# **Final Project Report: Airport Information System**

Table of Contents

[Final Project Report: Airport Information System 1](#_Toc39787432)

[STAGE 1: CONCEPTUAL DESIGN 2](#_Toc39787433)

[STAGE 2: LOGICAL DESIGN 3](#_Toc39787434)

[STAGE 3: TEST DATA 4](#_Toc39787435)

[Table: Model 4](#_Toc39787436)

[Table: Airplane 4](#_Toc39787437)

[Table: Employee 4](#_Toc39787438)

[Table: Technician 5](#_Toc39787439)

[Table: TrafficController 5](#_Toc39787440)

[Table: QualityTest 5](#_Toc39787441)

[Table: TestHistory 5](#_Toc39787442)

[STAGE 4: PROGRAM EXECUTION AND OUTPUT 7](#_Toc39787443)

[Query 1: Insert a new technician into the database (Menu option 5 in program) 7](#_Toc39787444)

[Query 2: Delete an existing airplane from the database (Menu option (6) in program) 10](#_Toc39787445)

[Query 3: Update the expertise of an existing technician (Menu option 7 in program) 13](#_Toc39787446)

[Query 4: List the details of the technician whose salary is greater than the average of the salary of all technicians (Menu option 8 in program) 16](#_Toc39787447)

[Query 5: List all the model numbers that a given technician has the expertise, along with their capacity and weight information (Menu option 9 in program) 17](#_Toc39787448)

[Query 6: List the total number of technicians who are experts in each model (Menu option 10 in program) 18](#_Toc39787449)

[Query 7: List the details (test number, test name, maximum score, etc.) of the FAA tests for a given airplane, sorted by the maximum scores (Menu option 11 in program) 19](#_Toc39787450)

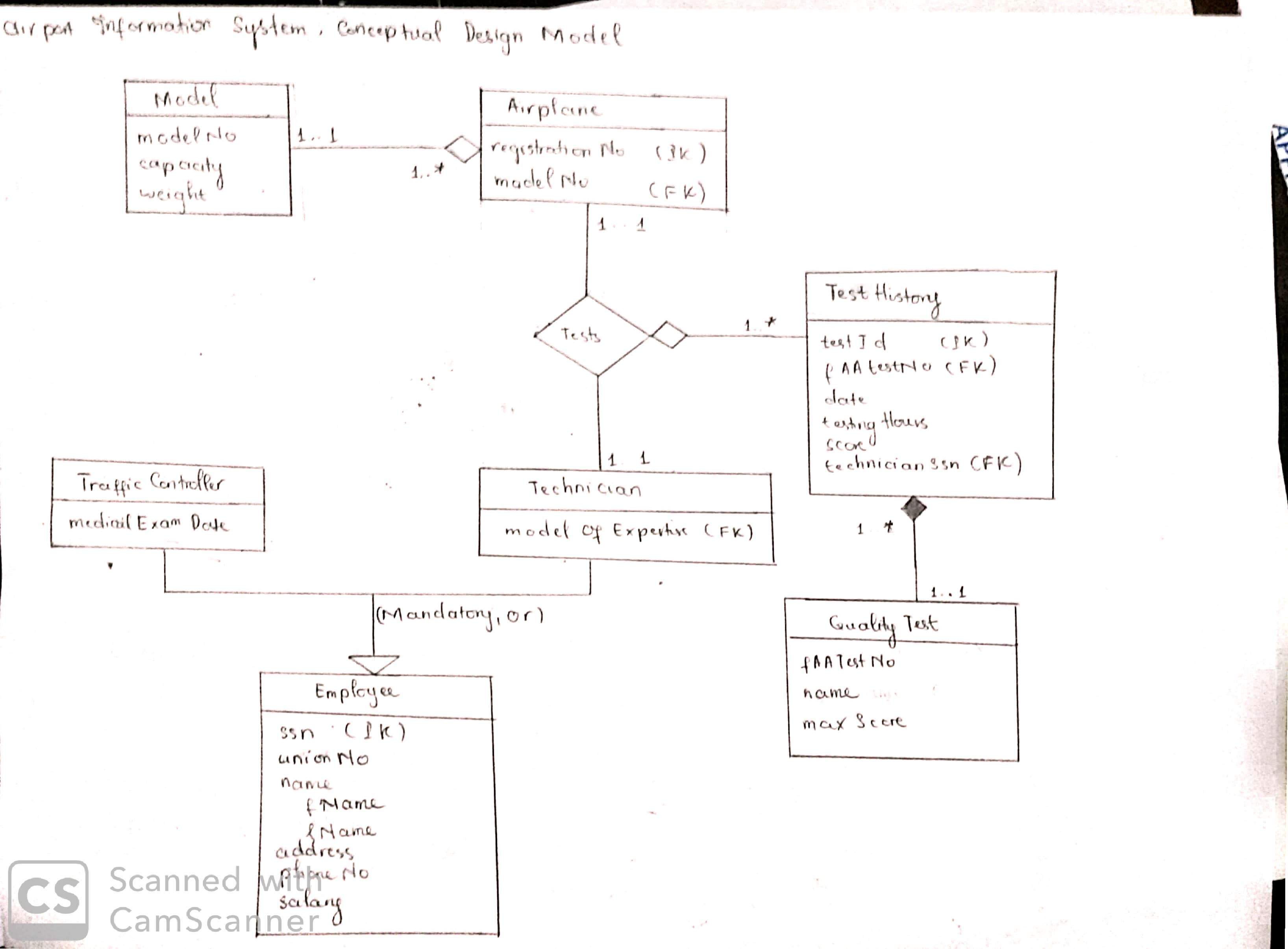
[Query 8: List the most recent annual medical examination and his/her union membership number for each traffic controller (Menu option 12 in program) 20](#_Toc39787451)

[Query 9: List the total number of tests done by each technician for a given airplane (Menu option 13 in program) 21](#_Toc39787452)

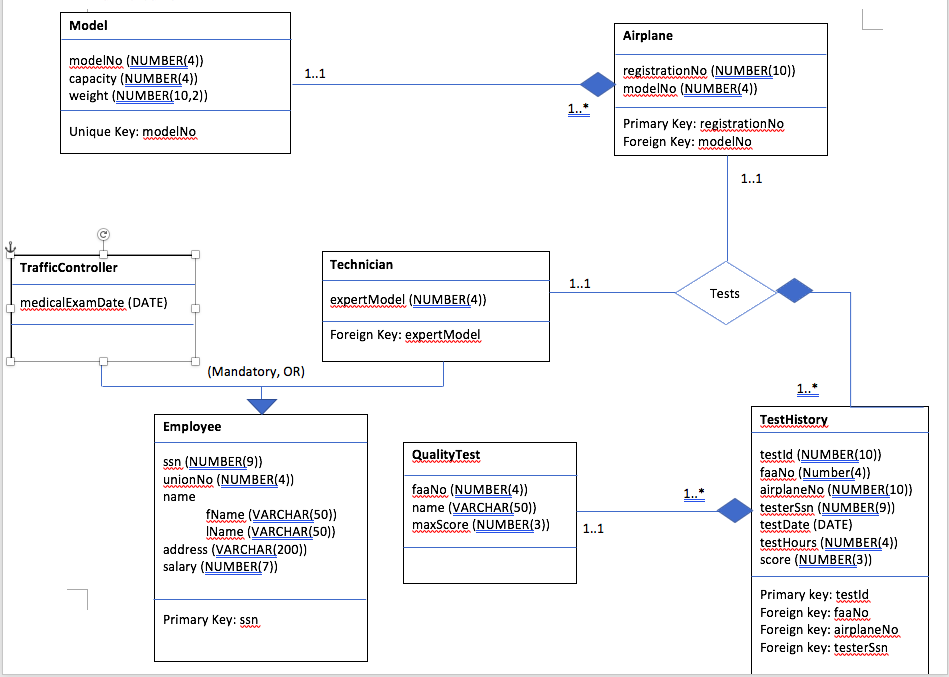
[Query 10: List the name of the technician, the registration number of the airplane, and the FAA number of those tests done between September 2015 and December 2015, sorted by the FAA numbers (Menu option 14 in program) 22](#_Toc39787453)

[STAGE 5: CONCLUSION 23](#_Toc39787454)

## STAGE 1: CONCEPTUAL DESIGN



## STAGE 2: LOGICAL DESIGN



## STAGE 3: TEST DATA

Here is a snapshot of the sample data that is used to initialize the data table:

### Table: Model

|  |  |  |
| --- | --- | --- |
| **ModelNo** | **Capacity** | **Weight** |
| 1000 | 500 | 3000.5 |
| 7275 | 200 | 2310 |
| 9999 | 50 | 1000 |
| 4449 | 700 | 5000 |

### Table: Airplane

|  |  |
| --- | --- |
| **RegistrationNo** | **ModelNo** |
| 123456789 | 4449 |
| 987543621 | 7275 |
| 111222333 | 4449 |
| 444555888 | 1000 |
| 999222444 | 7275 |

### Table: Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SSN** | **UnionNo** | **FirstName** | **LastName** | **Address** | **Salary** |
| 721074426 | 0406 | Alice | Wonderland | 1320 S Dixie Hwy, Miami, FL 33146 | 65000 |
| 264153998 | 0125 | Martina | Velichovska | 39 Stockton St, San Francisco, CA 94108 | 80000 |
| 170119497 | 2030 | Gia | Do | Boulevard de Parc, 77700 Coupvray, France | 72000 |
| 241032961 | 0406 | Brad | Pitt | 6701 Red Rd, Coral Gables, FL 33143 | 70000 |
| 619866062 | 0125 | Clemson | Martynev | 6620 SW 57th Ave, South Miami, FL 3314 | 82000 |

### Table: Technician

|  |  |
| --- | --- |
| **SSN** | **ExpertModel** |
| 721074426 | 7275 |
| 721074426 | 1000 |
| 721074426 | 4449 |
| 264153998 | 7275 |
| 264153998 | 4449 |
| 170119497 | 7275 |

### Table: TrafficController

|  |  |
| --- | --- |
| **SSN** | **MedExamDate** |
| 241032961 | 04/04/2020 |
| 619866062 | 04/01/2020 |

### Table: QualityTest

|  |  |  |
| --- | --- | --- |
| **FAANo** | **Name** | **MaxScore** |
| 1237 | Wings Test | 100 |
| 7321 | Fuel Test | 50 |
| 4421 | Wheels Test | 60 |
| 3445 | Windows Test | 20 |

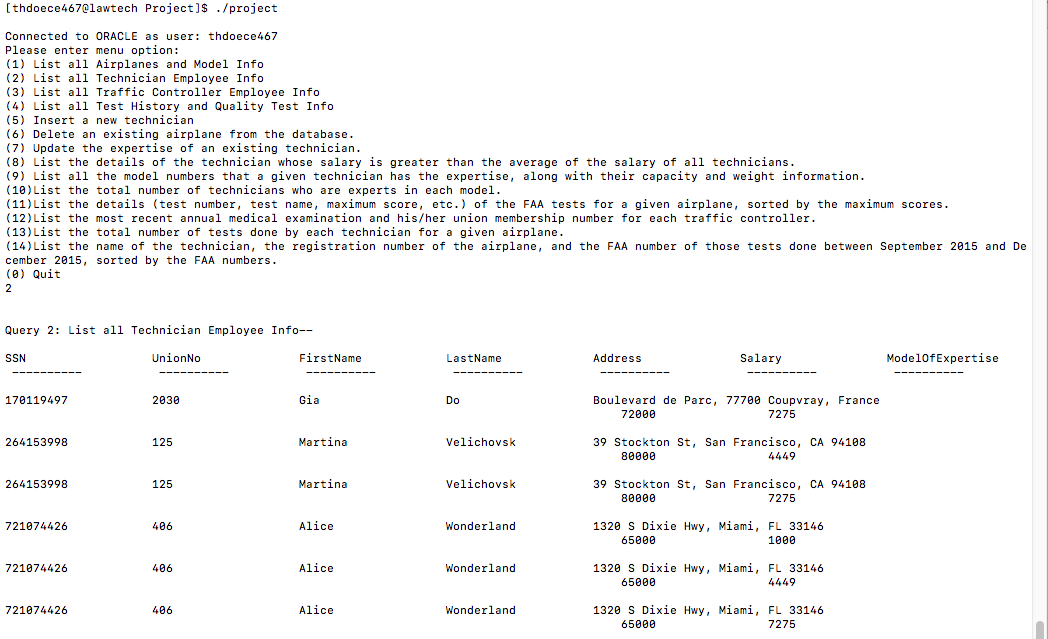
### Table: TestHistory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TestID** | **FAANo** | **AirplaneNo** | **TesterSSN** | **TestDate** | **TestHours** | **Score** |
| 0123456789 | 7321 | 123456789 | 721074426 | 04/04/2020 | 4 | 99 |
| 0023456789 | 4421 | 111222333 | 264153998 | 09/08/2015 | 14 | 40 |
| 0003456789 | 3445 | 999222444 | 721074426 | 02/02/2019 | 9 | 17 |
| 0000456789 | 1237 | 987543621 | 170119497 | 09/08/2019 | 10 | 88 |
| 5222456789 | 3445 | 111222333 | 264153998 | 04/26/2015 | 12 | 78 |
| 777777777 | 7321 | 111222333 | 264153998 | 07/18/2015 | 5 | 14 |
| 888888988 | 3445 | 987543621 | 170119497 | 02/14/2014 | 20 | 20 |

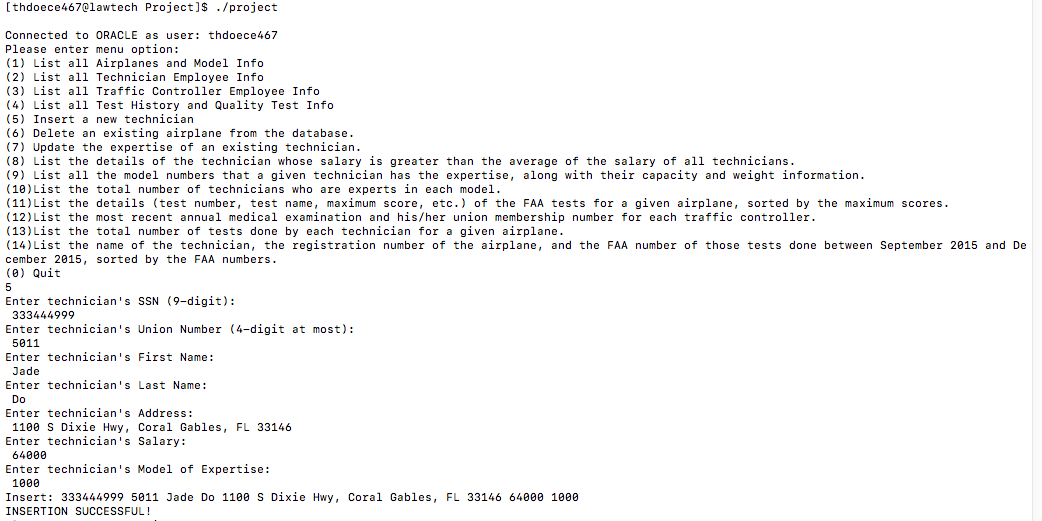
## STAGE 4: PROGRAM EXECUTION AND OUTPUT

### Query 1: Insert a new technician into the database (Menu option 5 in program)

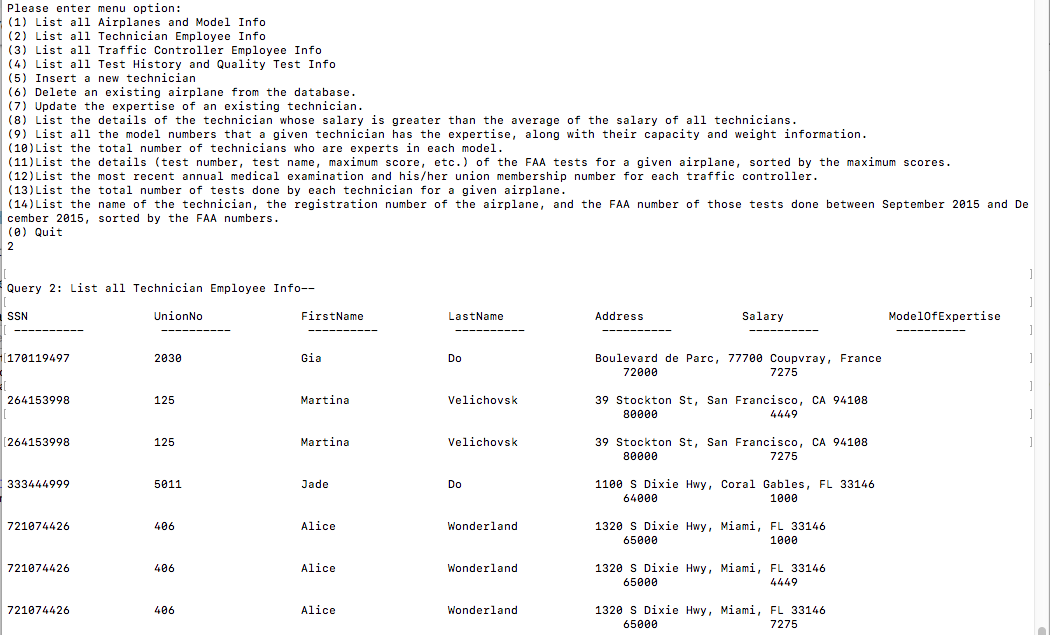
##### Data table before Insertion:



##### Insertion:



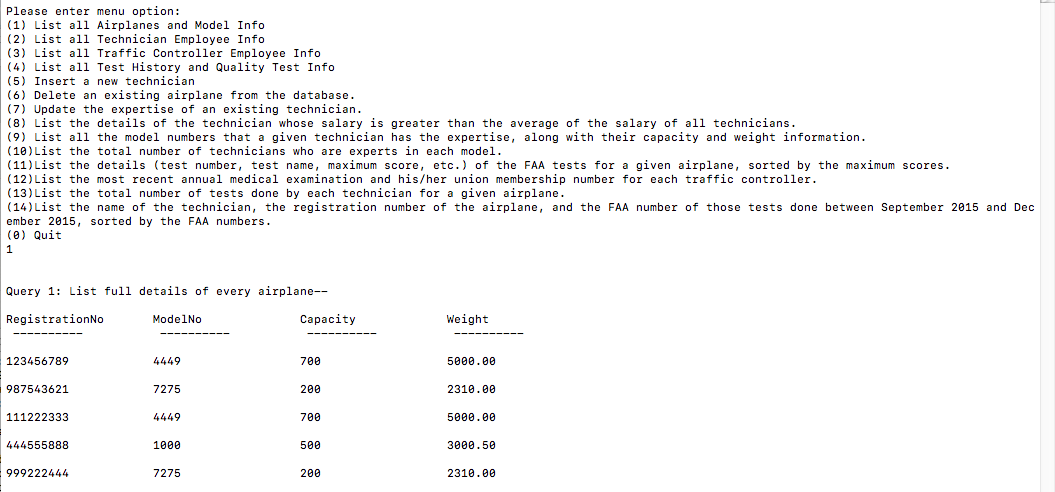
##### Data table after insertion:



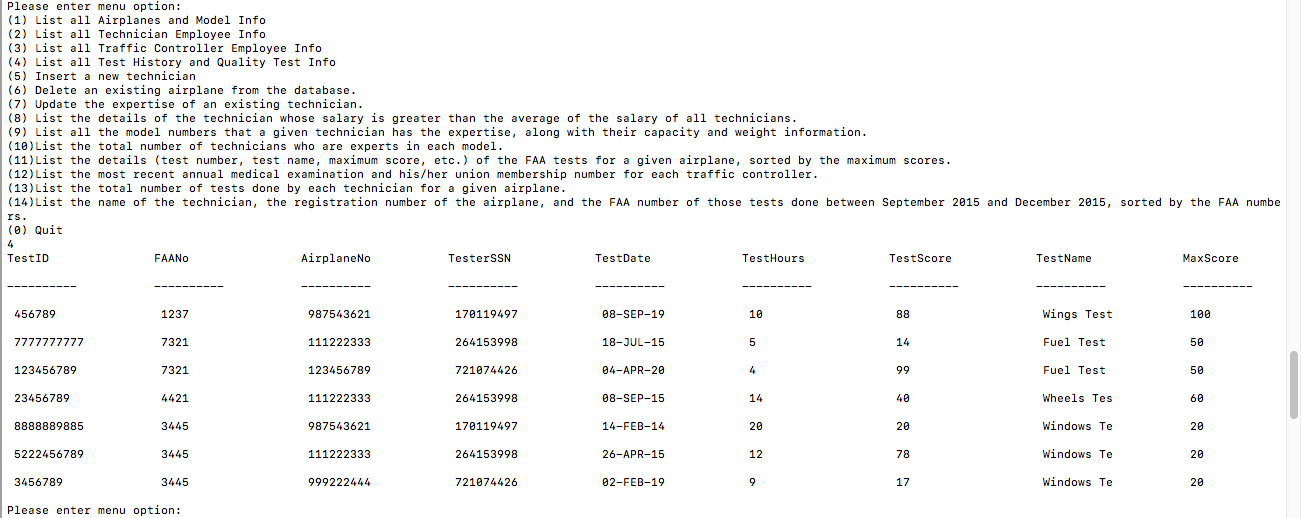
### Query 2: Delete an existing airplane from the database (Menu option (6) in program)

##### Before deletion:

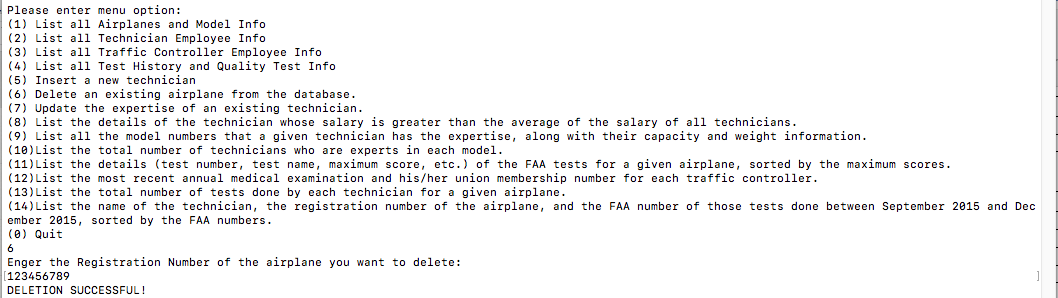
Airplane table:



TestHistory table:

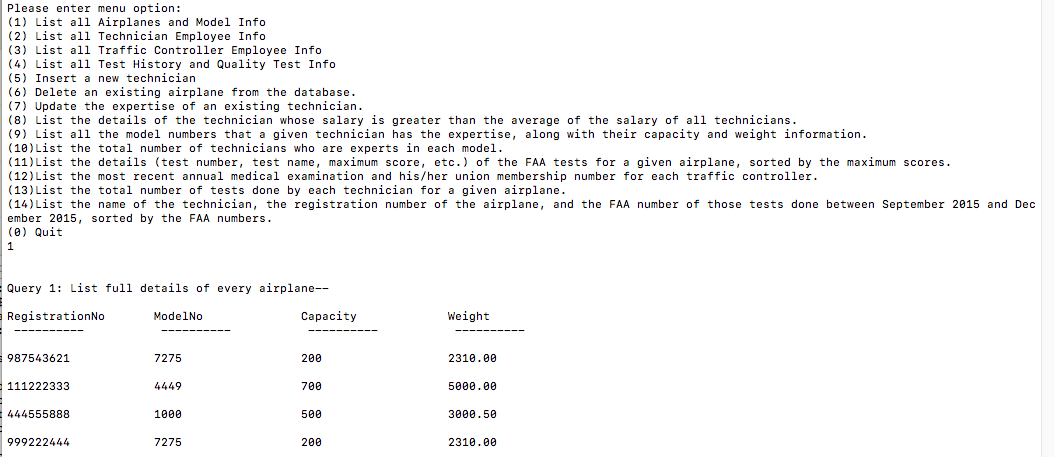


##### Deletion:

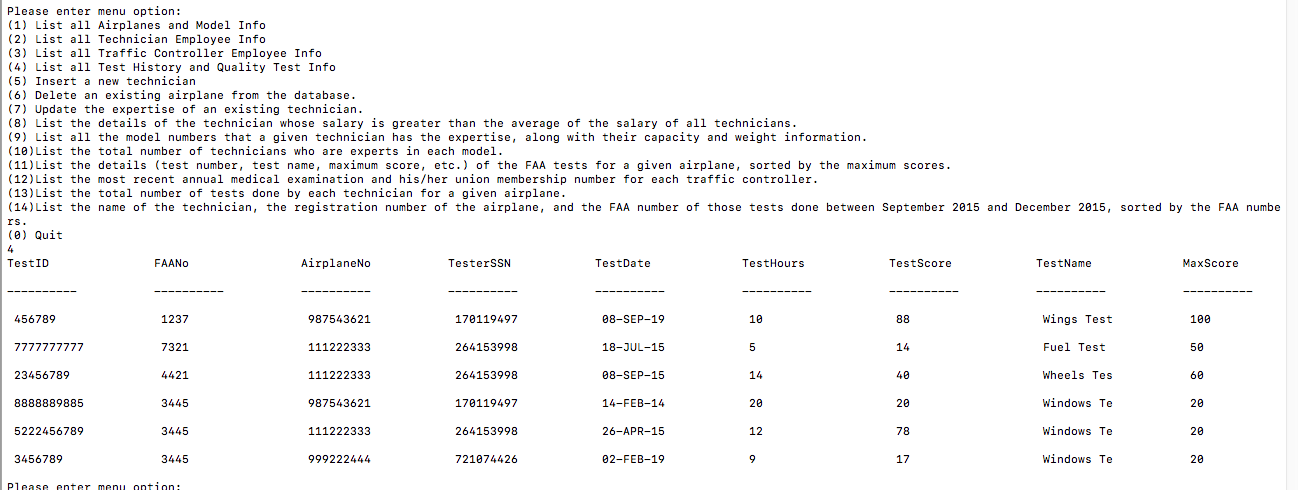


##### After deletion:

We can see that the entry for the airplane is deleted:

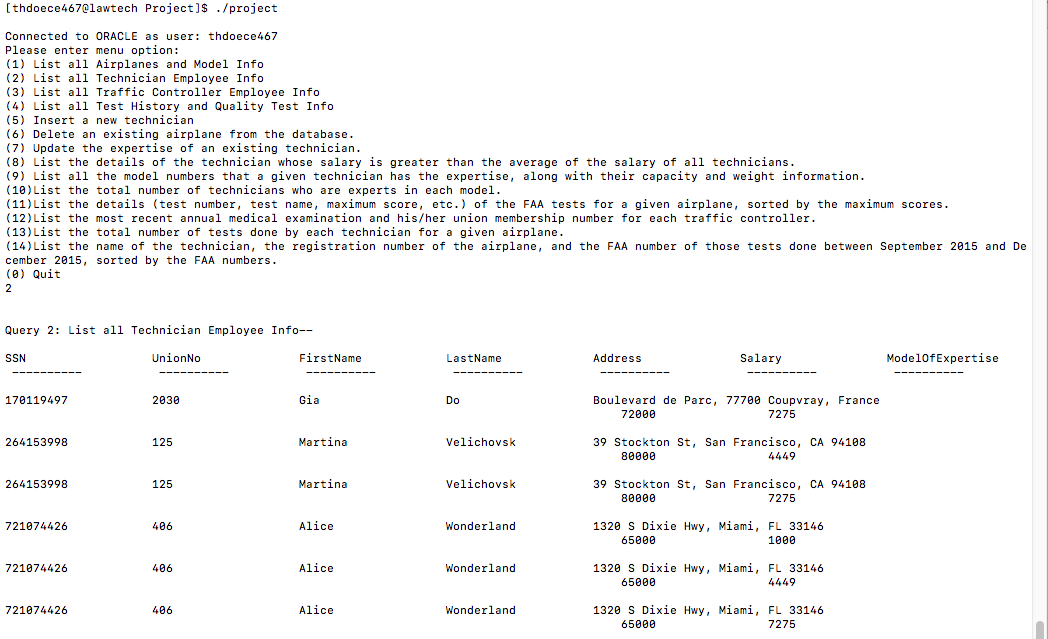


Similarly, the corresponding test history record of the deleted airplane is also deleted:

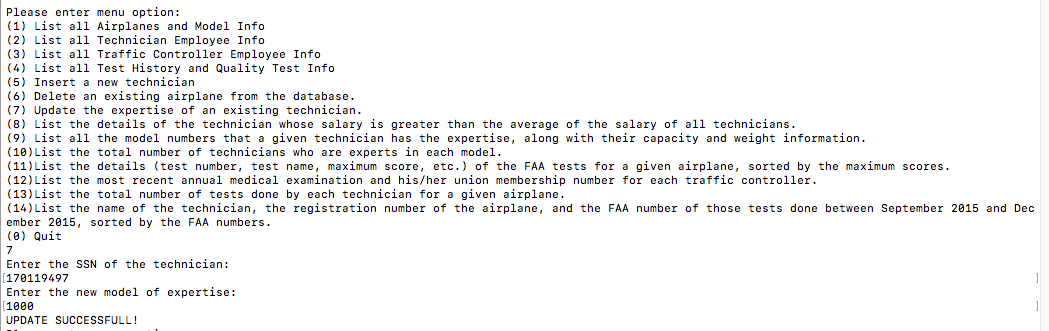


### Query 3: Update the expertise of an existing technician (Menu option 7 in program)

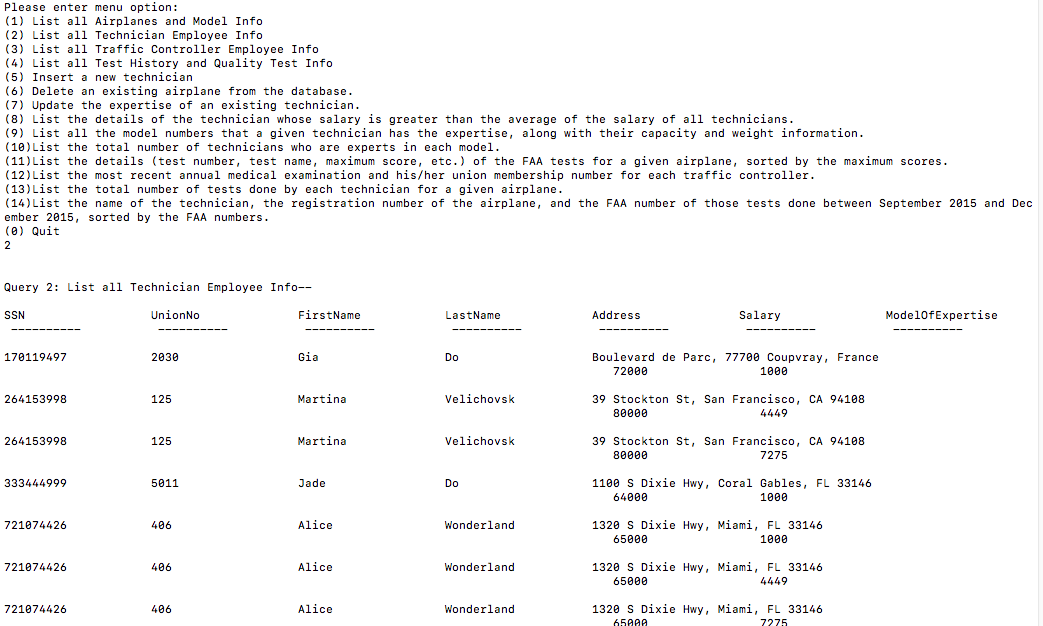
##### Data table before update:



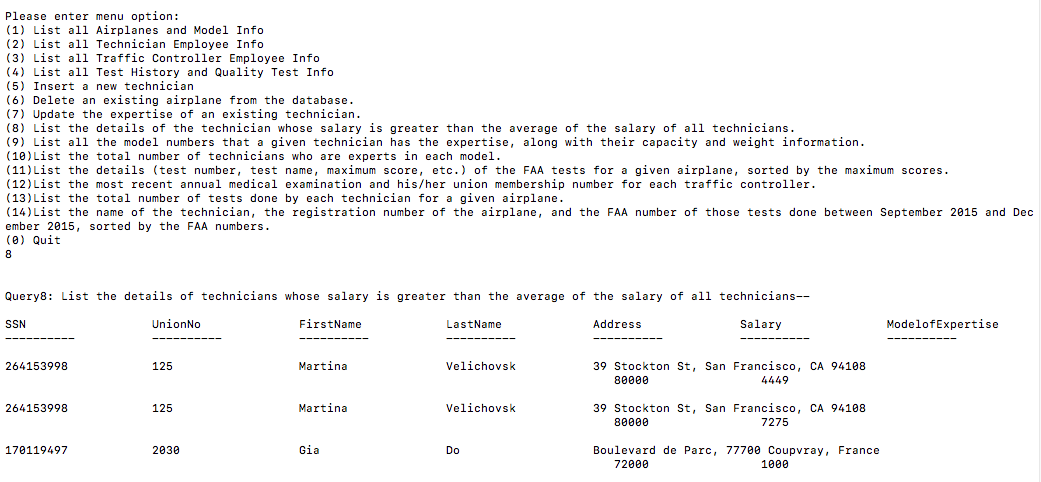
##### Update:



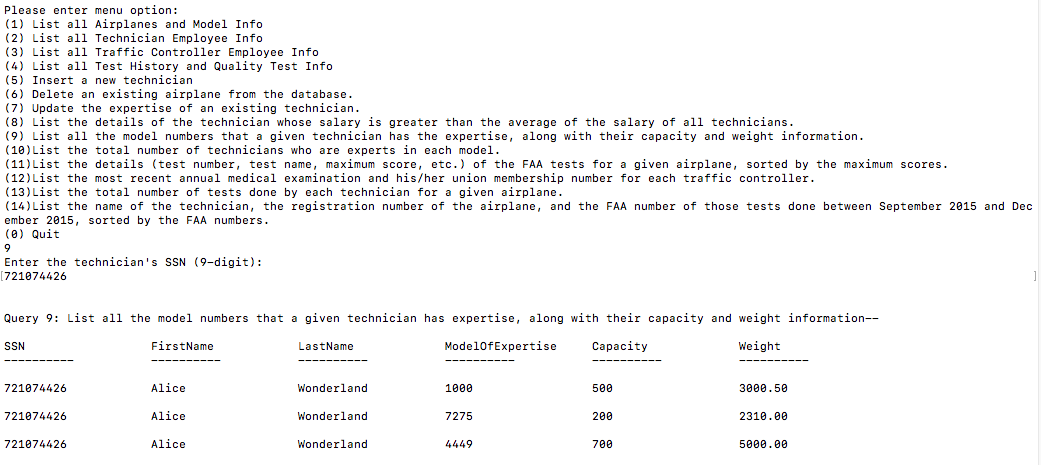
##### Data table after update:



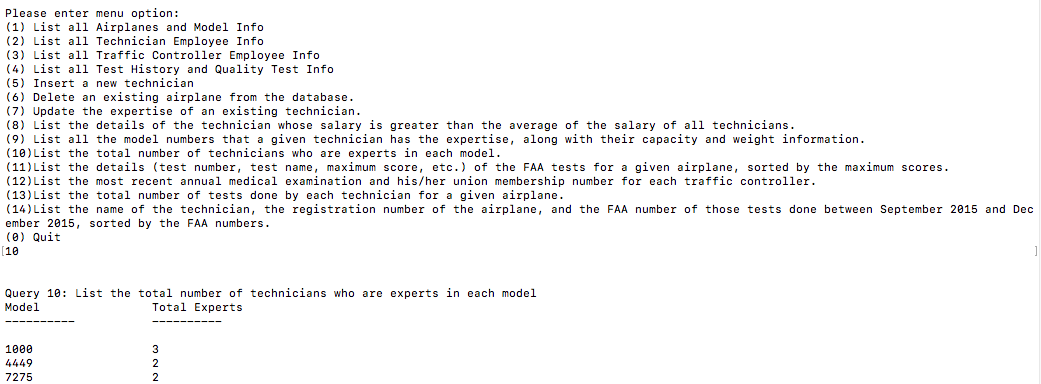
### Query 4: List the details of the technician whose salary is greater than the average of the salary of all technicians (Menu option 8 in program)



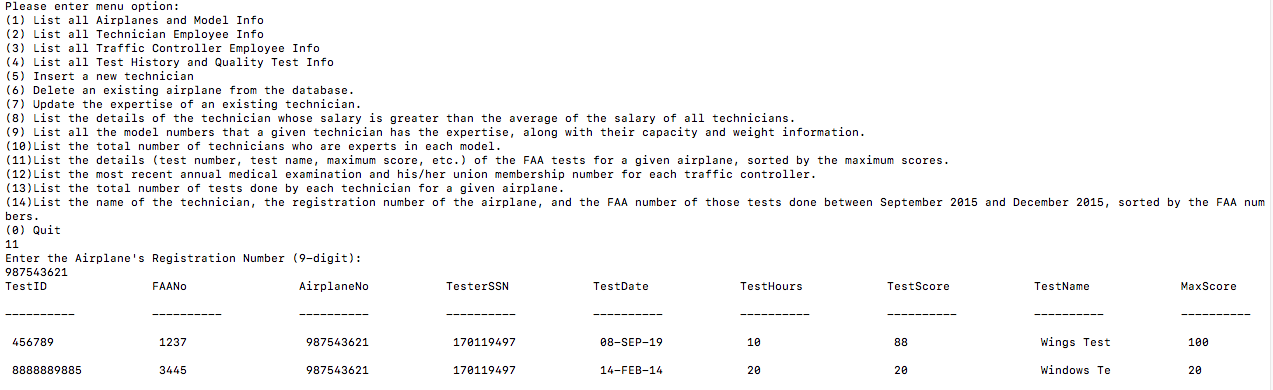
### Query 5: List all the model numbers that a given technician has the expertise, along with their capacity and weight information (Menu option 9 in program)



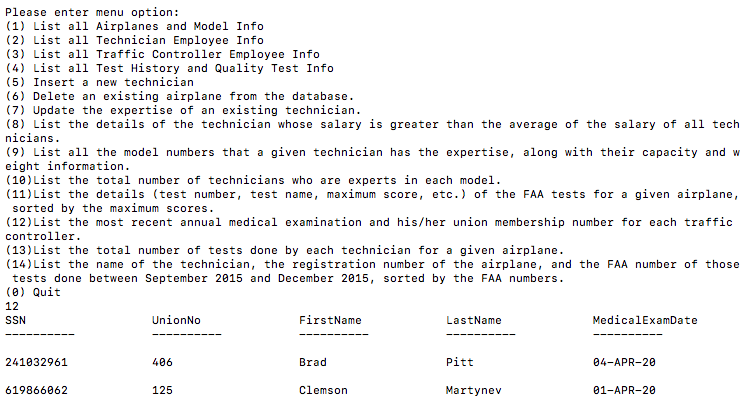
### Query 6: List the total number of technicians who are experts in each model (Menu option 10 in program)



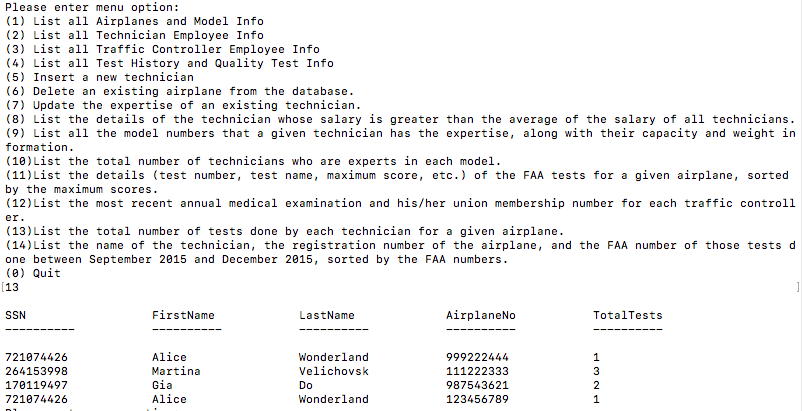
### Query 7: List the details (test number, test name, maximum score, etc.) of the FAA tests for a given airplane, sorted by the maximum scores (Menu option 11 in program)



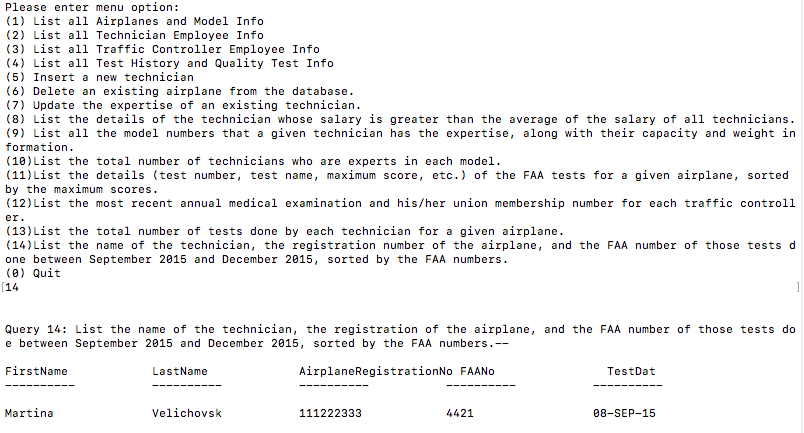
### Query 8: List the most recent annual medical examination and his/her union membership number for each traffic controller (Menu option 12 in program)



### Query 9: List the total number of tests done by each technician for a given airplane (Menu option 13 in program)



### Query 10: List the name of the technician, the registration number of the airplane, and the FAA number of those tests done between September 2015 and December 2015, sorted by the FAA numbers (Menu option 14 in program)



## STAGE 5: CONCLUSION

This has been a very interesting and creative project. I learnt how to embed SQL script in a C program, and therefore see the the complexity and practicality of a program that has both the features of a primitive language and a functional language.

However, Oracle Embedded SQL is not perfect, and below is a listing of some limitations I encountered that I believe the programmers at Oracle can improve on:

1. Structure Record Pointer: Whenever the cursor fetches a record, it places the record into a structure pointer. I realize that the struct record pointer has to have the exact fields that the cursor fetches, no more and no less. This is inflexible, as I ended up having to design a new struct data type for each query that I attempt to execute, because using an existing struct data type with more fields will generate an error, while designing the SELECT statement to fetch more column than necessary in order to fit in an existing struct data type would cause a waste of memory and a slower run-time for the program. This would be a big hindrance for larger database with millions of rows. However, had the program been more complex, designing a new struct for every potential query is not sustainable either, because it would negatively affect program readability and a slower compile time. Therefore, I believe that Oracle can improve on this by allow for SQL cursor to fetch and place its data values into a struct that has more fields than the number of columns it retrieved.
2. Join Operations: I was surprised to find out that Oracle embedded SQL does not support the INNER JOIN, OUTER JOIN, LEFT JOIN and RIGHT JOIN operations. In order to join multiples, I had to use a WHERE clause and match their foreign keys together. This, I believe, should also be improved.
3. Group By: The GROUP BY clause in Oracle Embedded SQL also has more limitations than the GROUP BY clause in a SQL. In Oracle Embedded SQL, I can’t simply use GROUP BY by specifying the column(s) that I want to group by, but I had to either put all columns specified in the SELECT clause in the GROUP BY clause, or use functions on the columns in the SELECT clause to compress the results of those columns into a single value, such as MIN, MAX or SUM. Although this does not affect program performance, it does affect code readability and I believe that the code would look more elegant if this issue is worked on.

Last but not least, thank you Professor Shyu and her TAs for having taught us!