GROUP NUMBER: Group 22

GROUP MEMBERS: Colby Frison, Brayden Garner, Brendan Ford, Jackson Dunlap

GRADED HOMEWORK NUMBER: 3

COURSE: CS/DSA-4513 - DATABASE MANAGEMENT

SECTION: 002

SEMESTER: FALL 2025

INSTRUCTOR: EGAWATI PANJEI

SCORE:

Contents

1 Problem 1: JDBC and Stored Procedures (GQ1)

1.1 Problem Description

Write a Java program using JDBC and Azure SQL Database to manage a relational database with the following schema:

- **Passenger** (pid: integer, pname: string, tier: string, age: integer)
- **Pilot** (pIid: integer, pIname: string, hours: real)
- **Flight** (fnum: string, origin: string, destination: string, dep_time: string, arrival_time: string, pIid: integer)
- **Booking** (pid: integer, fnum: string)

The program provides a menu-driven interface to perform the following operations on the Pilot relation:

1. **Insert a Pilot (Hours Based on Average Flight Time):** Insert a new pilot with hours calculated as the average flight time of all flights.
2. **Insert a Pilot (Hours Based on Passenger Tier):** Insert a new pilot with hours calculated as the average flight time of flights piloted by pilots who have at least one passenger of a specified tier.
3. **Display All Pilots:** Display all records in the Pilot table sorted by pIid.
4. **Quit:** Exit the program.

1.2 Implementation Overview

1.2.1 Transact-SQL Stored Procedures

Two stored procedures were created in Azure SQL Database to implement Options 1 and 2:

InsertPilotAvgFlightTime Accepts `pIid` and `pIname` as parameters. Calculates hours as the average flight time of all flights by using `DATEDIFF(MINUTE, dep_time, arrival_time)` divided by 60.0. Uses `ISNULL()` to return 0 if no flights exist.

InsertPilotByPassengerTier Accepts `pIid`, `pIname`, and `tier` as parameters. Calculates hours as the average flight time of flights piloted by pilots who have at least one passenger of the specified tier. Uses same time calculation and NULL handling as the first procedure.

1.2.2 Java JDBC Implementation

The Java program uses JDBC to connect to Azure SQL Database and provides a menu-driven interface. Key features:

- Uses `CallableStatement` to execute stored procedures
- Uses `Statement` and `ResultSet` to query and display pilot data
- Continues until user selects Option 4 (Quit)
- Includes proper exception handling for SQL errors

2 Compilation and Execution

2.1 Compilation

The Java program was compiled using:

```
javac Problem1.java
```

2.2 Stored Procedure Creation

The Transact-SQL stored procedures were created in Azure SQL Database by executing the `Problem1_StoredProcedures.sql` file in the Azure SQL Query Editor.

2.3 Program Execution

The program was executed using:

```
java -cp ".;sqljdbc_13.2.1.0_enu\sqljdbc_13.2\enu\jars\mssql-jdbc-13.2.1.jre11.jar"
     Problem1
```

3 Test Execution Results

3.1 Program Start and Main Menu

```
PS C:\Users\colby\OneDrive\Documents\notes\200 School\205 Fall 2025\CS 4513 - DBMS\Assignments\Homework
3\src>  java -cp ".;sqljdbc_13.2.1.0_enu\sqljdbc_13.2\enu\jars\mssql-jdbc-13.2.1.jre11.jar" Problem1

========================================
   Welcome to Pilot Management System!
   CS 4513 - Homework 3, Problem 1
========================================
```

Figure 1: Program Start - Connection to Azure SQL Database

```
========================================
     PILOT MANAGEMENT SYSTEM
========================================
Please select one of the options below:
1) Insert Pilot (Hours Based on Average Flight Time)
2) Insert Pilot (Hours Based on Passenger Tier)
3) Display All Pilots
4) Quit
========================================
Enter your choice:
```

Figure 2: Main Menu - Four menu options displayed

The program successfully connects to Azure SQL Database and displays the menu-driven interface with all four options.

3.2 Option 1: Insert Pilot Based on Average Flight Time

Option 1 inserts a new pilot with hours calculated based on the average flight time of all flights in the database. The stored procedure `InsertPilotAvgFlightTime` is called with pilot ID and name as parameters.

3.2.1 Execution 1

```
Enter your choice: 1

--- Insert Pilot (Average Flight Time) ---
Enter Pilot ID (pIid): 11
Enter Pilot Name (pIname): Name1-1

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on average flight time of all flights.
```

Figure 3: Option 1 Execution 1 - Inserting pilot with pIid=100, pIname=name1

```
================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 4: Option 1 Result 1 - Pilot table after insertion (Option 3 display)

Note: Pilots 201-204 (Smith, Chen, Garcia, Patel) are from the original dataset. The new pilot (Name1-X) shows the calculated hours based on average flight time.

3.2.2 Execution 2

```
Enter your choice: 1

--- Insert Pilot (Average Flight Time) ---
Enter Pilot ID (pIid): 12
Enter Pilot Name (pIname): Name1-2

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on average flight time of all flights.
```

Figure 5: Option 1 Execution 2 - Inserting another pilot with average flight time calculation

```
================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 6: Option 1 Result 2 - Pilot table showing both new insertions

3.2.3 Execution 3

```
Enter your choice: 1

--- Insert Pilot (Average Flight Time) ---
Enter Pilot ID (pIid): 13
Enter Pilot Name (pIname): Name1-3

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on average flight time of all flights.
```

Figure 7: Option 1 Execution 3 - Third insertion using average flight time

```
================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
13         | Name1-3                       | 4.36
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 8: Option 1 Result 3 - Pilot table with all three new pilots from Option 1

Observation: All three pilots inserted via Option 1 have the same hours value, demonstrating that the calculation is based on the current average flight time across all flights in the database.

3.3 Option 2: Insert Pilot Based on Passenger Tier

Option 2 inserts a new pilot with hours calculated based on the average flight time of flights piloted by pilots who have at least one passenger of the specified tier. The stored procedure `InsertPilotByPassengerTier` is called with pilot ID, name, and tier as parameters.

3.3.1 Execution 1 - Gold Tier

```
Enter your choice: 2

--- Insert Pilot (By Passenger Tier) ---
Enter Pilot ID (pIid): 21
Enter Pilot Name (pIname): Name2-1
Enter Passenger Tier (e.g., Gold, Silver, Bronze): Gold

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on flights with Gold tier passengers.
```

Figure 9: Option 2 Execution 1 - Inserting pilot based on Gold tier passengers

```
================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
13         | Name1-3                       | 4.36
21         | Name2-1                       | 4.92
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 10: Option 2 Result 1 - Pilot table after Gold tier insertion

Note: Pilots 201-204 (Smith, Chen, Garcia, Patel) and pilots 11-13 (Name1-X) are from the original dataset. The new pilot (Name2-X) show the insertion of pilot based on tier.

3.3.2 Execution 2 - Silver Tier

```
Enter your choice: 2

--- Insert Pilot (By Passenger Tier) ---
Enter Pilot ID (pIid): 22
Enter Pilot Name (pIname): Name2-2
Enter Passenger Tier (e.g., Gold, Silver, Bronze): Silver

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on flights with Silver tier passengers.
```

Figure 11: Option 2 Execution 2 - Inserting pilot based on Silver tier passengers

```
======================================================================
                    ALL PILOTS IN DATABASE
======================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
13         | Name1-3                       | 4.36
21         | Name2-1                       | 4.92
22         | Name2-2                       | 4.56
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
======================================================================
End of pilot records.
```

Figure 12: Option 2 Result 2 - Pilot table showing Silver tier calculation

3.3.3 Execution 3 - Bronze Tier

```
Enter your choice: 2

--- Insert Pilot (By Passenger Tier) ---
Enter Pilot ID (pIid): 23
Enter Pilot Name (pIname): Name2-3
Enter Passenger Tier (e.g., Gold, Silver, Bronze): Bronze

Connecting to database...
Executing stored procedure...

[SUCCESS] Pilot inserted successfully!
Rows affected: 1
Hours calculated based on flights with Bronze tier passengers.
```

Figure 13: Option 2 Execution 3 - Inserting pilot based on Bronze tier passengers

```
================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
13         | Name1-3                       | 4.36
21         | Name2-1                       | 4.92
22         | Name2-2                       | 4.56
23         | Name2-3                       | 4.64
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 14: Option 2 Result 3 - Pilot table with all tier-based calculations

Observation: Different passenger tiers produce different hours values, demonstrating that the tier-based calculation correctly filters flights based on passenger membership level.

3.4 Option 3: Display All Pilots

Option 3 displays all pilots currently in the database sorted by pId. This option was executed multiple times throughout testing to verify the results of Options 1 and 2 insertions.

```
========================================
     PILOT MANAGEMENT SYSTEM
========================================
Please select one of the options below:
1) Insert Pilot (Hours Based on Average Flight Time)
2) Insert Pilot (Hours Based on Passenger Tier)
3) Display All Pilots
4) Quit
========================================
Enter your choice: 3

Connecting to database...
Retrieving pilot records...

================================================================
                    ALL PILOTS IN DATABASE
================================================================
Pilot ID   | Pilot Name                    | Hours
----------------------------------------------------------------
11         | Name1-1                       | 4.36
12         | Name1-2                       | 4.36
13         | Name1-3                       | 4.36
21         | Name2-1                       | 4.92
22         | Name2-2                       | 4.56
23         | Name2-3                       | 4.64
201        | Smith                         | 12000.00
202        | Chen                          | 9500.00
203        | Garcia                        | 8000.00
204        | Patel                         | 7000.00
================================================================
End of pilot records.
```

Figure 15: Option 3 - Final display showing all pilots in database sorted by pId

The final display shows:

- Original pilots (IDs 201-204): Smith, Chen, Garcia, and Patel from the initial dataset
- New pilots from Option 1 (IDs 11-13): Name1-1, Name1-2, Name1-3 with hours based on average flight time
- New pilots from Option 2 (IDs 21-23): Name2-1 (Gold), Name2-2 (Silver), Name2-3 (Bronze) with hours based on tier

3.5 Option 4: Program Termination

Option 4 allows the user to exit the program cleanly. When selected, the program displays a termination message and exits.

```
========================================
     PILOT MANAGEMENT SYSTEM
========================================
Please select one of the options below:
1) Insert Pilot (Hours Based on Average Flight Time)
2) Insert Pilot (Hours Based on Passenger Tier)
3) Display All Pilots
4) Quit
========================================
Enter your choice: 4

========================================
   Exiting Pilot Management System.
   Thank you for using our system!
========================================
```

Figure 16: Option 4 - Program termination with exit message

The program successfully terminates with an appropriate farewell message, demonstrating that the menu loop only exits when Option 4 is explicitly selected by the user.

4 Summary

4.1 Requirements Verification

All requirements are met:

- **JDBC Connection:** Program successfully connects to Azure SQL Database
- **Menu-driven interface:** Four-option menu displayed and functional
- **Program termination:** Program only exits when Option 4 is selected
- **Stored procedures:** Options 1 and 2 implemented as T-SQL stored procedures accepting required parameters (pIid, pIname, and tier for Option 2)
- **Flight time calculation:** Stored procedures calculate flight time by subtracting dep_time from arrival_time and converting to hours
- **Testing requirement:** Option 3 executed before and after each insertion to display changes
- **Execution count:** Options 1 and 2 each executed three times with different values
- **Option 4 execution:** Executed to verify proper program termination
- **Code comments:** Both Java and SQL files include detailed comments

5 Conclusion

The Java program and Transact-SQL stored procedures were successfully implemented and tested according to all assignment requirements. The implementation demonstrates:

- Effective use of JDBC for database connectivity
- Proper implementation of stored procedures in T-SQL
- Menu-driven user interface with appropriate input/output handling
- Correct calculation of flight times and hours
- Comprehensive testing of all program features