Si Thu Zaw
G5223816T
BTID11701A
T7177793

Relational and NoSQL Databases ICA

Report 1

23 August 2018

Teesside University

Relational and NoSQL Databases (**CIS2017-N**)

Student ID: T7177793

Student Name: Si Thu Zaw

Lecturer Name: Dr. Loo Poh Kok

Si Thu Zaw
G5223816
BTID11701A
T7177793

Table of Contents

List of Figures

1.Introduction

The following document outlines the specifications for the creation of a database system to serve the needs of the aviation school, including the requirements of the system, the functions it is designed to perform, the relationship between entities in the database, the schema of the system, the queries for creating tables, procedures and performing stated functions and other documentation of the system.

1.1 Given Project Situation

The system being developed is intended for The Aviation School to be used to maintain information needed for operation of the school as it transactions from filing physical records to digital records. The school offers various courses for its students, operates multiple aircraft and employs instructors required to teach the courses.

As part of the physical filing system, many of the courses taught by the school vary from basic knowledge courses to preparatory flight training courses for licences and as such, have varying medical and qualification requirements that need to be checked for those enrolling in the courses. The instructors also need to be checked that their licences are still valid before being assigned to teach the students.

Other day to day operations include registering new student pilots, scheduling lesson and tests between instructors and student pilots, maintaining records of students and instructors, maintaining records of aircraft and their servicing information, recording the number of hours the instructors have worked, etc. Three types of users handle operations in the aviation school. They are the manager, staff and instructors.

Student pilots can be classified into 3 categories. They can be independent students, members of the aviation school or students sponsored by their companies. They qualify for different fees for registration into courses per their type.

1.2 Additional Considerations and Assumptions

Some tasks the staff normally do can be replaced by automation in the system such as checking availability of an instructor for booking. Therefore, these functions such as recording the number of hours the instructor has worked will be automated when the lesson or the test is booked. The instructor and aircraft assignments are done when the booking for tests or lessons. The validity of the licences of the instructors will also be automatically checked in the process of booking. These are intended to streamline the operation of the aviation school.

During the enrolment process for the student pilots, the checking of medical requirements and required qualifications to enrol for a specific course will be left up to the staff to confirm in person. The qualification certificates can come from many different sources and accommodating the database to be able to store all types of them would make it very complex. As for medical requirements, the required class of medical certificate for each course will be stored in the database for reference by the staff and the staff will be able to store the confirmation of whether the student has obtained the certificate or not in the student's enrolment record.

Si Thu Zaw
G5223816
BTID11701A
T7177793

2.0 Requirements

2.1 Use Cases

The following diagram (Figure 1) illustrates the use cases identified from the case study of the aviation school provided. Some functions that are stated in the case study (Appendix A,1), are not present in the diagram as they have been integrated as part of the process for booking tests or lessons in this system as explained in the above section.

Figure 1: Use Case Diagram of the Aviation School System

Link to Full Size Use Case Diagram: AviationSchoolUCD

2.2 Entity Relationships

The diagram (Figure 2) shows the main entities that are identified to be required to store the required data and their relationships. The requirements are derived from the specifications from the original case study and the use cases identified.

Subtypes of entities are utilised in this case due to the requirements of the aviation school needing to store information of similar entities with slight differences in attributes. The subtypes of Lesson, Exam and Instructor use total, disjoint constraints as they must be either of the subtypes. However, Sponsored Pilot subtype uses a partial, overlapping constraint as a student pilot can be optionally sponsored or independent.

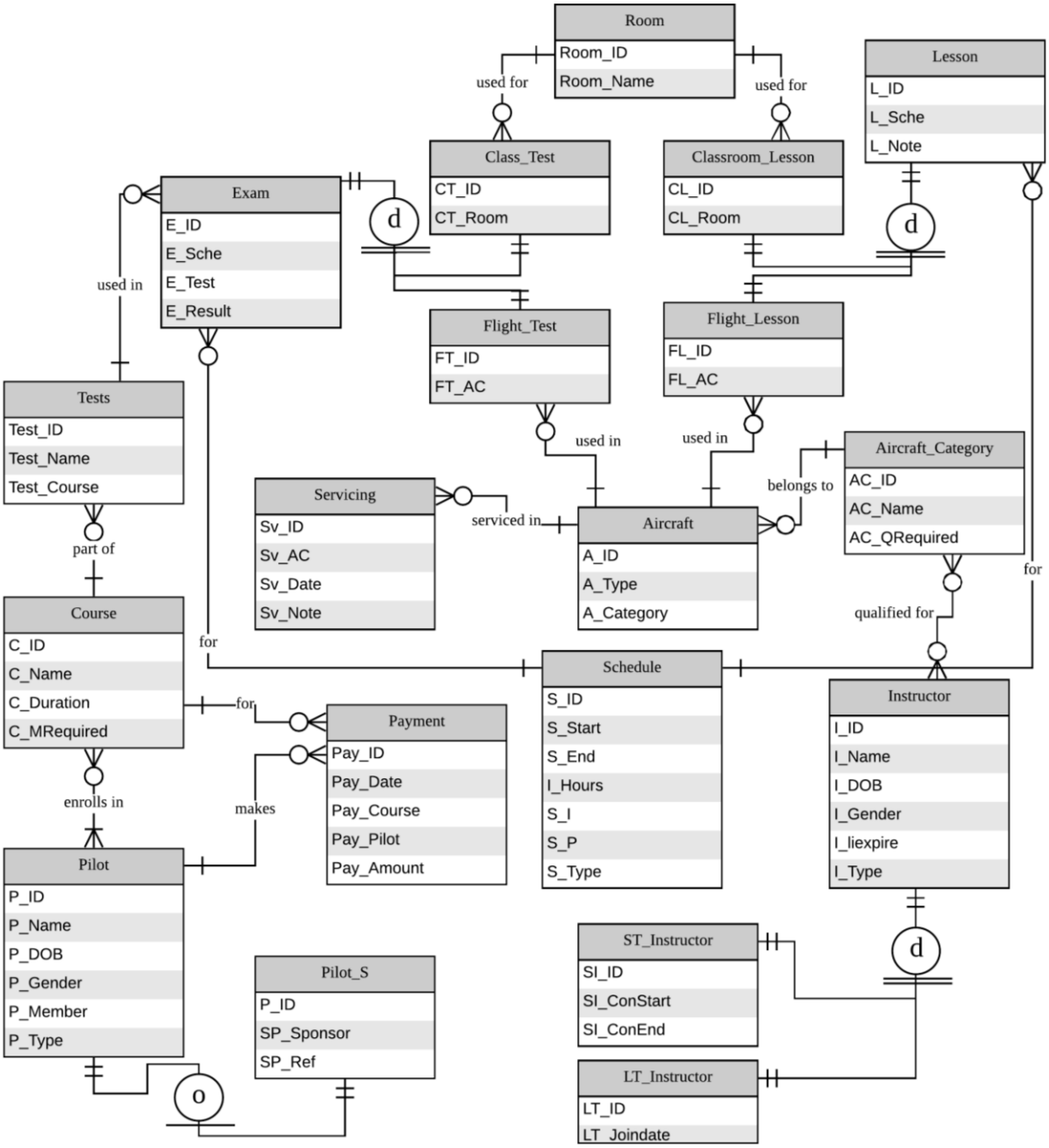

Figure 2: Entity Relationship Diagram of the database for the Aviation School System

Link to Full Size ERD: AviationSchoolERD

Si Thu Zaw
G5223816
BTID11701A
T7177793

3.0 Schema

The implementation of the ERD in the SQL Server Database after normalisation and resolving many-to-many relationships results in the database structure shown below. It consists of 21 tables that are all interconnected (Figure 3). The subtypes detailed in the ERD are formed into corresponding tables to be formed into logical tables later.

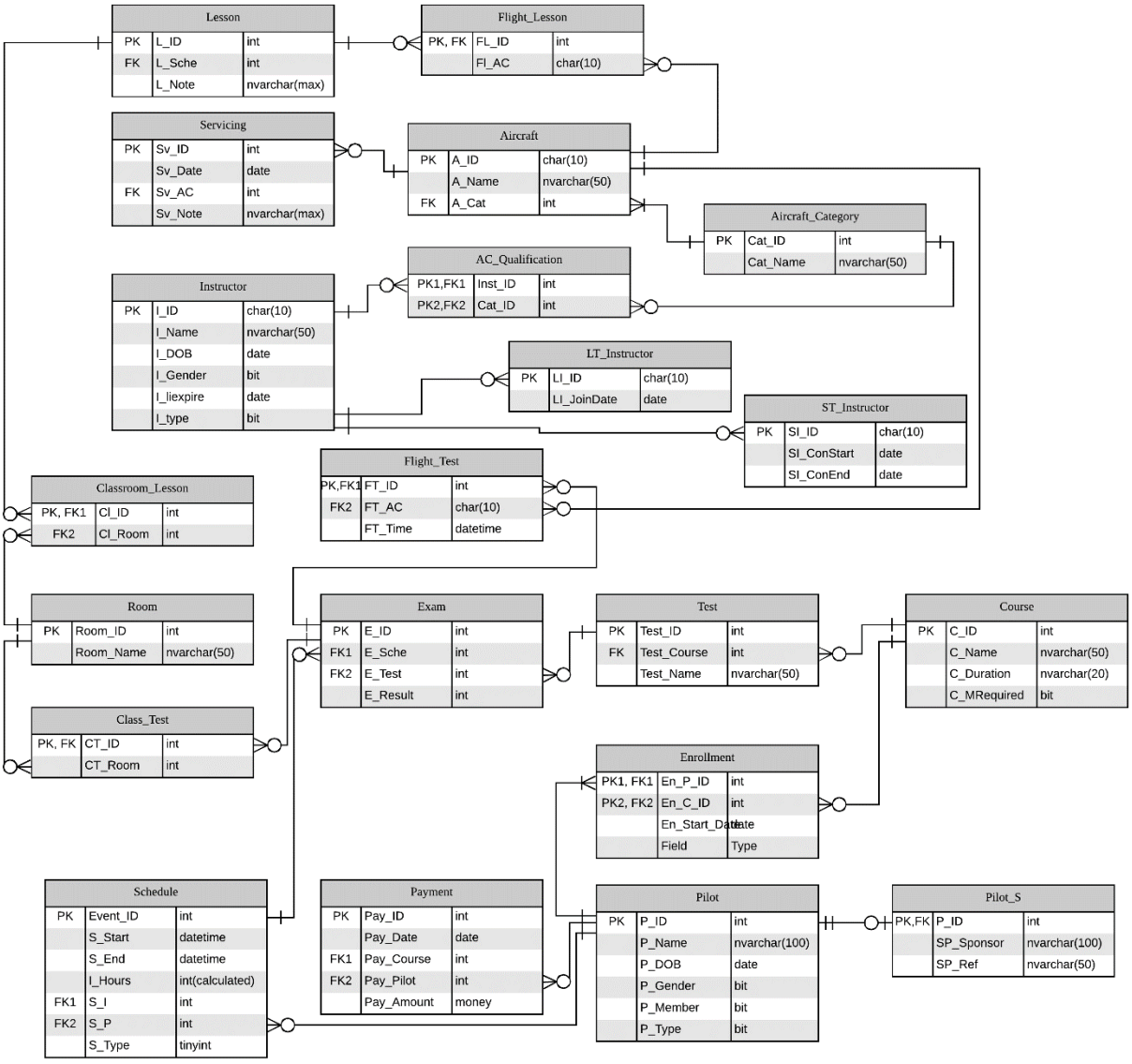

Figure 3: Schema for Aviation School System

Link to Full Size Schema: [AviationSchoolSchema]

4.0 Creating Tables in Database (SQL Data Definition Language Code)

4.1 Create Table Queries

Create Table queries (Figure 4) are used to generate the tables in the database. They specify the table name, column names, their respective datatypes and default values and the constraints for keys. The database creation script contains all queries required to create the database 'Aviation School', 21 tables and their associations and indexes increase query performance in the database. Some tables also have the cascade delete function enabled to allow automatic deletion of child elements on deletion of the parent.

```
CREATE TABLE Servicing(
	[Sv_ID] int identity(1,1) NOT NULL,
	[Sv_Date] datetime NOT NULL DEFAULT GETDATE(),
	[Sv_AC] char(10) NOT NULL,
	[Sv_Note] nvarchar(MAX) NOT NULL,
	CONSTRAINT PK_Sv PRIMARY KEY (Sv_ID),
	CONSTRAINT FK_Sv_AC FOREIGN KEY (Sv_AC) REFERENCES Aircraft(A_ID) ON DELETE
CASCADE
);
GO
```

Figure 4: An Example of a Create Table Query

The diagram below is generated in Microsoft SQL Server Management Studio and shows the structure of the database after the 'Create Database' script has been run (Figure 5).

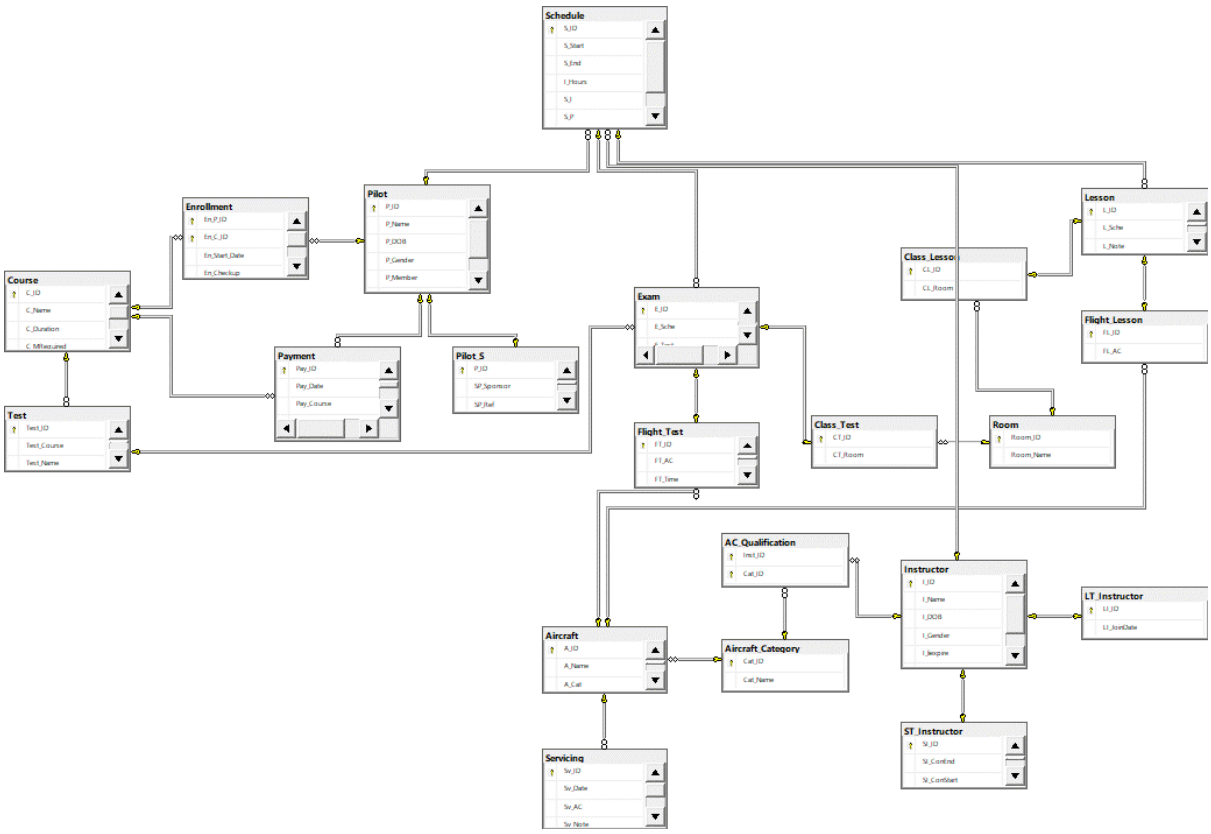

Figure 5: SQL Diagram Output of Database

Link to Database Creation Script: SSMS Scripts\1. Aviation School Create All Tables and Indexes.sql

Link to Full Resolution Diagram: SQL Diagram Output.png

4.2 Create Procedure Queries

Multiple procedures are utilised in this database where multiple queries need to be run with the similar input or a logical set of data (such as details for a flight lesson, for example) to achieve an outcome. They include checks to ensure validity of dates and checking of the entered time and date are available for scheduling procedures. The procedure also is built to roll back changes to the database if it encounters an error and should ensure better data integrity.

```
CREATE PROCEDURE dbo.newclasstest
          @Start               datetime = NULL,
          @End                 datetime = NULL,
          @Instructor          char(10) = NULL,
          @Pilot               int = NULL,
          @Test                int = NULL,
          @Room                int = NULL,
          @ID                  int = NULL
AS
IF EXISTS(SELECT * FROM ((schedule INNER JOIN exam ON S_ID = E_Sche) INNER JOIN class_test ON E_ID = CT_ID)WHERE (S_Start BETWEEN @Start
AND @END) OR (S_End BETWEEN @START AND @END) AND (CT_Room = @Room))
BEGIN
          (SELECT 'The selected room is in use in the given time' AS 'Error');
          RETURN
END
IF EXISTS(SELECT * FROM schedule WHERE (S_Start BETWEEN @Start AND @END) OR (S_End BETWEEN @START AND @END) AND (S_I = @Instructor))
BEGIN
          (SELECT 'Another event exists for selected instructor in the time selected' AS 'Error');
          RETURN
END
IF        ((SELECT I_liexpire FROM Instructor WHERE I_ID=@Instructor) < @Start)
BEGIN
          (SELECT 'Entered instructor"s licence expires before the event starting time' AS 'Error');
          RETURN
END
ELSE
BEGIN
BEGIN TRANSACTION
BEGIN TRY
     SET NOCOUNT ON

     INSERT INTO dbo.Schedule
          (
            S_Start,
            S_End,
            S_I,
            S_P,
            S_Type
          )
           VALUES
          (
            @start,
            @end,
            @Instructor,
            @Pilot,
            '3'
          );
           SET @ID = SCOPE_IDENTITY();
           INSERT INTO dbo.Exam
                     (
                               E_Sche,
                               E_Test
                     )
           VALUES
                     (
                               @ID,
                               @Test
                     );
           SET @ID = SCOPE_IDENTITY();
           INSERT INTO dbo.Class_Test
                     (
                               CT_ID,
                               CT_Room
                     )
           VALUES
                     (
                               @ID,
                               @Room
                     );
           COMMIT TRANSACTION
END TRY
BEGIN CATCH

SELECT ERROR_MESSAGE()'Error Message';
ROLLBACK TRANSACTION

END CATCH
END
GO
```

Figure 6: An Example of a Create Procedure Query Used In Aviation School System

Link to Script for Creating Procedures: SSMS Scripts\2. Aviation School Stored Procedures.sql

4.3 Insert and Execute Queries

The insert queries (Figure 7) are used to populate the database with data initially and to insert further data afterwards in the aviation school system. Execute queries are also used for the purpose in this system as the system relies on procedures to insert consistent data into multiple tables simultaneously due to having to maintain information about multiple subtypes of entities, such as long-term and short-term instructors.

```
INSERT INTO Pilot([P_Name], [P_DOB], [P_Member], [P_Gender])
VALUES
('Dylan Rosa','2000-12-16',1,1);
```

Figure 7: An Example of an Insert Query

```
EXEC dbo.newlongterminstructor
	@ID = 'LT005',
	@Name = 'Bob',
	@DOB = '1982-04-11',
	@Gender = '1',
	@Liexpire = '2020-12-02',
	@LiJoinDate = '2017-01-22'
```

Figure 8: An Example of an Execute Query

Link to Database Population Script: SSMS Scripts\3. Aviation School Populate And Select All.sql

4.4 Views for Subtype Entities

Due to the design of the database to accommodate storing data for subtypes of entities, such as long-term and short-term instructors, the data is spread out over multiple physical tables (Figure 10). Views are utilised to consolidate these data into a virtual table (Figure 9, 11). This increases readability and simplifies queries that would use the output of the views, requiring a simpler statement to get the result rather than having to include the full select query.

```
CREATE VIEW [Sponsored Pilot] AS
SELECT
P_ID AS 'ID',
P_Name AS 'Name',
P_DOB AS 'Date of Birth',
SP_Sponsor AS 'Sponsor Name',
SP_Ref AS 'Sponsor Reference No.',
CASE P_Gender WHEN 1 THEN 'Male' WHEN 0 THEN 'Female' END AS Gender,
CASE P_Member WHEN 1 THEN 'Yes' WHEN 0 THEN 'No' END AS 'Member?'
FROM (Pilot INNER JOIN Pilot_S ON Pilot.P_ID = Pilot_S.P_ID);
GO
```

Figure 9: An Example of a Create View Query

	P_ID	P_Name	P_DOB	P_Gender	P_Member	P_Type
1	25	Halla Frye	1980-11-30	0	0	1
2	26	Rigel Rivas	1992-01-20	1	0	1

	P_ID	SP_Sponsor	SP_Ref
1	25	RyanAir	30582617
2	26	SilkAir	23379265

Figure 10: Separate Tables for Storing Sponsored Pilot Data

	Name	Date of Birth	Sponsor Name	Sponsor Reference No.	Gender	Member?
1	Halla Frye	1980-11-30	RyanAir	30582617	Female	No
2	Rigel Rivas	1992-01-20	SilkAir	23379265	Male	No

Figure 11: Logical Table for Sponsored Pilot Created by View

5.0 Queries for Functions (SQL Data Manipulation Language Code)

The queries below aim to fulfil the functions identified in the case study and developed as shown in the use case diagram. Each query or procedure is intended to perform one or part of a use case.

5.1 Maintain Aircraft Record

This use case is an aggregate of create, read, update and delete functions to the list of aircraft stored in the database.

5.1.1 Add New Aircraft

This query adds a new aircraft to the list of aircraft owned by the aviation school.

```
INSERT INTO Aircraft ([A_ID], [A_Name], [A_Cat])
VALUES
('CA1962', 'Piper Tripacer', 2);
```

7	CA1962	Piper Tripacer	2

Figure 12: Data Input by Add New Aircraft Query

5.1.2 Read Aircraft

This query is used to read the list of aircraft with more complete information about the aircraft.

```
SELECT
A_ID AS 'Registration No.',
A_Name AS 'Name',
Cat_Name AS 'Aircraft Category'
FROM (Aircraft inner join Aircraft_Category on A_Cat = Cat_ID);
```

	Registration No.	Name	Aircraft Category
1	AW2134	Extra 230	Aerobatic Sport-Single-Engine
2	AW2350	Extra 230	Aerobatic Sport-Single-Engine
3	BA0245	Cessna 414	Medium Two-Engine
4	CA1027	Cessna 210	Light 4 Setaer-Single-Engine
5	CA1246	Beechcraft Bonanza Model 35D	Light 2 Seater-Single-Engine
6	CA1838	Cessna 172	Light 4 Setaer-Single-Engine
7	CA1962	Piper Tripacer	Light 2 Seater-Single-Engine
8	CA2931	Piper J-3 Cub	Light 2 Seater-Single-Engine
9	CA3413	Piper Tripacer	Light 2 Seater-Single-Engine
10	CA5013	Piper Super Cub	Light 2 Seater-Single-Engine
11	CA5023	Mooney M20C	Light 2 Seater-Single-Engine
12	CA5184	Cessna T182 Turbo Skylane	Light 4 Setaer-Single-Engine
13	CA7032	Cessna 172	Light 4 Setaer-Single-Engine
14	CA7214	Cessna 182	Light 4 Setaer-Single-Engine
15	CA9310	Cirrus SR22 G2	Light 4 Setaer-Single-Engine
16	DE9483	Yak-52	Aerobatic Sport-Single-Engine
17	IN6429	Beech Duke B60	Medium Two-Engine

Figure 13: List of Aircraft Called by Read Aircraft Query

5.1.3 Update Aircraft

This query is used to change an attribute of an aircraft where in this example is the name of aircraft.

```
UPDATE [Aircraft]
SET
[A_Name] = 'Cessna 189'
WHERE
[A_ID] = 'CA1962';
```

7	CA1962	Piper Tripacer	2

Figure 14: Previously Inserted Aircraft Record

7	CA1962	Cessna 189	2

Figure 15: Aircraft Record Changed by Update Aircraft Query

5.1.4 Delete Aircraft

This query is used to delete an aircraft from the list of aircraft. This also delete the associated servicing records of the aircraft due to cascading deletion on the Servicing table.

```
DELETE FROM [Aircraft]
WHERE [A_ID] = 'CA1962';
```

6	CA1838	Cessna 172	3
7	CA1962	Cessna 189	2
8	CA2931	Piper J-3 Cub	2

Figure 16: Aircraft List before Query

6	CA1838	Cessna 172	3
7	CA2931	Piper J-3 Cub	2

Figure 17: Aircraft List After Delete Aircraft Query

5.2 Maintain Student Record

This use case is an aggregate of create, read, update and delete functions to the list of student records stored in the database.

5.2.1.1 Read Independent Student

This query retrieves the list of independent pilots and their information.

```
SELECT
P_Name AS 'Name',
P_DOB AS 'Date of Birth',
CASE P_Gender WHEN 1 THEN 'Male' WHEN 0 THEN 'Female' END As Gender,
CASE P_Member WHEN 1 THEN 'Yes' WHEN 0 THEN 'No' END As 'Member?'
FROM [Pilot]1;
```

	Name	Date of Birth	Gender	Member?
1	Jana Trevino	1973-09-08	Female	Yes
2	Stacey Strickland	1996-03-15	Female	No
3	Stone Ryan	1982-11-21	Male	No
4	Elliott Taylor	1981-06-06	Male	Yes
5	Kelsie Fry	1997-10-11	Female	No
6	Nasim Macdonald	1984-07-23	Male	No
7	Katell Hobbs	1989-09-17	Male	No
8	Yael Richards	1969-10-13	Male	No
9	Susan Stokes	1993-07-10	Female	Yes
10	Dylan Rosa	2000-12-16	Male	Yes

Figure 18: List of Independent Students Called by Read Independent Student Query

5.2.1.2 Read Sponsored Student

This query retrieves the list of sponsored pilots and their information by joining the Pilot and Sponsored Pilot tables.

```
SELECT
P_Name AS 'Name',
P_DOB AS 'Date of Birth',
SP_Sponsor AS 'Sponsor Name',
SP_Ref AS 'Sponsor Reference No.',
CASE P_Gender WHEN 1 THEN 'Male' WHEN 0 THEN 'Female' END AS Gender,
CASE P_Member WHEN 1 THEN 'Yes' WHEN 0 THEN 'No' END AS 'Member?'
FROM (Pilot INNER JOIN Pilot_S ON Pilot.P_ID = Pilot_S.P_ID);
```

Alternatively, the view for Sponsored Pilot can also be selected to get the same results.

```
SELECT * FROM [Sponsored Pilot];
```

	Name	Date of Birth	Sponsor Name	Sponsor Reference No.	Gender	Member?
1	Halla Frye	1980-11-30	RyanAir	30582617	Female	No
2	Rigel Rivas	1992-01-20	SilkAir	23379265	Male	No

Figure 19: List of Sponsored Student Pilots called by Read Sponsored Student Query

5.2.2 Update Student

This query is used to change a selected attribute of a selected student. The member status is updated in this example.

```
UPDATE [Pilot]
SET
[P_Member] = 1
WHERE
[P_Name] = 'Halla Frye';
```

	Name	Date of Birth	Sponsor Name	Sponsor Reference No.	Gender	Member?
1	Halla Frye	1980-11-30	RyanAir	30582617	Female	Yes
2	Rigel Rivas	1992-01-20	SilkAir	23379265	Male	No

Figure 20: Student's Record Changed by Update Student Query

5.2.3 Delete Student

This query is used to remove a selected student's record entirely from the database. As cascade delete is enabled on the relationships linking back to pilot records, the simple command achieves the deletion of all records in tables associated with the pilot record along with the record itself (Figure 21, 22).

```
DELETE FROM [Pilot] WHERE P_ID = 1;
```

	P_ID	P_Name	P_DOB	P_Gender	P_Member	P_Type
1	1	Jana Trevino	1973-09-08	0	1	0
2	2	Stacey Strickland	1996-03-15	0	0	0
3	3	Stone Ryan	1982-11-21	1	0	0

	En_P_ID	En_C_ID	En_Start_Date	En_Checkup
1	1	1	2017-01-22 00:00:00.000	1
2	2	2	2017-02-05 00:00:00.000	1
3	3	3	2017-02-09 00:00:00.000	1

Figure 21: Pilot and Enrolment Tables Before Deletion

	P_ID	P_Name	P_DOB	P_Gender	P_Member	P_Type
1	2	Stacey Strickland	1996-03-15	0	0	0
2	3	Stone Ryan	1982-11-21	1	0	0

	En_P_ID	En_C_ID	En_Start_Date	En_Checkup
1	2	2	2017-02-05 00:00:00.000	1
2	3	3	2017-02-09 00:00:00.000	1

Figure 22: Pilot and Enrolment Tables After Deletion

5.3 Add Long Term Instructor

This command is used to call the procedure 'newlongterminstructor' and provides the parameters to use in the procedure. This adds entries into the 'Instructor' and 'LT_Instructor' tables to form a complete record for a long-term instructor. This procedure checks whether the date of birth entered is in the past and that the licence expiry date entered is in the future.

```
EXEC dbo.newlongterminstructor
        @ID = 'LT005',
        @Name = 'Bob',
        @DOB = '1982-04-11',
        @Gender = '1',
        @Liexpire = '2020-12-02',
        @LiJoinDate = '2017-01-22'
```

	Name	Date of Birth	Gender	Licence Expiration Date	Join Date
1	Bob	1982-04-11	Male	2020-12-02	2016-06-25

Figure 23: Record Inserted by Add Long Term Instructor Procedure

5.4 Add Short Term Instructor

This command is used to call the procedure 'newshortterminstructor' and provides the parameters to use in the procedure. This adds entries into the 'Instructor' and 'ST_Instructor' tables to form a complete record for a short-term instructor. This procedure checks whether the date of birth entered is in the past and that the licence expiry date entered is in the future.

```
EXEC dbo.newshortterminstructor
       @ID = 'ST002',
       @Name = 'Dave',
       @DOB = '1982-04-11',
       @Gender = '1',
       @Liexpire = '2020-12-02',
       @SI_ConStart = '2016-06-25',
       @SI_ConEnd = '2018-06-25'
```

	Name	Date of Birth	Gender	Licence Expiration Date	Contract Starting Date	Contract Ending Date
1	Dave	1982-04-11	Male	2020-12-02	2016-06-25	2017-11-05

Figure 24: Record Inserted by Add Short Term Instructor Procedure

5.5 Update Student Record

This query is used to change a selected attribute of a selected student. The member status is updated in this example (Figure 25).

```
UPDATE [Pilot]
SET
[P_Member] = 1
WHERE
[P_Name] = 'Halla Frye';
```

	Name	Date of Birth	Sponsor Name	Sponsor Reference No.	Gender	Member?
1	Halla Frye	1980-11-30	RyanAir	30582617	Female	Yes
2	Rigel Rivas	1992-01-20	SilkAir	23379265	Male	No

Figure 25: Student's Record Changed by Update Student Query

5.6 Update Training Record

This query is used to make changes to an existing training record. The time of the end of a flight lesson is changed in this example (Figure 26, 27).

```
UPDATE [Schedule]
SET
[S_End] = '2018-03-29 16:00:00.000'
WHERE
S_ID = 7;
SELECT * FROM [Flight Lesson];
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Aircraft
1	7	2018-03-29 12:00:00.000	2018-03-29 15:00:00.000	Duncan	Elliott Taylor	Takeoff and landing	CA5184

Figure 26: Flight Lesson Entry Before Update

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Aircraft
1	7	2018-03-29 12:00:00.000	2018-03-29 16:00:00.000	Duncan	Elliott Taylor	Takeoff and landing	CA5184

Figure 27: Flight Lesson Entry After Update

5.7 Record Payments

This query is used to input a payment record containing relevant information into the Payment table to maintain records of payment.

```
INSERT INTO Payment ([Pay_Pilot], [Pay_Course], [Pay_Date], [Pay_Amount])
VALUES
(8, 8, '2017-09-09', 15000.00);
```

	Pay_ID	Pay_Date	Pay_Course	Pay_Pilot	Pay_Amount
1	8	2017-09-09 00:00:00.000	8	8	15000.00

Figure 28: Payment Record input by Record Payments Query

5.8 Register Independent Pilot

This query is used to add a new independent pilot to the list of students.

```
INSERT INTO Pilot([P_Name], [P_DOB], [P_Member], [P_Gender])
VALUES
('Dylan Rosa','2000-12-16',1,1);
```

10	Dylan Rosa	2000-12-16	Male	Yes

Figure 29: Data Entered by Register Independent Query

5.9 Register Sponsored Pilot

This execute command is used to register a new sponsored pilot as data for them are stored in two different tables. The procedure adds records to the table Pilot and Pilot_S simultaneously to maintain data for Sponsored Pilots. This procedure checks that date of birth entered is in the past.

```
EXEC dbo.newsponsoredpilot
        @Name                ='Halla Frye',
        @DOB                 ='1980-11-30',
        @Gender              ='0',
        @Member              ='0',
        @Sponsor             ='RyanAir',
        @SponsorRef          ='30582617'
```

	Name	Date of Birth	Sponsor Name	Sponsor Reference No.	Gender	Member?
1	Halla Frye	1980-11-30	RyanAir	30582617	Female	Yes

Figure 30: Record Inserted by Sponsored Pilot Procedure

5.10 Book Course

This query registers a student to a course and stores the record in the enrolment table.

```
INSERT INTO Enrollment ([En_P_ID], [En_C_ID], [En_Start_date])
VALUES
(10, 1, '2017-10-07');
```

	En_P_ID	En_C_ID	En_Start_Date	En_Checkup
1	10	1	2017-10-07 00:00:00.000	1

Figure 31: Record Inserted by Book Course Query

5.11 Confirm Medical Status

This query changes the marker stating whether the student's medical status is still pending or not.

```
UPDATE [Enrolment]
SET
[En_Checkup] = 0
WHERE
[En_P_ID] = 2;
```

	En_P_ID	En_C_ID	En_Start_Date	En_Checkup
1	2	2	2017-02-05 00:00:00.000	1

Figure 32: Course Enrolment Entry Before Update by Confirm Medical Status Query

	En_P_ID	En_C_ID	En_Start_Date	En_Checkup
1	2	2	2017-02-05 00:00:00.000	0

Figure 33: Course Enrolment Entry After Update by Confirm Medical Status Query

5.12 Print Lesson Schedule

This query is used to call all scheduled lessons within a given time frame for a selected instructor within a given time frame.

```
SELECT
S_Start AS 'Start Time',
S_End AS 'End Time',
I_Name AS 'Instructor',
P_Name AS 'Pilot',
CASE S_Type WHEN 1 THEN 'Flight Lesson' WHEN 2 THEN 'Class Lesson' WHEN 3 THEN 'Flight
Test' WHEN 4 THEN 'Class Test' END AS 'Type'
FROM ((Schedule INNER JOIN Instructor ON S_I = I_ID)
INNER JOIN Pilot ON P_ID = S_P)
WHERE S_Start BETWEEN '2018-03-01' AND '2018-09-01';
```

	Start Time	End Time	Instructor	Pilot	Type
1	2018-03-16 18:00:00.000	2018-03-16 19:00:00.000	Zac	Kelsie Fry	Class Lesson
2	2018-04-05 09:00:00.000	2018-04-05 12:00:00.000	Amy	Rigel Rivas	Class Lesson
3	2018-05-23 11:00:00.000	2018-05-23 13:00:00.000	Silas	Nasim Macdonald	Class Lesson
4	2018-03-29 12:00:00.000	2018-03-29 16:00:00.000	Duncan	Elliott Taylor	Flight Lesson
5	2018-03-19 10:00:00.000	2018-03-19 13:00:00.000	Zac	Kelsie Fry	Flight Lesson

Figure 34: Output of Print Lesson Schedule Query

5.13 Print Pilot Record

This procedure retrieves all record associated with the pilot registered at the aviation school. A procedure is used in this case to simplify retrieval as only the Pilot's ID only need to be entered once.

```
EXEC dbo.retrievepilotrecord
       @PilotID              = 6
```

	Name	Date of Birth	Gender	Pilot Type	Member?
1	Nasim Macdonald	1984-07-23	Male	Independent	No

	Course	Course Start Date	Medical Status
1	IMC Instrument Flying Rating	2017-05-29 00:00:00.000	Pending

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Room
1	5	2018-05-23 11:00:00.000	2018-05-23 13:00:00.000	Silas	Nasim Macdonald	Night flying theory	Durian

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Aircraft
1	10	2018-05-25 18:00:00.000	2018-05-25 20:00:00.000	Silas	Nasim Macdonald	Night flying practice	IN6429

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Room	Result
1	15	2018-08-11 20:00:00.000	2018-08-11 22:00:00.000	Silas	Nasim Macdonald	IMC Rating Theoratical Test	Citrus	83

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Aircraft	Test Starting Time	Result
1	20	2018-08-15 18:00:00.000	2018-08-15 22:30:00.000	Silas	Nasim Macdonald	IMC Rating Practical Test	IN6429	2018-08-15 19:00:00.000	85

Figure 35: Output Retrieved by Print Pilot Record Query

5.14 Check Instructor Schedule

This query is used to call all scheduled lessons and tests associated with an instructor.

```
SELECT
CASE S_Type WHEN 1 THEN 'Class Lesson' WHEN 2 THEN 'Flight Lesson' WHEN 3 THEN 'Class
Test' WHEN 4 THEN 'Flight Test' END AS 'Type',
P_Name AS 'Pilot Assigned',
S_Start AS 'Starting Time',
S_End AS 'Ending Time'
FROM (Schedule INNER JOIN Pilot ON S_P = P_ID) WHERE S_I = 'LT007';
```

	Type	Pilot Assigned	Starting Time	Ending Time
1	Flight Lesson	Nasim Macdonald	2018-05-23 11:00:00.000	2018-05-23 13:00:00.000
2	Class Lesson	Nasim Macdonald	2018-05-25 18:00:00.000	2018-05-25 20:00:00.000
3	Class Test	Nasim Macdonald	2018-08-11 20:00:00.000	2018-08-11 22:00:00.000
4	Flight Test	Nasim Macdonald	2018-08-15 18:00:00.000	2018-08-15 22:30:00.000

Figure 36: Instructor Schedule Called by Check Instructor Schedule Query

5.15 Record Test Result

This query is used to input the results of a class test or a flight test to an existing event. Flight tests or class test events are created with no result entered by default. A flight test entry is changed for this example (Figure 37, 38).

```
UPDATE [Exam]
SET
[E_Result] = 86
WHERE
[E_ID] = 11;
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Aircraft	Test Starting Time	Result
1	21	2018-08-20 18:00:00.000	2018-08-20 22:30:00.000	Silas	Nasim Macdonald	IMC Rating Practical Test	IN6429	2018-08-20 19:00:00.000	NULL

Figure 37: Flight Test Entry Before Update by Record Test Result Query

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Aircraft	Test Starting Time	Result
1	21	2018-08-20 18:00:00.000	2018-08-20 22:30:00.000	Silas	Nasim Macdonald	IMC Rating Practical Test	IN6429	2018-08-20 19:00:00.000	86

Figure 38: Flight Test Entry After Update by Record Test Result Query

5.16 Book Flight Lesson

This execute command is used to insert a record for a flight lesson. The procedure inserts records into the tables Schedule, Lesson and Flight Lesson tables and creates a full flight lesson record. This procedure checks if the instructor is available in the time frame and whether the instructor's teaching licence is still valid before the start of the schedule.

```
EXEC dbo.newflightlesson
        @Start              ='2018-03-19 10:00:00.000',
        @End                ='2018-03-19 13:00:00.000',
        @Instructor         ='LT003',
        @Pilot              ='5',
        @Note               ='Flying practice',
        @Aircraft           ='YS3680'
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Aircraft
1	8	2018-03-19 10:00:00.000	2018-03-19 13:00:00.000	Zac	Kelsie Fry	Flying practice	YS3680

Figure 39: Flight Lesson Record Inserted by Book Flight Lesson Query

5.17 Book Class Lesson

This execute command is used to insert a record for a class lesson. The procedure inserts records into the tables Schedule, Lesson and Class Lesson tables and creates a full class lesson record. This procedure checks if the room or the instructor is available in the time frame and whether the instructor's teaching licence is still valid before the start of the schedule.

```
EXEC dbo.newclasslesson
        @Start                  ='2018-03-16 18:00:00.000',
        @End                    ='2018-03-16 19:00:00.000',
        @Instructor             ='LT003',
        @Pilot                  ='5',
        @Note                   ='Radio Telephony theory',
        @Room                   ='1'
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Lesson Notes	Room
1	3	2018-03-16 18:00:00.000	2018-03-16 19:00:00.000	Zac	Kelsie Fry	Radio Telephony theory	Apple

Figure 40: Class Lesson Record Inserted by Book Class Lesson Query

5.18 Book Class Test

This execute command is used to insert a record for a class test. The procedure inserts records into the tables Schedule, Exam and Class Test tables and creates a full class test record. This procedure checks if the room or the instructor is available in the time frame and whether the instructor's teaching licence is still valid before the start of the schedule.

```
EXEC dbo.newclasstest
        @Start                  ='2018-09-12 14:00:00.000',
        @End                    ='2018-09-12 15:30:00.000',
        @Instructor             ='LT003',
        @Pilot                  ='5',
        @Test                   ='13',
        @Room                   ='4'
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Room	Result
1	13	2018-09-12 14:00:00.000	2018-09-12 15:30:00.000	Zac	Kelsie Fry	EU PPL Theoratical Test	Durian	82

Figure 41: Class Test Record Inserted by Book Class Test Query

5.19 Book Flight Test

This execute command is used to insert a record for a flight test. The procedure inserts records into the tables Schedule, Exam and Flight Test tables and creates a full flight test record. This procedure checks if the instructor is available in the time frame and whether the instructor's teaching licence is still valid before the start of the schedule. It also automatically calculates the test starting time as the actual flight test starts an hour later than the reporting time to allow the student to prepare for it.

```
EXEC dbo.newflighttest
        @Start                  ='2018-09-15 11:00:00.000',
        @End                    ='2018-09-15 14:30:00.000',
        @Instructor             ='LT003',
        @Pilot                  ='5',
        @Test                   ='12',
        @Aircraft               ='YS3680'
```

	Event ID	Start Time	Ending Time	Instructor	Pilot	Test Name	Aircraft	Test Starting Time	Result
1	18	2018-09-15 11:00:00.000	2018-09-15 14:30:00.000	Zac	Kelsie Fry	EU PPL Flight Test	YS3680	2018-09-15 12:00:00.000	91

Figure 42: Flight Test Record Inserted by Book Flight Test Query

6.0 Emerging SQL Features

SQL as with many other computing technologies is ever-evolving and new features are introduced relatively frequently. The SQL features explored below are additional features to the Microsoft SQL Server that allows for functions from using the data to be managed more efficiently to being able to generate easily digestible reports from the existing data in the database. These services are part of the Microsoft SQL Server Services Suite.

6.1 SQL Server Reporting Services

SQL Server Reporting Services (SSRS) is a solution that allows the users to generate graphical reports based on the data on their systems and make them accessible via a web interface that is accessible on both desktop computers and mobile devices. The software is run on the server and integrates with Microsoft Visual Studio to design the web application that form the front-end user interface (Microsoft, 2018). This tutorial will outline the incorporation of SSRS in an existing database to generate graphical representations of the data.

6.1.1 Installing SSRS on a Database

SSRS installation package can be downloaded from the Microsoft website (Figure 43). The SQL Server Database Engine must be installed separately beforehand for SSRS to work. After running the installer and following the on-screen instructions, choose an edition of your choice (Express edition is being used in this tutorial). After the installation has been completed, the server needs to be configured to work.

Figure 43: SSRS Installer

6.1.2 Configuring SSRS And Report Builder

After installation, enter the Report Server Configuration Manager. The report server requires its own database to store server metadata and objects (Microsoft, 2018).

By default, SSRS uses a virtual service account to create the databases therefore, if the software is deployed on a separate server from the server containing the database engine, the login credentials for the server need to be entered in the Service Account section (Figure 44).

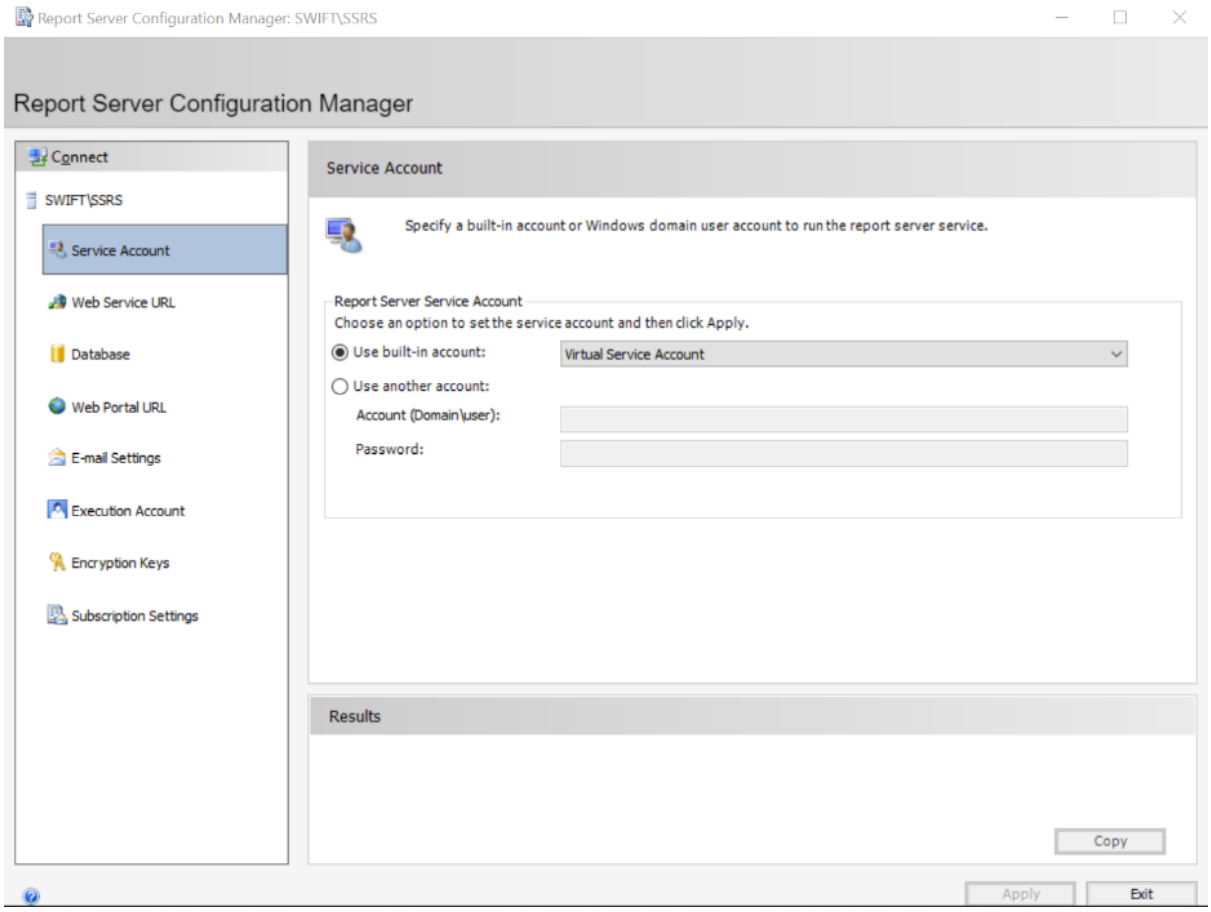

Figure 44: Report Server Configuration Manager Service Account section

To create the report server database, the Report Server Database Configuration Wizard can be used to create the database. The wizard can be run by clicking on 'Change Database' on the Database tab (Figure 45). In the wizard, the specifications for the server, login credentials, the database settings and the service credentials (or other credential types if the SQL server is on a remote location) need to be entered to create the required database. The ports for the connections also need to be enabled on the firewalls of servers. This creates two databases, ReportServer and ReportServerTempDB.

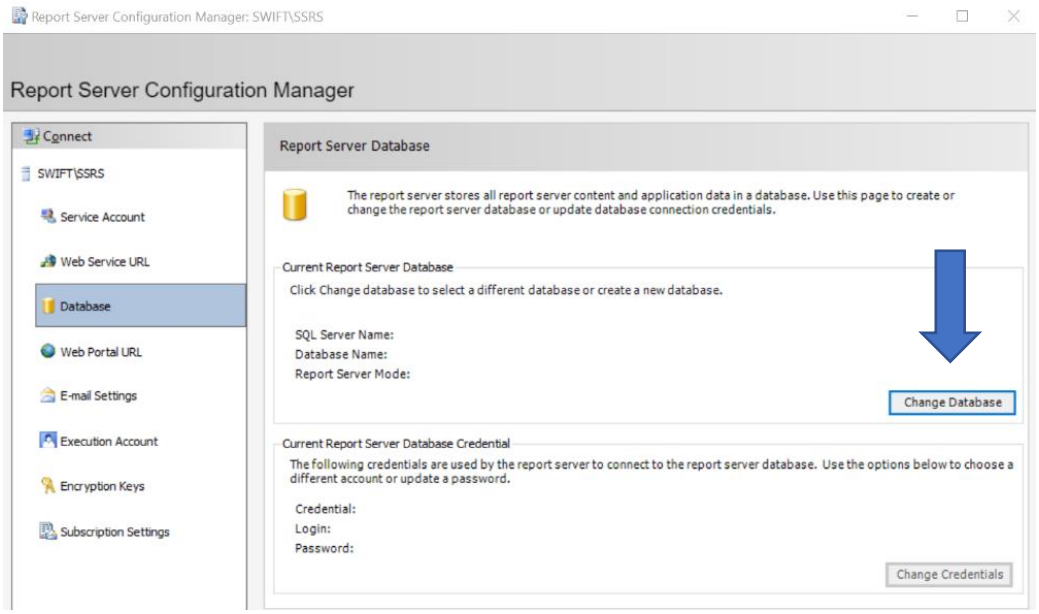

Figure 45: SSRS Database Configuration

Next, the configuration for the Web Service URL needs to be set so that the server is accessible for testing from web browsers. The configuration manager provides default values initially, but they can be reconfigured to suit the IPs, ports and security requirements of the network the server is being set up on (Figure 46). The Advanced tab also allows for configuration of multiple instances of SSRS to be run on separate IPs and ports. The default values will be applied for the purposes of this tutorial.

Afterwards, the URL of the web portal needs to be configured to allow for accessing from devices in a similar way (Figure 47).

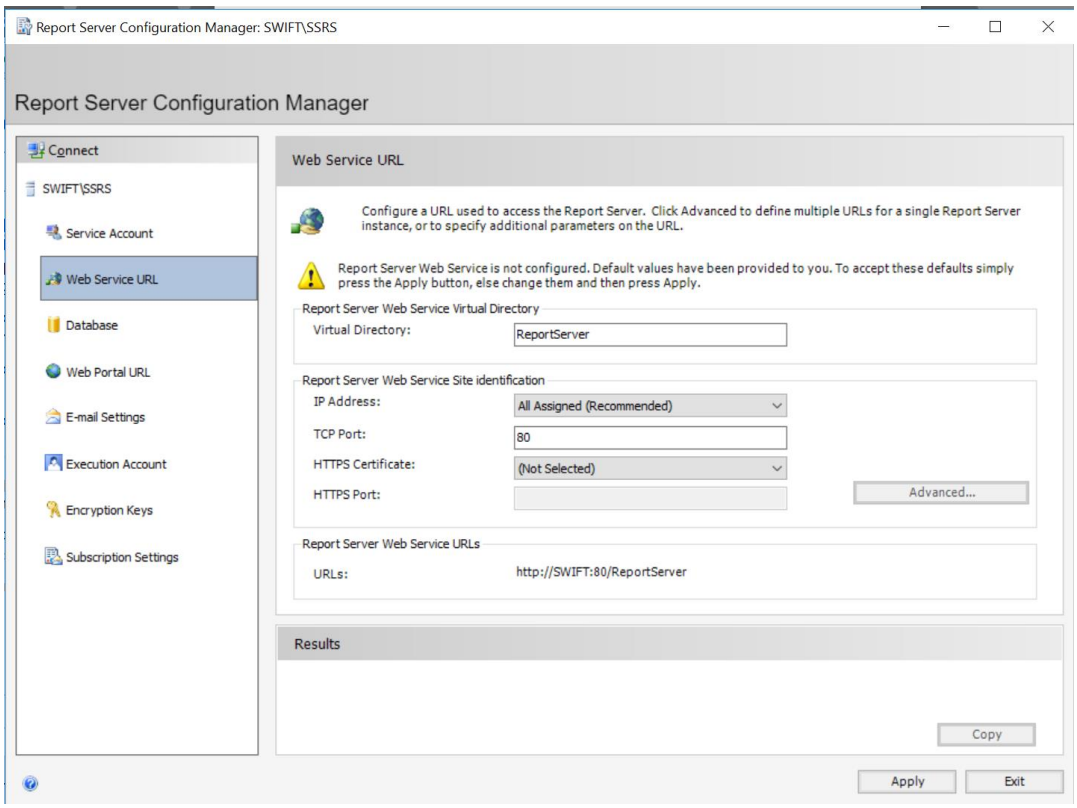

Figure 46: SSRS Web Service URL Configuration

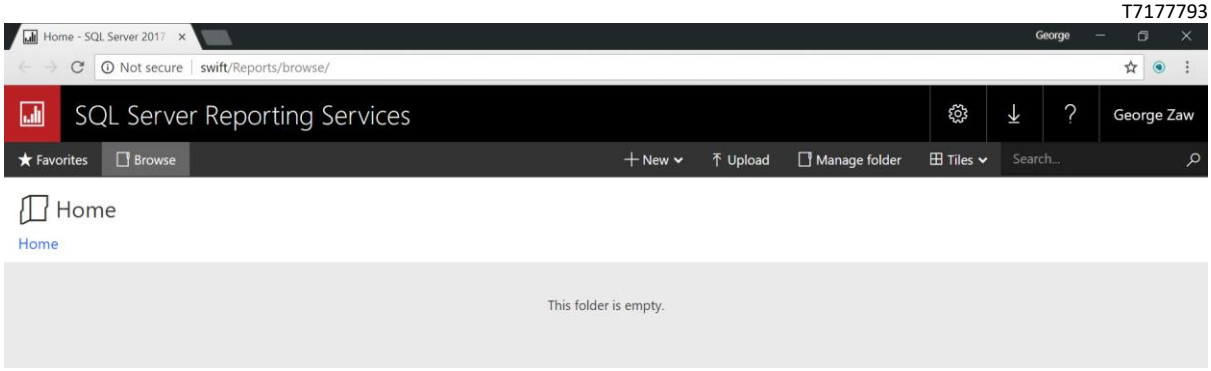

Figure 47: Functioning SQL Report Server Page

6.1.3 Generating and Deploying Reports to The Report Server

The report builder can be used to generate queries to create graphical representations of the data present in the databases connected to it that can be represented on the SSRS server. These reports are updated dynamically as the data on the database changes (Microsoft, 2017).

With the report server now set up, the reports page should now have a functional user interface. The report builder now needs to be installed to generate reports to be deployed onto the report server. The download link for the report builder can be reached through the reports page itself. During installation, the URL of the report server can be specified as default (Figure 48).

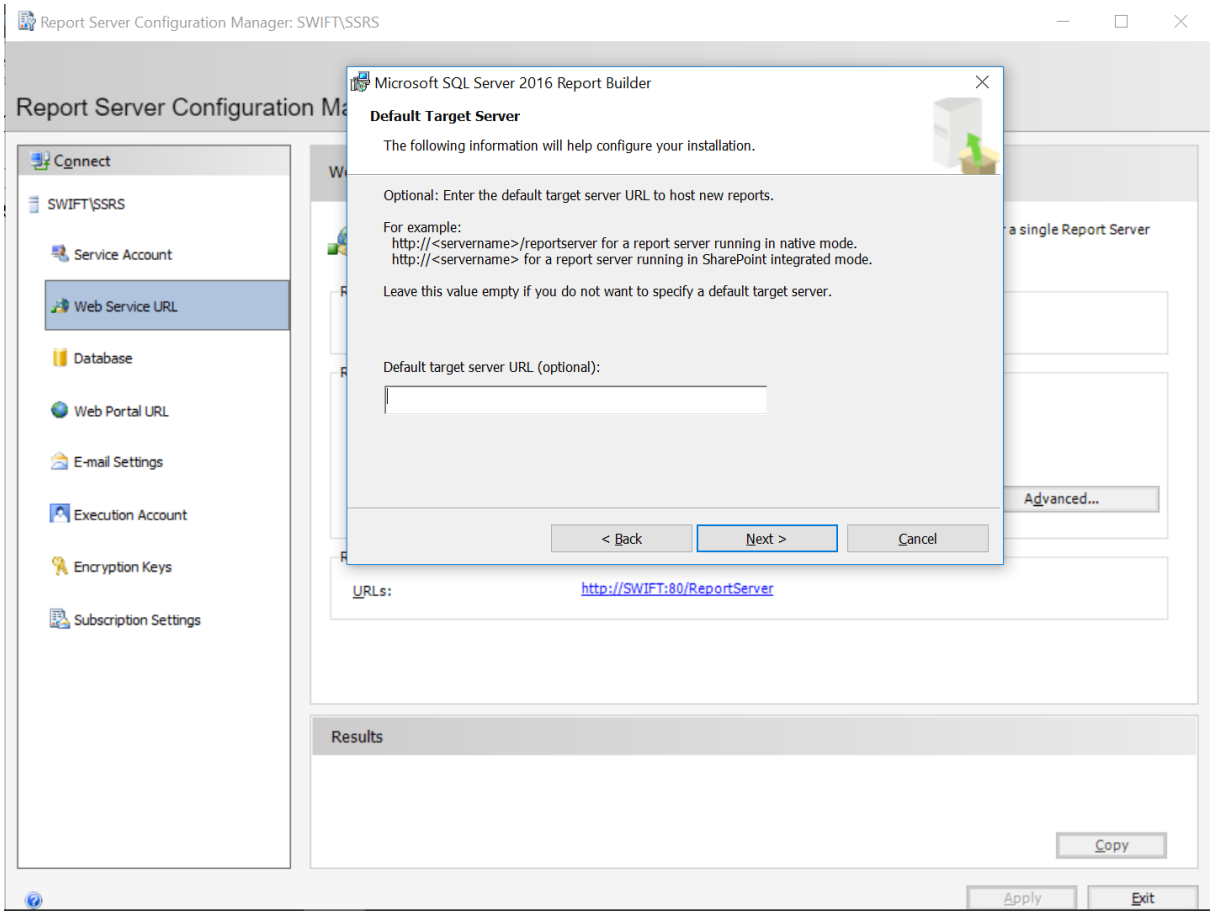

Figure 48: SSRS Report Builder Setup

The report build has multiple wizards to create different types of reports to be deployed. In this case, a chart will be made. First, a new data source must be created to allow for retrieval of the data from the database to use. This involves the creation of a new connection string that specifies the SQL Server and the database (Figure 49). The database for the Aviation School System will be used in this example. After the data source is created, it can be used in any of the wizards to generate corresponding reports (Microsoft, 2017).

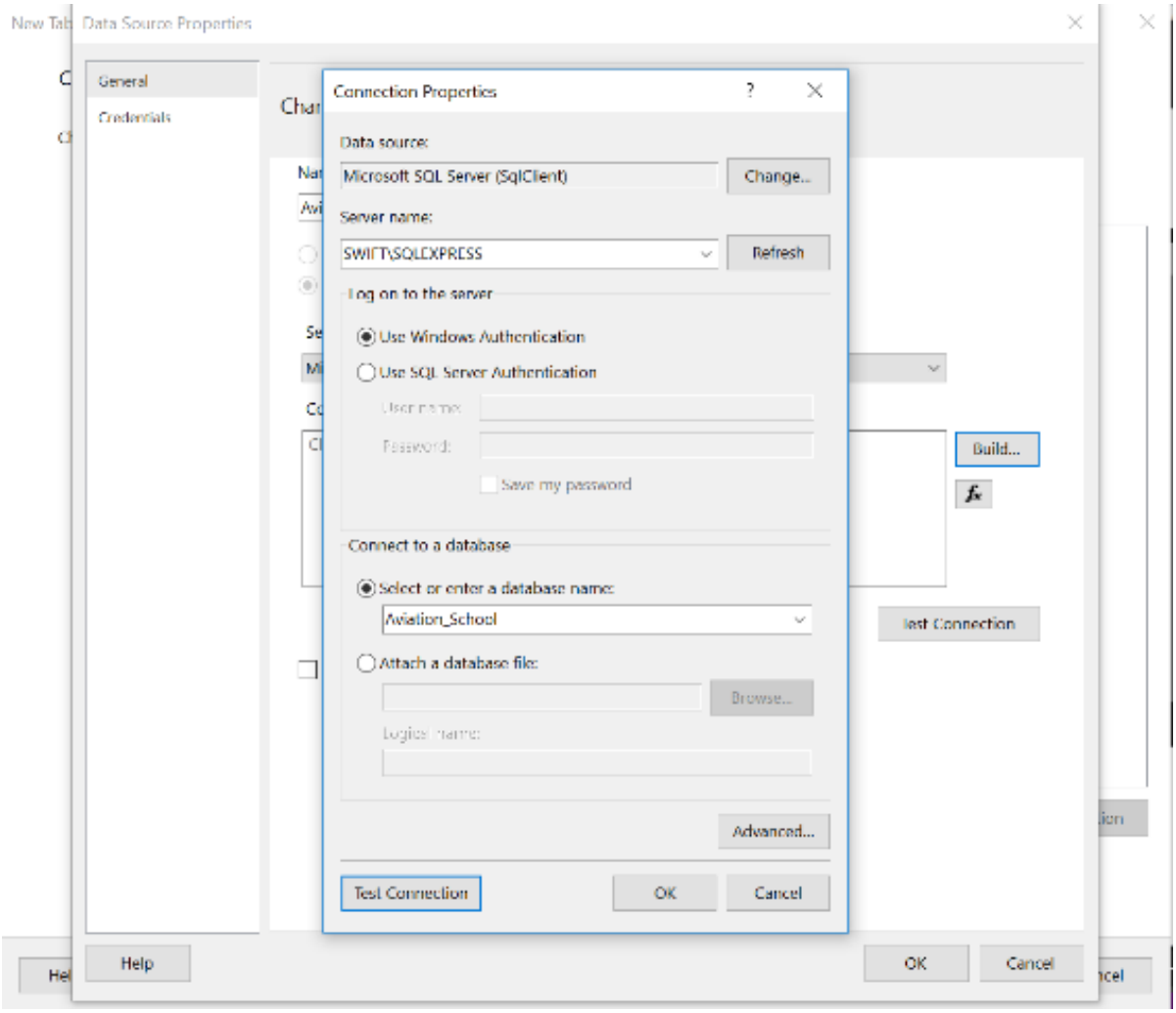

Figure 49: SSRS Report Building Creating New Connection String

Once the connection to the database has been set up, a query can be designed using the built-in graphical user interface to retrieve the necessary data to represent in the chart. The attributes or tables to be displayed can be select on the Database View pane and the selected fields can have functions such as sum or order by applied to. The relationship between the chosen attributes can be manually specified but the wizard also includes an auto detect function to create the relationship. Filters can also be applied to the dataset to limit the data represented (Figure 50).

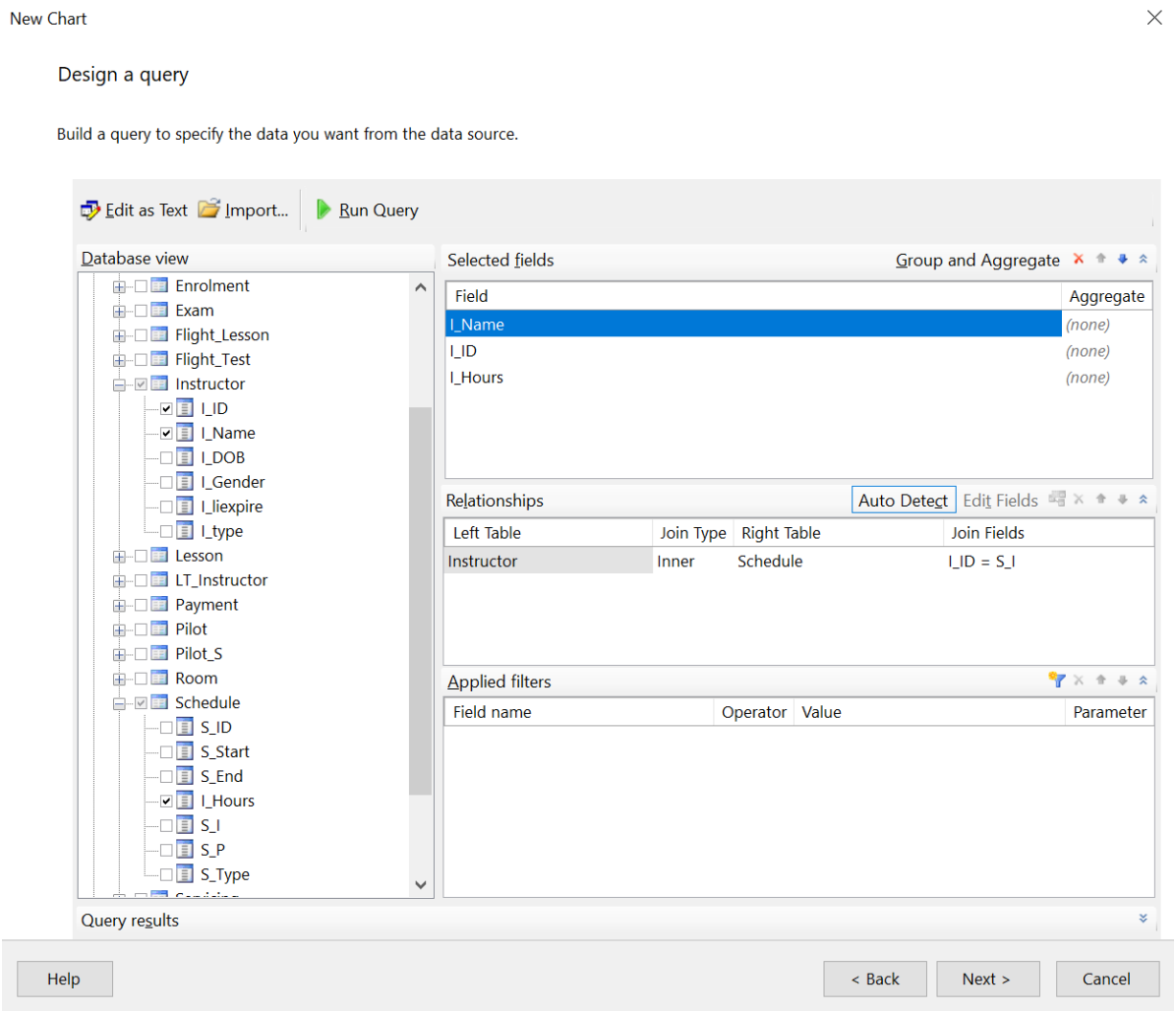

Figure 50: Creating A Query for Retrieving Data In GUI

Once the dataset is complete, the type of chart is chosen then the chosen attributes are arranged on the axes on the chart so that they are displayed correctly (Figure 51). The generated template can then be modified to add titles, change styles of fonts, layouts, etc. If the version of SSRS Server running were a standard version or higher, the generated report can be directly uploaded through File -> Publish Report Parts. However, as the tutorial is using the Express version of SSRS, this feature is not enabled, as it is only for standard version and above, but a workaround can be done. The file can be saved onto the local drive in the .rdl format then uploaded onto the SSRS via the web portal. This allows the dynamically updated report to be viewed through the web portal (Figure 52).

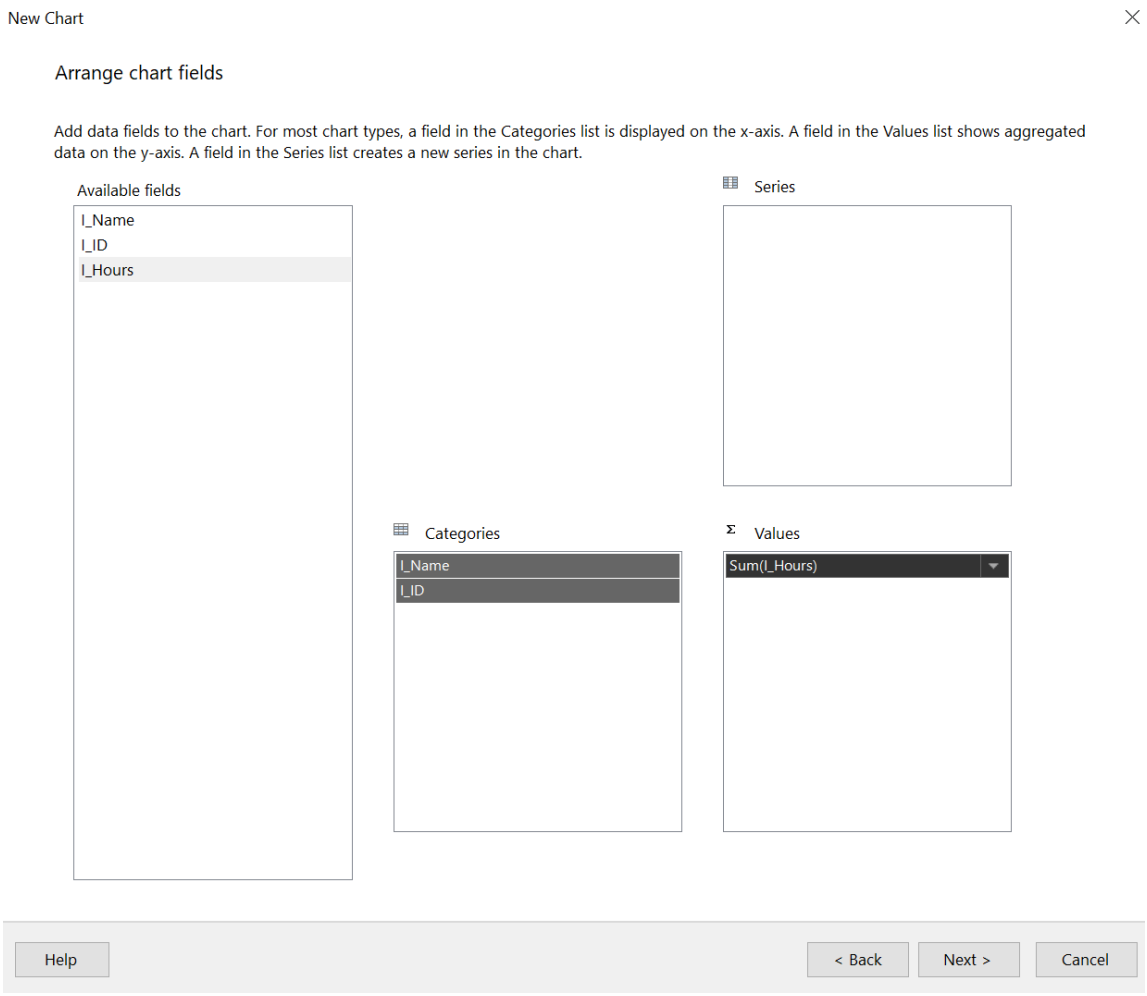

Figure 51: Arranging Attributes to be Shown on Axes

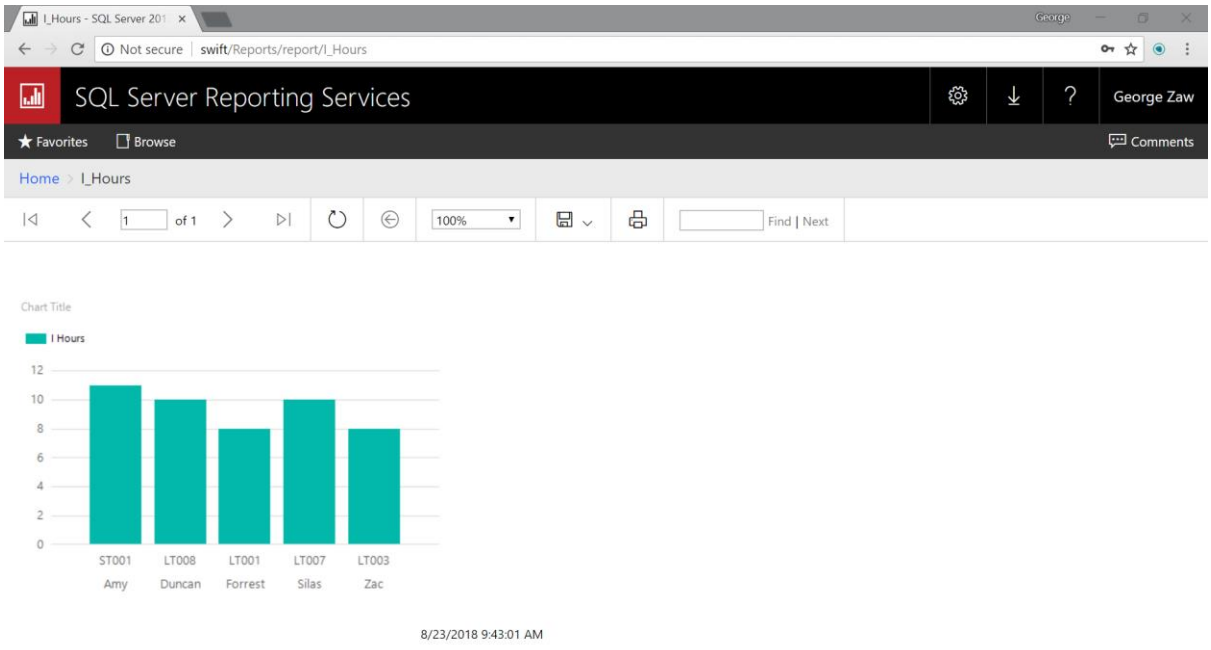

Figure 52: Deployed Report on SSRS Server Web Portal

6.1.4 Further SSRS Functionality

6.1.4.1 Creating a Shared Data Source

With the previous example, the data source configured only applies to the specific report and would not be reusable for others. A shared data source can be created via SSRS web portal to allow for reusing of the data source that would be commonly used by the users on the SSRS portal. The new data source page can be reached via the New tab (Figure 53).

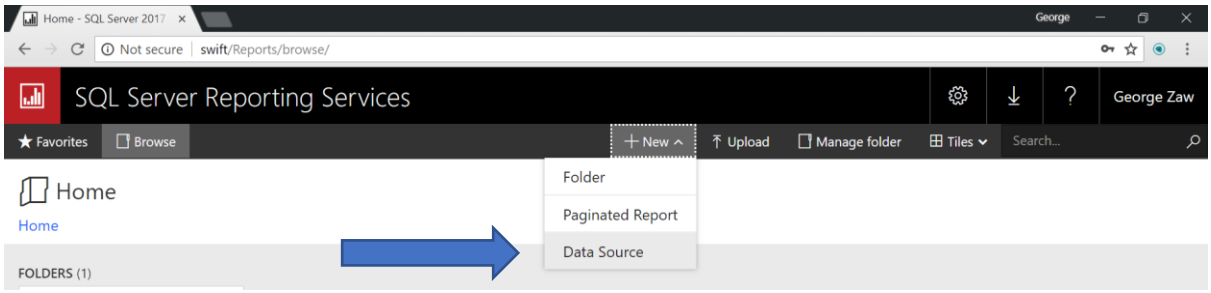

Figure 53: Creating a New Data Source on SSRS Web Portal

In the New Data Source Form, the name and the description must be entered. Options for hiding the data source on the web portal and enabling the data source are also present below the description. A connection string is required to link the data source to the database to be connected to. The syntax of the connection string to a SQL Server is:

```
Data Source=<hostname>\<server name>; Initial Catalog=<database name>
```

The login credentials for connection to the database also need to be entered to be able to connect to the database. This needs to be configured accordingly to the structure of the network and security policies of the firm (Figure 54).

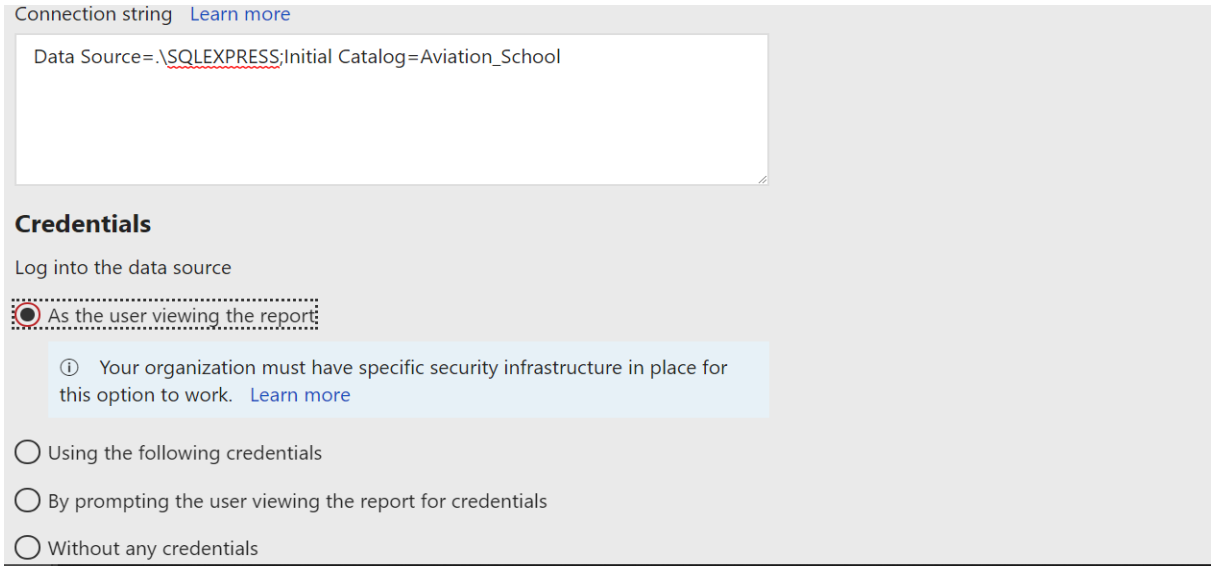

Figure 54: Configuration of a New Data Source

After the configuration is complete and the SQL Server is verified to be contactable via the connection string, the new data source can be saved to be used for creating new reports (Figure 55).

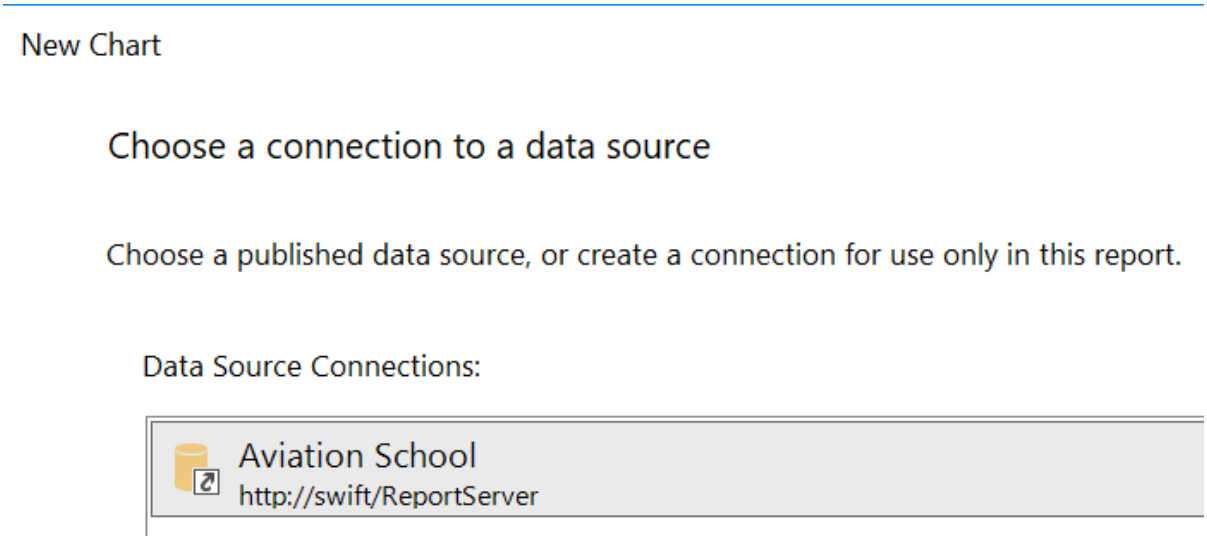

Figure 55: Now Accessible Shared Data Source

6.4.1.2 Adding Role-Based Security for User Management

SSRS contains functionality for giving sets of permissions on the SSRS web portal to security groups or individual users. First, the groups or users must be assigned user roles on the server. This gives or limits administrative functions to the users. These can be assigned in the Site Settings -> Security tab (Figure 56).

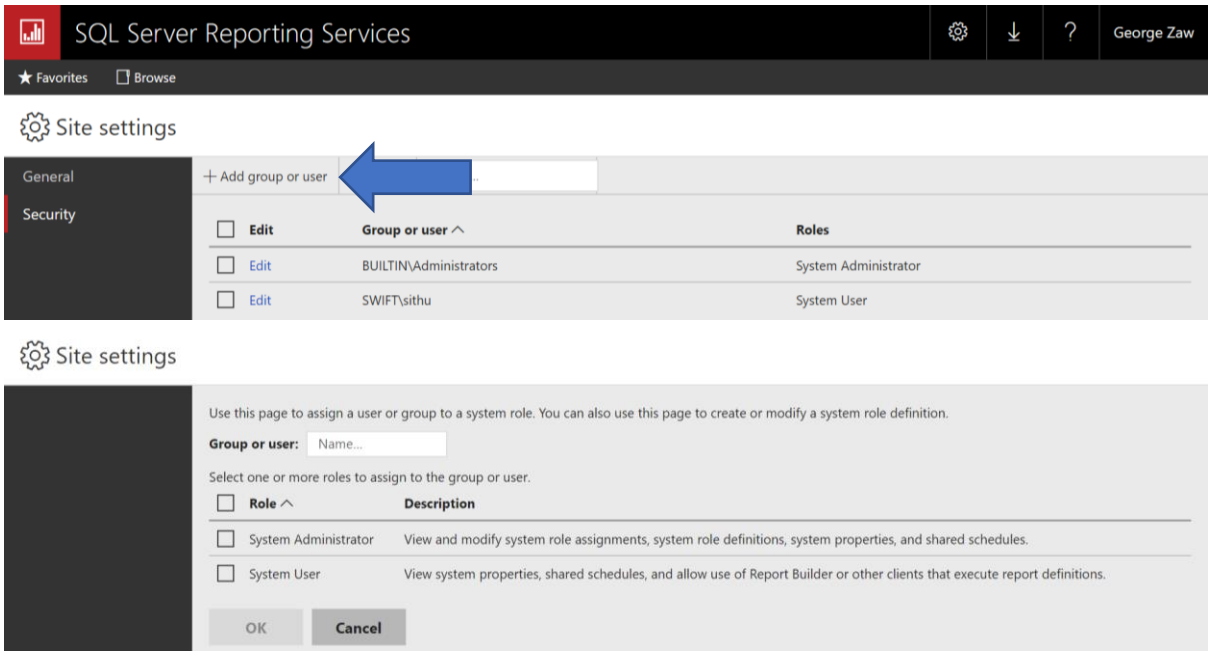

Figure 56: SSRS Web Portal Security Tab and Assigning Roles

Once the user roles have been assigned, the individual permissions for each folder can be applied. This allows for more granular control of managing the use of SSRS web portal for firms with large numbers of employees, such as permissions for viewing contents, making changes to the content, publishing reports and viewing a report's definitions. The individual folders' settings can be changed via the Manage Folder button on each folder (Figure 57).

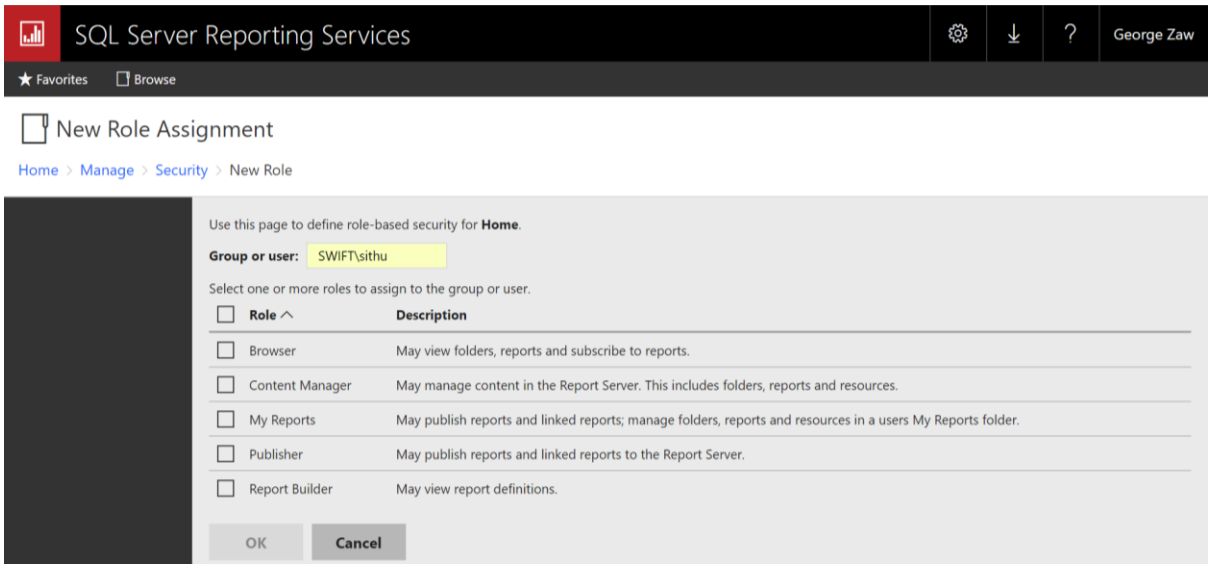

Figure 57: SSRS Permissions for Folders

7.0 Conclusion and Critical Review

The database designed for the Aviation School should serve it well for a long time to come and should cover the business needs. The documentation required for maintenance of the database has been included in the report. The following is a critical review on the different aspects of this assignment based on the criteria given in the marking scheme based on my opinion.

7.1 Criteria A: Case Study, Report and Critical Review

Developing the Aviation School System has been very intriguing, and I have learnt much about SQL in the process. I chose to develop the case study because of my fascination with aviation. Identifying the necessary requirements for the system from the case study has been challenging as I found that there are many nuances in the functions given that greatly affected the design of the tables depending on how they were interpreted. After many scrapped use case diagrams, I feel that I have arrived at the specifications that outlines most, if not all the requirements of the aviation school. I have also made attempts at improving the current system rather than creating a replica of the physical system in the form of a database by automating and/or combining functions to streamline the workflow.

The technical report covers the requirements and the specifications for the creation of the database for the aviation school and documents the functionality built into the system. It fully documents the functionality of the system, fully supported by the SQL codes and screenshots of it working. Therefore, I feel that the report is of first class quality.

7.2 Criteria B: ERD, Tables, Views and Keys

The entity relationship diagram and the tables of the database were tricky to design due to the aforementioned problem I had with identifying the functions the system was supposed to perform from the case study. When I finally arrived at the design utilising subtypes of multiple entities that need to be stored, the tables also had to undergo major changes to accommodate them. Nevertheless, I am pleased with the overall result that is achieved within the given timeframe and the knowledge I gained in the process of making the system. In my opinion, the ERD reflects the requirements of the scenario in terms of the entities and the relationships between them. The tables are also implemented to the specification of the ERD with suitable mix of single and composite keys to ensure data consistency throughout the schema, as demonstrated by the ability to insert to or read data from multiple tables without conflicts in constraints. Accompanying views are also implemented to make retrieval of data for subtype entities simpler. As such, I feel that I have achieved first class in these criteria.

7.3 Criteria C: Implementation of Database and SQL Statements

The database implemented contains constraints as described in the ERD and are all functional. I feel that he scripts for populating the database with test data that are included provide enough data to demonstrate the capabilities of the database and the stored procedures. I have used various data manipulation language in the queries and stored procedures intended to fulfil the functions outlined on the use case diagram. These consist of a mix of SQL queries and SQL statements to utilise stored procedures. I had the most trouble creating the stored procedures as they combine many functions, it was also harder to ensure that they ran smoothly and had adequate safeguards to ensure integrity of the database in event of a failed procedure due to problems such as bad input. I managed to resolve that by adding transaction rollbacks to the procedures. If I had more time to work on the DML, I would certainly like to add more to the stored procedures to further automate the operation of the database to a higher degree and to put additional functionality to the database than it currently has. With the

amount of functionality implemented and the diverse functions employed in the database, I feel that the work is within the range of first class.

7.4 Criteria D: Emerging SQL Features

For this criterion, I explored the function of Microsoft SQL Server Reporting Services feature in detail. I went through step by step thoroughly from the installation and set-up of the SSRS server to creating reports and configuring security settings on the web portal, supported by screenshots. I have learnt a lot about SSRS in the process of writing the tutorial since I also had to set up the server and the report builder to a functional state. As I was able to cover the purpose, functionality and demonstrate a good command of SSRS, I feel that my work is of first class.

References

Microsoft, 2017. *Add a New or Existing Report to a Report Project (SSRS).* [Online]
Available at: https://docs.microsoft.com/en-us/sql/reporting-services/tools/add-a-new-or-existing-report-to-a-report-project-ssrs?view=sql-server-2017
[Accessed 18 08 2018].

Microsoft, 2017. *Report Builder in SQL Server 2016.* [Online]
Available at: https://docs.microsoft.com/en-us/sql/reporting-services/report-builder/report-builder-in-sql-server-2016?view=sql-server-2017
[Accessed 18 08 2018].

Microsoft, 2018. *Create a Report Server Database.* [Online]
Available at: https://docs.microsoft.com/en-us/sql/reporting-services/install-windows/ssrs-report-server-create-a-report-server-database?view=sql-server-2017
[Accessed 18 08 2018].

Microsoft, 2018. *SQL Server Integration Services.* [Online]
Available at: https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-2017
[Accessed 18 08 2018].

Microsoft, 2018. *What is SQL Server Reporting Services (SSRS)?.* [Online]
Available at: https://docs.microsoft.com/en-us/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports?view=sql-server-2017
[Accessed 18 08 2018].

Appendices

Appendix A: Links to related documents

1. Case Study Document: Aviation School: ica.dd.aviationschool.pdf
2. Link to Full Size Use Case Diagram: AviationSchoolUCD
3. Link to Full Size ERD: AviationSchoolERD
4. Link to Full Size Schema: AviationSchoolSchema
5. Link to Database Creation Script: SSMS Scripts\1. Aviation School Create All Tables and Indexes.sql
6. Link to Full Resolution Diagram: SQL Diagram Output.png
7. Link to Script for Creating Procedures: SSMS Scripts\2. Aviation School Stored Procedures.sql
8. Link to Database Population Script: SSMS Scripts\3. Aviation School Populate And Select All.sql