**Project Component 3: SQL Implementation and Java Database Connectivity (JDBC)**

**Dr. Wassim Itani**

**Project Topic:** Design of a database system for a FrequentFlier program

**Project Collaboration Model:** This project component should be developed in a **team of 2 members minimum and 4 members maximum**. **You cannot work independently without a team.**

**Due Date/Time:** 04/07/2023 11:59:59 PM.

1. Deliver the .ZIP file (other formats won’t be accepted) containing your ***insertions.sql*** (refer to part 1), ***queries.sql*** (refer to part 2), ***querydisplay.jar*** (refer to part 3), and ***Readme.txt*** files on Blackboard by the above due date/time. The Readme.txt file should contain the name of the group and the names and GMU IDs of the members. Each member should submit the same copy of the ZIP file on Blackboard to secure an entry in the Grade Center. Your .zip file should be named as follows: P3\_[Your Last Name]\_[Your GMUID].zip. For instance, if John Smith with GMUID: G12345678 were to submit this file, John would name it: P3\_Smith\_G12345678.zip

**Project Component 3 Description**

The Project Component 3 consists of 3 parts:

**Part 1 (25%):**

1. Execute the ***FrequentFlier\_Impl.sql*** script (posted on Blackboard) to generate the FrequentFlier database tables.
2. Use the SQL INSERT INTO statement to Insert sample records into the FrequentFlier generated database according to the following guidelines:

* *Insert 40 passenger records into the Passengers table. Assume that all the passengers added are part of the FrequentFlier program.* (5%)
* *Insert the passengers addresses and phone numbers into the Addresses and Phones tables respectively. At least 5 passengers must have 2 addresses and 2 phone numbers.* (3%)
* *Insert the login information for your passengers. You should insert 40 rows corresponding to the number of passengers you have in the Passengers table (1-to-1 relationship)* (2%)
* *Populate the Point\_Accounts table. It should contain 40 records corresponding to the number of passengers you have in the Passengers table (Each passenger should have an entry in the Point\_Accounts table since all the passengers are assumed to be part of the FrequentFlier program).* (2%)
* *Insert data into the Flights, Trips, and Flights\_Trips tables. You should have a minimum of 10 Trips and 10 Flights. Let some Flights contain some common Trips with other Flights).* (3%)
* *Insert the card information into the Cards table. Each passenger must have a valid card. Let at least 5 passengers have a second expired (non-valid) card.* (2%)
* *Insert the information of at least 10 promotions into the Promotions table.* (2%)
* *For simplicity, leave the Flight\_Promotions table empty.*
* *Create a set of 5 exchange centers.* (1%)
* *Create a list of 15 awards.* (1%)
* *Populate the Redemption\_History table with sample data linking the Passengers, Awards, Point\_Accounts, and Exchg\_Centers tables. You should have a minimum of 20 records in the Redemption\_History table.* (4%)

Store the SQL INSERT INTO statements in a file named: ***insertions.sql***. Make sure that your insertions.sql script executes successfully on the FrequentFlier database on the GMU Oracle Server.

**Part 2 (50%):**

Provide the SQL SELECT statements satisfying the following queries. Include your queries in a file named ***queries.sql***. Make sure that your queries.sql script executes successfully on the FrequentFlier database on the GMU Oracle Server. **Number your queries using SQL Comments to make it easier on your GTA to grade your project. Not following this guideline would result in a deduction of 10%.**

1) *Select the promotion id, promotion action, and promotion period provided to a particular flight id.* (2.5%)

2) *Display all the flight ids, flight points, and the flight arrival dates for a particular passenger name.* (2.5%)

3) *Find the Flight ids and the number of promotions provided to each flight id.* (2.5%)

4) Find the passengers ids and names who arrived to Berlin between March 1st and March 15th 2023. (2.5%)

5) *Display for a particular flight id, the flight id, source, destination, the number of points collected, and the trip ids and arrival dates included in the flight.* (2.5%)

6) *Find the number of expired cards available in the database.* (2.5%)

7) *Find the passenger with the maximum number of expired cards.* (2.5%)

8) *Find the redemption history of a particular passenger name. You should display the award ID, award description, passenger name,center id, and number of points redeemed.* (3%)

9) *Display the name and occupation of the passengers living in Fairfax.* (3%)

10) *Display the sum of points of the passengers living in Fairfax.* (3%)

11) *Display the passenger name with the maximum number of collected points.* (3%)

12) *Find the total number of points redeemed on a particular date.* (3%)

13) *Find the number of awards redeemed by a particular passenger id.* (3%)

14) *Find the number of passengers who redeemed awards from a particular center id.* (3%)

15) *Find the total number of awards in the database.* (3%)

16) *Display a list of passenger names living in Fairfax and whose occupation is Engineer.* (2.5%)

17) *Find the list of trips not included in any flight.* (3%)

18) *Find the trip booked the most by passengers* (3%)

**Part 3 (25%):**

In this part, the FrequentFlier developers ask you to develop a simple standalone application in Java to test the results of Oracle SQL queries. The application consists of 2 main Java Swing JFrames. The first JFrame in Figure 1 contains the connection parameters for accessing the Oracle database server.

Graphical user interface, application

Description automatically generated

Table

Description automatically generatedFigure 1: The database connection parameters

Figure 2: The query execution JFrame

Note that the user can enter any Oracle Server parameters (not necessarily the GMU Oracle Server parameters as demonstrated below). Once the Connect button is clicked and the connection is successful, the application displays the JFrame demonstrated in Figure 2. Very simply, the user can enter any SQL DML query statement (SELECT, DELETE, INSERT INTO, UPDATE) and execute it. If the SQL statement is a SELECT statement, the result is displayed in a JTable component. The JTable should dynamically figure out the number and names of columns in the SQL result and display it on the screen. You may use the ojdbc10.jar JDBC drivers (posted on Blackboard) from Oracle for supporting the database connections. You should package your application in a single Java JAR file containing all the application dependencies including the JDBC drivers needed for the connection to the database. Name the file ***querydisplay.jar***. The GTAs should be able to execute the command:

*java -jar querydisplay.jar*

to run the application.