.Name, 'N/A') AS 'Airport Name'
FROM ((AIRLINE AS AR LEFT OUTER JOIN ARLARP AS ARP ON AR.ARLID = ARP.ARLID)
      LEFT OUTER JOIN AIRPORT AS AP ON ARP.ARPID = AP.ARPID);
```

Query Screenshot:

```
SELECT AR.ARLID AS "Airline ID", AR.Name AS "Airline Name", IFNULL(AP.ARPID, 'N/A') AS "Airport ID", IFNULL(AP.Name, 'N/A') AS "Airport Name"
FROM ((AIRLINE AS AR LEFT OUTER JOIN ARLARP AS ARP ON AR.ARLID = ARP.ARLID) LEFT OUTER JOIN AIRPORT AS AP ON ARP.ARPID = AP.ARPID);
```

Query Output:

	Airline ID	Airline Name	Airport ID	Airport Name
▶	AA	American Airlines	AIA	Athens International Airport
	AA	American Airlines	ATL	Hartsfield-Jackson Atlanta International Airport
	AA	American Airlines	DFW	Dallas/Fort Worth Internatioal Airport
	AA	American Airlines	LAX	Los Angeles International Airport
	AC	Air Canada	YYZ	Toronto Pearson Interantional Airport
	DL	Delta Air Lines	ORD	O'Hare International Airport
	EK	Emirates	ORD	O'Hare International Airport
	LX	SWISS International Air Lines	N/A	N/A
	OA	Olympic Air	AIA	Athens International Airport
	UA	United Airlines	N/A	N/A

Action Output:

```
12 10:27:23 SELECT AR.ARLID AS "Airline ID", AR.Name AS "Airline Name", IFNULL(AP.ARPID, 'N/A') AS "Airport ID", I... 10 row(s) returned
```


9) Write a query that will calculate for all tickets how many days left until the flight date. Ignore past tickets. (Hint: Do not hardcode today's day. There is a DBMS functionality that automatically returns the today's date).

I used the DBMS function CURDATE() to obtain the current date.

Query Text:

```
SELECT S.TravelDate - CURDATE() AS 'Days Left'
FROM SEQUENCE S, TICKET T
WHERE S.TCKID = T.TCKID AND (S.TravelDate-CURDATE()) >= 0;
```

Query Screenshot:

```
SELECT S.TravelDate - CURDATE() AS "Days Left"
FROM SEQUENCE S, TICKET T
WHERE S.TCKID = T.TCKID AND (S.TravelDate-CURDATE()) >= 0;
```

Query Output:

	Days Left
▶	7
	85
	112

Action Output:

2 13:45:28 SELECT S.TravelDate - CURDATE() AS "Days Left" FROM SEQUENCE S, TICKET T WHERE S.TCKID = T.T... 3 row(s) returned 0.015 sec / 0.000 sec

10) Write a query that will return the airline id, flight id, and all aircraft classes, along with the number of seats, the change fee and the fare. Order your results by the aircraft id.

Query Text:

```
SELECT FG.ARLID, FG.FLGID, AC.CLASS, AC.NumofSeats, AC.ChangeFee, FG.Fare
FROM ((FLGCLASS AS FG JOIN FLIGHT AS F ON F.FLGID=FG.FLGID)
JOIN ARCCCLASS AS AC ON AC.CLASS=FG.CLASS)
WHERE F.FLGID=FG.FLGID AND AC.ARCID=F.ARCID
ORDER BY AC.ARCID;
```

Query Screenshot:

```
SELECT FG.ARLID, FG.FLGID, FG.CLASS, AC.NumofSeats, AC.ChangeFee, FG.Fare
FROM ((FLGCLASS AS FG JOIN FLIGHT AS F ON F.FLGID=FG.FLGID)
JOIN ARCCCLASS AS AC ON AC.CLASS=FG.CLASS)
WHERE F.FLGID=FG.FLGID AND AC.ARCID=F.ARCID
ORDER BY AC.ARCID;
```

Query Output:

	ARLID	FLGID	CLASS	NumofSeats	ChangeFee	Fare
▶	AA	2459	Economy	171	50.00	60.00
	AA	2459	First	16	0.00	300.00
	AA	2733	Economy	171	50.00	57.00
	AA	2733	First	16	0.00	300.00
	AA	242	Economy	141	50.00	63.00
	AA	242	Extra	30	50.00	75.00
	AA	242	First	16	0.00	500.00
	AA	2403	Economy	141	50.00	63.00
	AA	2403	Extra	30	50.00	80.00
	AA	2403	First	16	0.00	450.00
	AA	2737	Economy	141	50.00	60.00
	AA	2737	Extra	30	50.00	75.00
	AA	2737	First	16	0.00	300.00
	UA	589	Economy	96	50.00	90.00
	UA	589	First	12	0.00	600.00
	UA	589	Plus	42	50.00	110.00
	UA	775	Economy	96	50.00	45.00
	UA	775	First	12	0.00	250.00
	UA	775	Plus	42	50.00	50.00

Action Output:

18 15:06:52 SELECT FG.ARLID, FG.FLGID, FG.CLASS, AC.NumofSeats, AC.C... 19 row(s) returned 0.000 sec / 0.000 sec

Query Output and Action Output:

100%	7:7	Result Grid	Filter Rows: Search	Edit: [Icons]	Export/Import: [Icons]	Result Grid
USRID	Name	email	Phone	Type		
1	Lula Wiley	wiley@gmail.com	+1-202-555-0177	1		
2	Isabell Horn	horn@gmail.com	+1-202-555-0136	1		
3	Usman Hook	hook@hotmail.com	+1-219-555-0126	1		
4	Abdullah Singleton	singleton@outlook.com	+1-404-555-0199	1		
5	Sumaiya Dean	dean@yahoo.com	+1-512-555-0161	1		
6	August Phillips	phillips@gmail.com	+1-213-555-0128	1		
7	Student Test	test@mavs.uta.edu	+1-837-721-8965	1		
8	Alexander S. Smith	AlexanderSSmith@airline.com	+1-818-617-5676	2		
9	Dorothy J. Gregory	DorothyJGregory@airline.com	+1-360-302-4081	1		
10	Theresa G. Hathaway	TheresaGHathaway@airline.com	+1-301-609-7990	2		
11	Brandon A. Garcia	BrandonAGarcia@airline.us	+1-763-477-7105	3		
12	Helen D. Cole	HelenDCole@airline.com	+1-407-685-8024	3		
13	Iliana J. Manley	IlianaJManley@airline.com	+1-774-453-9545	2		
14	Christal H. Rodriguez	CRodriguez@teleworm.us	+1-215-627-8663	1		
NULL	NULL	NULL	NULL	NULL		

USERS 3 Apply Revert

Time	Action	Response	Duration / Fetch Time
41 14:52:04	UPDATE USERS AS U SET Active=0 WHERE U.Type...	2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0	0.015 sec
42 14:54:18	ALTER TABLE USERS DROP COLUMN Active	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.238 sec
43 14:54:26	SELECT * FROM Flight_Reservations2020Ph3.USER...	14 row(s) returned	0.00033 sec / 0.000...
44 14:54:41	SELECT * FROM Flight_Reservations2020Ph3.USER...	14 row(s) returned	0.00040 sec / 0.000...
45 14:54:41	ALTER TABLE USERS ADD COLUMN Active SMALLI...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.029 sec
46 14:54:41	SET SQL_SAFE_UPDATES=0	0 row(s) affected	0.00023 sec
47 14:54:41	UPDATE USERS AS U SET Active=1 WHERE U.USRI...	7 row(s) affected Rows matched: 7 Changed: 7 Warnings: 0	0.0082 sec
48 14:54:41	UPDATE USERS AS U SET Active=0 WHERE U.Type...	2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0	0.0021 sec
49 14:55:57	SELECT * FROM Flight_Reservations2020Ph3.USER...	14 row(s) returned	0.00056 sec / 0.000...

Task 1b: For this task, you need to return the percentage of active customers.

Query Text:

```
SET @Total_Active=(SELECT COUNT(Active)
                    FROM USERS
                    WHERE Active=1);
```

```
SELECT (@Total_Active/COUNT(*)) AS Percent_Active
FROM USERS;
```

Query Screenshot, Query Output and Action Output:

23	SET @Total_Active=(SELECT COUNT(Active)
24	FROM USERS
25	WHERE Active=1);
26	
27	SELECT (@Total_Active/COUNT(*)) AS Percent_Active
28	FROM USERS;
29	
30	
31	
32	

100%	1:15	Result Grid	Filter Rows: Search	Export: [Icon]	Result Grid
Percent_Active					
0.5000					

Result 9 Read Only

Time	Action	Response	Duration / Fetch Time
1 16:00:54	SET @Total_Active=(SELECT COUNT(Active)...	0 row(s) affected	0.0033 sec
2 16:00:54	SELECT (@Total_Active/COUNT(*)) AS Percent_Acti...	1 row(s) returned	0.00071 sec / 0.0000...

Task 2: By an error to the system, the 'Type' attribute from the table 'Ticket' is set to '1'(One-way) for all trips. You need to write a query to fix this problem. Your task is to update the 'Type' attribute from table TICKET to the corresponding number {1: One-way, 2: Round-Trip}. An easy way to do that is by counting how many flights booked per ticket.

Query Text:

```
UPDATE TICKET
SET Type=2
WHERE TCKID IN (SELECT TCKID
                FROM SEQUENCE
                GROUP BY TCKID
                HAVING COUNT(TCKID) >=2);
```

Query Screenshot, Query Output and Action Output:

UPDATE TICKET
SET Type=2
WHERE TCKID IN (SELECT TCKID
FROM SEQUENCE
GROUP BY TCKID
HAVING COUNT(TCKID) >=2);

TCKID	USRID	Type	TotalFare	PurchaseDateTime
1	1	2	140.00	2020-04-02 01:39:43
2	2	2	143.00	2020-01-23 11:18:16
3	7	1	60.00	2020-04-14 00:18:43
4	14	1	110.00	2020-01-18 09:19:16
5	14	1	45.00	2020-01-18 10:23:09
6	6	2	155.00	2019-12-28 08:45:30
7	3	2	135.00	2019-11-18 08:31:06
8	9	1	45.00	2020-04-18 01:34:26

TICKET 1

Apply Revert

	Time	A. Response	Duration / Fetch Time
✓ 11	16:31:02	U 4 row(s) affected Rows matched: 4 Changed: 4 Warnings: 0	0.034 sec
✓ 12	16:31:10	S 8 row(s) returned	0.0014 sec / 0.00003...
✓ 13	16:31:18	S 12 row(s) returned	0.0016 sec / 0.00001...

Task 3: Here you need to create a view named as 'viewTicketInfo', that retrieves all useful information associated with a ticket. Specifically, this view should have the following attributes:

- ticketPurchaseDate (Keep only the date part) – in an ascending order
- ticketID • ticketType – if it is one-way or round trip. Your query needs to return the description of the type
- custName
- custPhone
- flightID – in one cell
- travelDate
- flightClass
- flightFare – You need to append the US dollar sign (\$) before the amount

Query Text:

```
CREATE VIEW viewTicketInfo(ticketPurchaseDate, ticketID, ticketType, custName, custPhone,
                           flightID, travelDate, flightClass, flightFare)
AS SELECT    date(T.PurchaseDateTime), T.TCKID, IF(T.Type=1,'Round-Trip','One-Way'),
              U.Name,U.Phone, concat(S.ARLID,S.FLGID), S.TravelDate, S.CLASS,
              concat('$',T.TotalFare)
FROM          TICKET AS T, USERS AS U, SEQUENCE AS S
WHERE         U.USRID=T.USRID AND S.TCKID= T.TCKID
ORDER BY     T.PurchaseDateTime ASC;
```

Query Screenshot and Action Output:

The screenshot shows a database query editor with a SQL query to create a view named 'viewTicketInfo'. The query is as follows:

```
CREATE VIEW viewTicketInfo(ticketPurchaseDate, ticketID, ticketType, custName, custPhone, flightID, travelDate, flightClass, flightFare)
AS SELECT    date(T.PurchaseDateTime), T.TCKID, IF(T.Type=1,'Round-Trip','One-Way'), U.Name,U.Phone, concat(S.ARLID,S.FLGID), S.TravelDate, S.CLASS,
              concat('$',T.TotalFare)
FROM          TICKET AS T, USERS AS U, SEQUENCE AS S
WHERE         U.USRID=T.USRID AND S.TCKID= T.TCKID
ORDER BY     T.PurchaseDateTime ASC;
```

Below the query editor, the 'Action Output' tab is selected, showing the following action:

	Time	Action	Response
1	18:10:10	DROP VIEW `Flight_Reservatio...	0 row(s) affected

```
SELECT * FROM Flight_Reservations2020Ph3.viewticketinfo;
```

Query Output:

1	SELECT * FROM Flight_Reservations2020Ph3.viewticketinfo;							
100%	1:1							
Result Grid	Filter Rows:	Search	Export:					
ticketPurchaseDate	ticketID	ticketType	custName	custPhone	flightID	travelDate	flightClass	flightFare
2019-11-18	7	One-Way	Usman Hook	+1-219-555-0126	UA589	2020-04-24	Economy	\$135.00
2019-11-18	7	One-Way	Usman Hook	+1-219-555-0126	UA775	2020-05-02	Economy	\$135.00
2019-12-28	6	One-Way	August Phillips	+1-213-555-0128	UA589	2020-04-24	Plus	\$155.00
2019-12-28	6	One-Way	August Phillips	+1-213-555-0128	UA775	2020-05-02	Economy	\$155.00
2020-01-18	4	Round-Trip	Christal H. Rodriguez	+1-215-627-8663	UA589	2020-04-24	Plus	\$110.00
2020-01-18	5	Round-Trip	Christal H. Rodriguez	+1-215-627-8663	UA775	2020-05-02	Economy	\$45.00
2020-01-23	2	One-Way	Isabell Horn	+1-202-555-0136	AA242	2020-03-15	Economy	\$143.00
2020-01-23	2	One-Way	Isabell Horn	+1-202-555-0136	AA2403	2020-03-13	Extra	\$143.00
2020-04-02	1	One-Way	Lula Wiley	+1-202-555-0177	UA589	2020-04-24	Economy	\$140.00
2020-04-02	1	One-Way	Lula Wiley	+1-202-555-0177	UA775	2020-05-02	Plus	\$140.00
2020-04-14	3	Round-Trip	Student Test	+1-837-721-8965	AA2459	2020-05-29	Economy	\$60.00
2020-04-18	8	Round-Trip	Dorothy J. Gregory	+1-360-302-4081	UA775	2020-05-02	Economy	\$45.00
viewticketinfo 1								
Action Output								
	Time	Action		Response				
✓ 1	18:10:10	CREATE VIEW viewTicketInfo(...		0 row(s) affected				
✓ 2	18:14:57	SELECT * FROM Flight_Reserva...		12 row(s) returned				

Task 4: Write a query that retrieves data from your stored 'viewTicketInfo' view. This query needs to summarize information about the flight fare per ticket. Order your results from the maximum amount to the minimum and then by the customer name. Your query needs to return the following attributes:

- ticketPurchaseDate renamed to 'Purchase Date'
- ticketID as 'Ticket' • ticketType as 'Ticket Type'
- custName as 'Customer'
- custPhone as 'Customer Number'
- ticketFare as 'Ticket fare'

I used the SELECT command to retrieve the information required. I used the ORDER BY command to ensure that the information displayed is in the required order.

Query Text:

```
SELECT ticketPurchaseDate AS "Purchase Date", ticketID AS "Ticket", ticketType AS "Ticket Type", custName AS "Customer", custPhone AS "Customer Number", flightFare AS "Ticket Fare"
FROM viewTicketInfo
ORDER BY flightFare DESC, custName;
```

Query Screenshot:

```
SELECT ticketPurchaseDate AS "Purchase Date", ticketID AS "Ticket", ticketType AS "Ticket Type", custName AS "Customer", custPhone AS "Customer Number", flightFare AS "Ticket Fare"
FROM viewticketinfo
ORDER BY flightFare DESC, custName;
```

Query Output:

	Purchase Date	Ticket	Ticket Type	Customer	Customer Number	Ticket Fare
▶	2020-04-14	3	Round-Trip	Student Test	+1-837-721-8965	\$60.00
	2020-01-18	5	Round-Trip	Christal H. Rodriquez	+1-215-627-8663	\$45.00
	2020-04-18	8	Round-Trip	Dorothy J. Gregory	+1-360-302-4081	\$45.00
	2019-12-28	6	One-Way	August Phillips	+1-213-555-0128	\$155.00
	2019-12-28	6	One-Way	August Phillips	+1-213-555-0128	\$155.00
	2020-01-23	2	One-Way	Isabell Horn	+1-202-555-0136	\$143.00
	2020-01-23	2	One-Way	Isabell Horn	+1-202-555-0136	\$143.00
	2020-04-02	1	One-Way	Lula Wiley	+1-202-555-0177	\$140.00
	2020-04-02	1	One-Way	Lula Wiley	+1-202-555-0177	\$140.00
	2019-11-18	7	One-Way	Usman Hook	+1-219-555-0126	\$135.00
	2019-11-18	7	One-Way	Usman Hook	+1-219-555-0126	\$135.00
	2020-01-18	4	Round-Trip	Christal H. Rodriquez	+1-215-627-8663	\$110.00

Action Output:

3 11:54:27 SELECT ticketPurchaseDate AS "Purchase Date", ticketID AS "Ticket", ticketType AS "Ticket Type", custNa... 12 row(s) returned

Task 5: In this task, we want to count how many seats left for two specific flights. You need to write one query that returns:

- the flight number (merge in one cell the Airline Id + Flight Code),
- the travel date,
- Ids and names from the departure and destination airports,
- departure and arrival time,
- class description,
- the initially available number of seats,
- the total number of tickets sold, and
- the total number of available seats [Available = Initial-Sold].

In your query you need to consider the following two flights for the specific dates:

- UA589 [2020-04-24], and
- UA775 [2020-05-02].

Order your results by flight number and class. Do not forget to name your attributes to something meaningful!

I have joined tables TICKET, SEQUENCE, FLIGHT, AIRPORT (2x), ARCCCLASS in the order mentioned to obtain the necessary information. I have grouped the flight id and class to get the number of seats for the flights mentioned in the question.

Query Text:

```
SELECT concat(F.ARLID, F.FLGID) AS "Flight Number", S.TravelDate AS "Travel Date",
F.DepARPID AS "Departure Airport ID",
AR.Name AS "Departure Airport", F.DesARPID AS "Destination Airport ID", ARP.Name AS
"Destination Airport",
F.DepTime AS "Departure Time", F.DesTime AS "Arrival Time", S.CLASS AS "Class",
ARC.NumOfSeats AS "Initial Number Of Seats",
COUNT(*) AS "Tickets Sold", ARC.NumOfSeats - COUNT(*) AS "Available Seats"
FROM (((((TICKET AS T JOIN SEQUENCE AS S ON T.TCKID = S.TCKID)
      JOIN FLIGHT AS F ON (F.ARLID = S.ARLID AND F.FLGID = S.FLGID))
     JOIN AIRPORT AS AR ON F.DepARPID = AR.ARPID)
    JOIN AIRPORT AS ARP ON F.DesARPID = ARP.ARPID)
   JOIN ARCCCLASS AS ARC ON (F.ARCID = ARC.ARCID AND S.CLASS = ARC.CLASS))
WHERE S.ARLID = "UA" AND ((S.FLGID = "589" AND S.TravelDate = "2020-04-24") OR
(S.FLGID = "775" AND S.TravelDate = "2020-05-02"))
GROUP BY S.FLGID, S.CLASS
ORDER BY S.FLGID, S.CLASS;
```

Query Screenshot:

```
SELECT concat(F.ARLID, F.FLGID) AS "Flight Number", S.TravelDate AS "Travel Date", F.DepARPID AS "Departure Airport ID",
AR.Name AS "Departure Airport", F.DesARPID AS "Destination Airport ID", ARP.Name AS "Destination Airport",
F.DepTime AS "Departure Time", F.DesTime AS "Arrival Time", S.CLASS AS "Class", ARC.NumOfSeats AS "Initial Number Of Seats",
COUNT(*) AS "Tickets Sold", ARC.NumOfSeats - COUNT(*) AS "Available Seats"
FROM (((TICKET AS T JOIN SEQUENCE AS S ON T.TCKID = S.TCKID)
      JOIN FLIGHT AS F ON (F.ARLID = S.ARLID AND F.FLGID = S.FLGID))
      JOIN AIRPORT AS AR ON F.DepARPID = AR.ARPID)
      JOIN AIRPORT AS ARP ON F.DesARPID = ARP.ARPID)
      JOIN ARCCCLASS AS ARC ON (F.ARCID = ARC.ARCID AND S.CLASS = ARC.CLASS))
WHERE S.ARLID = "UA" AND ((S.FLGID = "589" AND S.TravelDate = "2020-04-24") OR (S.FLGID = "775" AND S.TravelDate = "2020-05-02"))
GROUP BY S.FLGID, S.CLASS
ORDER BY S.FLGID, S.CLASS;
```

Query Output:

Flight Number	Travel Date	Departure Airport ID	Departure Airport	Destination Airport ID	Destination Airport	Departure Time	Arrival Time	Class	Initial Number Of Seats	Tickets Sold	Available Seats
UA589	2020-04-24	DFW	Dallas/Fort Worth International Airport	ORD	O'Hare International Airport	17:56:00	20:23:00	Economy	96	2	94
UA589	2020-04-24	DFW	Dallas/Fort Worth International Airport	ORD	O'Hare International Airport	17:56:00	20:23:00	Plus	42	2	40
UA775	2020-05-02	ORD	O'Hare International Airport	DFW	Dallas/Fort Worth International Airport	12:50:00	15:28:00	Economy	96	4	92
UA775	2020-05-02	ORD	O'Hare International Airport	DFW	Dallas/Fort Worth International Airport	12:50:00	15:28:00	Plus	42	1	41

Action Output:

✓ 1 14:13:07 SELECT concat(F.ARLID, F.FLGID) AS "Flight Number", S.TravelDate AS "Travel Date", F.DepARPID AS "De... 4 row(s) returned

Task 6: For this task, you need to provide statistical information for our customers. Specifically, your query needs to return information for all customers' names and phones, the total number of tickets we issued along with the total number of flights per specific customer, and the total fare value. The CFO also requested to provide information about an average flight fare estimation per customer [total fare value by the number of flights]. Also, in the same query, you need to include customers with no travel history. Order your results based on fare value and customer name.

I used LEFT OUTER JOIN on USERS, TICKET, SEQUENCE, and FLIGHT tables to achieve the desired output. I grouped by the user id to the statistics.

Query Text:

```
SELECT U.Name, U.Phone, COUNT(DISTINCT T.TCKID) AS "Number of Tickets",
COUNT(T.TCKID) AS "Number of Flights",
IFNULL(SUM(T.TotalFare), 0.00) AS "Total Fare", IFNULL(SUM(T.TotalFare)/COUNT(*),
0.000000) AS "Average Flight Fare"
FROM (((USERS AS U LEFT OUTER JOIN TICKET AS T ON U.USRID = T.USRID)
      LEFT OUTER JOIN SEQUENCE AS S ON T.TCKID = S.TCKID)
     LEFT OUTER JOIN FLIGHT AS F ON (S.ARLID = F.ARLID AND S.FLGID = F.FLGID))
GROUP BY U.USRID
ORDER BY U.Name, T.TotalFare;
```

Query Screenshot:

```
SELECT U.Name, U.Phone, COUNT(DISTINCT T.TCKID) AS "Number of Tickets", COUNT(T.TCKID) AS "Number of Flights",
IFNULL(SUM(T.TotalFare), 0.00) AS "Total Fare", IFNULL(SUM(T.TotalFare)/COUNT(*), 0.000000) AS "Average Flight Fare"
FROM (((USERS AS U LEFT OUTER JOIN TICKET AS T ON U.USRID = T.USRID)
      LEFT OUTER JOIN SEQUENCE AS S ON T.TCKID = S.TCKID)
     LEFT OUTER JOIN FLIGHT AS F ON (S.ARLID = F.ARLID AND S.FLGID = F.FLGID))
GROUP BY U.USRID
ORDER BY U.Name, T.TotalFare;
```

Query Output:

Name	Phone	Number of Tickets	Number of Flights	Total Fare	Average Flight Fare
Abdullah Singleton	+1-404-555-0199	0	0	0.00	0.000000
Alexander S. Smith	+1-818-617-5676	0	0	0.00	0.000000
August Phillips	+1-213-555-0128	1	2	310.00	155.000000
Brandon A. Garcia	+1-763-477-7105	0	0	0.00	0.000000
Christal H. Rodriguez	+1-215-627-8663	2	2	155.00	77.500000
Dorothy J. Gregory	+1-360-302-4081	1	1	45.00	45.000000
Helen D. Cole	+1-407-685-8024	0	0	0.00	0.000000
Iliana J. Manley	+1-774-453-9545	0	0	0.00	0.000000
Isabell Horn	+1-202-555-0136	1	2	286.00	143.000000
Lula Wiley	+1-202-555-0177	1	2	280.00	140.000000
Student Test	+1-837-721-8965	1	1	60.00	60.000000
Sumaiya Dean	+1-512-555-0161	0	0	0.00	0.000000
Theresa G. Hathaway	+1-301-609-7990	0	0	0.00	0.000000
Usman Hook	+1-219-555-0126	1	2	270.00	135.000000

Action Output:

2 15:12:23 SELECT U.Name, U.Phone, COUNT(DISTINCT T.TCKID) AS "Number of Tickets", COUNT(T.TCKID) AS "Nu... 14 row(s) returned

Task 7: Due to the current situation with COVID-19, our 'Online travel reservation system' CEO decided to grant to all upcoming round-trip flights (travel date after April 20, 2020) a 15% discount. You need to write a query to update only those ticket fares that fulfill those requirements.

I updated the TotalFare of the tickets that have the type = 2 (i.e., Round Trip) and whose travel date was after April 20, 2020.

Query Text:

```
UPDATE TICKET
SET TotalFare = TotalFare - (TotalFare*0.15)
WHERE Type = 2 and TCKID IN (SELECT S.TCKID
                             FROM SEQUENCE AS S
                             WHERE S.TravelDate > "2020-04-20");
```

Query Screenshot:

```
UPDATE TICKET
SET TotalFare = TotalFare - (TotalFare*0.15)
WHERE Type = 2 and TCKID IN (SELECT S.TCKID
                             FROM SEQUENCE AS S
                             WHERE S.TravelDate > "2020-04-20");
```

Query Output:

SELECT * FROM flight_reservations2020ph3.ticket;

TCKID	USRID	Type	TotalFare	PurchaseDateTime
1	1	2	119.00	2020-04-02 01:39:43
2	2	2	143.00	2020-01-23 11:18:16
3	7	1	60.00	2020-04-14 00:18:43
4	14	1	110.00	2020-01-18 09:19:16
5	14	1	45.00	2020-01-18 10:23:09
6	6	2	131.75	2019-12-28 08:45:30
7	3	2	114.75	2019-11-18 08:31:06
8	9	1	45.00	2020-04-18 01:34:26

Action Output:

14 15:46:34 UPDATE TICKET SET TotalFare = TotalFare - (TotalFare*0.15) WHERE Type = 2 and TCKID IN (SELECT S.... 3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0

Task 8: Our table that associates airline companies with airports (ARLARP) is outdated. You need to make sure that the ARLARP table utilizes all the information existing on the flight's table. In this task, you are only required to insert new records based on the flights' table information, do not delete any airline airport association.

I used the SQL phrase "NOT IN" to find the aircraft and airport associations not yet inserted into the ARLARP table and inserted them.

Query Text:

```
INSERT INTO ARLARP (ARLID,ARPID)
SELECT FLIGHT.ARLID, FLIGHT.DepARPID
FROM FLIGHT
WHERE (FLIGHT.ARLID,FLIGHT.DepARPID)NOT IN
(SELECT arlarp.arlid,arlarp.arpid
FROM ARLARP);
```

Query Screenshot:



```
INSERT INTO ARLARP (ARLID,ARPID)
SELECT FLIGHT.ARLID, FLIGHT.DepARPID
FROM FLIGHT
WHERE (FLIGHT.ARLID,FLIGHT.DepARPID)NOT IN
(SELECT arlarp.arlid,arlarp.arpid
FROM ARLARP);
```

Query Output:

	ARLID	ARPID
▶	AA	AIA
	OA	AIA
	AA	ATL
	AA	DFW
	UA	DFW
	AA	LAX
	DL	ORD
	EK	ORD
	UA	ORD
	AC	YYZ

Action Output:

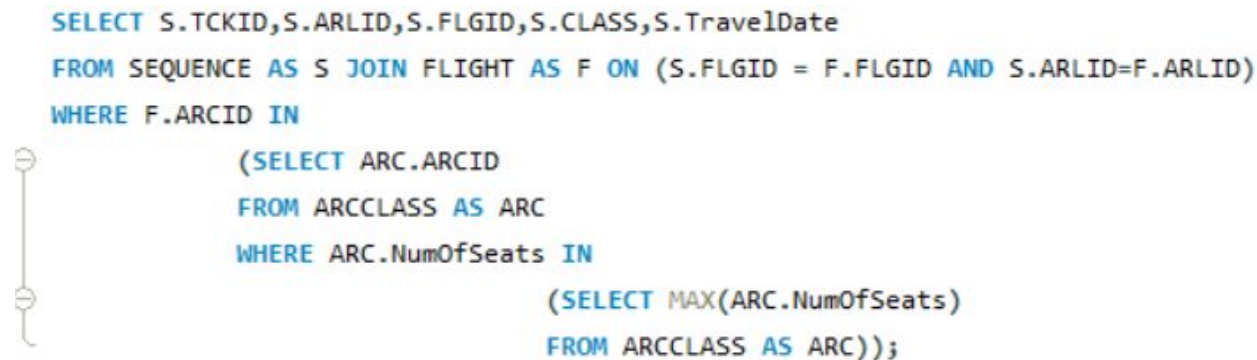
✓ 109 14:50:00 INSERT INTO ARLARP (ARLID,ARPID) SELECT FLI... 2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0 0.188 sec

Task 9: For this task, you need to write a query that returns from the table 'Sequence' the routes (booked flights) that are operated from the biggest airplane, in terms of seat capacity.

Query Text:

```
SELECT S.TCKID,S.ARLID,S.FLGID,S.CLASS,S.TravelDate
FROM SEQUENCE AS S JOIN FLIGHT AS F ON (S.FLGID = F.FLGID AND
S.ARLID=F.ARLID)
WHERE F.ARCID IN
      (SELECT ARC.ARCID
      FROM ARCCLASS AS ARC
      WHERE ARC.NumOfSeats IN
            (SELECT MAX(ARC.NumOfSeats)
            FROM ARCCLASS AS ARC));
```

Query Screenshot:



```
SELECT S.TCKID,S.ARLID,S.FLGID,S.CLASS,S.TravelDate
FROM SEQUENCE AS S JOIN FLIGHT AS F ON (S.FLGID = F.FLGID AND S.ARLID=F.ARLID)
WHERE F.ARCID IN
      (SELECT ARC.ARCID
      FROM ARCCLASS AS ARC
      WHERE ARC.NumOfSeats IN
            (SELECT MAX(ARC.NumOfSeats)
            FROM ARCCLASS AS ARC));
```

Query Output:

	TCKID	ARLID	FLGID	CLASS	TravelDate
▶	3	AA	2459	Economy	2020-05-29

Action Output:

✓ 47 17:33:15 SELECT S.TCKID,S.ARLID,S.FLGID,S.CLASS,S.Trav... 1 row(s) returned

0.000 sec / 0.000 sec

Task 10: American Airlines issued a new price policing, effective today, and we need to update our prices accordingly. The policy dictates the following:

- an increase 10% to all 'Economy' fares round up to the nearest integer value,
- an increase of 20% to 'Extra Space' class compare to the new 'Economy' fare for that specific flight, round up to the nearest integer value
- an increase of 120% for the 'First' class compare to the 'Extra' fare for that specific flight, round up to the nearest integer value

Query Text:

```
UPDATE FLGCLASS
```

```
SET FLGCLASS.Fare=ROUND(FLGCLASS.Fare+(FLGCLASS.Fare*.1))
```

```
WHERE FLGCLASS.ARLID='AA' AND FLGCLASS.Class='Economy';
```

```
SET @check=(SELECT IF(EXISTS(SELECT *
    FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
    WHERE f1.arlid='AA' AND f2.arlid='AA'
    AND (f1.class='economy' AND (f2.class='extra space' OR f2.class='plus'))),1,0));
```

```
UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare+(f1.fare*.2))
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra space' OR f2.class='plus'))
AND @check<>0;
```

```
SET @check1=(SELECT IF(EXISTS(SELECT *
    FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
    WHERE f1.arlid='AA' AND f2.arlid='AA'
    AND (f1.class='economy' AND (f2.class='extra'))),1,0));
```

```
UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare*1.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra'))
AND @check1<>0;
```

```
SET @check2=(SELECT IF(EXISTS(SELECT *
    FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
    WHERE f1.arlid='AA' AND f2.arlid='AA'
    AND (f1.class='extra' AND (f2.class='first'))),1,0));
```

```

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare*2.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='extra' AND (f2.class='first'))
AND @check2<>0;

```

```

SET @check3=(SELECT IF(EXISTS(SELECT *
    FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
    WHERE f1.arlid='AA' AND f2.arlid='AA'
    AND (f1.class='economy' AND (f2.class='first')
    AND f1.class <> 'extra' AND f2.class <> 'extra')),1,0));

```

```

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND((ROUND(f1.fare*1.2))*2.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='first'))
AND @check3<>0;

```

Query Screenshot:

```

UPDATE FLGCLASS
SET FLGCLASS.Fare=ROUND(FLGCLASS.Fare+(FLGCLASS.Fare*.1))
WHERE FLGCLASS.ARLID='AA' AND FLGCLASS.Class='Economy';

SET @check=(SELECT IF(EXISTS(SELECT *
FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra space' OR f2.class='plus'))),1,0));

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare+(f1.fare*.2))
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra space' OR f2.class='plus'))
AND @check<>0;

SET @check1=(SELECT IF(EXISTS(SELECT *
FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra'))),1,0));

```



```

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare*1.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='extra'))
AND @check1<>0;

```

① SET @check2=(SELECT IF(EXISTS(SELECT *

```

FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='extra' AND (f2.class='first'))),1,0));

```

```

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND(f1.fare*2.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='extra' AND (f2.class='first'))
AND @check2<>0;

```

① SET @check3=(SELECT IF(EXISTS(SELECT *

```

FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='first')
AND f1.class <> 'extra' AND f2.class <> 'extra'))),1,0));

```

②

```

UPDATE FLGCLASS AS F
inner join flgclass AS f1 on F.flgid=f1.flgid
inner join flgclass AS f2 on f1.flgid=f2.flgid
SET f2.Fare = ROUND((ROUND(f1.fare*1.2))*2.2)
WHERE f1.arlid='AA' AND f2.arlid='AA'
AND (f1.class='economy' AND (f2.class='first'))
AND @check3<>0;

```

Query Output:

ARLID	FLGID	CLASS	Fare
AA	242	Economy	69.00
AA	242	Extra	83.00
AA	242	First	183.00
AA	2403	Economy	69.00
AA	2403	Extra	83.00
AA	2403	First	183.00
AA	2459	Economy	66.00
AA	2459	First	174.00
AA	2733	Economy	63.00
AA	2733	First	167.00
AA	2737	Economy	66.00
AA	2737	Extra	79.00
AA	2737	First	174.00

Action Output:

✓	3	21:25:56	UPDATE FLGCLASS SET FLGCLASS.Fare=ROUND(FLGCLASS.Fare/(FLGCLASS.Fare*1)) WHERE FLGCLASS.ARLID=...	5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0	0.094 sec
✓	4	21:25:56	SET @check=(SELECT IF(EXISTS(SELECT * FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid WHERE f1.arlid=...	0 row(s) affected	0.000 sec
✓	5	21:25:56	UPDATE FLGCLASS AS F inner join flgclass AS f1 on F.flgid=f1.flgid inner join flgclass AS f2 on f1.flgid=f2.flgid SET f2.Fare ...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
✓	6	21:25:56	SET @check1=(SELECT IF(EXISTS(SELECT * FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid WHERE f1.arlid=...	0 row(s) affected	0.000 sec
✓	7	21:25:56	UPDATE FLGCLASS AS F inner join flgclass AS f1 on F.flgid=f1.flgid inner join flgclass AS f2 on f1.flgid=f2.flgid SET f2.Fare ...	3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0	0.312 sec
✓	8	21:25:56	SET @check2=(SELECT IF(EXISTS(SELECT * FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid WHERE f1.arlid=...	0 row(s) affected	0.000 sec
✓	9	21:25:56	UPDATE FLGCLASS AS F inner join flgclass AS f1 on F.flgid=f1.flgid inner join flgclass AS f2 on f1.flgid=f2.flgid SET f2.Fare ...	3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0	0.188 sec
✓	10	21:25:56	SET @check3=(SELECT IF(EXISTS(SELECT * FROM flgclass AS f1 JOIN flgclass AS f2 on f1.flgid=f2.flgid WHERE f1.arlid=...	0 row(s) affected	0.000 sec
✓	11	21:25:56	UPDATE FLGCLASS AS F inner join flgclass AS f1 on F.flgid=f1.flgid inner join flgclass AS f2 on f1.flgid=f2.flgid SET f2.Fare ...	2 row(s) affected Rows matched: 5 Changed: 2 Warnings: 0	0.078 sec