CSE 3330-002

Flight Reservations Phase 3

May 04, 2020

Team 1

Chi Shing Poon

Jacinto Mendoza

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.

Student Signature: Jacinto J. Mendoza Date: May 04 2020

Student Signature: Chi Shing Poon Date: May 04 2020

Phase 1:

Introduction:

This project's goal is to create a flight reservation system using MySQL under a set of requirements. Phase 1 of the Flight Reservation Project in CSE3330. Phase 1 begins the ER diagram design and its mapping into a relational database schema. Later phases will begin implementation of the system into an actual database with demo.

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).

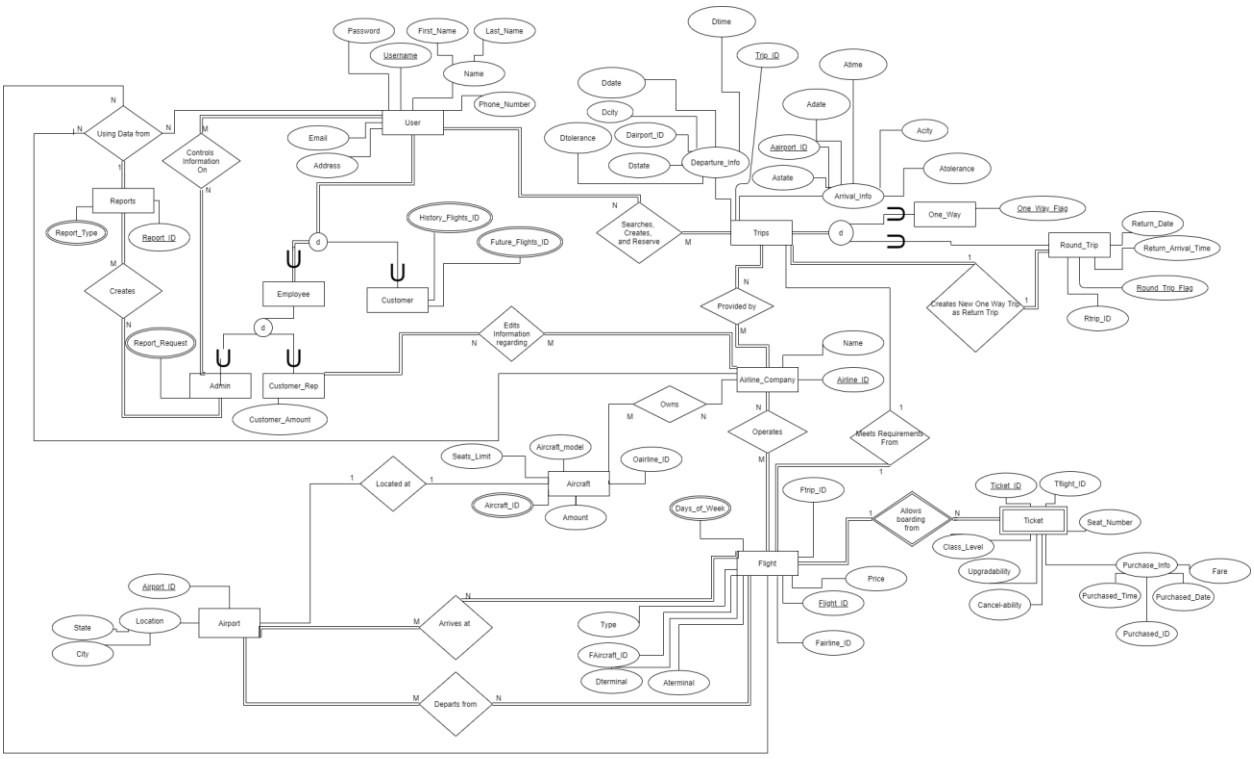


Figure 1. ERD of Flight Reservation Database

ERD Explanation/Description:

All users should be able to search, reserve and manage their flights by entering information regarding their trips. DB should store the name, some private information, and login information of all users. **Each email can only have one account associated with it.** Each User is either a customer, admin, or customer rep. (User is a superclass, with the others as subclasses) Each level has different functionalities and relationships with entities in the DB.

Employees can request different types of reports from admins. Customer Rep. has a number of customers to care for. Customers have a number of future flights they've booked.

Different admins can create many reports, multiple customer rep can care for various different customers. All users can search, create, and reserve trips; but their trips are not only limited to themselves. All reports are created by admins, and a report made by admin can be using data from various users, airlines, and/or flights, but does not have to include all users, airlines, and/or flights for each report.

All trips have departure and arrival information (Date, Time, Location, etc). Trips are either one way or round trips, with round trips containing information of the return flight. Different trips are provided by different airline companies, and they are not mutually exclusive. If a trip is a round trip, then it will have one and only one return trip.

Airline Companies operate all flights and own all aircrafts. Different companies may own different aircrafts and operate different flights. Each aircraft can only be located at one airport. All aircrafts are stored at an airport, unless they are in-flight.

Flights are scheduled at different times and prices, with different aircrafts. Each flight must match the requirements from its corresponding trip. Each Flight has a corresponding ticket that has all information regarding the trip, plus more information such as seat number, purchase date, time, and its class level (Economy, Business, or First), etc. Flights also have types, which is either international or domestic. **Flights can have multiple days of operation due to the flight being defined as the routes it takes. All flights are assumed to be direct, and without any connecting flights.** Each flight allows multiple passengers with the correct ticket, up to its aircraft's seat limit. All flights depart from and arrive at different airports, and all of them must depart and arrive from, and at, an airport.

Tickets are created to serve each flight, therefore they are dependent on the flight it's related to. Tickets for the same flight must be used only for that flight.

Attributes Explanation:

Attribute	**Entities it belongs to**	**Type**	**Explanation/Format/Constraints**
Report_Request	Admin	String	Types of reports needed to be created by the Admin. Multivariable Attribute due to the different types of reports can be made.
Amount_of_Customers	Customer_Rep	Int	Number of customers the rep takes care of
Future_Flight_ID	Customer	Int	Points to Flight_ID of future flights for this customer. Multivariable Attribute due to the possibility of multiple future flights booked.
History_Flight_ID	Customer	String	Past Flights of Customers. Multivariable Attribute due to the possibility of having various flights flown before.
Password	User	String	Self-Explanatory
Username	User	String	Self-Explanatory
First_Name	User	String	Self-Explanatory
Last_name	User	String	Self-Explanatory

Phone_Number	User	float	Self-Explanatory
Email	User	String	Self-Explanatory
Address	User	String	Self-Explanatory
Trip_ID	Trips	Int	ID of Trip with its specific requirements/Conditions
Atime	Trips	Int	24Hrs format. (Ex. 1300 = 1PM, 1440 = 2:40PM)
Aairport_ID	Trips	String	Airport_ID for this trip on Arrival
Astate	Trips	String	State for this trip on Arrival
Acity	Trips,	String	City For this trip on Arrival
Atolerance	Trips	int	Days which the users have tolerance on regarding the range around the selected dates. Min: 0, Max: 3
~~Arrival_ID~~	~~Trips~~	~~String~~	~~Corresponds to the other information in Arrival_Info. Format: A-XXXXX. XXXXX is a combination of integers~~
Adate	Trips	String	Date of arrival. MM/DD/YYYY
Ddate	Trips	String	Date of Departure. MM/DD/YYYY
Dtime	Trips	Int	24Hrs format. (Ex. 1300 = 1PM, 1440 = 2:40PM)
Dairport_ID	Trips	String	Airport_ID for this trip on Departure
Dstate	Trips	String	State for this trip on Departure
Dcity	Trips	String	City for this trip on Departure
Dtolerance	Trips	int	Days which the users have tolerance on regarding the range around the selected dates. Min: 1, Max: 3
~~Departure_ID~~	~~Trips~~	~~String~~	~~Corresponds to the other information in Departure_Info. Format: D-XXXXX. XXXXX is a combination of integers~~
One_Way_Flag	One_way	Boolean	if True, this is a one way trip
Round_Trip_Flag	Round_Trip	Boolean	if True, this is a round trip
Rtrip_ID	Round_Trip	String	Points to the trip ID of the return flight
Return_Date	Round_Trip	String	MM/DD/YYYY

Return_Arrival_Time	Round_Trip	Int	Time of arrival on Return Flight. 24Hrs format. (Ex. 1300 = 1PM, 1440 = 2:40PM)
Name	Airline_Company	String	name of Airline
Airline_ID	Airline_Company	String	Unique ID to the Airline. (Ex. AA = American Airline)
Aircraft_ID	Aircraft	String	Self-Explanatory
Oairline_ID	Aircraft	String	Points to Airline_ID of the airline that owns the aircraft
Amount	Aircraft	Int	Amount of the same aircraft model
Aircraft_model	Aircraft	String	Model Number of Aircraft
Seats_Limit	Aircraft	int	Amount of seats available in the aircraft
Airport_ID	Airport	String	Correspondence to Airport. Format: JFK, ...
State	Airport	String	Corresponds to the State of Airport. Format: TX, NY, ...
City	Airport	String	Corresponds to the City of Airport.
Flight_ID	Flight	String	Unique ID of Flight. Format: Airline_ID + XXXXX
Faircraft_ID	Flight	String	Points to Aircraft_ID used for this flight
Dterminal	Flight	String	Terminal of departure. Format: X-YY Ex. A-02, B-10
Dairport_ID	Flight	String	Correspondence to Airport. Format: JFK, ... (Points to Airport_ID)
Aterminal	Flight	String	Terminal of Arrival. Format: X-YY Ex. A-02, B-10
Aairport_ID	Flight	String	Correspondence to Airport. Format: JFK, ... (Points to Airport_ID)
Fairline_ID	Flight	String	Points to the airline_ID of the airline operating this flight
Price	Flight	Float	Price of Flight
Type	Flight	String	Type of flight. International or Domestic
Days_of_Week	Flight	String	Days of the week where this flight is operating. Format: Monday, Tuesday, … Multivariable Attribute due to how this flight can have various days of operations
Departure_Time	Flight	String	Time on Departure. 24Hrs format. (Ex. 1300 = 1PM, 1440 =

			2:40PM)
Arrival_Time	Flight	String	Time on Arrival. 24Hrs format. (Ex. 1300 = 1PM, 1440 = 2:40PM)
Ftrip_ID	Flight	String	Points to the Trip_ID used for this flight
Ticket_ID	Ticket	String	Unique ID to the Ticket
Tflight_ID	Ticket	String	Points to the flight ID that corresponds to this Ticket
Class_level	Ticket	String	Class of the ticket: Economy, Business, or First
Upgradability	Ticket	Boolean	if True, Upgradable without fees
Cancel-Ability	Ticket	Boolean	if True, cancellable without fees
Seat_Number	Ticket	String	Seat Number. Format: X-YY. Ex. A-02, B-10
Fare	Ticket	Float	Final Price on Ticket
Purchased_Time	Ticket	String	Time when ticket is purchased. 24Hrs format. (Ex. 1300 = 1PM, 1440 = 2:40PM)
Purchased_Date	Ticket	String	Date when ticket is purchased. Format: MM/DD/YYYY
Purchased_ID	Ticket	Int	Unique ID to ticket purchase order
Report_Type	Reports	String	Type of report needed for admin
Report_ID	Reports	String	Unique ID for report created. Format: R-XXXX

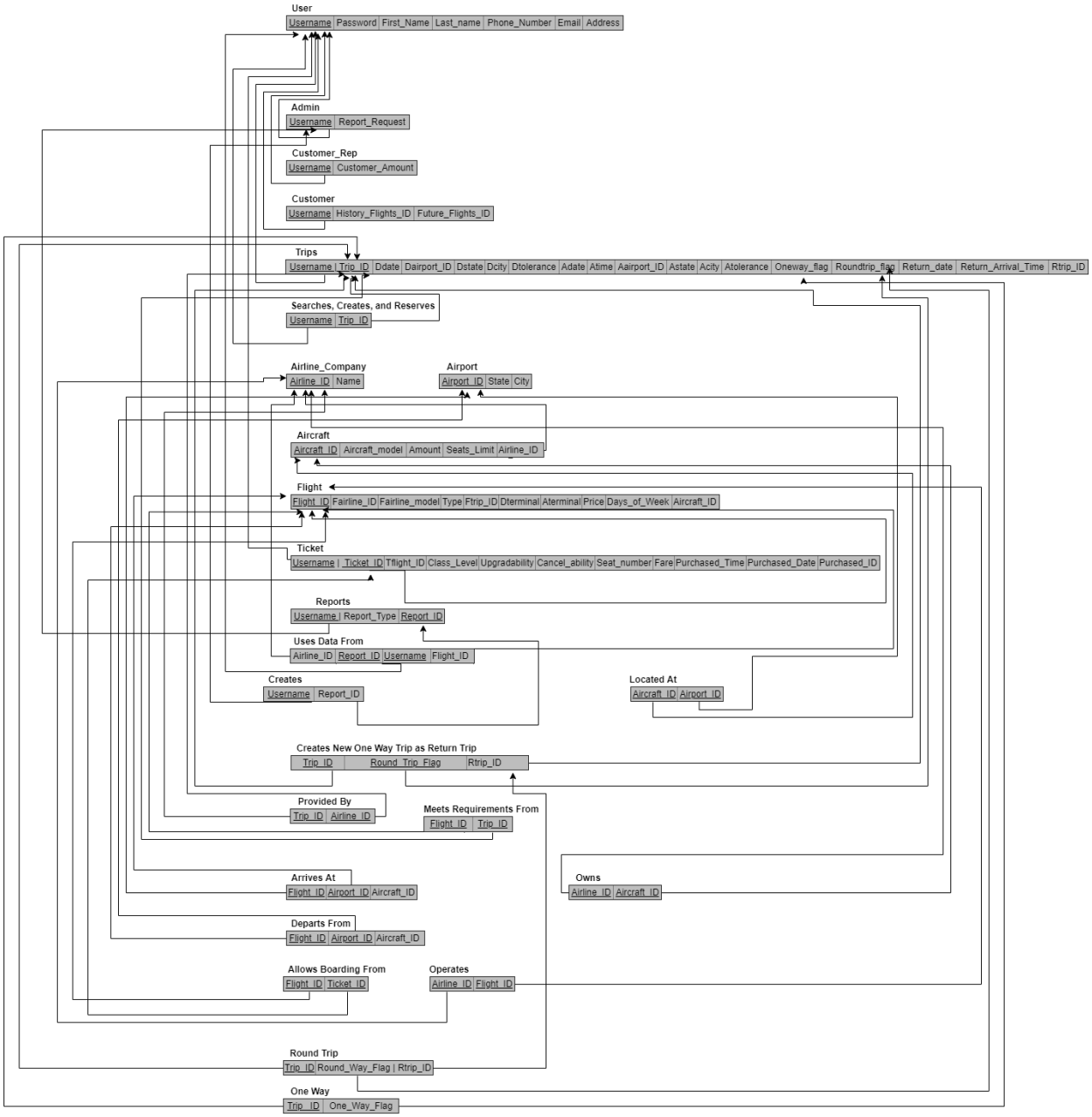


Figure 2. Relational Database Schema

Relational Schema Explanation:

The Relational Database Schema is organized mostly based on username, flight, ticket and trips, most of the tables comprising entities. The primary keys are underlined which are mostly IDs and usernames. All of the tables in the schema are at least interconnected with foreign keys from another table. All relationships are also organized within the tables with two entities being the primary and foreign key to each relationship.

Phase 2:

Task 1: Creating the Schema for the Flight Reservation Project Database

1.1 Query Code for Creating the Flight Reservation Schema

```sql
CREATE SCHEMA `flight_reservation` ;

CREATE TABLE `USERS`(

    `USRID` INT NOT NULL,
    `Name` VARCHAR(30) NOT NULL,
    `email` VARCHAR(30) NOT NULL,
    `Phone` VARCHAR(15) NULL,
    `Type` INT NOT NULL,
    PRIMARY KEY(`USRID`)
);

CREATE TABLE `AIRLINE`(

    `ARLID` VARCHAR(10) NOT NULL,
    `Name` VARCHAR(30) NOT NULL,
    PRIMARY KEY(`ARLID`)
);

CREATE TABLE `AIRCRAFT`(

    `ARCID` INT NOT NULL,
    `Name` VARCHAR(30) NOT NULL,
    `ARLID` VARCHAR(10) NOT NULL,
    PRIMARY KEY(`ARCID`),
    CONSTRAINT `ARLID`
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE RESTRICT          -- restrict as you cannot have an aircraft w/o airline
        ON UPDATE CASCADE
    );

CREATE TABLE `AIRPORT`(

    `ARPID` VARCHAR(10) NOT NULL,
    `Name` VARCHAR(50) NOT NULL,
    PRIMARY KEY(`ARPID`)
);

CREATE TABLE `ARCCLASS`(

    `ARCID` INT NOT NULL,
    `CLASS` VARCHAR(30) NOT NULL,
    `ChangeFee` DECIMAL(9,2) NULL,
    `NumofSeats` INT NULL,
    PRIMARY KEY(ARCID, CLASS),
    INDEX `CLASS` (`CLASS` ASC) VISIBLE,
    CONSTRAINT `ARCID`
        FOREIGN KEY (`ARCID`)
        REFERENCES `AIRCRAFT` (`ARCID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
```

```sql
);

CREATE TABLE `ARLARP`(

    `ARLID` VARCHAR(10) NOT NULL,
    `ARPID` VARCHAR(30) NOT NULL,
    PRIMARY KEY(ARLID, ARPID),
    CONSTRAINT `ARLID_ARLARP`
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `ARPID`
        FOREIGN KEY (`ARPID`)
        REFERENCES `AIRPORT` (`ARPID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE `FLIGHT`(

    `ARLID` VARCHAR(10) NOT NULL,
    `FLGID` INT NOT NULL,
    `Type` INT NOT NULL,
    `DepARPID` VARCHAR(10) NOT NULL,
    `DepTime` TIME NOT NULL,
    `DesARPID` VARCHAR(10) NOT NULL,
    `DesTime` TIME NOT NULL,
    `ARCID` INT NOT NULL,
    PRIMARY KEY(ARLID,FLGID),
    INDEX `FLGID` (`FLGID` ASC) VISIBLE,
    INDEX `Type` (`Type` ASC) VISIBLE,
    CONSTRAINT `ARLID_FLIGHT`
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `ARCID_FLIGHT`
        FOREIGN KEY (`ARCID`)
        REFERENCES `AIRCRAFT` (`ARCID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE `FLGCLASS`(

    `ARLID` VARCHAR(10) NOT NULL,
    `FLGID` INT NOT NULL,
    `CLASS` VARCHAR(30) NOT NULL,
    `Fare` DECIMAL(9,2) NOT NULL,
    PRIMARY KEY(ARLID,FLGID,CLASS),
    CONSTRAINT `ARLID_FLGCLASS`
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `FLGID`
        FOREIGN KEY (`FLGID`)
        REFERENCES `FLIGHT` (`FLGID`)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
```

```sql
    CONSTRAINT `CLASS`
        FOREIGN KEY (`CLASS`)
        REFERENCES `ARCCLASS` (`CLASS`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE `FLGDAYS`(

    `ARLID` VARCHAR(10) NOT NULL,
    `FLGID` INT NOT NULL,
    `DAY` VARCHAR(3),
    PRIMARY KEY(ARLID,FLGID,DAY),
    CONSTRAINT `ARLID_FLGDAYS`
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `FLGID_FLGDAYS`
        FOREIGN KEY (`FLGID`)
        REFERENCES `FLIGHT` (`FLGID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE `TICKET`(

    `TCKID` INT NOT NULL AUTO_INCREMENT,  -- increments tickets by 1 for each new one
    `USRID` INT NOT NULL,
    `Type` INT NOT NULL,
    `TotalFare` DECIMAL(9,2) NOT NULL, -- tickets are no more than $9m and have cents
    `PurchaseDateTime` TIMESTAMP DEFAULT NULL,
    PRIMARY KEY(TCKID),
    CONSTRAINT `USRID`
        FOREIGN KEY (`USRID`)
        REFERENCES `USERS` (`USRID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `Type`
        FOREIGN KEY (`Type`)
        REFERENCES `FLIGHT` (`Type`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    );


CREATE TABLE `SEQUENCE`(

    `TCKID` INT NOT NULL,
    `ARLID` VARCHAR(30) NOT NULL,
    `FLGID` INT NOT NULL,
    `CLASS` VARCHAR(30) NOT NULL,
    `TravelDate` DATE NULL,
    PRIMARY KEY(TCKID,ARLID,FLGID),                -- primary composite key for multiple
primary keys
    CONSTRAINT `TCKID`
        FOREIGN KEY (`TCKID`)
        REFERENCES `TICKET` (`TCKID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `ARLID_SEQUENCE`
```

```
        FOREIGN KEY (`ARLID`)
        REFERENCES `AIRLINE` (`ARLID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `FLGID_SEQUENCE`
        FOREIGN KEY (`FLGID`)
        REFERENCES `FLIGHT` (`FLGID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT `CLASS_SEQUENCE`
        FOREIGN KEY (`CLASS`)
        REFERENCES `ARCCLASS` (`CLASS`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

1.2 Description of Creating the Schema

The above schema was created based on the table headers specified on the project handout. Each table are coordinated to one and another based on the foreign keys that are that are members of other tables, most of which are the primary keys of the parent table. If they a foreign key isn't a primary key of any table, then the methodology of what's decided as the parent table for this column is based on what needs to exist in order for the other table to exist, for example the table, TICKET, references FLIGHT for 'Type' as a ticket can not exist without a flight, therefore flight is the parent table of ticket for the 'Type' column. Because of the foreign key constraints, the CREATE TABLE commands were made in an order from top to bottom the tables that can exist without other tables firstly, the parent tables, and the child tables at the bottom.

All attributes that are the primary keys for each relation have a NOT NULL as a tuple cannot exist without such primary key being specified; this same NOT NULL rule applies to primary composite keys as well. Other attributes that are not primary keys are given NULL values including those that are foreign keys.

Most attributes use ON DELETE CASCADE and ON UPDATE CASCADE as most of foreign key attributes on the child relation can simply be modified when changing the attribute on the parent relation.

Task 2: Loading Data to the Corresponding Tables

2.1 Description of Inserting Data into the Corresponding Tables

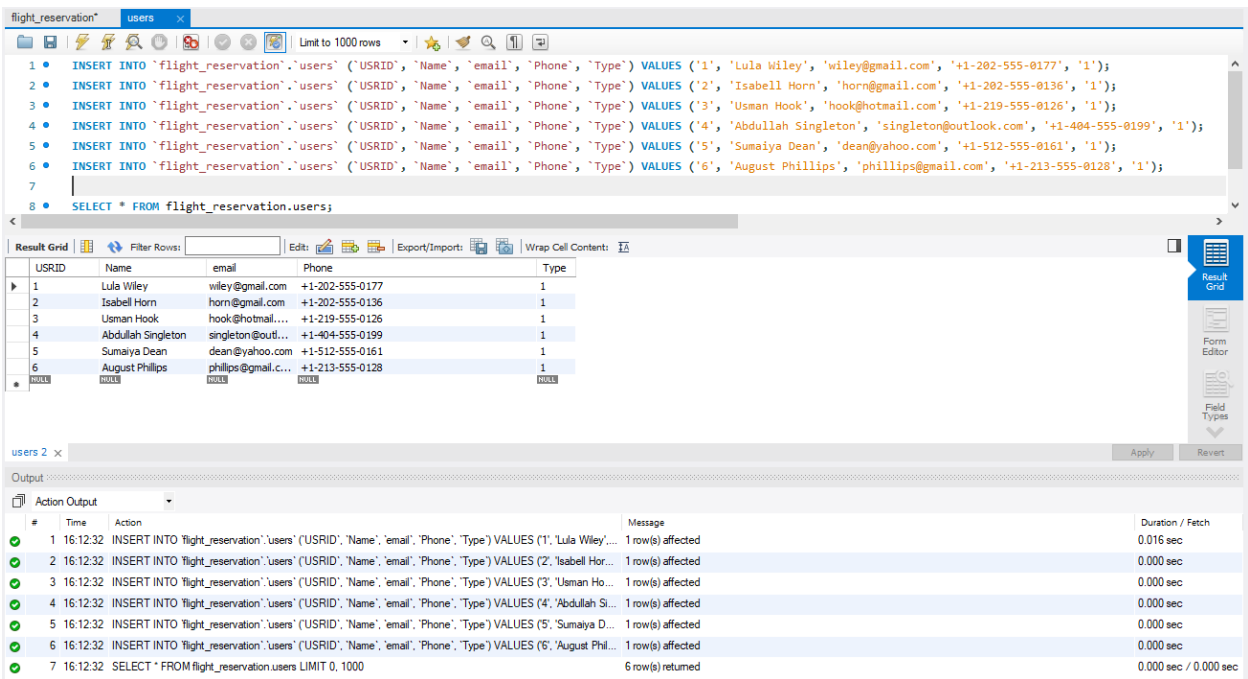
All the data was taken from the handout's and placed into comma separated value (CSV) files. With the data in proper format in the given CSV files, all rows except for title row was highlighted from the file using Notepad++ and taken into MySQL Workbench. From here, the corresponding table was selected using the option, 'Select Rows' and down at where it displays the result grid, the data is then pasted at the blank space left of the actual table on the first row. The 'Apply' option is then used to finish applying the data of the given table by executing INSERT INTO commands. This process is repeated for the other remaining empty tables. Additionally, typing these commands on the query was also used, but not as extensively as the apply method.

While all is now correct and functional, there were initial challenges being faced when loading the data, such as incorrect format of the tables. The tables were eventually modified and updated to correctly accommodate the data being inserted.

2.2 Data Insertion of Every Table

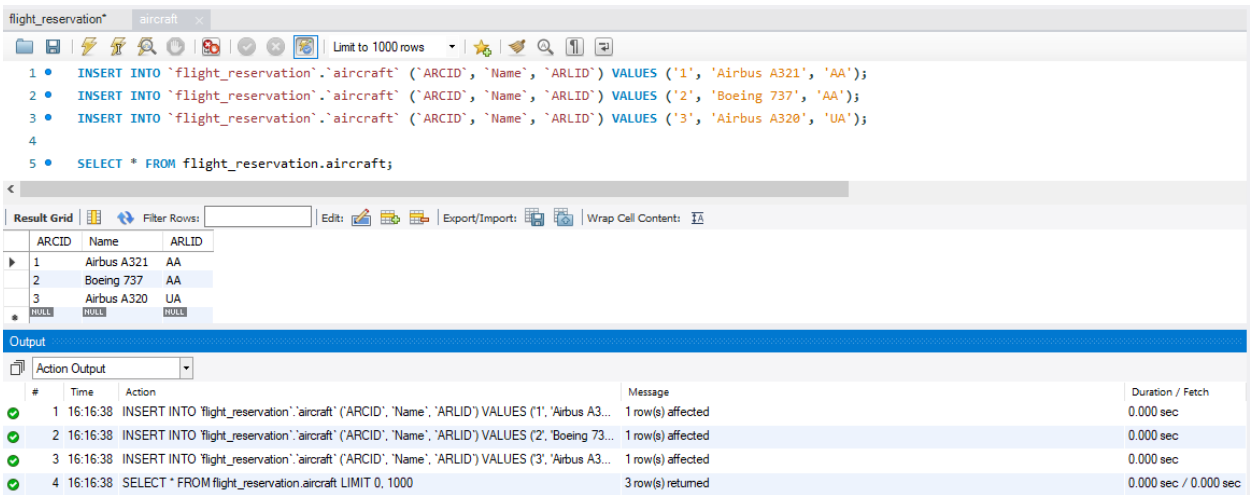
USERS

```sql
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('1', 'Lula Wiley', 'wiley@gmail.com', '+1-202-555-0177', '1');
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('2', 'Isabell Horn', 'horn@gmail.com', '+1-202-555-0136', '1');
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('3', 'Usman Hook', 'hook@hotmail.com', '+1-219-555-0126', '1');
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('4', 'Abdullah Singleton', 'singleton@outlook.com', '+1-404-555-0199', '1');
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('5', 'Sumaiya Dean', 'dean@yahoo.com', '+1-512-555-0161', '1');
INSERT INTO `flight_reservation`.`users` (`USRID`, `Name`, `email`, `Phone`, `Type`) VALUES ('6', 'August Phillips', 'phillips@gmail.com', '+1-213-555-0128', '1');

SELECT * FROM flight_reservation.users;
```



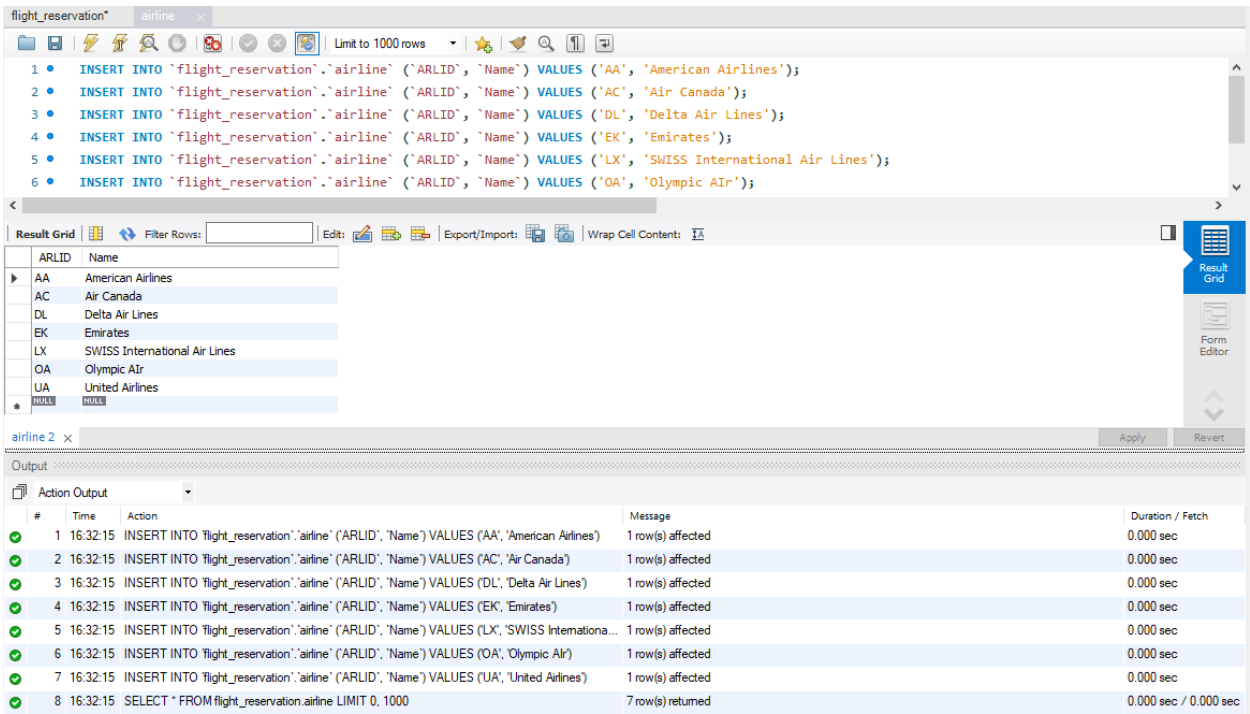
AIRCRAFT

```
INSERT INTO `flight_reservation`.`aircraft` (`ARCID`, `Name`, `ARLID`) VALUES ('1', 'Airbus A
321', 'AA');
INSERT INTO `flight_reservation`.`aircraft` (`ARCID`, `Name`, `ARLID`) VALUES ('2', 'Boeing 7
37', 'AA');
INSERT INTO `flight_reservation`.`aircraft` (`ARCID`, `Name`, `ARLID`) VALUES ('3', 'Airbus A
320', 'UA');

SELECT * FROM flight_reservation.aircraft;
```



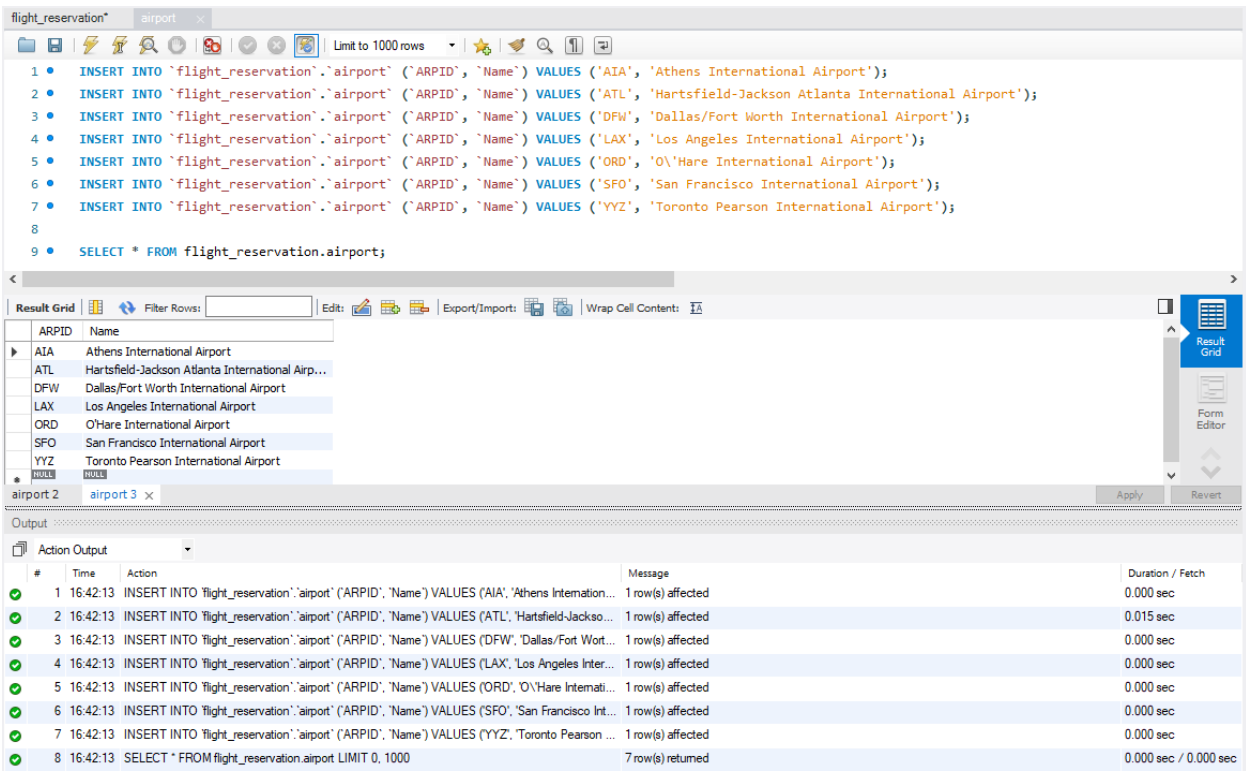
AIRLINE

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



AIRPORT

```
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('AIA', 'Athens Internatio
nal Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('ATL', 'Hartsfield-
Jackson Atlanta International Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('DFW', 'Dallas/Fort Worth
 International Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('LAX', 'Los Angeles Inter
national Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('ORD', 'O\'Hare Internati
onal Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('SFO', 'San Francisco Int
ernational Airport');
INSERT INTO `flight_reservation`.`airport` (`ARPID`, `Name`) VALUES ('YYZ', 'Toronto Pearson I
nternational Airport');

SELECT * FROM flight_reservation.airport;
```



ARCCLASS

```
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('1', 'Economy', '50.00', '171');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('1', 'First', '0.00', '16');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'Economy', '50.00', '114');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'Extra', '50.00', '30');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'First', '0.00', '16');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'Economy', '50.00', '96');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'First', '0.00', '12');
INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'Plus', '50.00', '42');

SELECT * FROM flight_reservation.arcclass;
```

flight_reservation* | arcclass

Limit to 1000 rows

```
1  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('1', 'Economy', '50.00', '171');
2  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('1', 'First', '0.00', '16');
3  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'Economy', '50.00', '114');
4  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'Extra', '50.00', '30');
5  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('2', 'First', '0.00', '16');
6  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'Economy', '50.00', '96');
7  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'First', '0.00', '12');
8  INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) VALUES ('3', 'Plus', '50.00', '42');
9
10 SELECT * FROM flight_reservation.arcclass;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ARCID	CLASS	ChangeFee	NumofSeats
1	Economy	50.00	171
1	First	0.00	16
2	Economy	50.00	114
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
NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types

arcclass 2 | Apply | Revert

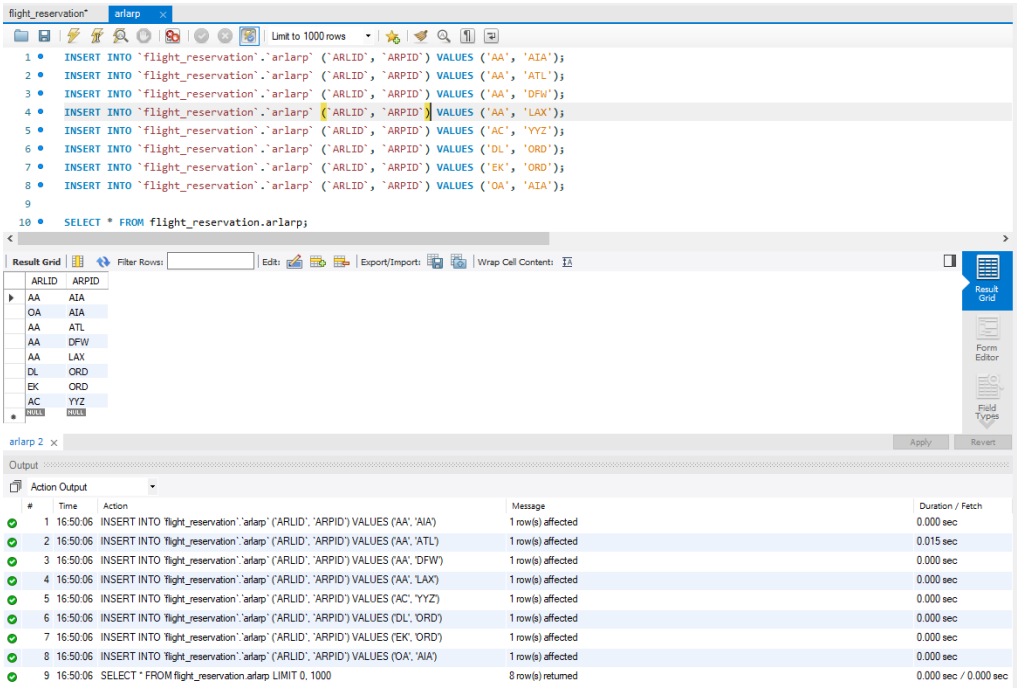
Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
2	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
3	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
4	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
5	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
6	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
7	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
8	16:46:53	INSERT INTO `flight_reservation`.`arcclass` (`ARCID`, `CLASS`, `ChangeFee`, `NumofSeats`) V...	1 row(s) affected	0.000 sec
9	16:46:53	SELECT * FROM flight_reservation.arcclass LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

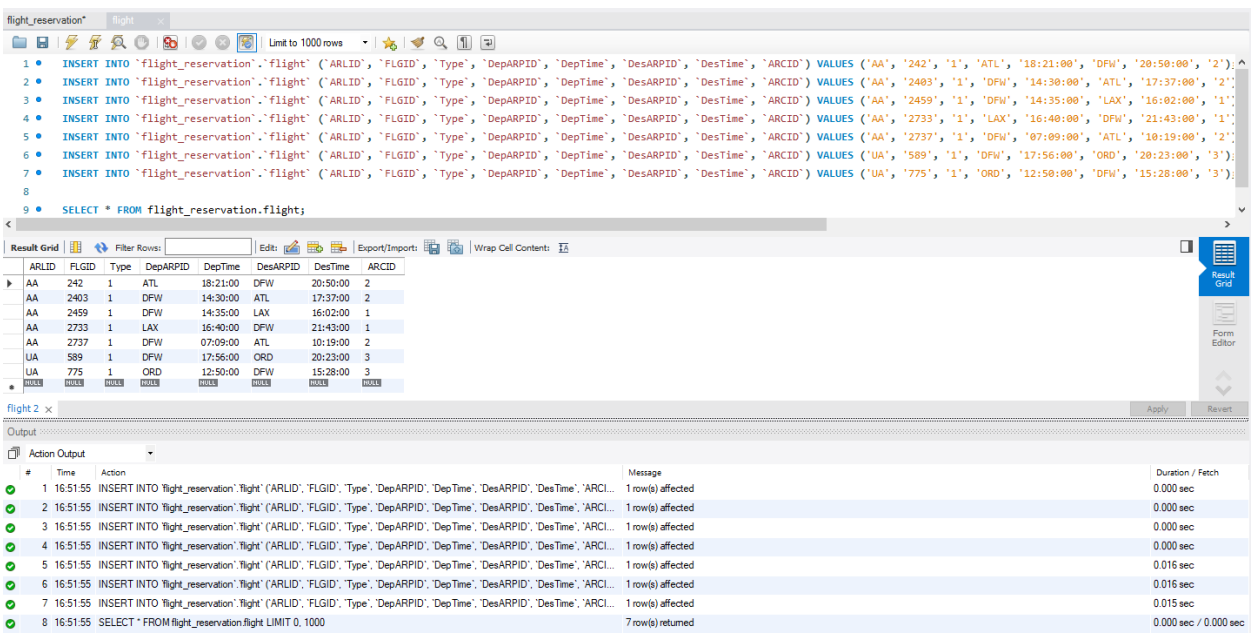
ARLARP

```sql
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('AA', 'AIA');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('OA', 'AIA');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('AA', 'ATL');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('AA', 'DFW');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('AA', 'LAX');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('DL', 'ORD');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('EK', 'ORD');
INSERT INTO `flight_reservation`.`arlarp` (`ARLID`, `ARPID`) VALUES ('AC', 'YYZ');

SELECT * FROM flight_reservation.arlarp;
```



FLIGHT

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

SELECT * FROM flight_reservation.flight;
```



FLGCLASS

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

SELECT * FROM flight_reservation.flgclass;
```

flight_reservation* flgclass

Limit to 1000 rows

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

SELECT * FROM flight_reservation.flgclass;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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
NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

flgclass 2 | Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
11	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
12	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
13	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
14	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
15	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.015 sec
16	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.016 sec
17	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.015 sec
18	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.016 sec
19	16:54:10	INSERT INTO 'flight_reservation'.'flgclass' ('ARLID', 'FLGID', 'CLASS',...	1 row(s) affected	0.000 sec
20	16:54:10	SELECT * FROM flight_reservation.flgclass LIMIT 0, 1000	19 row(s) returned	0.000 sec / 0.000 sec

FLGDAYS

```
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '242', 'Sat
');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '242', 'Sun
');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('UA', '589', 'Fri
');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('UA', '775', 'Fri
');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('UA', '775', 'Sat
');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2403', 'Mo
n');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2403', 'Tu
e');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2459', 'Fr
i');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2459', 'Th
u');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2733', 'Fr
i');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2733', 'Th
u');
INSERT INTO `flight_reservation`.`flgdays` (`ARLID`, `FLGID`, `DAY`) VALUES ('AA', '2737', 'We
d');
SELECT * FROM flight_reservation.flgdays;
```

flight_reservation* | flgdays

Limit to 1000 rows

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

SELECT * FROM flight_reservation.flgdays;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Result Grid | Form Editor | Field Types

flgdays 2 | Apply | Revert

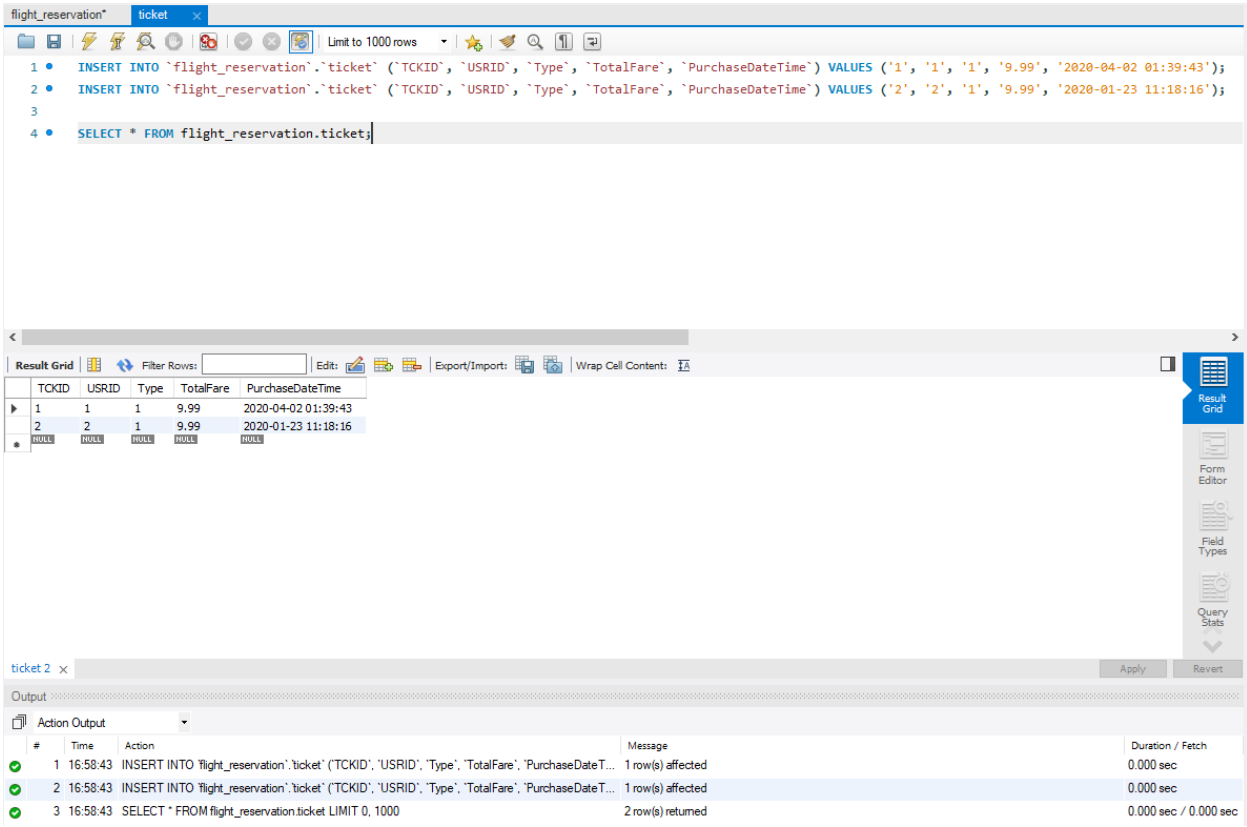
Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.016 sec
2	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
3	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
4	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
5	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
6	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
7	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
8	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
9	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
10	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
11	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
12	16:56:23	INSERT INTO 'flight_reservation'.'flgdays' ('ARLID', 'FLGID', 'DAY') VAL...	1 row(s) affected	0.000 sec
13	16:56:23	SELECT * FROM flight_reservation.flgdays LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec

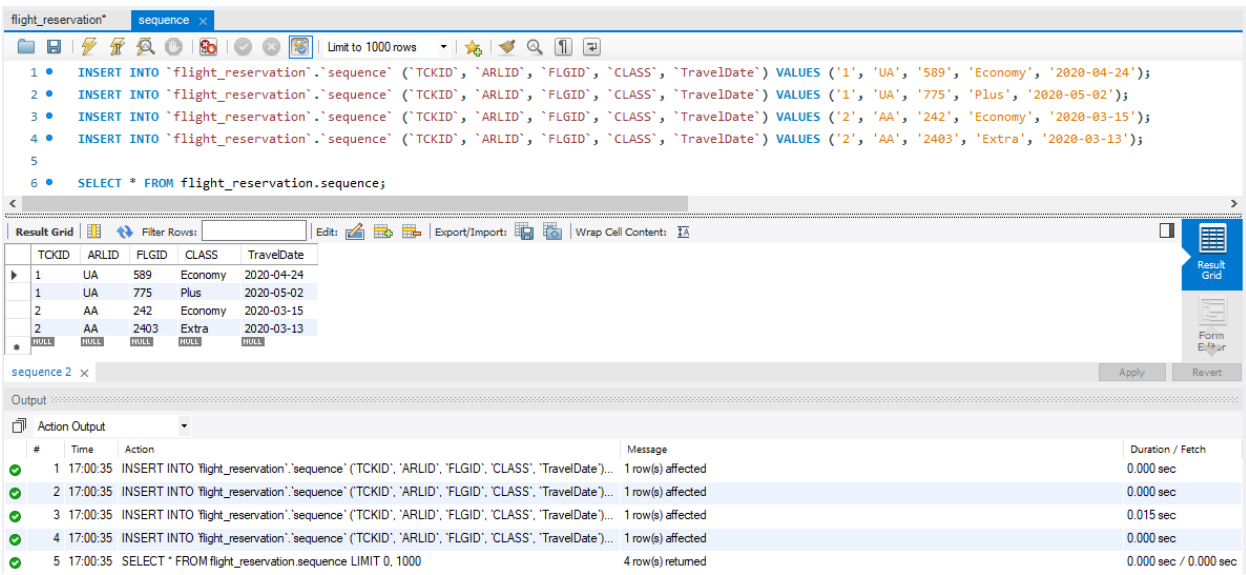
TICKET

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

SELECT * FROM flight_reservation.ticket;
```



SEQUENCE

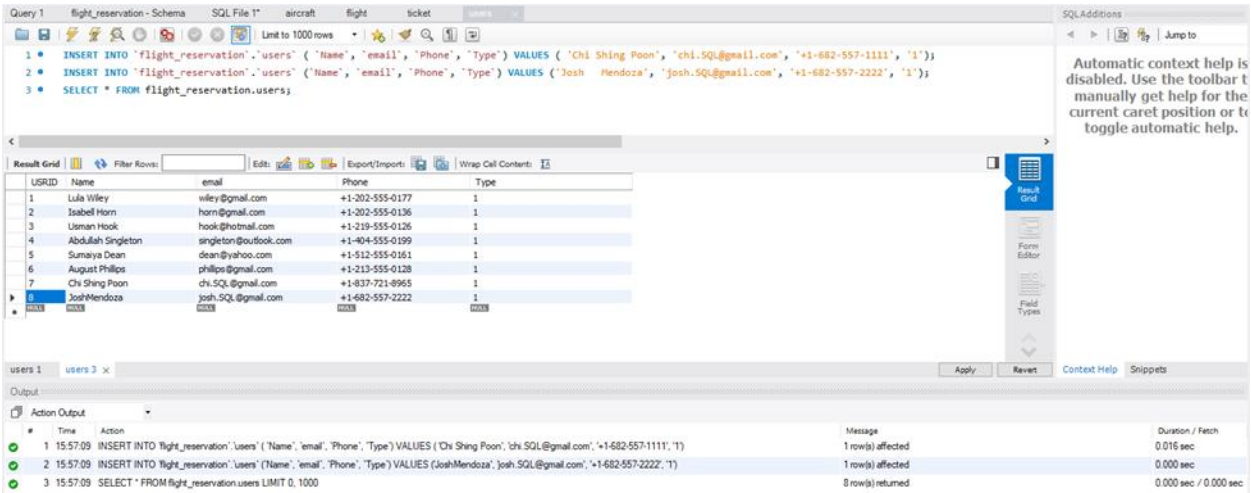
```
INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`, `CLASS`, `TravelDate`)
 VALUES ('1', 'UA', '589', 'Economy', '2020-04-24');
INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`, `CLASS`, `TravelDate`)
 VALUES ('1', 'UA', '775', 'Plus', '2020-05-02');
INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`, `CLASS`, `TravelDate`)
 VALUES ('2', 'AA', '242', 'Economy', '2020-03-15');
INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`, `CLASS`, `TravelDate`)
 VALUES ('2', 'AA', '2403', 'Extra', '2020-03-13');

SELECT * FROM flight_reservation.sequence;
```



Task 3: Executing Queries on Database Tables

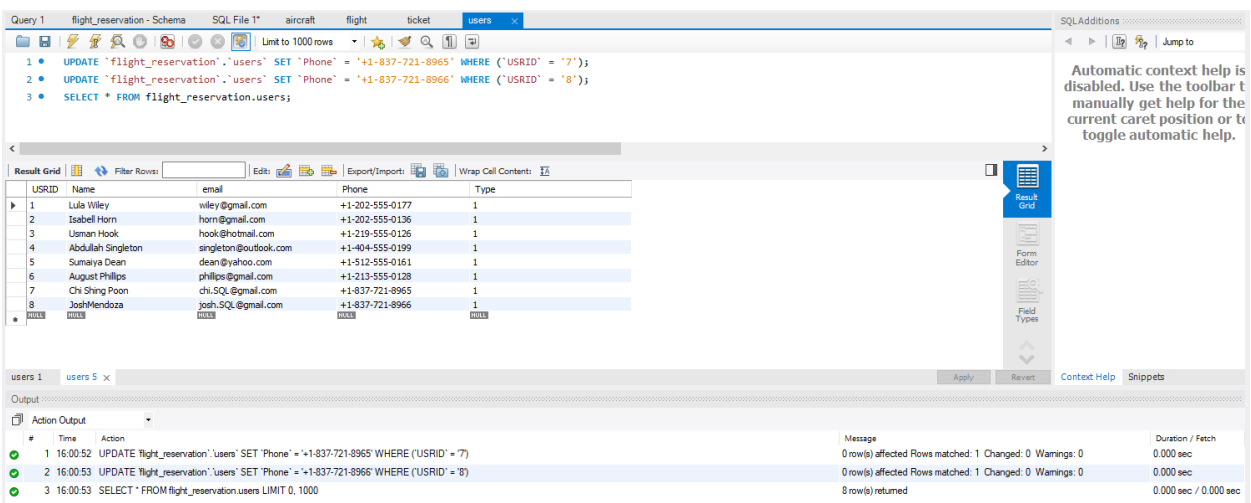
Question 1: Insert yourself as a New Customer. Do not provide the customer id in your query.

```
INSERT INTO `flight_reservation`.`users` ( `Name`, `email`, `Phone`, `Type`) VALUES ( 'Chi
 Shing Poon', 'chi.SQL@gmail.com', '+1-682-557-1111', '1');
INSERT INTO `flight_reservation`.`users` (`Name`, `email`, `Phone`, `Type`) VALUES ('Josh
  Mendoza', 'josh.SQL@gmail.com', '+1-682-557-2222', '1');
```



Question 2: Update your phone number to +1-837-721-8965

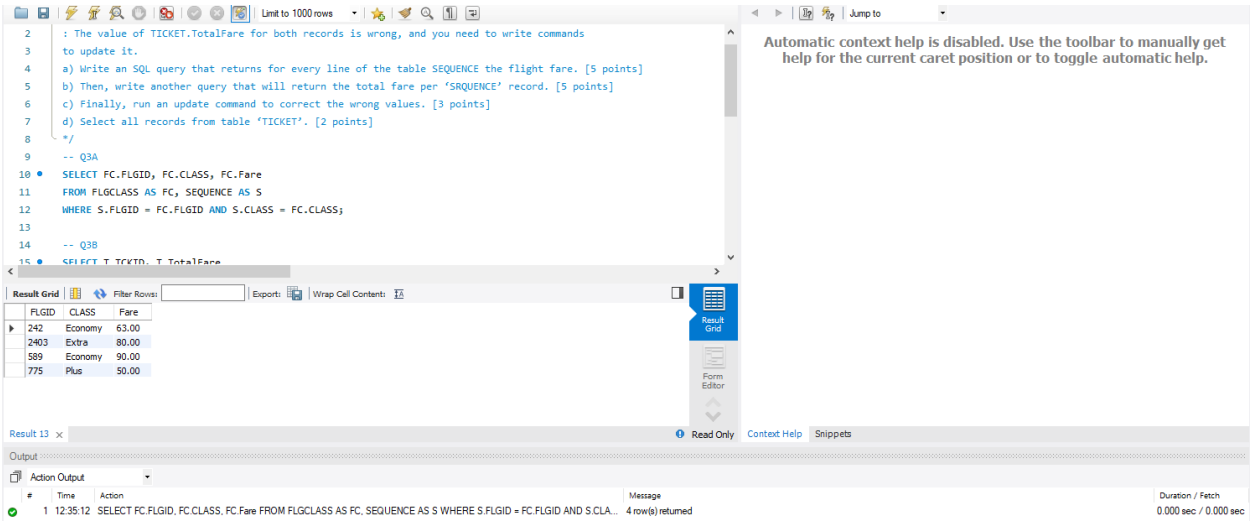
```
UPDATE `flight_reservation`.`users` SET `Phone` = '+1-837-721-
8965' WHERE (`USRID` = '7');
UPDATE `flight_reservation`.`users` SET `Phone` = '+1-837-721-
8966' WHERE (`USRID` = '8');
```



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

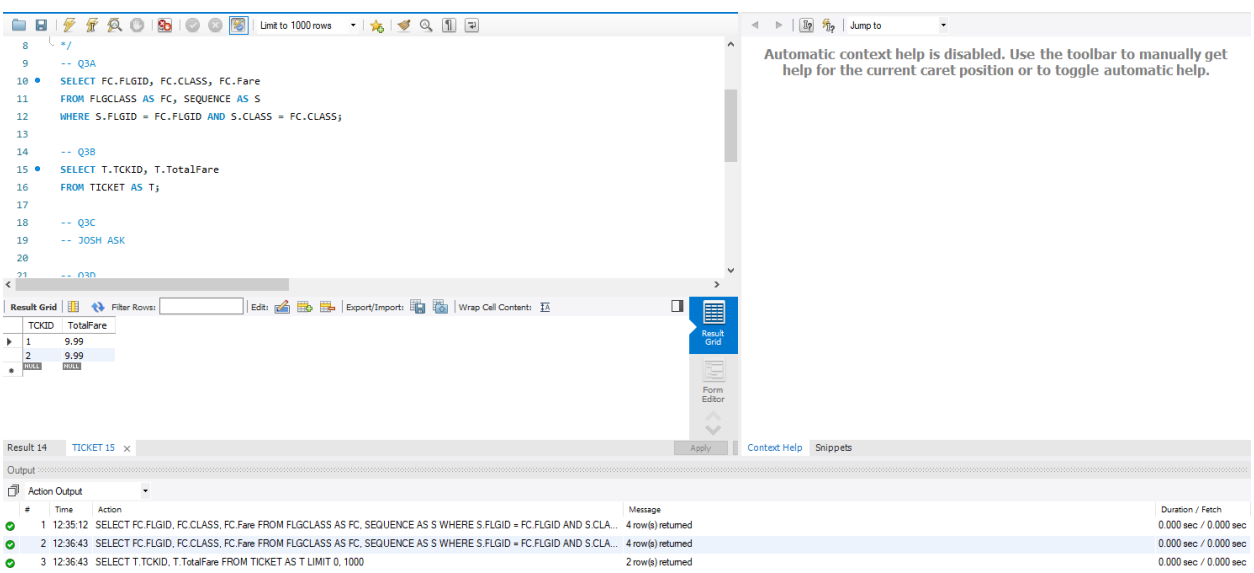
a) Write an SQL query that returns for every line of the table SEQUENCE the flight fare.

```
SELECT FC.FLGID, FC.CLASS, FC.Fare
FROM FLGCLASS AS FC, SEQUENCE AS S
WHERE S.FLGID = FC.FLGID AND S.CLASS = FC.CLASS;
```



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

```
SELECT T.TCKID, T.TotalFare
FROM TICKET AS T;
```

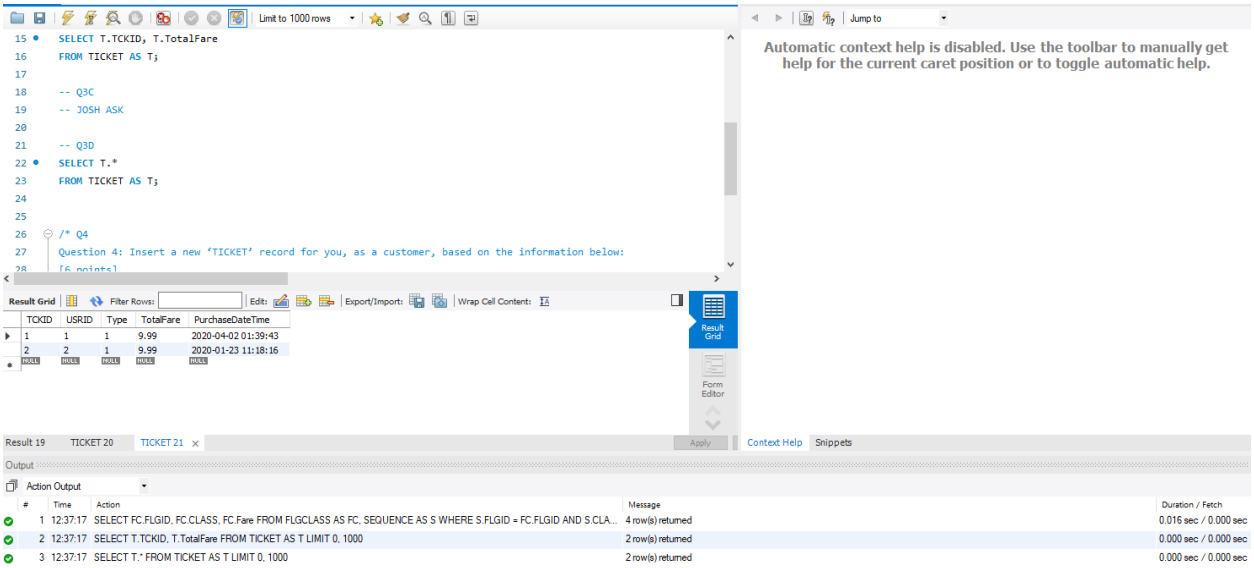


c) Finally, run an update command to correct the wrong values.

```
UPDATE `flight_reservation`.`ticket` SET `TotalFare` =
(
SELECT FC.Fare
From FLGCLASS  AS FC, SEQUENCE AS S -- TICKET AS T
WHERE S.TCKID = TCKID AND FC.CLASS = S.CLASS AND FC.FLGID = S.FLGID);
```

d) Select all records from table ‘TICKET’.

```
SELECT T.*
FROM TICKET AS T;
```



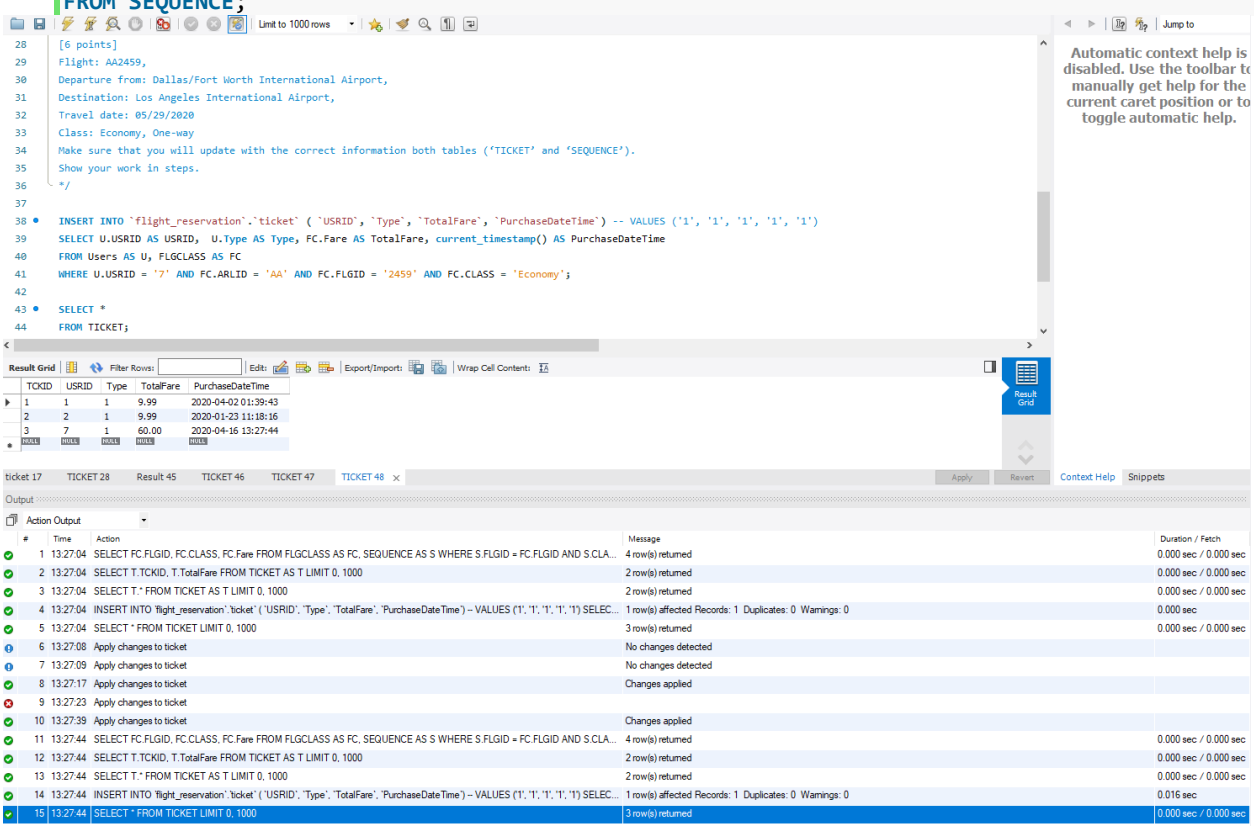
Question 4: Insert a new ‘TICKET’ record for you, as a customer, based on the information below:

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 both tables (‘TICKET’ and ‘SEQUENCE’). Show your work in steps.

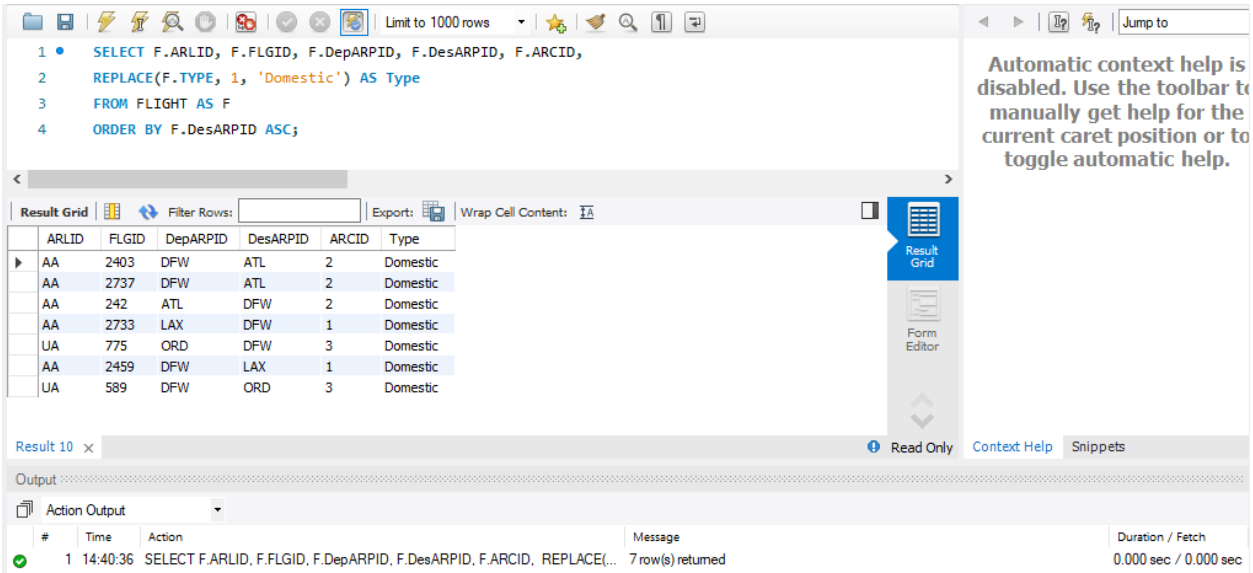
Note: One of the difficulties was trying to come up with the right condition before figuring out we can have the correct result

```
INSERT INTO `flight_reservation`.`ticket` ( `USRID`, `Type`, `TotalFare`, `PurchaseDateTim
e`) -- VALUES ('1', '1', '1', '1', '1')
SELECT U.USRID AS USRID,  U.Type AS Type, FC.Fare AS TotalFare, current_timestamp() AS Pur
chaseDateTime
FROM Users AS U, FLGCLASS AS FC
WHERE U.USRID = '7' AND FC.ARLID = 'AA' AND FC.FLGID = '2459' AND FC.CLASS = 'Economy';
SELECT *
FROM TICKET;

INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`, `CLASS`, `TravelDa
te`) -- VALUES ('1', '1', '1', '1', '1');
SELECT T.TCKID AS TCKID, FC.ARLID AS ARLID, FC.FLGID AS FLGID, FC.CLASS AS CLASS, '2020-
05-20' AS TravelDate
FROM Ticket AS T, FLGCLASS AS FC
WHERE T.TCKID = '3' AND FC.FLGID = '2459' AND FC.Fare = '60.00';

SELECT *
FROM SEQUENCE;
```



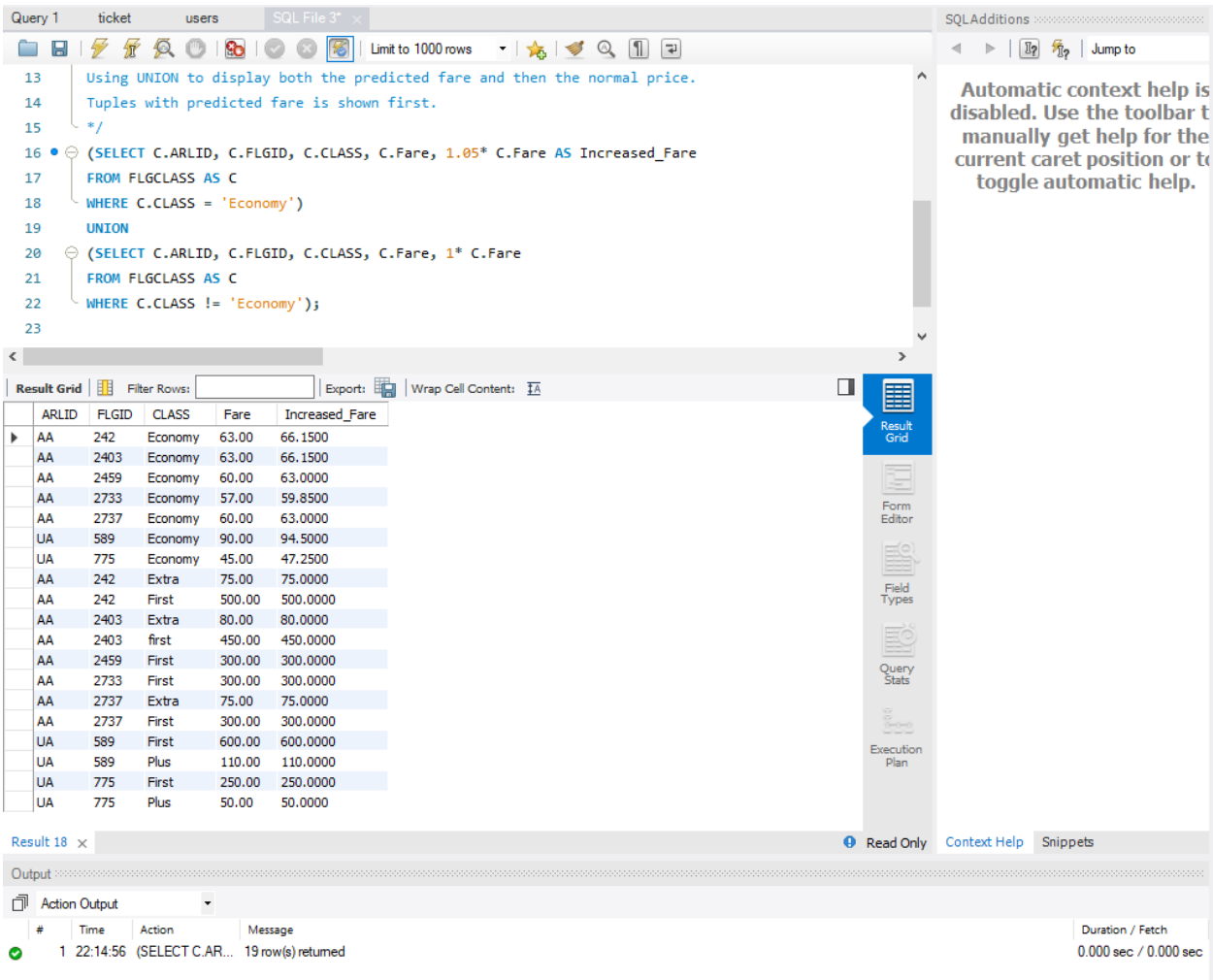
Question 5: Return for every flight, the airline id, the flight id, the departure and destination airport description, and the aircraft description. Furthermore, on the same query, return the type, but instead of 1, return ‘Domestic’. Order your results by the departure airport id.

```
SELECT F.ARLID, F.FLGID, F.DepARPID, F.DesARPID, F.ARCID,
REPLACE(F.TYPE, 1, 'Domestic') AS Type
FROM FLIGHT AS F
ORDER BY F.DesARPID ASC;
```



Question 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.

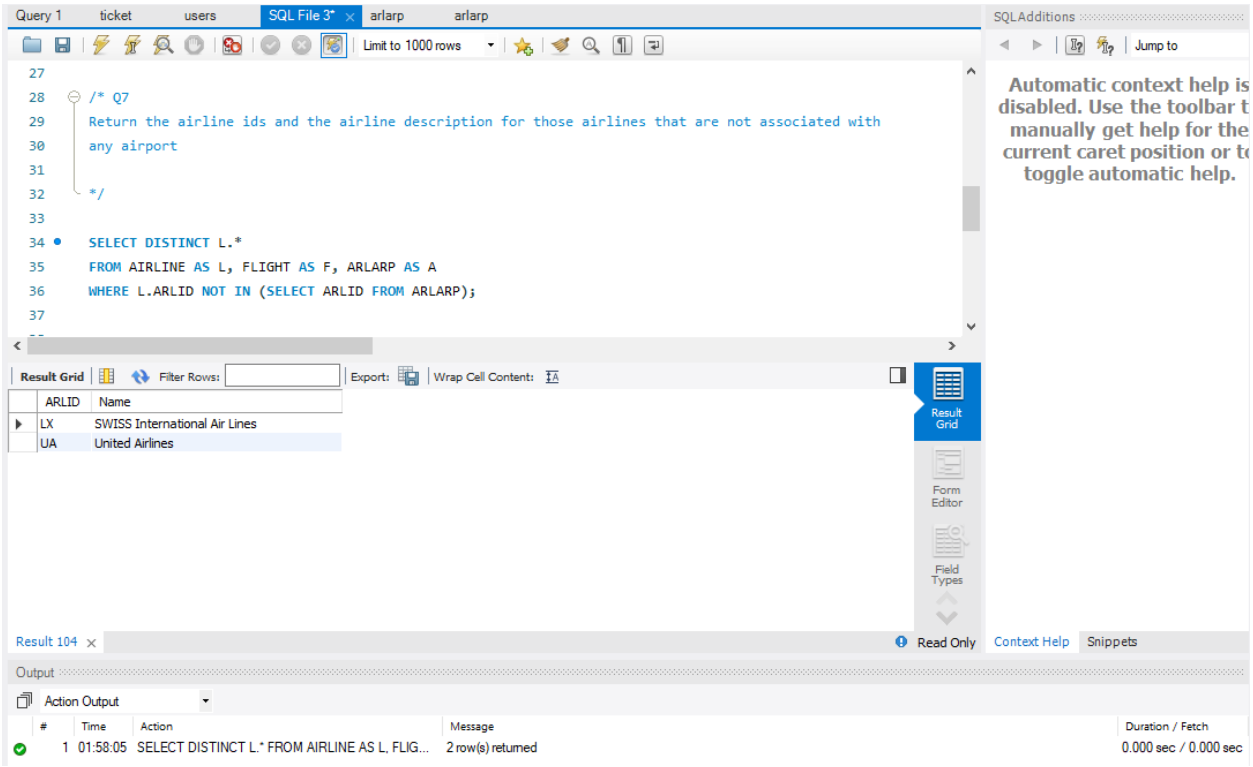
```
(SELECT C.ARLID, C.FLGID, C.CLASS, C.Fare, 1.05* C.Fare AS Increased_Fare
FROM FLGCLASS AS C
WHERE C.CLASS = 'Economy')
UNION
(SELECT C.ARLID, C.FLGID, C.CLASS, C.Fare, 1* C.Fare
FROM FLGCLASS AS C
WHERE C.CLASS != 'Economy');
```



ARLID	FLGID	CLASS	Fare	Increased_Fare
AA	242	Economy	63.00	66.1500
AA	2403	Economy	63.00	66.1500
AA	2459	Economy	60.00	63.0000
AA	2733	Economy	57.00	59.8500
AA	2737	Economy	60.00	63.0000
UA	589	Economy	90.00	94.5000
UA	775	Economy	45.00	47.2500
AA	242	Extra	75.00	75.0000
AA	242	First	500.00	500.0000
AA	2403	Extra	80.00	80.0000
AA	2403	first	450.00	450.0000
AA	2459	First	300.00	300.0000
AA	2733	First	300.00	300.0000
AA	2737	Extra	75.00	75.0000
AA	2737	First	300.00	300.0000
UA	589	First	600.00	600.0000
UA	589	Plus	110.00	110.0000
UA	775	First	250.00	250.0000
UA	775	Plus	50.00	50.0000

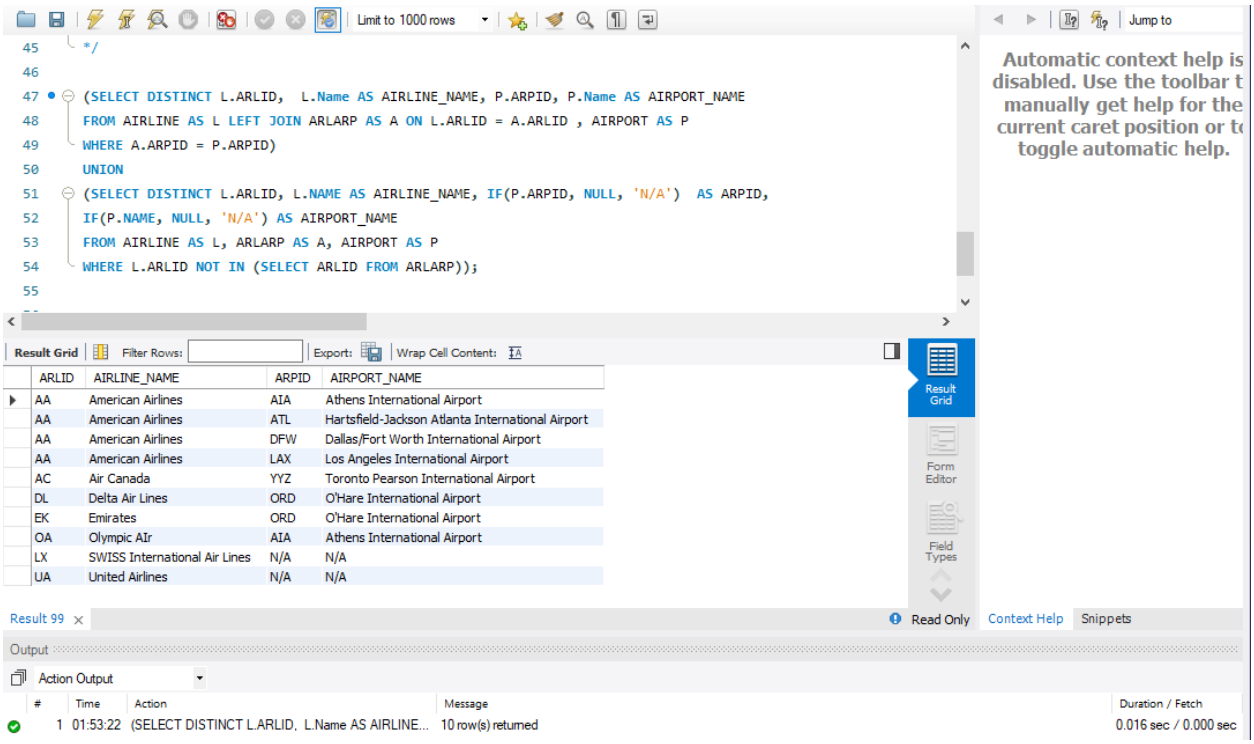
Question 7: Return the airline ids and the airline description for those airlines that are not associated with any airport.

```sql
SELECT DISTINCT L.*
FROM AIRLINE AS L, AIRPORT AS P
WHERE L.ARLID NOT IN (SELECT ARLID FROM ARLARP);
-- AND L.ARLID NOT IN(SELECT ARLID FROM FLIGHT); <-- Accounted for Flight as well
```



Question 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).

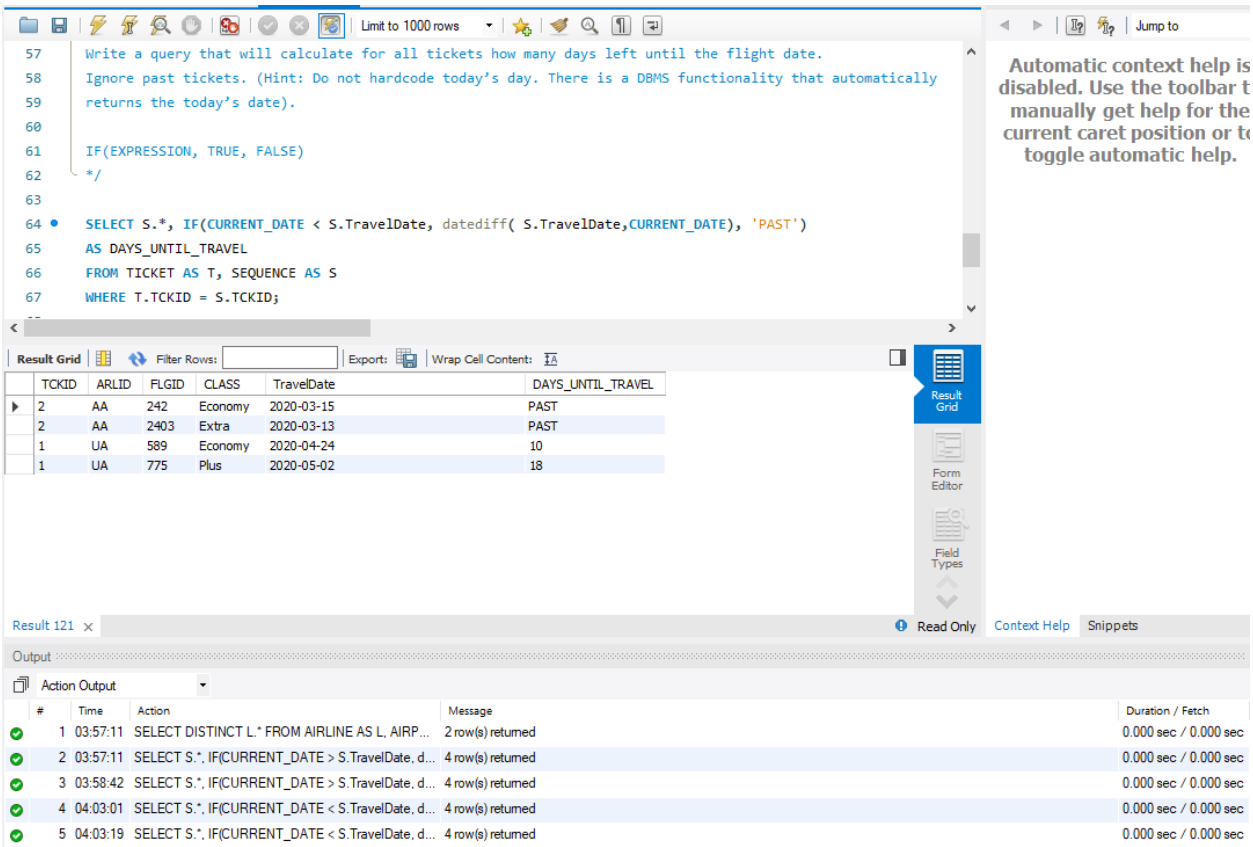
```
(SELECT DISTINCT L.ARLID,  L.Name AS AIRLINE_NAME, P.ARPID, P.Name AS AIRPORT_NAME
FROM AIRLINE AS L LEFT JOIN ARLARP AS A ON L.ARLID = A.ARLID , AIRPORT AS P
WHERE A.ARPID = P.ARPID)
UNION
(SELECT DISTINCT L.ARLID, L.NAME AS AIRLINE_NAME, IF(P.ARPID, NULL, 'N/A')  AS ARPID,
IF(P.NAME, NULL, 'N/A') AS AIRPORT_NAME
FROM AIRLINE AS L, ARLARP AS A, AIRPORT AS P
WHERE L.ARLID NOT IN (SELECT ARLID FROM ARLARP));
```



ARLID	AIRLINE_NAME	ARPID	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 International Airport
AA	American Airlines	LAX	Los Angeles International Airport
AC	Air Canada	YYZ	Toronto Pearson International Airport
DL	Delta Air Lines	ORD	O'Hare International Airport
EK	Emirates	ORD	O'Hare International Airport
OA	Olympic AIr	AIA	Athens International Airport
LX	SWISS International Air Lines	N/A	N/A
UA	United Airlines	N/A	N/A

Question 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).

```
SELECT S.*, IF(CURRENT_DATE < S.TravelDate, datediff( S.TravelDate,CURRENT_DATE), 'PAST')

AS DAYS_UNTIL_TRAVEL
FROM TICKET AS T, SEQUENCE AS S
WHERE T.TCKID = S.TCKID;
```



Question 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.

```
-- DISPLAYS ARLID, FLGID, AND THEIR RESPECTIVE CLASS OPTIONS
SELECT DISTINCT F.ARLID, F.FLGID, G.CLASS, C.ChangeFee, G.Fare, F.ARCID
 IF( C.ARCID = F.ARCID AND C.CLASS = G.CLASS,C.NumofSeats, 'N/A') AS NumofSeats
/*

CASE
    WHEN C.ARCID = F.ARCID AND C.CLASS = G.CLASS THEN C.NumofSeats
    END*/
FROM FLIGHT AS F , ARCCLASS AS C, FLGCLASS AS G
WHERE F.ARLID = G.ARLID AND C.CLASS = G.CLASS AND F.FLGID = G.FLGID
ORDER BY F.ARLID
;
```

flight_reservation* | flight

Limit to 1000 rows

```
-- DISPLAYS ARLID, FLGID, AND THEIR RESPECTIVE CLASS OPTIONS
SELECT DISTINCT F.ARLID, F.FLGID, G.CLASS, G.Fare, C.ChangeFee, F.ARCID,
 if ( C.ARCID = F.ARCID AND C.CLASS = G.CLASS,C.NumofSeats, 'N/A') AS NumofSeats
/*

CASE
    WHEN C.ARCID = F.ARCID AND C.CLASS = G.CLASS THEN C.NumofSeats
    END*/
FROM FLIGHT AS F, ARCCLASS AS C, FLGCLASS AS G
WHERE F.ARLID = G.ARLID AND C.CLASS = G.CLASS AND F.FLGID = G.FLGID
ORDER BY F.ARLID
;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ARLID	FLGID	CLASS	Fare	ChangeFee	ARCID	NumofSeats
AA	242	Economy	63.00	50.00	2	N/A
AA	242	Economy	63.00	50.00	2	114
AA	242	Extra	75.00	50.00	2	30
AA	242	First	500.00	0.00	2	N/A
AA	242	First	500.00	0.00	2	16
AA	2403	Economy	63.00	50.00	2	N/A
AA	2403	Economy	63.00	50.00	2	114
AA	2403	Extra	80.00	50.00	2	30
AA	2403	first	450.00	0.00	2	N/A
AA	2403	first	450.00	0.00	2	16
AA	2459	Economy	60.00	50.00	1	171
AA	2459	Economy	60.00	50.00	1	N/A
AA	2459	First	300.00	0.00	1	16
AA	2459	First	300.00	0.00	1	N/A
AA	2733	Economy	57.00	50.00	1	171
AA	2733	Economy	57.00	50.00	1	N/A
AA	2733	First	300.00	0.00	1	16
AA	2733	First	300.00	0.00	1	N/A
AA	2737	Economy	60.00	50.00	2	N/A
AA	2737	Economy	60.00	50.00	2	114
AA	2737	Extra	75.00	50.00	2	30
AA	2737	First	300.00	0.00	2	N/A
AA	2737	First	300.00	0.00	2	16
UA	589	Economy	90.00	50.00	3	N/A
UA	589	Economy	90.00	50.00	3	96
UA	589	First	600.00	0.00	3	N/A
UA	589	First	600.00	0.00	3	12
UA	589	Plus	110.00	50.00	3	42
UA	775	Economy	45.00	50.00	3	N/A
UA	775	Economy	45.00	50.00	3	96
UA	775	First	250.00	0.00	3	N/A
UA	775	First	250.00	0.00	3	12
UA	775	Plus	50.00	50.00	3	42

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result 17 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	21:37:00	SELECT DISTINCT F.ARLID, F.FLGID, G.CLASS, G.Fare, C.ChangeFee, F.ARCID, if (C.A...	33 row(s) returned	0.000 sec / 0.000 sec

Phase 3

Task 1a:

In this task, we want you to identify the customers that used our system to book tickets. To do that, you need to create a new column to the USERS table, and name it as 'Active'; define this column as a small integer. Values for this column are '0' for non-active customers and '1' for active. You do not need to create an enumerated list. Consequently, for non-active customers, there is not any ticket information stored in our database. Your task is to update the 'Active' column with '1' for all active customers and with '0' for the non-active. This column must be left empty for the customer representatives and administrators.

Submit:

(1) Task description.

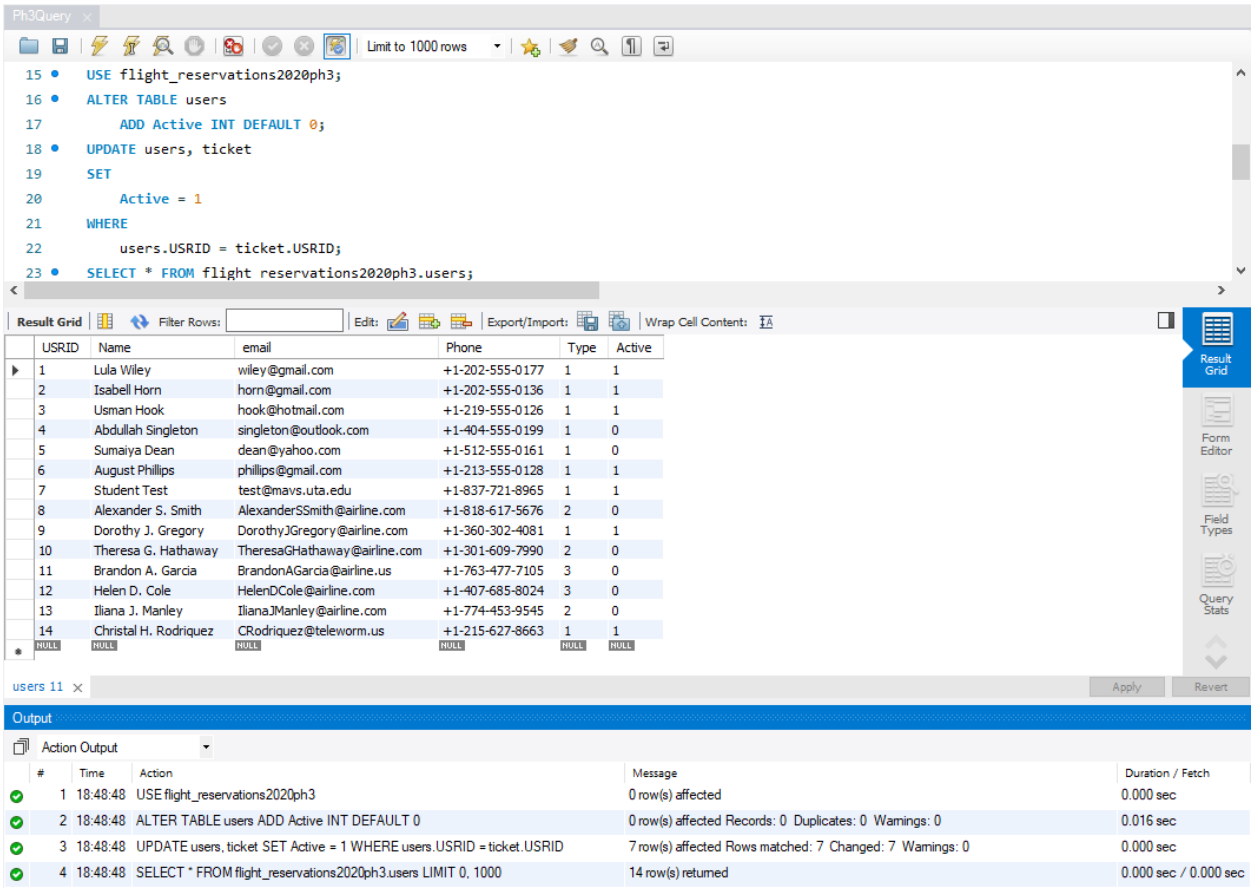
(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output.

(4) The select USERS typed command in your submission report file

(5) A screenshot with your query, result grid, and the action output.

```
/*
Task 1a: [8 points]
In this task, we want you to identify the customers that used our system
to book tickets. To do that,
you need to create a new column to the USERS table, and name it as
'Active'; define this column as a
small integer. Values for this column are '0' for non-active customers and
'1' for active. You do not
need to create an enumerated list. Consequently, for non-active customers,
there is not any ticket
information stored in our database. Your task is to update the 'Active'
column with '1' for all active
customers and with '0' for the non-active. This column must be left empty
for the customer
representatives and administrators.
*/

USE flight_reservations2020ph3;
ALTER TABLE users
       ADD Active INT DEFAULT 0;
UPDATE users, ticket
SET
       Active = 1
WHERE
       users.USRID = ticket.USRID;
SELECT * FROM flight_reservations2020ph3.users;
```



USRID	Name	email	Phone	Type	Active
1	Lula Wiley	wiley@gmail.com	+1-202-555-0177	1	1
2	Isabell Horn	horn@gmail.com	+1-202-555-0136	1	1
3	Usman Hook	hook@hotmail.com	+1-219-555-0126	1	1
4	Abdullah Singleton	singleton@outlook.com	+1-404-555-0199	1	0
5	Sumaiya Dean	dean@yahoo.com	+1-512-555-0161	1	0
6	August Phillips	phillips@gmail.com	+1-213-555-0128	1	1
7	Student Test	test@mavs.uta.edu	+1-837-721-8965	1	1
8	Alexander S. Smith	AlexanderSSmith@airline.com	+1-818-617-5676	2	0
9	Dorothy J. Gregory	DorothyJGregory@airline.com	+1-360-302-4081	1	1
10	Theresa G. Hathaway	TheresaGHathaway@airline.com	+1-301-609-7990	2	0
11	Brandon A. Garcia	BrandonAGarcia@airline.us	+1-763-477-7105	3	0
12	Helen D. Cole	HelenDCole@airline.com	+1-407-685-8024	3	0
13	Iliana J. Manley	IlianaJManley@airline.com	+1-774-453-9545	2	0
14	Christal H. Rodriquez	CRodriquez@teleworm.us	+1-215-627-8663	1	1
NULL	NULL	NULL	NULL	NULL	NULL

Task 1b:

For this task, you need to return the percentage of active customers.

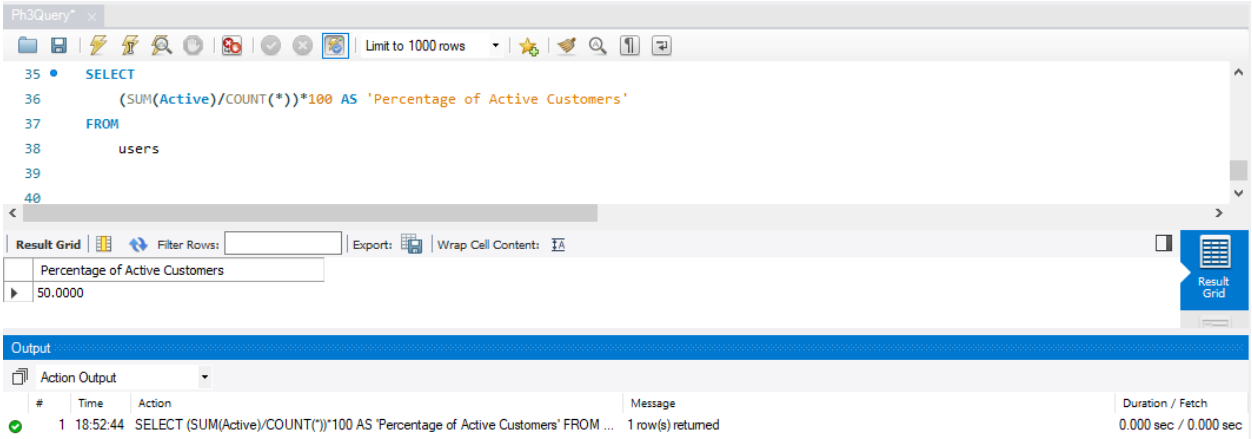
Submit:

(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output.

```
/*
Task 1b: [3 points]
For this task, you need to return the percentage of active customers.
*/

SELECT
        (SUM(Active)/COUNT(*))*100 AS 'Percentage of Active Customers'
FROM
        users;
```



Task 2:

By an error to the system, the 'Type' attribute from table 'Flight' 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.

Submit:

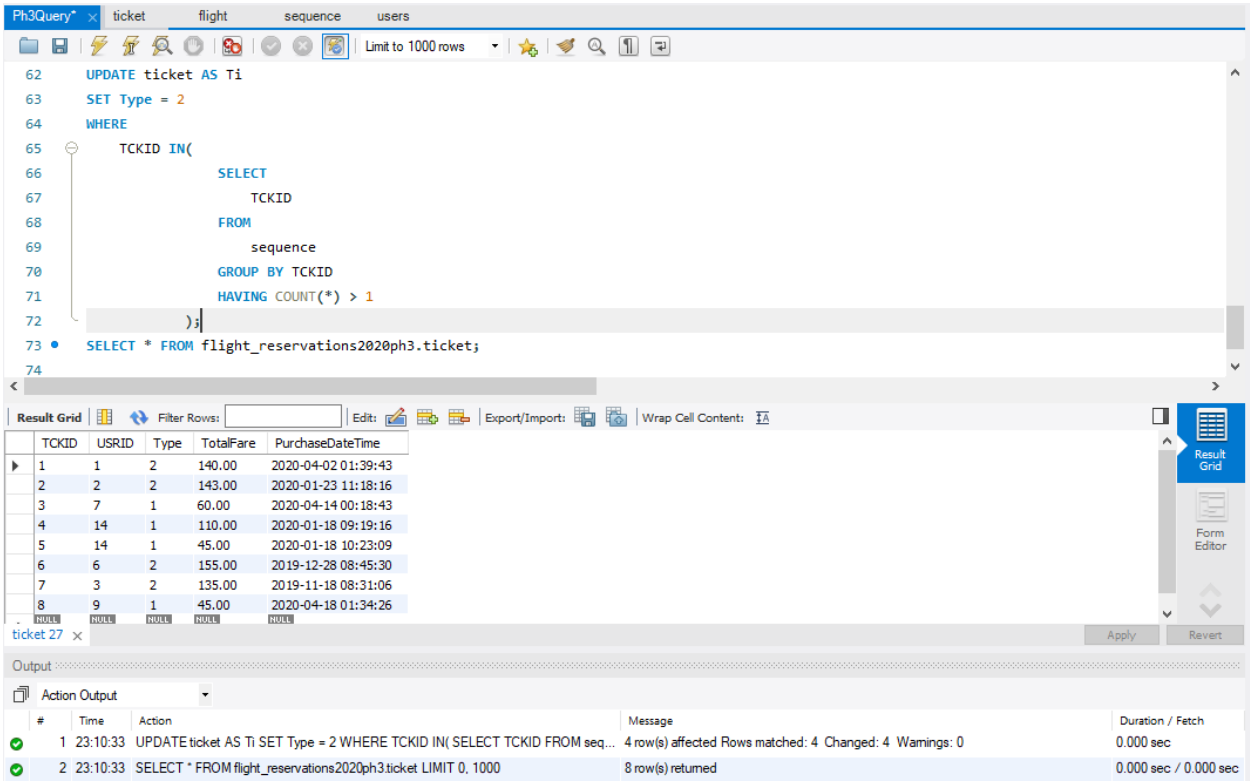
(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output.

```
/*
Task 2: [5 points]
By an error to the system, the 'Type' attribute from table 'Flight' 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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

UPDATE ticket AS Ti
SET Type = 2
WHERE
       TCKID IN(
                              SELECT
                                     TCKID
                              FROM
```

```
                                        sequence
                                GROUP BY TCKID
                                HAVING COUNT(*) > 1
                        );
SELECT * FROM flight_reservations2020ph3.ticket;
```



Task 3:

Here you need to create a view named as 'viewTicketIInfo', 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

Submit:

(1) Task description.

(2) Your 'Create view' command typed in your submission report file. Use only the above provided names for your attributes. Do NOT alter any name.

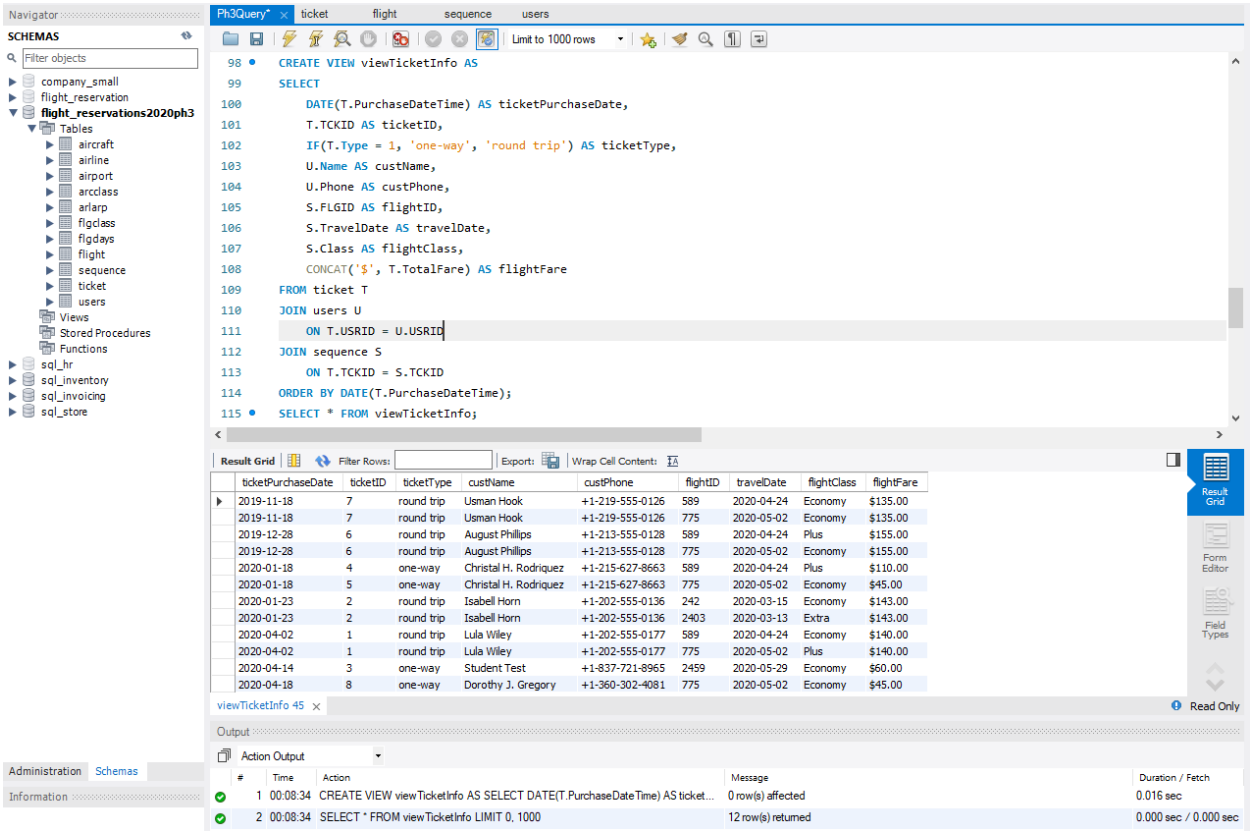
(3) A screenshot, with your query, result grid, and the action output.

(4) Then your 'Select view' command typed in your submission report file.

(5) A screenshot with your query, result grid, and the action output.

```
/*
Task 3: [10 points]
Here you need to create a view named as 'viewTicketlInfo', 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
Submit:
(1) Task description.
(2) Your 'Create view' command typed in your submission report file. Use
only the aboveprovided
names for your attributes. Do NOT alter any name.
(3) A screenshot, with your query, result grid, and the action output.
(4) Then your 'Select view' command typed in your submission report file.
(5) A screenshot with your query, result grid, and the action output.
*/

CREATE VIEW viewTicketInfo AS
SELECT
        DATE(T.PurchaseDateTime) AS ticketPurchaseDate,
    T.TCKID AS ticketID,
    IF(T.Type = 1, 'one-way', 'round trip') AS ticketType,
        U.Name AS custName,
    U.Phone AS custPhone,
    S.FLGID AS flightID,
```

```
    S.TravelDate AS travelDate,
    S.Class AS flightClass,
    CONCAT('$', T.TotalFare) AS flightFare
FROM ticket T
JOIN users U
        ON T.USRID = U.USRID
JOIN sequence S
        ON T.TCKID = S.TCKID
ORDER BY DATE(T.PurchaseDateTime);
SELECT * FROM viewTicketInfo;
```



Task 4:

Write a query that retrieves data from your stored 'viewTicketIInfo' 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’

Submit:

(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output.

```
/*
Task 4: [5 points]
Write a query that retrieves data from your stored ‘viewTicketlInfo’ 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’
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

SELECT
       v.ticketPurchaseDate as 'Purchase Date',
    v.ticketID as 'Ticket',
    v.ticketType as 'Ticket Type',
    v.custName as 'Customer',
    v.custPhone as 'Customer Number',
    v.flightFare as 'Ticket fare'
FROM
       viewTicketInfo v
ORDER BY v.flightFare ASC;
```

Ph3Query* ticket flight sequence users arcclass airport aircraft arlarp airline

Limit to 1000 rows

```
SELECT
    v.ticketPurchaseDate as 'Purchase Date',
    v.ticketID as 'Ticket',
    v.ticketType as 'Ticket Type',
    v.custName as 'Customer',
    v.custPhone as 'Customer Number',
    v.flightFare as 'Ticket fare'
FROM
    viewTicketInfo v
ORDER BY v.flightFare ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Purchase Date	Ticket	Ticket Type	Customer	Customer Number	Ticket fare
2020-01-18	4	one-way	Christal H. Rodriquez	+1-215-627-8663	$110.00
2019-11-18	7	round trip	Usman Hook	+1-219-555-0126	$114.75
2019-11-18	7	round trip	Usman Hook	+1-219-555-0126	$114.75
2020-04-02	1	round trip	Lula Wiley	+1-202-555-0177	$119.00
2020-04-02	1	round trip	Lula Wiley	+1-202-555-0177	$119.00
2019-12-28	6	round trip	August Phillips	+1-213-555-0128	$131.75
2019-12-28	6	round trip	August Phillips	+1-213-555-0128	$131.75
2020-01-23	2	round trip	Isabell Horn	+1-202-555-0136	$143.00
2020-01-23	2	round trip	Isabell Horn	+1-202-555-0136	$143.00
2020-01-18	5	one-way	Christal H. Rodriquez	+1-215-627-8663	$45.00
2020-04-18	8	one-way	Dorothy J. Gregory	+1-360-302-4081	$45.00
2020-04-14	3	one-way	Student Test	+1-837-721-8965	$60.00

Result Grid Form Editor Field Types Query Stats

viewTicketInfo 18 Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:14:48	SELECT v.ticketPurchaseDate as 'Purchase Date', v.ticketID as 'Ticket', v.tick...	12 row(s) returned	0.000 sec / 0.000 sec

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 fight 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!

Submit:

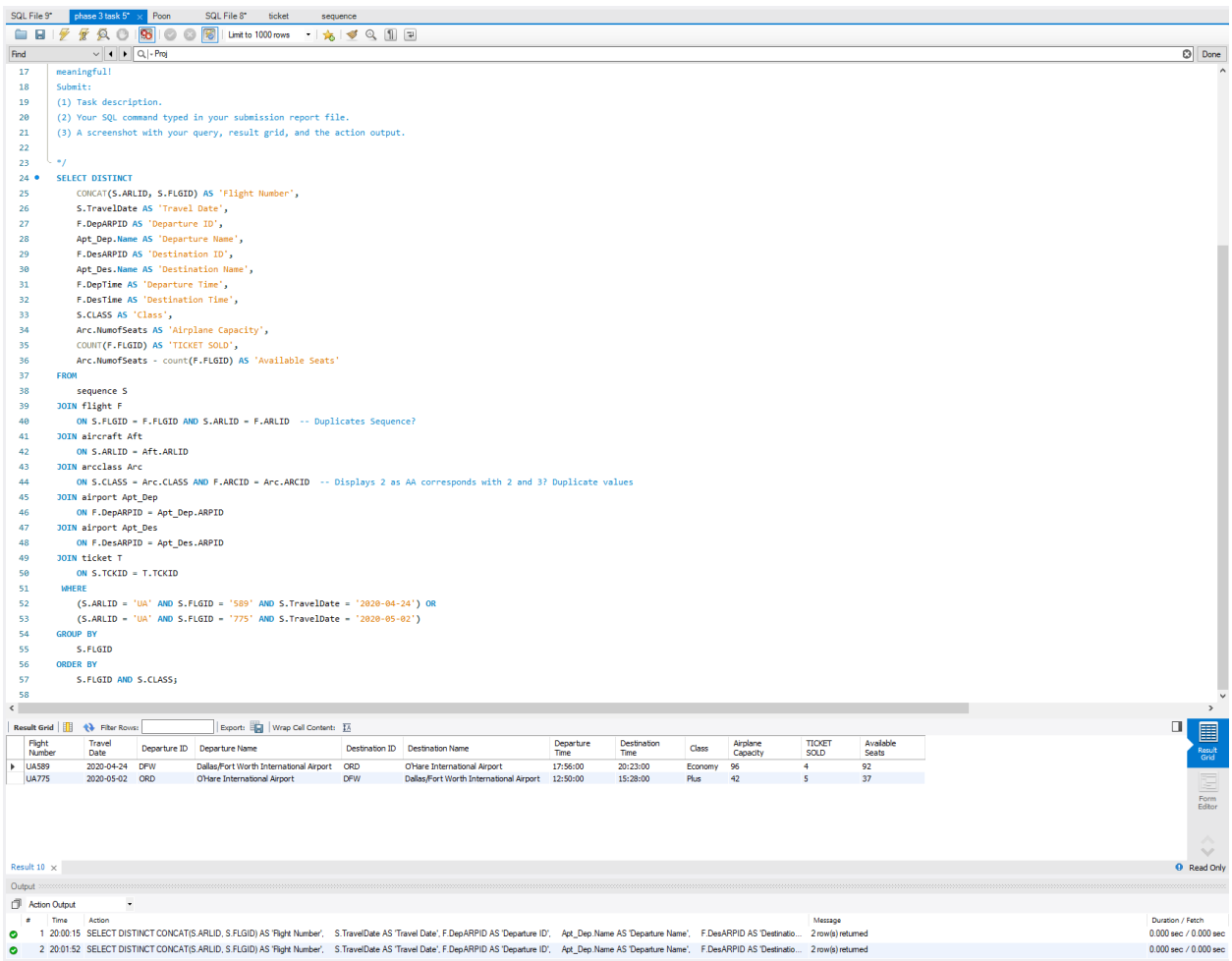
(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output.

```
/*
Task 5: [10 points]
In this task, we want to count how many seats left for two specific flights.
You need to write one query
that returns:
• the fight 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!
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.

*/
SELECT DISTINCT
        CONCAT(S.ARLID, S.FLGID) AS 'Flight Number',
    S.TravelDate AS 'Travel Date',
        F.DepARPID AS 'Departure ID',
    Apt_Dep.Name AS 'Departure Name',
    F.DesARPID AS 'Destination ID',
    Apt_Des.Name AS 'Destination Name',
    F.DepTime AS 'Departure Time',
    F.DesTime AS 'Destination Time',
    S.CLASS AS 'Class',
    Arc.NumofSeats AS 'Airplane Capacity',
    COUNT(F.FLGID) AS 'Tickets Sold',
    Arc.NumofSeats - count(F.FLGID) AS 'Available Seats'
FROM
        sequence S
JOIN flight F
        ON S.FLGID = F.FLGID AND S.ARLID = F.ARLID  -- Duplicates Sequence?
JOIN aircraft Aft
        ON S.ARLID = Aft.ARLID
```

```
JOIN arcclass Arc
        ON S.CLASS = Arc.CLASS AND F.ARCID = Arc.ARCID  -- Displays 2 as AA
corresponds with 2 and 3? Duplicate values
JOIN airport Apt_Dep
        ON F.DepARPID = Apt_Dep.ARPID
JOIN airport Apt_Des
        ON F.DesARPID = Apt_Des.ARPID
JOIN ticket T
        ON S.TCKID = T.TCKID
 WHERE
        (S.ARLID = 'UA' AND S.FLGID = '589' AND S.TravelDate = '2020-04-24')
OR
        (S.ARLID = 'UA' AND S.FLGID = '775' AND S.TravelDate = '2020-05-02')
GROUP BY
        S.FLGID
ORDER BY
        S.FLGID AND S.CLASS;
```



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.

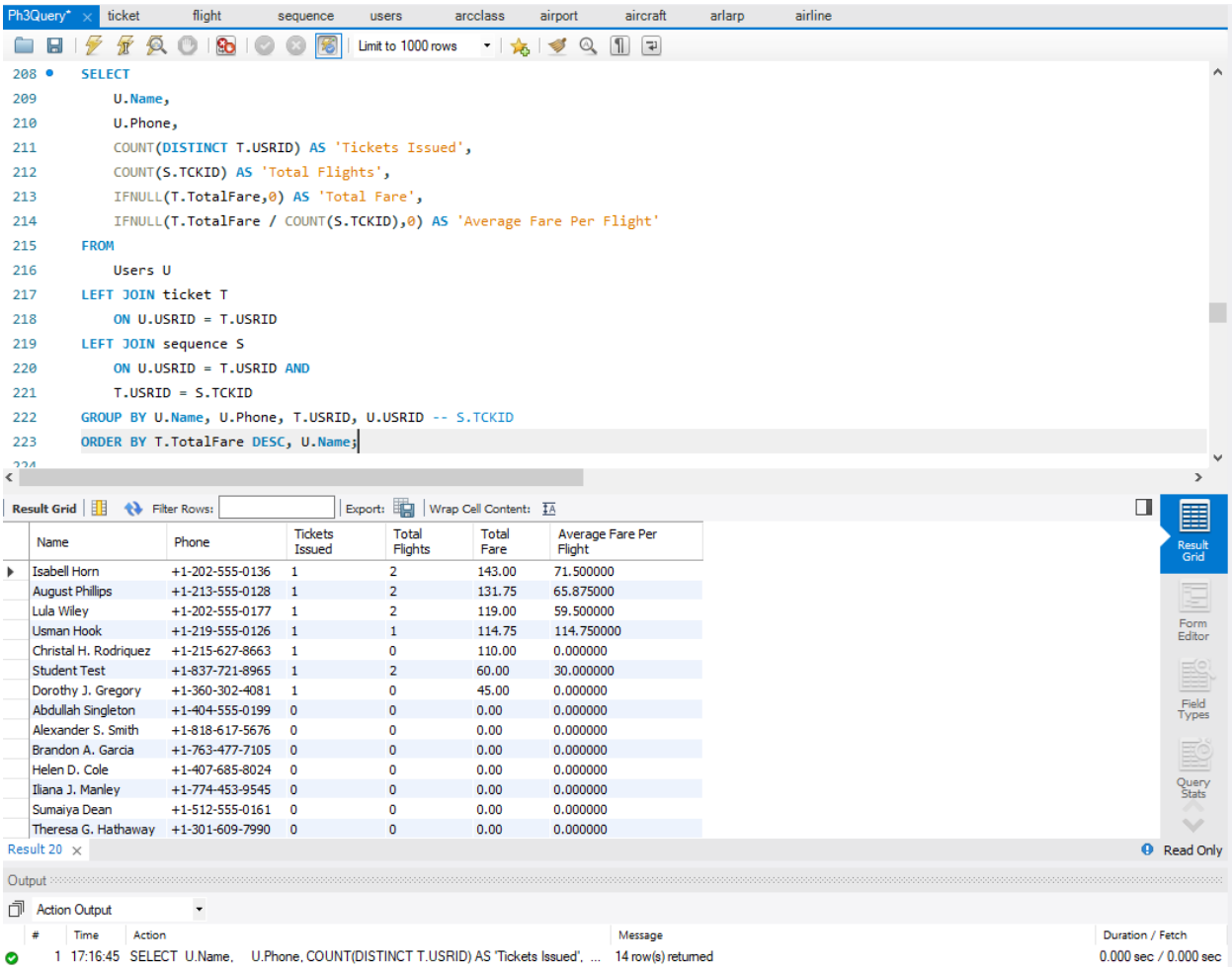
Submit:

(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output

```
/*
Task 6: [8 points]
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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

SELECT
        U.Name,
    U.Phone,
        COUNT(DISTINCT T.USRID) AS 'Tickets Issued',
        COUNT(S.TCKID) AS 'Total Flights',
    IFNULL(T.TotalFare,0) AS 'Total Fare',
    IFNULL(T.TotalFare / COUNT(S.TCKID),0) AS 'Average Fare Per Flight'
FROM
        Users U
LEFT JOIN ticket T
        ON U.USRID = T.USRID
LEFT JOIN sequence S
        ON U.USRID = T.USRID AND
    T.USRID = S.TCKID
GROUP BY U.Name, U.Phone, T.USRID, U.USRID -- S.TCKID
ORDER BY T.TotalFare DESC, U.Name;
```



Task 7:

Due to the current situation with COVID-19, our 'Online travel reservation system' CEO decided to grand 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.

Submit:

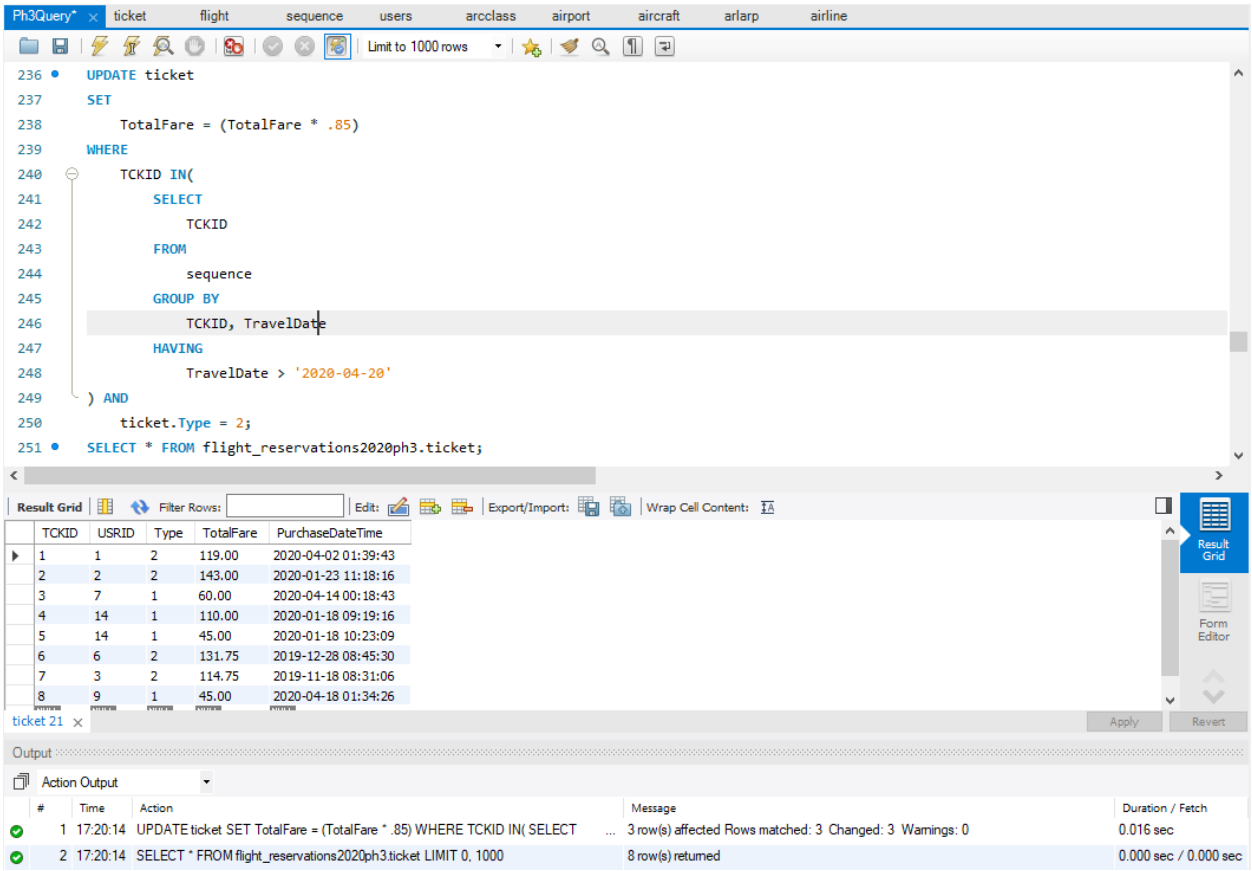
(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot with your query, result grid, and the action output

```
/*
Task 7: [8 points]
Due to the current situation with COVID-19, our 'Online travel reservation
system' CEO decided to
```

```
grand 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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

UPDATE ticket
SET
       TotalFare = (TotalFare * .85)
WHERE
       TCKID IN(
              SELECT
            TCKID
              FROM
                     sequence
              GROUP BY
                     TCKID, TravelDate
              HAVING
                     TravelDate > '2020-04-20'
) AND
       ticket.Type = 2;
SELECT * FROM flight_reservations2020ph3.ticket;
```



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.

Submit:

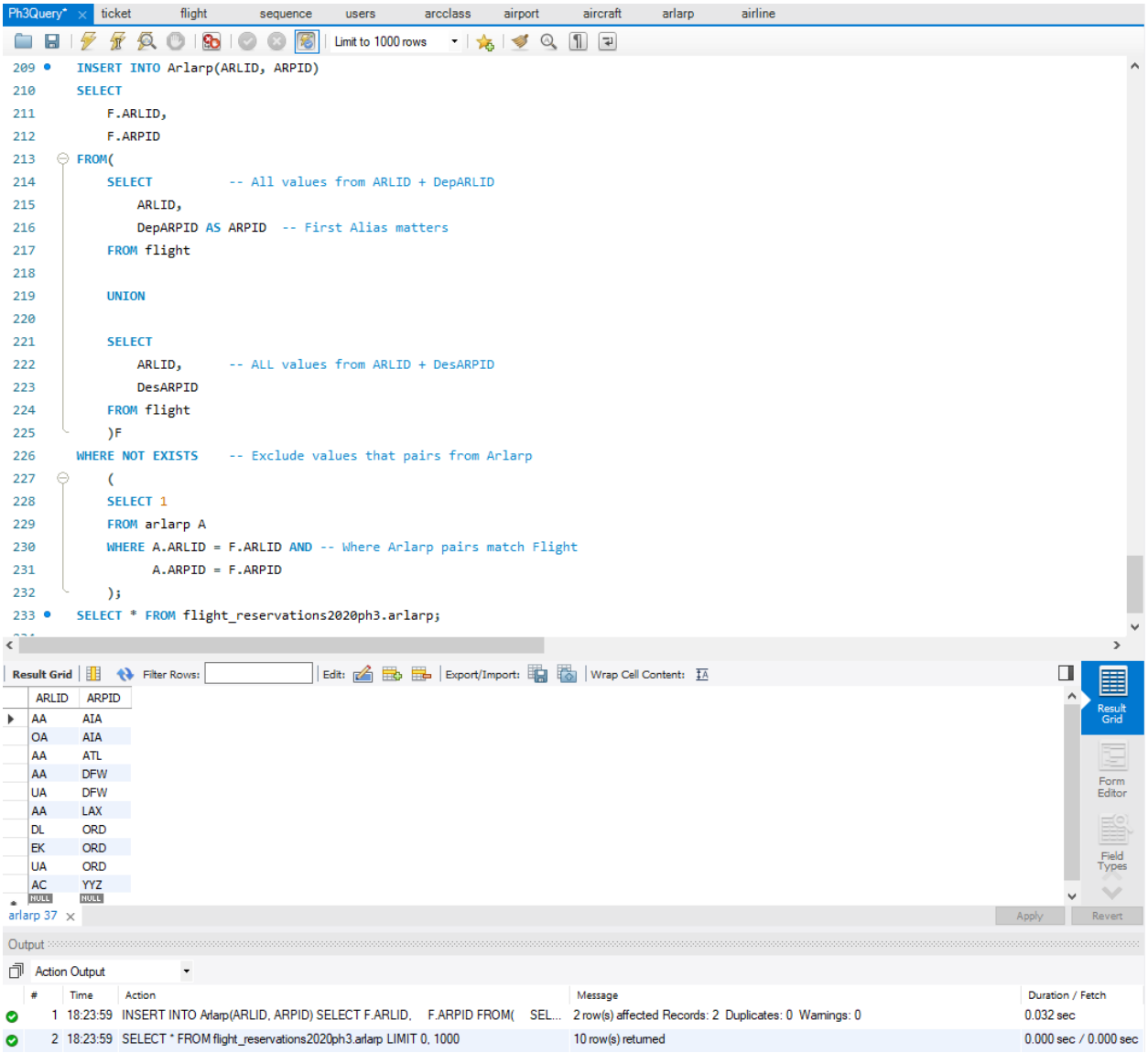
(1) Task description.

(2) Your SQL command(s) typed in your submission report file.

(3) The screenshot(s), with your query, result grid, and the action output.

```
/*
Task 8: [8 points]
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.
Submit:
(1) Task description.
(2) Your SQL command(s) typed in your submission report file.
(3) The screenshot(s), with your query, result grid, and the action
output.
*/

INSERT INTO Arlarp(ARLID, ARPID)
SELECT
       F.ARLID,
    F.ARPID
FROM(
    SELECT                   -- All values from ARLID + DepARLID
              ARLID,
        DepARPID AS ARPID  -- First Alias matters
       FROM flight

    UNION

       SELECT
              ARLID,         -- ALL values from ARLID + DesARPID
              DesARPID
       FROM flight
    )F
WHERE NOT EXISTS      -- Exclude values that pairs from Arlarp
        (
    SELECT 1
    FROM arlarp A
    WHERE A.ARLID = F.ARLID AND -- Where Arlarp pairs match Flight
                A.ARPID = F.ARPID
       );
SELECT * FROM flight_reservations2020ph3.arlarp;
```



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.

Submit:

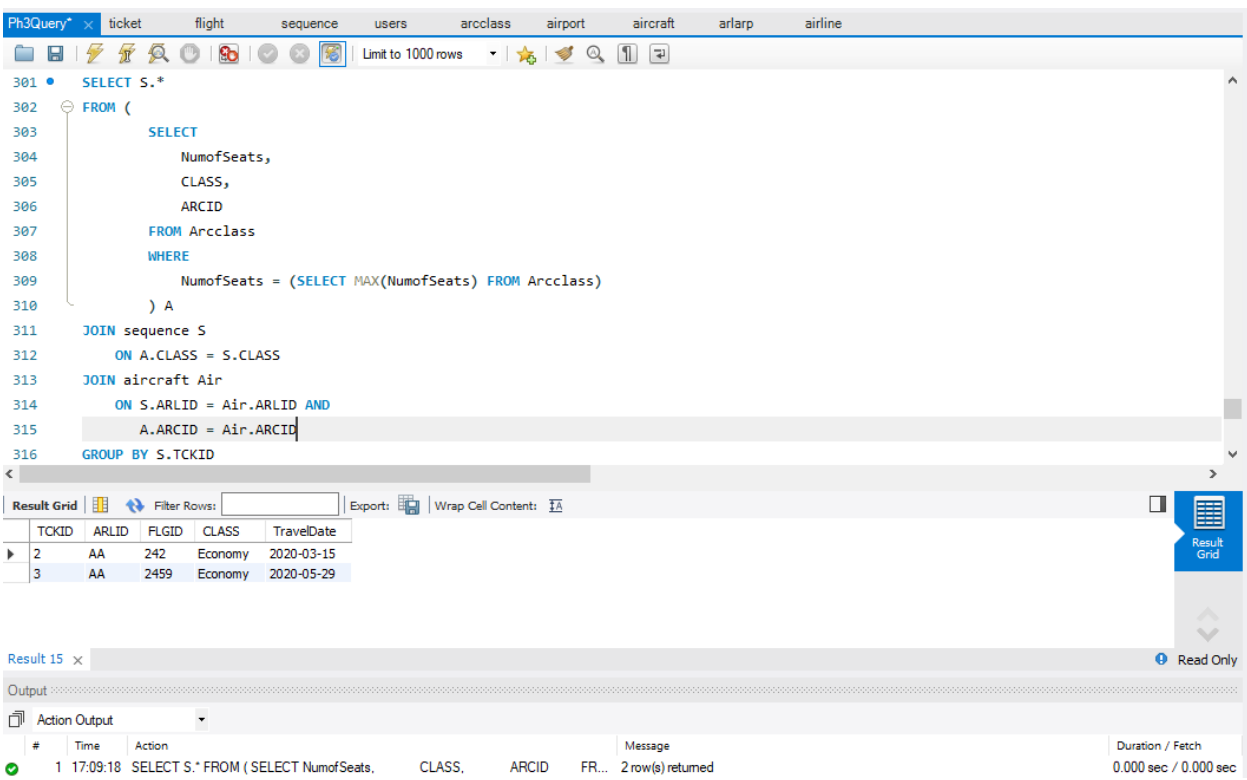
(1) Task description.

(2) Your SQL command typed in your submission report file.

(3) A screenshot(with your query, result grid, and the action output.

```
/*
```

```
Task 9: [5 points]
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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot( with your query, result grid, and the action output.
*/

SELECT S.*
FROM (
              SELECT
                     NumofSeats,
           CLASS,
           ARCID
        FROM Arcclass
              WHERE
                     NumofSeats = (SELECT MAX(NumofSeats) FROM Arcclass)
        ) A
JOIN sequence S
       ON A.CLASS = S.CLASS
JOIN aircraft Air
       ON S.ARLID = Air.ARLID AND
          A.ARCID = Air.ARCID
GROUP BY S.TCKID;
```



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.

For a better understanding of the task requirements, let's consider the following example for flight AA2459. Currently, the 'Economy' fare is $60.00, with the new policy will be $66.00. Hence, there is no information for the 'Extra' class, the 'First' class fare should be ($66.00*20%) = $79.20 rounded to $79.00 multiplied with 120% resulting $173.8 rounded to $174.00. So, the first-class fare for flight AA2459 will be $174.00.

For this task you can write more than query, but make sure that you will not hardcode the flight or class info.

Submit:

(1) Task description.

(2) Your SQL command(s) typed in your submission report file.

(3) The screenshot(s) with your query, result grid, and the action output.

```
/*
Task 10: [15 points]
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.
For a better understanding of the task requirements, let's consider the
following example for flight
AA2459. Currently, the 'Economy' fare is $60.00, with the new policy will
be $66.00. Hence, there is
no information for the 'Extra' class, the 'First' class fare should be
($66.00*20%) = $79.20 rounded to
```

```
$79.00 multiplied with 120% resulting $173.8 rounded to $174.00. So, the
first-class fare for flight
AA2459 will be $174.00.
For this task you can write more than query, but make sure that you will
not hardcode the flight or class
info.
Submit:
(1) Task description.
(2) Your SQL command(s) typed in your submission report file.
(3) The screenshot(s) with your query, result grid, and the action output.
*/


-- Update Economy
UPDATE flgclass
SET Fare = ROUND((Fare * .10) + Fare)
WHERE CLASS = 'Economy';

-- Update Extra
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((t2.Fare * .20) + t2.Fare)
WHERE  t1.CLASS = 'Extra' AND
              t2.CLASS = 'Economy' AND
              t1.FLGID = t2.FLGID AND
              t1.ARLID = t2.ARLID;

-- Update First
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((((t2.Fare * .20) + t2.Fare) * 1.2) + ((t2.Fare * .20)
+ t2.Fare))
WHERE  t1.CLASS = 'First' AND
              t2.CLASS = 'Economy' AND
              t1.FLGID = t2.FLGID AND
              t1.ARLID = t2.ARLID;

SELECT * FROM flgclass;
```

SQL File 3* | Ph3Query* | flgclass

Limit to 1000 rows

```sql
-- Update Economy
UPDATE flgclass
SET Fare = ROUND((Fare * .10) + Fare)
WHERE CLASS = 'Economy';

-- Update Extra
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((t2.Fare * .20) + t2.Fare)
WHERE   t1.CLASS = 'Extra' AND
        t2.CLASS = 'Economy' AND
        t1.FLGID = t2.FLGID AND
        t1.ARLID = t2.ARLID;

-- Update First
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((((t2.Fare * .20) + t2.Fare) * 1.2) + ((t2.Fare * .20) + t2.Fare))
WHERE   t1.CLASS = 'First' AND
        t2.CLASS = 'Economy' AND
        t1.FLGID = t2.FLGID AND
        t1.ARLID = t2.ARLID;

SELECT * FROM flgclass;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ARLID	FLGID	CLASS	Fare
AA	242	Economy	69.00
AA	242	Extra	83.00
AA	242	First	182.00
AA	2403	Economy	69.00
AA	2403	Extra	83.00
AA	2403	First	182.00
AA	2459	Economy	66.00
AA	2459	First	174.00
AA	2733	Economy	63.00
AA	2733	First	166.00
AA	2737	Economy	66.00
AA	2737	Extra	79.00
AA	2737	First	174.00
UA	589	Economy	99.00
UA	589	First	261.00
UA	589	Plus	110.00
UA	775	Economy	50.00
UA	775	First	132.00
UA	775	Plus	50.00
NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

flgclass 2 | Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:26:58	UPDATE flgclass SET Fare = ROUND((Fare * .10) + Fare) WHERE CLASS = 'Econo...	7 row(s) affected Rows matched: 7 Changed: 7 Warnings: 0	0.016 sec
2	01:26:58	UPDATE flgclass t1, flgclass t2 SET t1.Fare = ROUND((t2.Fare * .20) + t2.Fare) WH...	3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0	0.000 sec
3	01:26:58	UPDATE flgclass t1, flgclass t2 SET t1.Fare = ROUND((((t2.Fare * .20) + t2.Fare) * 1....	7 row(s) affected Rows matched: 7 Changed: 7 Warnings: 0	0.000 sec
4	01:26:58	SELECT * FROM flgclass LIMIT 0, 1000	19 row(s) returned	0.000 sec / 0.000 sec

Complete Code For Phase 2

```
/* Q1
Insert yourself as a New Customer. Do not provide the customer id in your
query.
*/
INSERT INTO `flight_reservation`.`users` ( `Name`, `email`, `Phone`, `Type`)
VALUES ( 'Chi Shing Poon', 'chi.SQL@gmail.com', '+1-682-557-1111', '1');
INSERT INTO `flight_reservation`.`users` (`Name`, `email`, `Phone`, `Type`)
VALUES ('Josh     Mendoza', 'josh.SQL@gmail.com', '+1-682-557-2222', '1');



/*Q2
Update your phone number to +1-837-721-8965
*/
UPDATE `flight_reservation`.`users` SET `Phone` = '+1-837-721-8965' WHERE
(`USRID` = '7');
UPDATE `flight_reservation`.`users` SET `Phone` = '+1-837-721-8966' WHERE
(`USRID` = '8');


/*Q3
: 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. [5 points]
b) Then, write another query that will return the total fare per 'SRQUENCE'
record. [5 points]
c) Finally, run an update command to correct the wrong values. [3 points]
d) Select all records from table 'TICKET'. [2 points]
*/
-- Q3A
SELECT FC.FLGID, FC.CLASS, FC.Fare
FROM FLGCLASS AS FC, SEQUENCE AS S
WHERE S.FLGID = FC.FLGID AND S.CLASS = FC.CLASS;

-- Q3B
SELECT T.TCKID, T.TotalFare
FROM TICKET AS T;

-- Q3C
UPDATE `flight_reservation`.`ticket` SET `TotalFare` =
(
SELECT FC.Fare
```

```sql
From FLGCLASS  AS FC, SEQUENCE AS S -- TICKET AS T
WHERE S.TCKID = TCKID AND FC.CLASS = S.CLASS AND FC.FLGID = S.FLGID);

-- Q3D
SELECT T.*
FROM TICKET AS T;


/* Q4
Question 4: Insert a new 'TICKET' record for you, as a customer, based on the
information below:
[6 points]
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 both tables
('TICKET' and 'SEQUENCE').
Show your work in steps.
*/

INSERT INTO `flight_reservation`.`ticket` ( `USRID`, `Type`, `TotalFare`,
`PurchaseDateTime`) -- VALUES ('1', '1', '1', '1', '1')
SELECT U.USRID AS USRID,  U.Type AS Type, FC.Fare AS TotalFare,
current_timestamp() AS PurchaseDateTime
FROM Users AS U, FLGCLASS AS FC
WHERE U.USRID = '7' AND FC.ARLID = 'AA' AND FC.FLGID = '2459' AND FC.CLASS =
'Economy';
SELECT *
FROM TICKET;

INSERT INTO `flight_reservation`.`sequence` (`TCKID`, `ARLID`, `FLGID`,
`CLASS`, `TravelDate`) -- VALUES ('1', '1', '1', '1', '1');
SELECT T.TCKID AS TCKID, FC.ARLID AS ARLID, FC.FLGID AS FLGID, FC.CLASS AS
CLASS, '2020-05-20' AS TravelDate
FROM Ticket AS T, FLGCLASS AS FC
WHERE T.TCKID = '3' AND FC.FLGID = '2459' AND FC.Fare = '60.00';

SELECT *
FROM SEQUENCE;



/* Q5
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.
*/

 SELECT F.ARLID, F.FLGID, F.DepARPID, F.DesARPID, F.ARCID,
REPLACE(F.TYPE, 1, 'Domestic') AS Type
FROM FLIGHT AS F
ORDER BY F.DesARPID ASC;
```

```
/* Q6
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.

Using UNION to display both the predicted fare and then the normal price.
Tuples with predicted fare is shown first.
*/


(SELECT C.ARLID, C.FLGID, C.CLASS, C.Fare, 1.05* C.Fare AS Increased_Fare
FROM FLGCLASS AS C
WHERE C.CLASS = 'Economy')
UNION
(SELECT C.ARLID, C.FLGID, C.CLASS, C.Fare, 1* C.Fare
FROM FLGCLASS AS C
WHERE C.CLASS != 'Economy');



/* Q7
Return the airline ids and the airline description for those airlines that
are not associated with
any airport

*/

SELECT DISTINCT L.*
FROM AIRLINE AS L, AIRPORT AS P
WHERE L.ARLID NOT IN (SELECT ARLID FROM ARLARP);
-- AND L.ARLID NOT IN(SELECT ARLID FROM FLIGHT); <-- Accounted for Flight as
well


/*Q8
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).
*/

(SELECT DISTINCT L.ARLID,  L.Name AS AIRLINE_NAME, P.ARPID, P.Name AS
AIRPORT_NAME
FROM AIRLINE AS L LEFT JOIN ARLARP AS A ON L.ARLID = A.ARLID , AIRPORT AS P
WHERE A.ARPID = P.ARPID)
UNION
(SELECT DISTINCT L.ARLID, L.NAME AS AIRLINE_NAME, IF(P.ARPID, NULL, 'N/A')
AS ARPID,
IF(P.NAME, NULL, 'N/A') AS AIRPORT_NAME
FROM AIRLINE AS L, ARLARP AS A, AIRPORT AS P
WHERE L.ARLID NOT IN (SELECT ARLID FROM ARLARP));
```

```
/*Q9
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).

IF(EXPRESSION, TRUE, FALSE)
*/

SELECT S.*, IF(CURRENT_DATE < S.TravelDate, datediff(
S.TravelDate,CURRENT_DATE), 'PAST')
AS DAYS_UNTIL_TRAVEL
FROM TICKET AS T, SEQUENCE AS S
WHERE T.TCKID = S.TCKID;


/* Q10
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.
*/

-- DISPLAYS ARLID, FLGID, AND THEIR RESPECTIVE CLASS OPTIONS
SELECT DISTINCT F.ARLID, F.FLGID, G.CLASS, C.ChangeFee, G.Fare, F.ARCID
-- IF( C.ARCID = F.ARCID AND C.CLASS = G.CLASS,C.NumofSeats, 'N/A') AS
NumofSeats
/*

CASE
      WHEN C.ARCID = F.ARCID AND C.CLASS = G.CLASS THEN C.NumofSeats
    END*/
FROM FLIGHT AS F , ARCCLASS AS C, FLGCLASS AS G
WHERE F.ARLID = G.ARLID AND C.CLASS = G.CLASS AND F.FLGID = G.FLGID
ORDER BY F.ARLID
;
```

Complete Code for Phase 3:

```
ALTER TABLE users
  DROP COLUMN Active;

/*
Task 1a: [8 points]
In this task, we want you to identify the customers that used our system
to book tickets. To do that,
you need to create a new column to the USERS table, and name it as
'Active'; define this column as a
small integer. Values for this column are '0' for non-active customers and
'1' for active. You do not
need to create an enumerated list. Consequently, for non-active customers,
there is not any ticket
information stored in our database. Your task is to update the 'Active'
column with '1' for all active
customers and with '0' for the non-active. This column must be left empty
for the customer
representatives and administrators.
*/

USE flight_reservations2020ph3;
ALTER TABLE users
       ADD Active INT DEFAULT 0;
UPDATE users, ticket
SET
       Active = 1
WHERE
       users.USRID = ticket.USRID;
SELECT * FROM flight_reservations2020ph3.users;

/*
Task 1b: [3 points]
For this task, you need to return the percentage of active customers.
*/

SELECT
       (SUM(Active)/COUNT(*))*100 AS 'Percentage of Active Customers'
FROM
       users;

/*
Task 2: [5 points]
By an error to the system, the 'Type' attribute from table 'Flight' 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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
```

```
(3) A screenshot with your query, result grid, and the action output.
*/

UPDATE ticket AS Ti
SET Type = 2
WHERE
       TCKID IN(
                                SELECT
                                       TCKID
                                FROM
                                       sequence
                                GROUP BY TCKID
                                HAVING COUNT(*) > 1
                        );
SELECT * FROM flight_reservations2020ph3.ticket;

/*
Task 3: [10 points]
Here you need to create a view named as 'viewTicketlInfo', 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
Submit:
(1) Task description.
(2) Your 'Create view' command typed in your submission report file. Use
only the aboveprovided
names for your attributes. Do NOT alter any name.
(3) A screenshot, with your query, result grid, and the action output.
(4) Then your 'Select view' command typed in your submission report file.
(5) A screenshot with your query, result grid, and the action output.
*/

CREATE VIEW viewTicketInfo AS
SELECT
       DATE(T.PurchaseDateTime) AS ticketPurchaseDate,
    T.TCKID AS ticketID,
    IF(T.Type = 1, 'one-way', 'round trip') AS ticketType,
       U.Name AS custName,
    U.Phone AS custPhone,
    S.FLGID AS flightID,
    S.TravelDate AS travelDate,
    S.Class AS flightClass,
    CONCAT('$', T.TotalFare) AS flightFare
```

```sql
FROM ticket T
JOIN users U
        ON T.USRID = U.USRID
JOIN sequence S
        ON T.TCKID = S.TCKID
ORDER BY DATE(T.PurchaseDateTime);
SELECT * FROM viewTicketInfo;

/*
Task 4: [5 points]
Write a query that retrieves data from your stored 'viewTicketlInfo' 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'
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

SELECT
        v.ticketPurchaseDate as 'Purchase Date',
    v.ticketID as 'Ticket',
    v.ticketType as 'Ticket Type',
    v.custName as 'Customer',
    v.custPhone as 'Customer Number',
    v.flightFare as 'Ticket fare'
FROM
        viewTicketInfo v
ORDER BY v.flightFare ASC;

/*
Task 5: [10 points]
In this task, we want to count how many seats left for two specific flights.
You need to write one query
that returns:
• the fight 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!
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.

*/
SELECT DISTINCT
        CONCAT(S.ARLID, S.FLGID) AS 'Flight Number',
    S.TravelDate AS 'Travel Date',
        F.DepARPID AS 'Departure ID',
    Apt_Dep.Name AS 'Departure Name',
    F.DesARPID AS 'Destination ID',
    Apt_Des.Name AS 'Destination Name',
    F.DepTime AS 'Departure Time',
    F.DesTime AS 'Destination Time',
    S.CLASS AS 'Class',
    Arc.NumofSeats AS 'Airplane Capacity',
    COUNT(F.FLGID) AS 'Tickets Sold',
    Arc.NumofSeats - count(F.FLGID) AS 'Available Seats'
FROM
        sequence S
JOIN flight F
        ON S.FLGID = F.FLGID AND S.ARLID = F.ARLID  -- Duplicates Sequence?
JOIN aircraft Aft
        ON S.ARLID = Aft.ARLID
JOIN arcclass Arc
        ON S.CLASS = Arc.CLASS AND F.ARCID = Arc.ARCID  -- Displays 2 as AA
corresponds with 2 and 3? Duplicate values
JOIN airport Apt_Dep
        ON F.DepARPID = Apt_Dep.ARPID
JOIN airport Apt_Des
        ON F.DesARPID = Apt_Des.ARPID
JOIN ticket T
        ON S.TCKID = T.TCKID
 WHERE
        (S.ARLID = 'UA' AND S.FLGID = '589' AND S.TravelDate = '2020-04-24')
OR
        (S.ARLID = 'UA' AND S.FLGID = '775' AND S.TravelDate = '2020-05-02')
GROUP BY
        S.FLGID
ORDER BY
        S.FLGID AND S.CLASS;


/*
Task 6: [8 points]
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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

SELECT
       U.Name,
    U.Phone,
       COUNT(DISTINCT T.USRID) AS 'Tickets Issued',
       COUNT(S.TCKID) AS 'Total Flights',
    IFNULL(T.TotalFare,0) AS 'Total Fare',
    IFNULL(T.TotalFare / COUNT(S.TCKID),0) AS 'Average Fare Per Flight'
FROM
       Users U
LEFT JOIN ticket T
       ON U.USRID = T.USRID
LEFT JOIN sequence S
       ON U.USRID = T.USRID AND
    T.USRID = S.TCKID
GROUP BY U.Name, U.Phone, T.USRID, U.USRID -- S.TCKID
ORDER BY T.TotalFare DESC, U.Name;

/*
Task 7: [8 points]
Due to the current situation with COVID-19, our 'Online travel reservation
system' CEO decided to
grand 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.
Submit:
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot with your query, result grid, and the action output.
*/

UPDATE ticket
SET
       TotalFare = (TotalFare * .85)
WHERE
       TCKID IN(
              SELECT
           TCKID
              FROM
                     sequence
              GROUP BY
                     TCKID, TravelDate
              HAVING
```

```
                        TravelDate > '2020-04-20'
) AND
       ticket.Type = 2;
SELECT * FROM flight_reservations2020ph3.ticket;

/*
Task 8: [8 points]
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.
Submit:
(1) Task description.
(2) Your SQL command(s) typed in your submission report file.
(3) The screenshot(s), with your query, result grid, and the action
output.
*/

INSERT INTO Arlarp(ARLID, ARPID)
SELECT
       F.ARLID,
    F.ARPID
FROM(
    SELECT                    -- All values from ARLID + DepARLID
              ARLID,
        DepARPID AS ARPID  -- First Alias matters
       FROM flight

    UNION

       SELECT
              ARLID,         -- ALL values from ARLID + DesARPID
              DesARPID
       FROM flight
    )F
WHERE NOT EXISTS      -- Exclude values that pairs from Arlarp
       (
    SELECT 1
    FROM arlarp A
    WHERE A.ARLID = F.ARLID AND -- Where Arlarp pairs match Flight
                A.ARPID = F.ARPID
       );
SELECT * FROM flight_reservations2020ph3.arlarp;

/*
Task 9: [5 points]
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.
Submit:
```

```
(1) Task description.
(2) Your SQL command typed in your submission report file.
(3) A screenshot( with your query, result grid, and the action output.
*/

SELECT S.*
FROM (
              SELECT
                     NumofSeats,
           CLASS,
           ARCID
        FROM Arcclass
              WHERE
                     NumofSeats = (SELECT MAX(NumofSeats) FROM Arcclass)
        ) A
JOIN sequence S
       ON A.CLASS = S.CLASS
JOIN aircraft Air
       ON S.ARLID = Air.ARLID AND
          A.ARCID = Air.ARCID
GROUP BY S.TCKID;

/*
Task 10: [15 points]
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.
For a better understanding of the task requirements, let's consider the
following example for flight
AA2459. Currently, the 'Economy' fare is $60.00, with the new policy will
be $66.00. Hence, there is
no information for the 'Extra' class, the 'First' class fare should be
($66.00*20%) = $79.20 rounded to
$79.00 multiplied with 120% resulting $173.8 rounded to $174.00. So, the
first-class fare for flight
AA2459 will be $174.00.
For this task you can write more than query, but make sure that you will
not hardcode the flight or class
info.
Submit:
(1) Task description.
(2) Your SQL command(s) typed in your submission report file.
(3) The screenshot(s) with your query, result grid, and the action output.
*/
```

```sql
-- Update Economy
UPDATE flgclass
SET Fare = ROUND((Fare * .10) + Fare)
WHERE CLASS = 'Economy';

-- Update Extra
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((t2.Fare * .20) + t2.Fare)
WHERE  t1.CLASS = 'Extra' AND
              t2.CLASS = 'Economy' AND
              t1.FLGID = t2.FLGID AND
              t1.ARLID = t2.ARLID;

-- Update First
UPDATE flgclass t1, flgclass t2
SET t1.Fare = ROUND((((t2.Fare * .20) + t2.Fare) * 1.2) + ((t2.Fare * .20)
+ t2.Fare))
WHERE  t1.CLASS = 'First' AND
              t2.CLASS = 'Economy' AND
              t1.FLGID = t2.FLGID AND
              t1.ARLID = t2.ARLID;

SELECT * FROM flgclass;
```