

Emerging Database Technologies ICA

Task A & B: Design and Implementation of Bus Management
Database

Task C: Research and Investigation into Power BI

30 April 2019

Teesside University / Management Development Institute of Singapore

Information Systems Security (**CIS3007-N**)

Student ID: T7177793

Student Name: Si Thu Zaw

Lecturer Name: Ms. Manian Latha

Table of Contents

1.0 Introduction	1
1.1 Project Scenario	1
1.2 Additional Considerations and Assumptions	1
2.0 Requirements.....	2
2.1 Use Cases	2
2.2 Architecture Diagram.....	2
3.0 Task A: Database Design	3
3.1 Complete Entity Relationship Diagram	3
3.2 Partial Entity Relationship Diagrams.....	4
3.2.1 Bus and Promotions Management	4
3.2.2 Human Resources	4
3.2.3 Route Management	5
3.2.4 Scheduling.....	5
3.2.5 Location Management	6
3.2.6 Analytics Collection.....	6
3.2.7 Payment Cards	7
4.0 Task B: Database Implementation	8
4.1 B1: Creating SQL Tables and Relationships.....	8
4.1.1 Implemented Database Diagrams	8
4.1.2 Database Creation Script	11
4.2 B2: Database Population Script	12
4.3 B3: Database View Creation.....	12
4.3.1 Database View Creation Script.....	12
4.4 B4: Creating Stored Procedures.....	13
4.4.1 Bus System Procedure Creation and Execution Scripts	13
4.4.2 'newstudentpass' Procedure	13
4.4.1 'getrecord' Procedure	14
4.4.3 'getfee' Procedure	14
4.4.4 'tripregister' Procedure.....	15
4.4.5 'loganalytics' Procedure.....	16
5.0 Task C: Microsoft Power BI	17
5.1 Data Integration	17
5.2 Data Processing and Visualisation	20
5.2.1 Query Editor.....	20

5.2.2 Creating a Graph/Chart.....	21
5.2.3 Build Report Dashboards	24
5.3 Sharing the report.....	25
References	25

Table of Figures

Figure 1: Use Case Diagram of University Bus Management System.....	2
Figure 2: Simplified Architecture diagram of the Bus Management System.....	2
Figure 3: Complete University Bus System Entity Relationship Diagram	3
Figure 4: Bus and Promotions Management Partial ERD	4
Figure 5: Human Resource Management Partial ERD	4
Figure 6: Route Management Partial ERD	5
Figure 7: Scheduling Partial ERD	5
Figure 8: Location Management Partial ERD	6
Figure 9: Analytics Collection Partial ERD	6
Figure 10: Payment Cards Partial ERD	7
Figure 11: Implemented Full Database Diagram	8
Figure 12: Implemented Bus and Promotions Diagram.....	8
Figure 13: Implemented Human Resource Management Diagram.....	9
Figure 14: Implemented Route Management Diagram.....	9
Figure 15: Implemented Scheduling Diagram.....	10
Figure 16: Implemented Location Management Diagram.....	10
Figure 17: Implemented Analytics Collection Diagram.....	11
Figure 18: Implemented Payment Cards Diagram.....	11
Figure 19: Output of Current Promotions View.....	12
Figure 20: Output of Bus Stops View	12
Figure 21: Output of Depots View	12
Figure 22: Output of Staff Detail View	12
Figure 23: Power BI Start-up Screen	17
Figure 24: Power BI Get Data Screen	18
Figure 25: Power BI SQL Server Database Connection Screen	18
Figure 26: Power BI SQL Server Credentials Screen Options	19
Figure 27: Power BI Data Import from SQL Server	19
Figure 28: Power BI Query Editor and Available Functions (in red highlight)	20
Figure 29: Visualisations Panel of Power BI	21
Figure 30: 'Ask A Question' Button on Power BI Toolbar	21
Figure 31: Query Field for Questions in Power BI.....	22
Figure 32: Generated Result In Table Form From Question in Power BI.....	22
Figure 33: Generated Result in Bar Chart Form From Question in Power BI	23
Figure 34: Previously Generated Chart Changed to Pie Chart.....	23
Figure 35: Power BI Report File	24
Figure 36: Exporting Options of Power BI.....	25

1.0 Introduction

The following document outlines the specifications for the creation of a database system to support a private-hire bus company, including the requirements of the system, the design of the database including the partial ERDs and the implementation of the database.

1.1 Project Scenario

The system is designed for a private-hire bus company to support the operations of the new bus service in a university campus. The company typically ran chartered bus services and stored its operational information on paper and Microsoft Excel sheets. As the operations are now expanding to include providing service for an intra-campus bus service for a university, a new database system is needed to manage the necessary data for the task. The company currently owns 15 buses of various sizes and operates from 2 depot locations.

The database needs to be designed to accommodate IOT features for the embedded systems onboard buses. They are intended to handle task such as logging and charging fares for different types of passengers, reporting the status of the bus every stop with information such as passengers onboard, bus stop code, time reached, etc so that the data can be used to provide information to users on the smartphones for time till next bus and also provide analytics data that can be analysed to improve the service in the future. The users can also use the mobile application to

The system also needs to support traditional recordkeeping functions that will replace the paper and Excel sheet records. The database will store necessary data for managing buses and their maintenance, scheduling, routes and stops, HR systems for managing staff and bus passes. Additionally, functions to manage promotions is also needed as the company runs various types of advertisements on the buses.

1.2 Additional Considerations and Assumptions

- Payment System:
 - The students can purchase unlimited bus passes from the school office that have a validity of 3 months to utilise the bus service.
 - The standard bus cards for city buses can be used by the general public to ride on the university bus service.
 - The standard fare will be calculated using this formula: \$ $(0.2 * x^{1/4})$, where x is distance travelled in kilometres.
- A mobile app can be used to enter the desired bus number and the current stop to get an estimate time of arrival of the next bus.
- The payment card system for public transport that already exists in the city is compatible with the system used for the university bus service.
- SQL Procedures:
 - The 'loganalytics' procedure is intended to be called by the bus embedded systems once they have detected that the bus has moved away from a stop.
 - The 'tripregister' procedure is intended to be called when a passenger taps into the bus using their card.

2.0 Requirements

2.1 Use Cases

The following use case diagram indicates the needs that the system is intended to fulfil as explained in the project scenario (Figure 1).

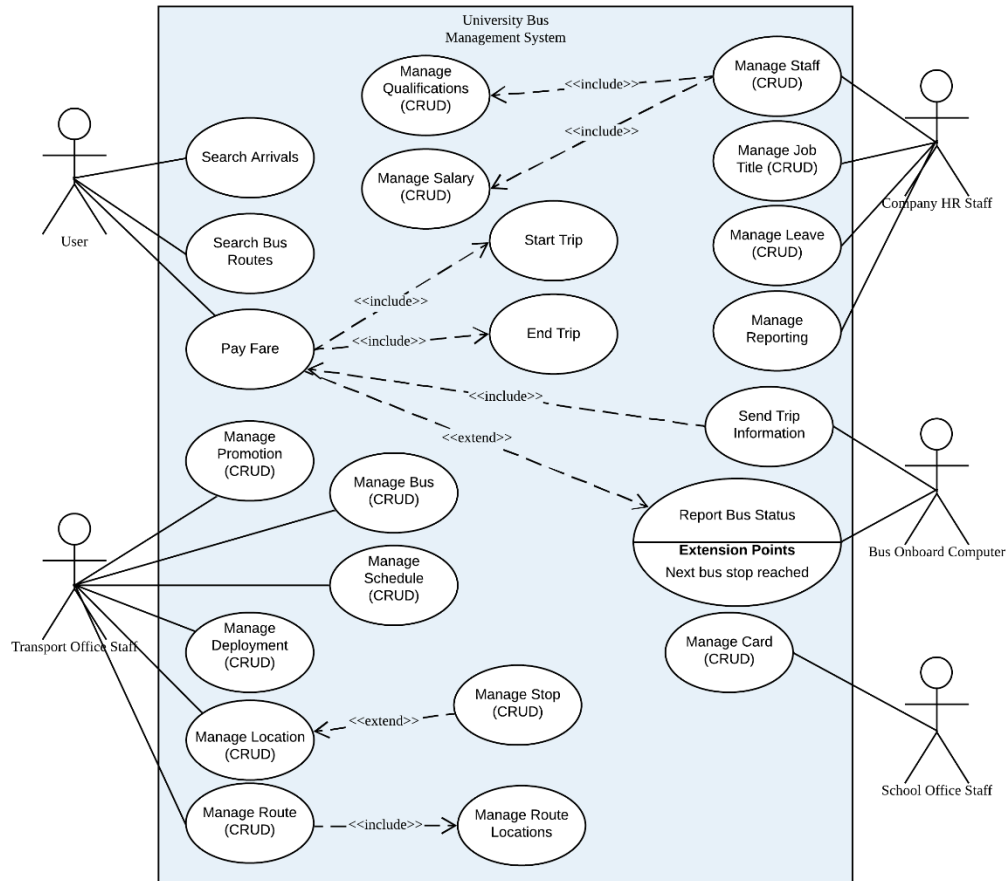


Figure 1: Use Case Diagram of University Bus Management System

2.2 Architecture Diagram

The following is the diagram showing the components involved in the system (Figure 2).

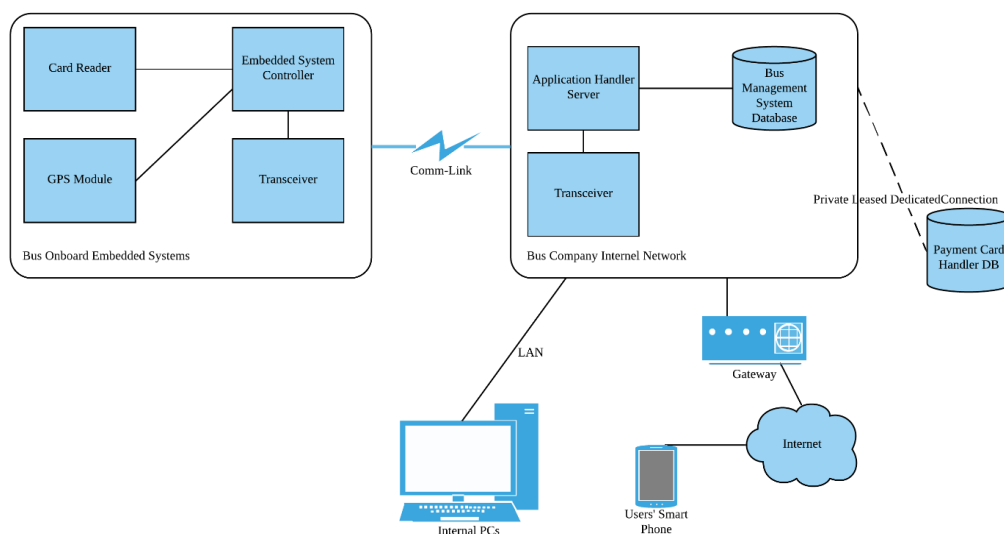


Figure 2: Simplified Architecture diagram of the Bus Management System

3.0 Task A: Database Design

3.1 Complete Entity Relationship Diagram

This Extended Entity Relationship Diagram (EERD) shows all aspects of the University Campus Bus System. The total system consists of sets of tables that handle separate functions (Figure 3).

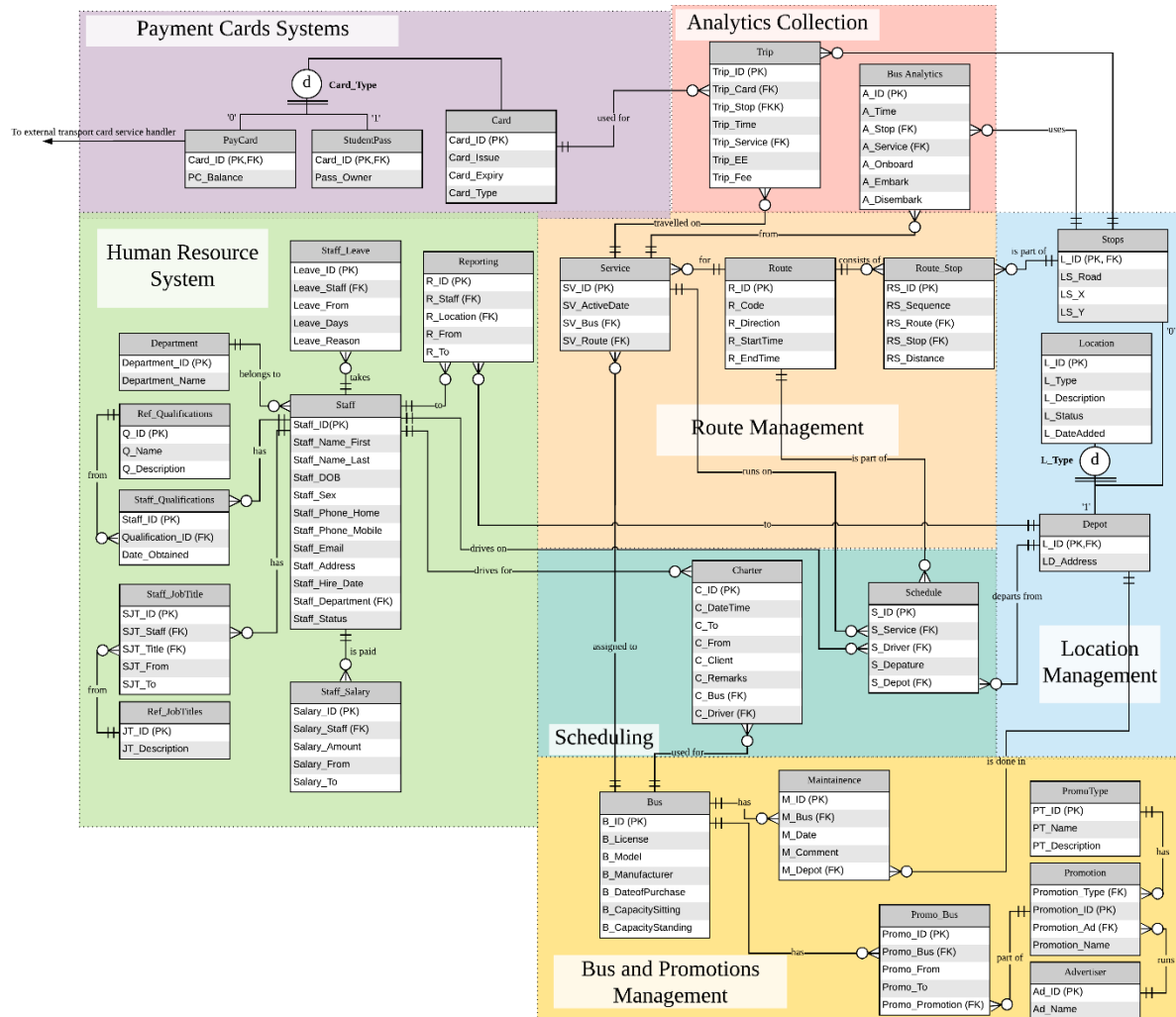


Figure 3: Complete University Bus System Entity Relationship Diagram

3.2 Partial Entity Relationship Diagrams

3.2.1 Bus and Promotions Management

This subsystem is for storing and managing data regarding the fleet of buses the company operates. This includes maintenance logs, and tables for storing information for brand promotions on buses (Figure 4).

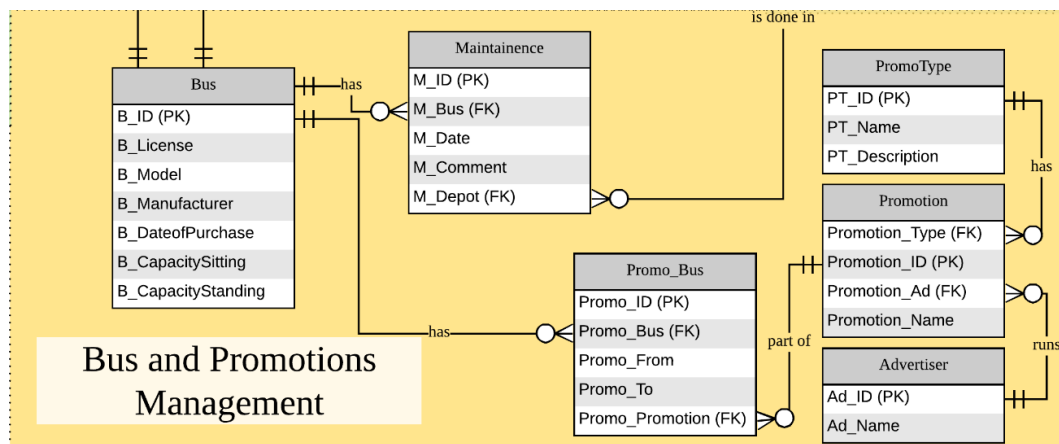


Figure 4: Bus and Promotions Management Partial ERD

3.2.2 Human Resources

This subsystem is for storing and managing data regarding the employees working at the company. This includes typical HR data such as the employee details, job titles, leaves, the bus services or chartered trips they have been assigned to, etc (Figure 5).

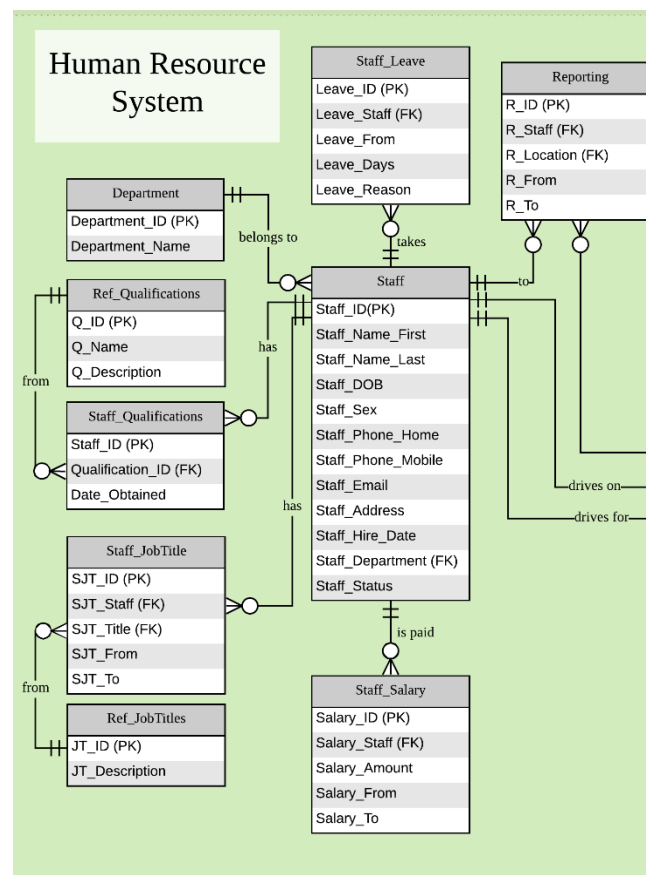


Figure 5: Human Resource Management Partial ERD

3.2.3 Route Management

The route management stores data about the university bus routes. This includes the stops and their sequences for each route, the distances for fee calculation and information about the typical start and stop time of each route, which is shown to the users on the mobile app. The service table stores information about which buses are in use for which route (Figure 6).

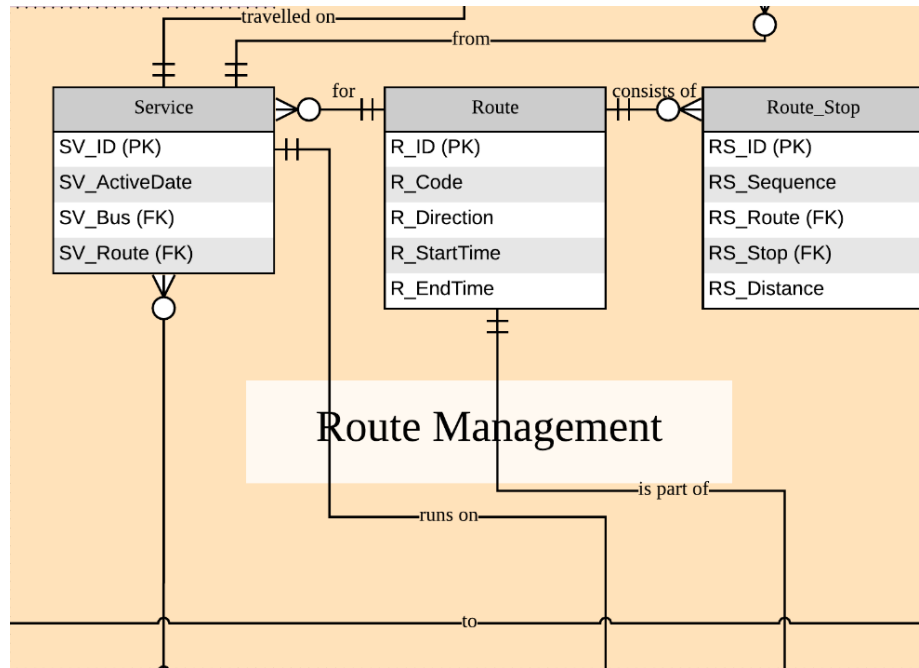


Figure 6: Route Management Partial ERD

3.2.4 Scheduling

The scheduling tables store details of the services and the chartered trips. Schedule table stores the times that each service is scheduled to depart from the depots and the driver on that run. The charter table stores similar data for private hire, one-way, trips by organisations (Figure 7).

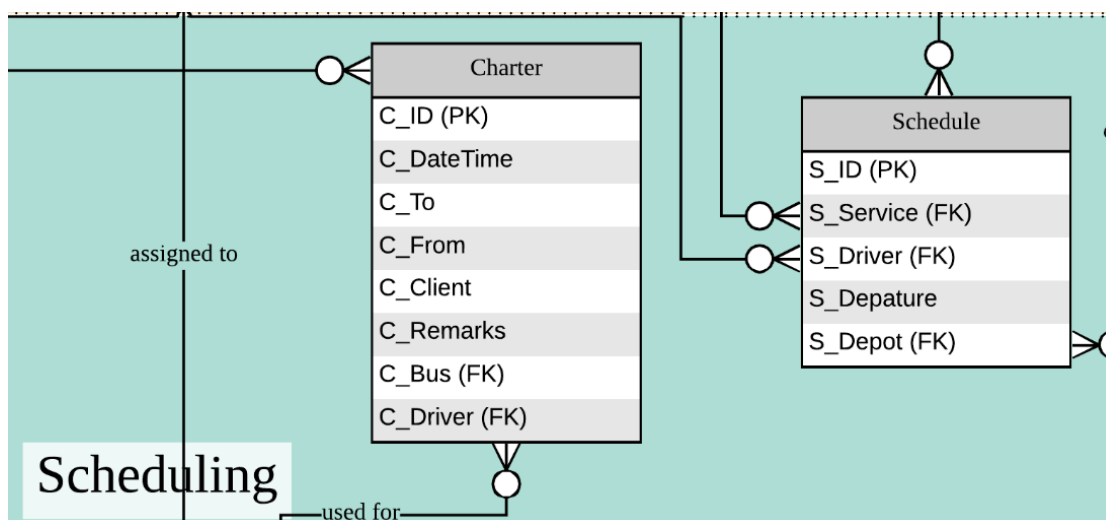


Figure 7: Scheduling Partial ERD

3.2.5 Location Management

The location management tables store information about all locations involved in the operation of the bus service. This includes general information such as the status and the description of the location, the precise location of the bus stops and the addresses of the bus depots (Figure 8).

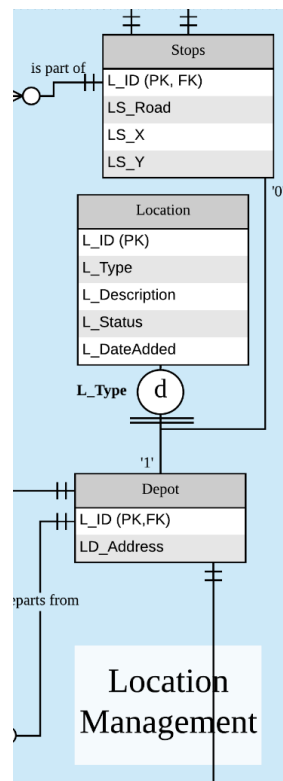


Figure 8: Location Management Partial ERD

3.2.6 Analytics Collection

The analytics collection tables store data that can be processed to generate information about the bus usage for both from the view of each passenger and overall. The trip table (and its accompanying procedures) are used for storing the trip information of each passenger, such as which service and stop they tapped in with their bus cards, and the cost for the trip upon exiting a bus. The bus analytics table stores the numbers of people getting on and off the bus on each service at each stop. The records for these tables are intended to be automatically generated upon leaving a bus stop using the bus's GPS as a trigger (Figure 9).

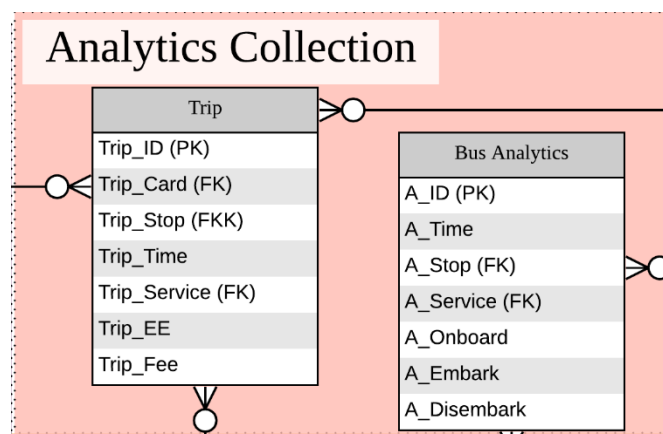


Figure 9: Analytics Collection Partial ERD

3.2.7 Payment Cards

The tables in this section hold data for the cards used to process payments for the bus service in the university bus system. There are two types of cards that can be used, student passes and payment cards. The system bills the passengers who use the standard payment cards, which is handled by the external company that runs the payment card systems. The passengers who use the student pass are not billed on the system (Figure10).

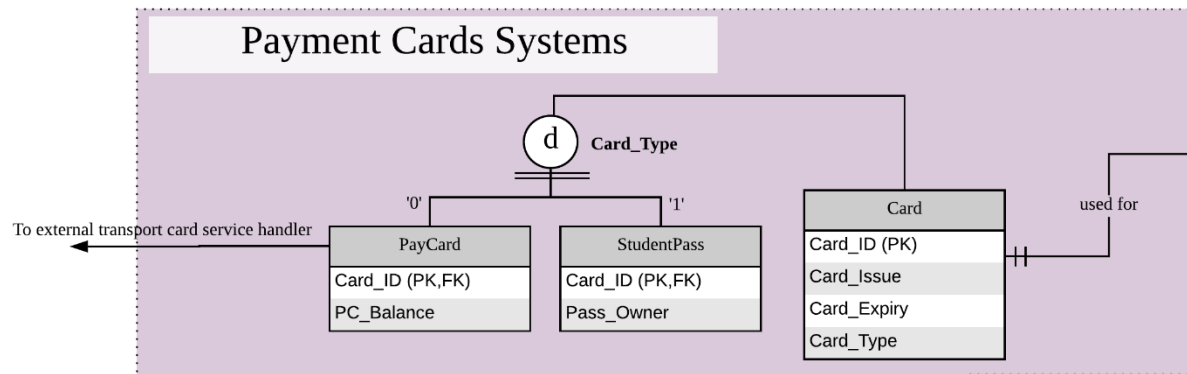


Figure 10: Payment Cards Partial ERD

4.0 Task B: Database Implementation

4.1 B1: Creating SQL Tables and Relationships

4.1.1 Implemented Database Diagrams

The following diagrams are generated in the SQL Server Management Studio after the implementation of the database. Figure 11 shows the entire database and Figures 12 to 18 shows the implementation of the partial ERDs as shown in the above section.

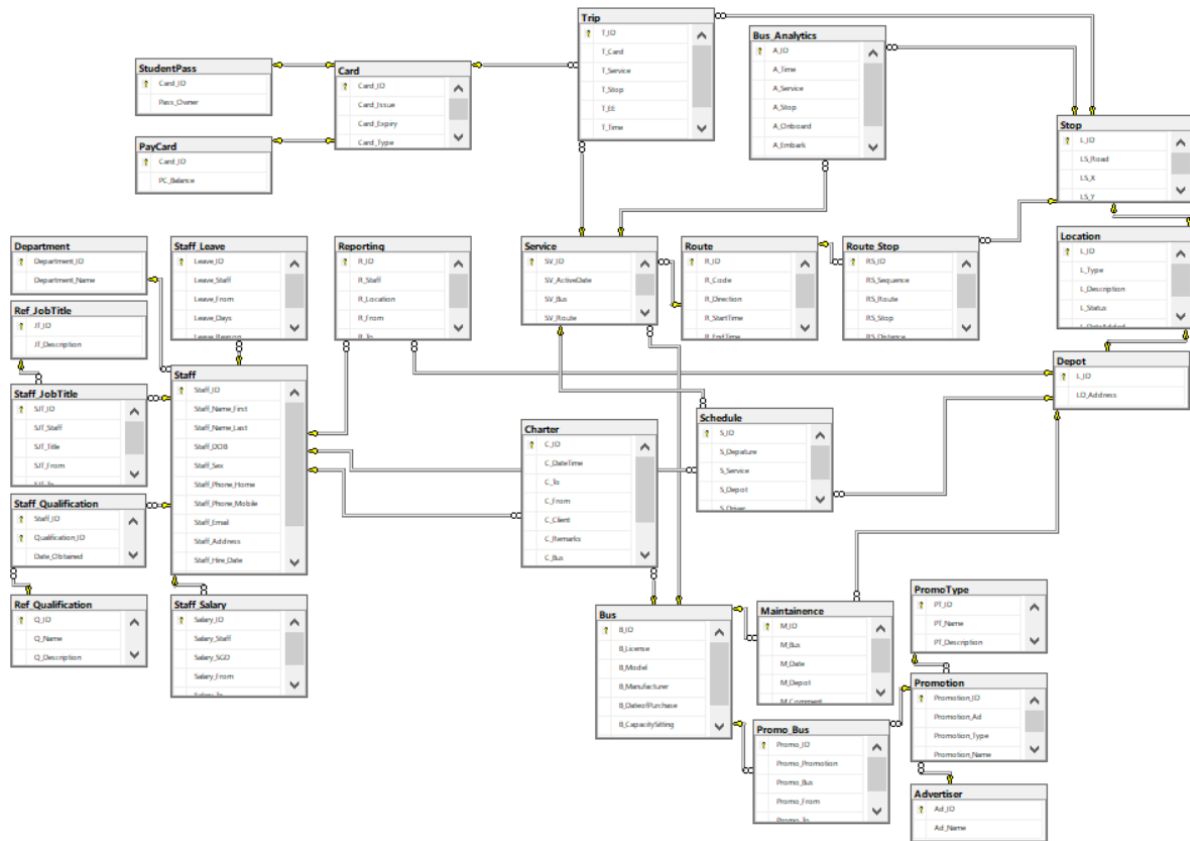


Figure 11: Implemented Full Database Diagram

4.1.1.1 Bus and Promotions Management

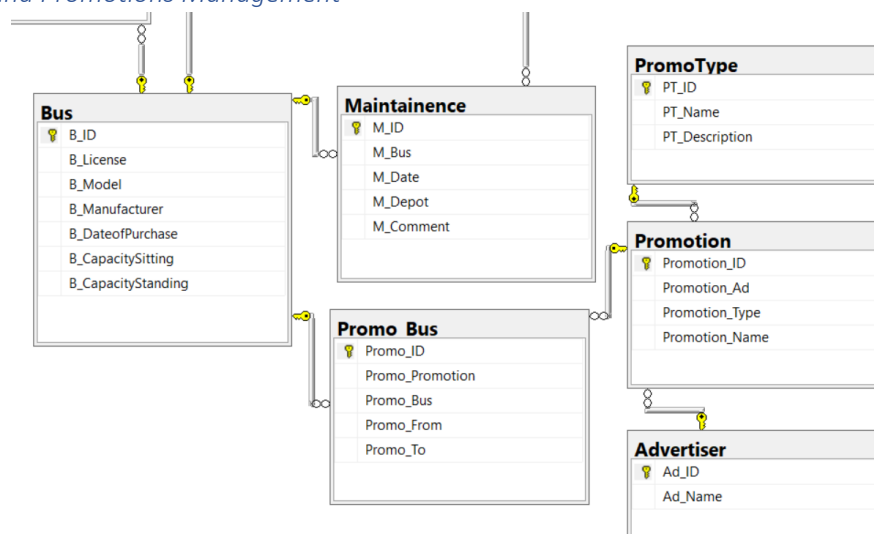


Figure 12: Implemented Bus and Promotions Diagram

4.1.1.2 Human Resources Management

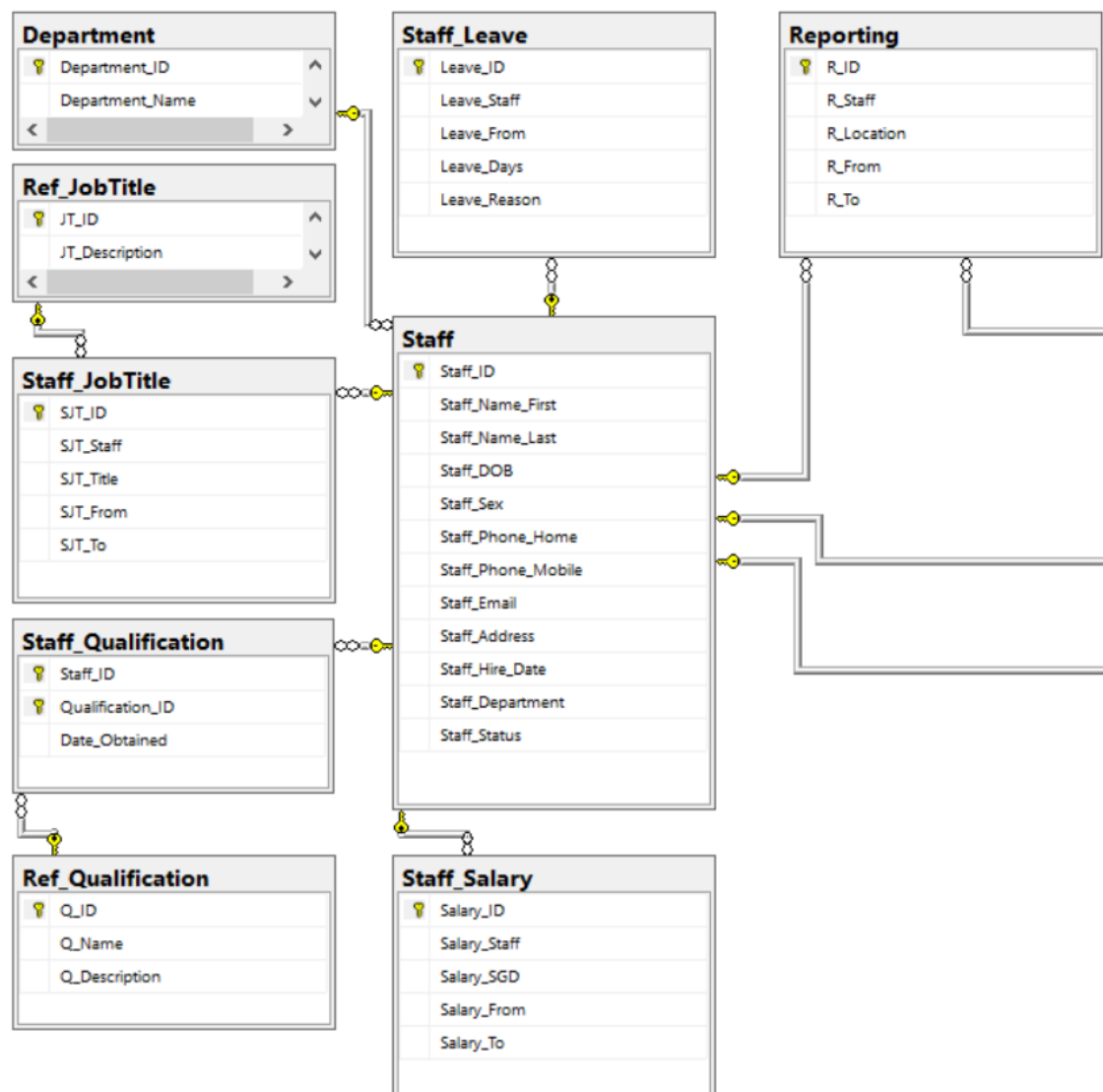


Figure 13: Implemented Human Resource Management Diagram

4.1.1.3 Route Management

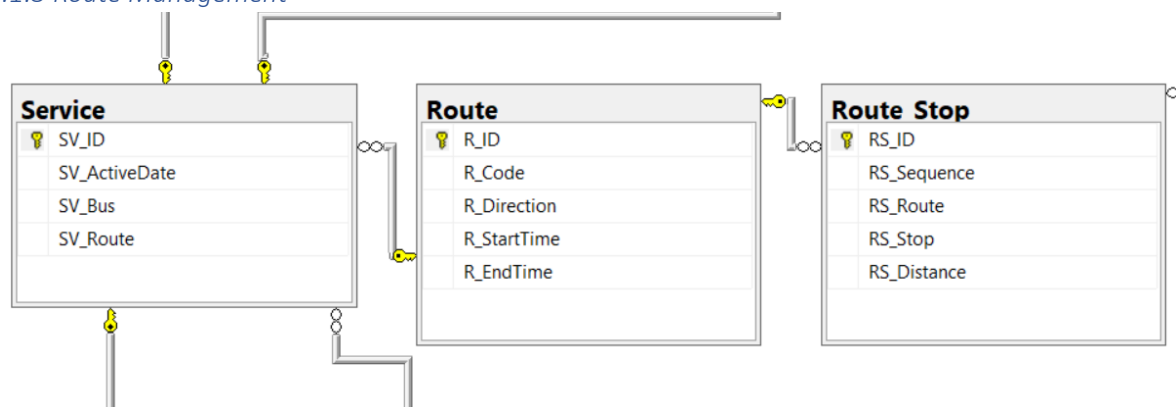


Figure 14: Implemented Route Management Diagram

4.1.1.4 Scheduling

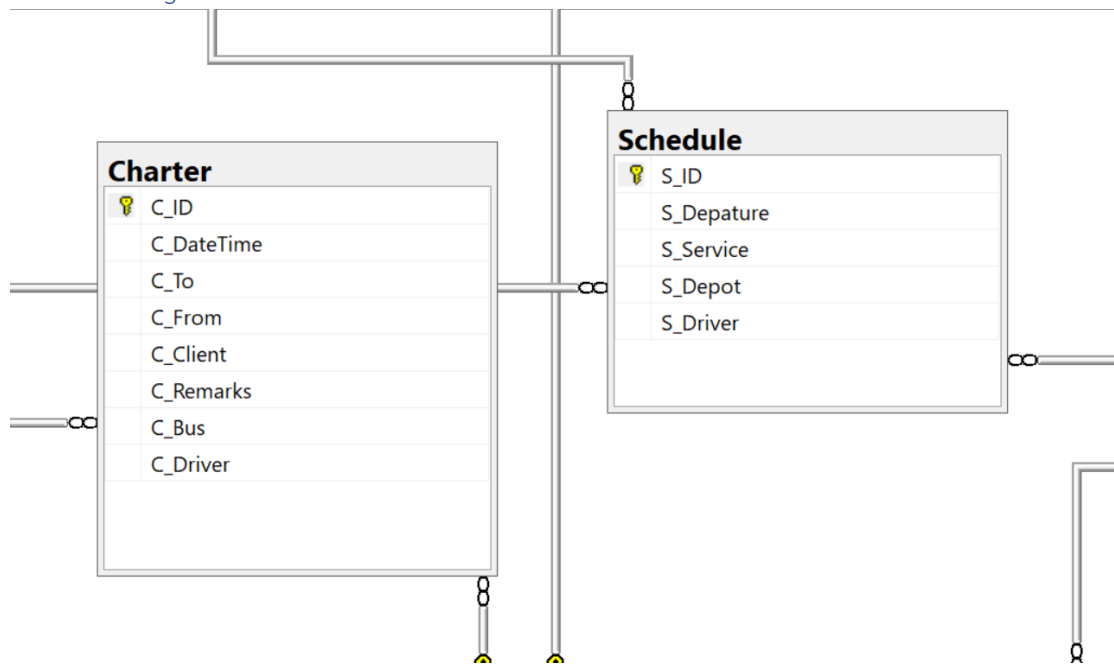


Figure 15: Implemented Scheduling Diagram

4.1.1.5 Location Management

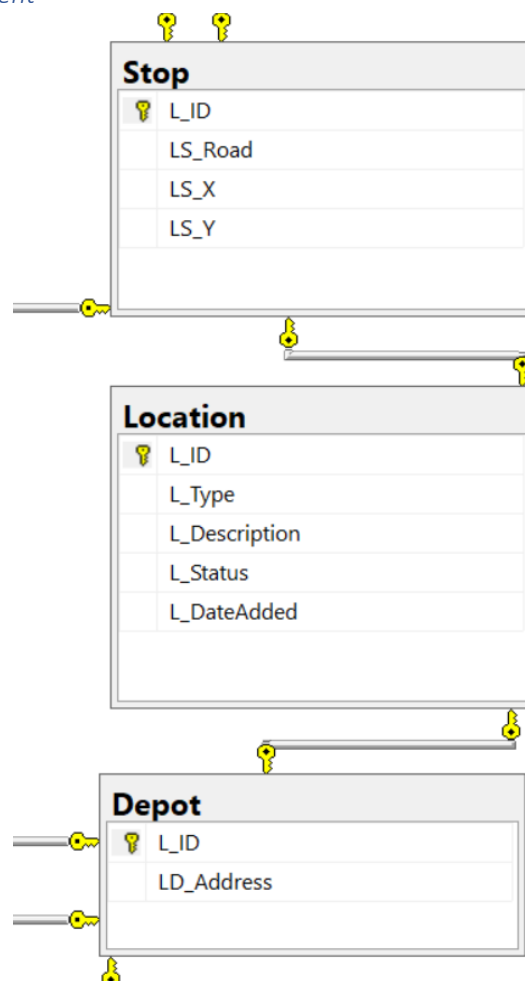


Figure 16: Implemented Location Management Diagram

4.1.1.6 Analytics Collection

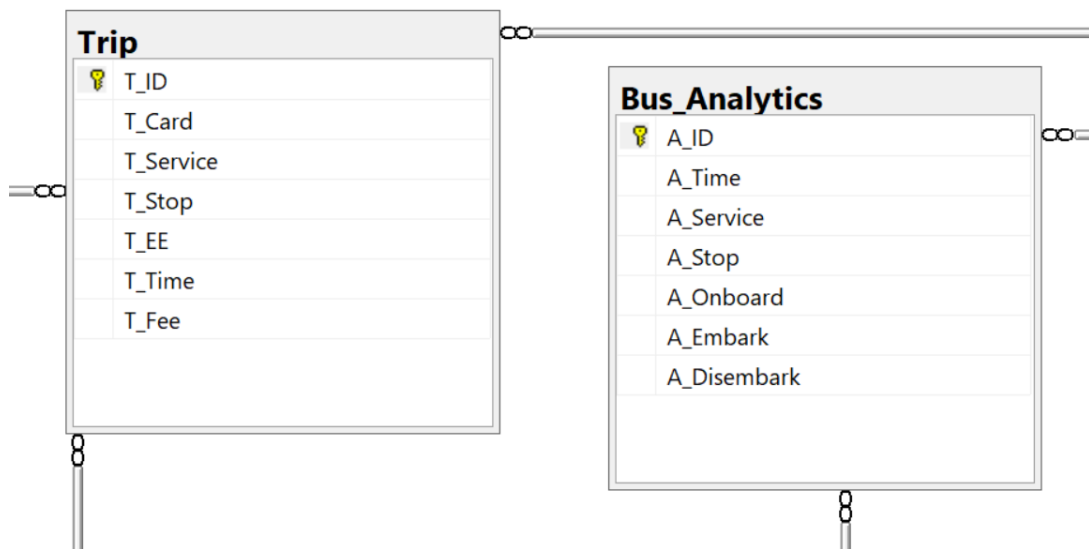


Figure 17: Implemented Analytics Collection Diagram

4.1.1.7 Payment Cards

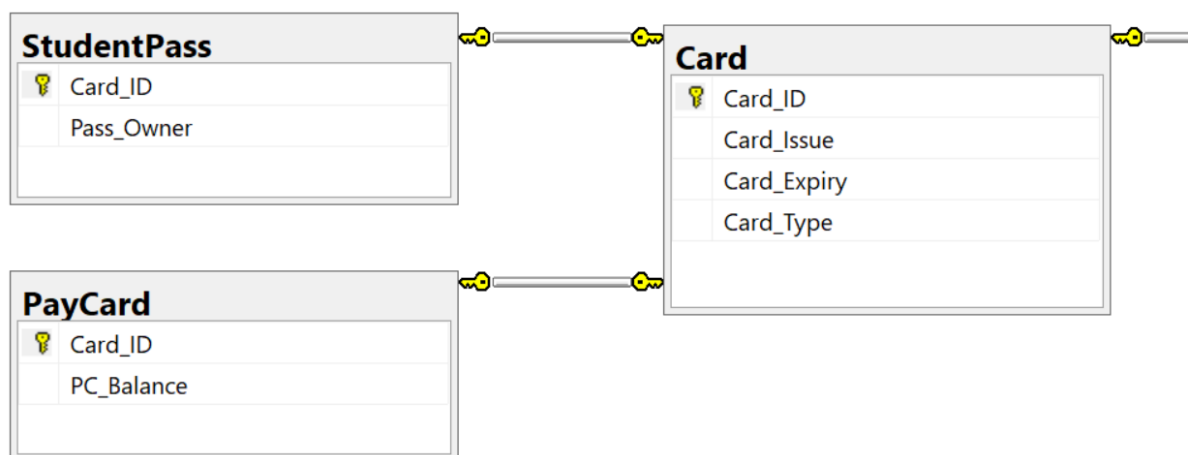




Figure 18: Implemented Payment Cards Diagram

4.1.2 Database Create and Drop Scripts

The file used for creation of the database, tables and relationships is attached below (double click to open attached files). A file to drop all tables is also attached.

- 

1. B1 Bus System
Create.sql



7. B1,2 Bus System
Drop All.sql

4.2 B2: Database Population Script

The following file contains the SQL queries to insert records into the database. The select all query can be used to preview all data in the database after inserts and procedure executions have completed.



4. B2 Bus System
Populate.sql



6. B2,2 Bus System
Select All.sql

4.3 B3: Database View Creation

The following are the outputs of the views implemented in the University Bus System

	Promotion ID	Promotion Type	Advertiser	Promotion Name
1	1	Paper Hangers	Febreeze	Summer Promotion
2	2	Small Interior Ad	Coca Cola	Because you cannot think of anything else to drink
3	3	Large Interior Ad	Pringles	Once you pop, you cant stop
4	4	Typical Exterior Ad	Circles.Life	20GB Mobile data for \$20
5	5	Atypical Exterior Ad	Universal Studios	Better with Friends

Figure 19: Output of Current Promotions View

	Location ID	Description	Current Status	Date Added	Road	Location X Coordinates	Location Y Coordinates
1	1	Building 1	Active	2019-04-30 14:07:38.120	Tempor Road	-4.45200	-73.11045
2	2	Building 2	Active	2019-04-30 14:07:38.120	Mus. Street	-18.30663	55.21410
3	3	Building 3	Active	2019-04-30 14:07:38.120	At St.	-33.70525	-88.56533
4	4	Building 4	Active	2019-04-30 14:07:38.123	Suspendisse Rd.	109.06053	49.47992
5	5	Building 5	Active	2019-04-30 14:07:38.123	A St.	49.60432	22.26814
6	6	Building 6	Active	2019-04-30 14:07:38.123	Eleifend, Avenue	-54.89987	-21.08217
7	7	Building 7	Active	2019-04-30 14:07:38.127	Nec, Av.	60.77428	22.22452
8	8	Building 8	Active	2019-04-30 14:07:38.127	Justo Street	-113.58967	55.92479
9	9	Building 9	Active	2019-04-30 14:07:38.127	Nunc Street	-84.92970	-13.25450
10	10	Building 10	Active	2019-04-30 14:07:38.130	Eget Rd.	65.09241	-89.42535

Figure 20: Output of Bus Stops View

	Location ID	Description	Current Status	Date Added	Location Address
1	11	Northeast Depot	Active	2019-04-30 14:07:38.130	445-1672 Ut Road
2	12	Downtown Depot	Active	2019-04-30 14:07:38.130	#126-1766 Ut Street
3	13	West Depot	Active	2019-04-30 14:07:38.133	689-8346 Ligula Av

Figure 21: Output of Depots View

	First Name	Last Name	Department	Job Title	Date of Birth	Staff Gender	Home Phone	Mobile Phone	Email Address	Salary
1	Allegra	Bennett	Reservations	Mechanic	1987-11-13	Female	68525828	94955172	ipsum@afeugiat.co.uk	3574
2	Maxwell	Reed	Maintenance	Drivers	1984-07-26	Female	69497153	85652903	metus.Aenean.sed@eudolor.edu	7407
3	Gemane	Bishop	Reservations	Receptionist	1992-04-18	Male	69427522	92512517	tellus.Phasellus.elt@aaliquet.org	7014
4	Lois	Evans	Finance	Drivers	1996-01-22	Male	63730640	94336906	elt.Aliquam.auctor@miDuis.edu	6888
5	Jesse	Perkins	Reservations	Scheduler	1982-11-14	Female	61331510	88464863	ut.nisi.a@porttitorerosnec.co.uk	7576
6	Desirae	Paul	Drivers	Accountant	1981-07-21	Male	64380942	80363612	ac.arcu.Nunc@liberoInteger.co.uk	6314
7	Peter	England	Finance	Drivers	1980-06-28	Female	67705927	84082079	ut.molestie@cursus.com	4466
8	Cyrus	Dalton	Reservations	Accountant	1981-03-21	Female	65697597	90908487	non@acmattis.com	7008
9	Elliott	Gamer	Finance	Receptionist	1980-06-19	Male	64613522	99919858	tempor.bibendum@sedliberoProin.co.uk	3062
10	Vera	Klein	Drivers	Route Manager	1994-10-29	Female	67105309	93576764	pede.ac.uma@ami.net	7767
11	Magee	Sellers	Human Resource	Drivers	1991-04-12	Female	68653539	83141042	pede.ac.uma@tibusDonecegestas.org	5067
12	Inez	Gallegos	Human Resource	Receptionist	1983-06-30	Female	64348216	92382775	Quisque@ultricesposuere.ca	4428
13	Walker	Simon	Finance	Accountant	1999-02-07	Female	68086137	92563440	magnis.dis@scelerisqueDuiSuspendisse.edu	3907
14	Rosalyn	Chen	Drivers	Mechanic	1985-03-04	Male	64056815	86769169	magnis.dis.parturiunt@sedorci.edu	7383
15	Pearl	Cantrell	Finance	Receptionist	1992-08-31	Female	62318896	99857076	Praesent@tristiqueque.co.uk	5464
16	Lavinia	Rutledge	Human Resource	Supervisor	1991-10-27	Male	62544997	96593586	at@nisl.edu	7092
17	Rhona	James	Human Resource	Scheduler	1987-07-04	Female	68710049	84439000	penatibus.et@mi.ca	3337

Figure 22: Output of Staff Detail View

4.3.1 Database View Creation Script



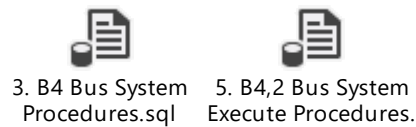
2. B3 Bus System
Views.sql

The following file contains the code for creating SQL views.

4.4 B4: Creating Stored Procedures

4.4.1 Bus System Procedure Creation and Execution Scripts

The file 'B4' contains the scripts for creating all stored procedure in the database. The file 'B4,2' contains the scripts to execute the stored procedures to generate and input new records into their respective tables.



4.4.2 'newstudentpass' Procedure

This procedure is called to create a new student bus pass registered to an owner id in the system. The procedure is necessary to implement the supertype/subtype relationship between 'Card' and 'StudentPass' tables as illustrated in the EERD. The procedure creates a new Card record, with the appropriate expiry date, then retrieves the ID of the newly generated record then enters another record into the 'StudentPass' table with the same ID to maintain relationship integrity between supertype and subtype tables.

This procedure involves similar commands as other procedures not shown in the report ('newpaycard', 'newdepot' and 'newstop').

```
CREATE PROCEDURE dbo.newstudentpass --Creates new procedure
    @ownerID          nvarchar(20) = NULL, -- creates enterable variables and datatypes
    @DATE             datetime = NULL,
    @EXPIRY           datetime = NULL,
    @ID               int = NULL
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        SET NOCOUNT ON
        SET @DATE = GETDATE();
        SET @EXPIRY = DATEADD(month, 3, @DATE) --calculates a 3 month expiry date from creation date
        INSERT INTO dbo.Card -- inserts record into Card, a supertype table
        (
            Card_Issue,
            Card_Expiry,
            Card_Type
        )
        VALUES
        (
            @DATE,
            @EXPIRY,
            1
        );
        SET @ID = SCOPE_IDENTITY(); -- takes the ID value of last inserted record
        INSERT INTO dbo.StudentPass -- inserts record into StudentPass, a subtype table with same primary key
        (
            Card_ID,
            Pass_Owner
        )
        VALUES
        (
            @ID,
            @ownerID
        );
        COMMIT TRANSACTION -- commits the transaction
    END TRY
    BEGIN CATCH -- begins if commit transaction fails
        SELECT ERROR_MESSAGE() 'Error encountered. Transaction not committed'; --selects error message
        ROLLBACK TRANSACTION
    END CATCH
END
GO
```


4.4.1 'getrecord' Procedure

This procedure is used to retrieve the trip records on a specific bus card. The procedure requires a card ID as input and generates a table of the card holder's trip record including the route taken, route direction, time, bus stop name and the bus fare.

```
CREATE PROCEDURE dbo.getrecord -- gets travel records of a particular card
    @card int = NULL
AS
BEGIN
    SELECT
        T_ID AS 'Trip ID',
        R_Code AS 'Route Code',
        CASE R_Direction WHEN 1 THEN 'Forward' WHEN 0 THEN 'Backwards' END AS 'Direction',
        T_Stop AS 'Logged Stop',
        CASE T_EE WHEN 1 THEN 'Entry' WHEN 0 THEN 'Exit' END AS 'Entry or Exit',
        T_Time AS 'Date and Time',
        T_Fee AS 'Billed'
    FROM ((Trip INNER JOIN Service ON T_Service = SV_ID)
    INNER JOIN Route ON SV_Route = R_ID) WHERE [T_Card] = @card;
END
GO
```

4.4.3 'getfee' Procedure

This procedure is used to calculate the bus fare of a trip from one bus stop to another. The procedure is not intended to be called standalone but is used by the 'tripregister' procedure, which inputs the starting and ending points of a trip and receives a fee as calculated by the predefined formula.

The formula for calculating the bus fare is: $\$(0.2 * x^{1/4})$

```
CREATE PROCEDURE dbo.getfee -- this procedure is called by 'tripregister' to calculate the fee for payment cards
    @routeF int = NULL,
    @startF int = NULL,
    @endF int = NULL,
    @fee decimal (38, 5) = NULL OUTPUT,
    @distA decimal (38, 5) = NULL,
    @distB decimal (38, 5) = NULL,
    @distance decimal (38, 5) = NULL
AS
BEGIN
    SET @distA = ( --takes the distance value of embarking bus stop
        SELECT [RS_Distance]
        FROM Route_Stop
        WHERE
            [RS_Route] = @routeF AND
            [RS_Stop] = @startF
    )
    SET @distB = ( -- takes the distance value of disembarking bus stop
        SELECT [RS_Distance]
        FROM Route_Stop
        WHERE
            [RS_Route] = @routeF AND
            [RS_Stop] = @endF
    )
    SET @distance = (@distB - @distA); -- calculates total distance travelled
    SET @fee = (SELECT ABS(0.2 + POWER(@distance, 0.25))) -- calculates fare according to preset formula
    SELECT @fee AS 'Fare'; -- returns calculated fare
END
GO
```

4.4.4 'tripregister' Procedure

This procedure is used to generate a trip record when a passenger taps in or out on a bus. It determines whether the record needs to be an embark or disembark entry based on the previous state of the card/service data pair on the Trip table. If a previous record where the passenger boarded the bus, the procedure creates a disembark record, calculates fare if a payment card is used then enters the record. If there is no previous embark record, an embark record is created with the provided parameters that include card ID, bus stop and the service.

```
CREATE PROCEDURE dbo.tripregister --procedure is called when a user taps in a bus
    @card int = NULL,
    @stop int = NULL,
    @service int = NULL,
    @start int = NULL,
    @route int = NULL,
    @datetime datetime = NULL
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        SET NOCOUNT ON
        SET @datetime = GETDATE();

        IF EXISTS ( --check if last tap was on same the service from with the card and was a trip start
            SELECT TOP 1 * FROM Trip WHERE
                (T_Card = @card) AND
                (T_Service = @service) AND
                (T_EE = 1)
            ORDER BY T_Time DESC)
        BEGIN
            SET @start = (SELECT TOP 1 T_Stop FROM Trip WHERE --grabs embarking bus stop
                (T_Card = @card) AND
                (T_Service = @service) AND
                (T_EE = 1)
            ORDER BY T_Time DESC)
            SET @route = (SELECT SV_Route FROM Service where SV_ID = @service) -- selects travelled route
            DECLARE @fee decimal (38,2) -- creates a fee variable
            IF ((SELECT Card_Type FROM Card WHERE Card_ID = @card) = 1) --checks if card is student card
            BEGIN
                SET @fee = 0 --if true, fare is set to 0
            END
            ELSE
            BEGIN
                EXEC dbo.getfee --if false, 'getfee' procedure is called to calculate fare
                    @route,
                    @start,
                    @stop,
                    @fee OUTPUT --stores output
                DECLARE @oldbalance decimal (38, 3) --creates an temporary balance variable
                SET @oldbalance = (SELECT [PC_Balance] FROM PayCard WHERE ([Card_ID] = @card)) --retrieves original balance of payment card
                DECLARE @newbalance decimal (38, 3 -- creates a new balance variable
                SET @newbalance = (@oldbalance - @fee)
                UPDATE PayCard SET [PC_Balance] = @newbalance WHERE Card_ID = @card; --updates the balance of the payment card with new value
            END
            INSERT INTO dbo.Trip ( --inserts trip end record
                T_Card,
                T_Service,
                T_Stop,
                T_EE,
                T_Time,
                T_Fee
            )
            VALUES (
                @card,
                @service,
                @stop,
                0,
                @datetime,
                @fee
            );
        END
        ELSE
        BEGIN
            INSERT INTO dbo.Trip ( --if last tap was not a trip start record, creates a trip start record
                T_Card,
                T_Service,
                T_Stop,
                T_EE,
                T_Time
            )
            VALUES (
                @card,
                @service,
                @stop,
                1,
                @datetime
            );
        END
    END
    COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        SELECT ERROR_MESSAGE() 'Error Message';
        ROLLBACK TRANSACTION
    END CATCH
END
GO
```

4.4.5 'loganalytics' Procedure

The following is the 'loganalytics' procedure is intended to be called by the bus's onboard embedded system when it detects that the bus has left a bus stop. It calculates the number of people embarked and disembarked at the previous stop, calculates the total number of people onboard and stores all the values in the *Bus_Analytics* table.

```
CREATE PROCEDURE dbo.loganalytics -- this procedure is called when onboard GPS detects that a bus has left a stop
    @service      int = NULL,
    @stop         int = NULL,
    @onboard      smallint = NULL,
    @embark       smallint = NULL,
    @disembark    smallint = NULL,
    @oldonboard   smallint = 0
AS
BEGIN
    BEGIN TRANSACTION
        BEGIN TRY
            SET NOCOUNT ON

            SET @embark = (SELECT COUNT (T_ID) FROM Trip
                WHERE T_stop = @stop AND T_EE = 1
                AND (DATEDIFF (n, T_Time, GETDATE()) <3))
            --calculates the number of embarking passengers

            SET @disembark = (SELECT COUNT (T_ID) FROM Trip
                WHERE T_stop = @stop AND T_EE = 0
                AND (DATEDIFF (n, T_Time, GETDATE()) <3))
            --calculates the number of disembarking passengers

            IF EXISTS (SELECT TOP 1 A_Onboard FROM Bus_Analytics
                WHERE A_Service = @service
                AND CONVERT(Date, A_Time) = CONVERT(Date, GETDATE()))
            --checks if the bus has previous analytics entry from same day

                BEGIN
                    -- if true, take the last recorded passengers onboard number
                    SET @oldonboard = (SELECT TOP 1 A_Onboard FROM Bus_Analytics
                        WHERE A_Service = @service
                        AND CONVERT(Date, A_Time) = CONVERT(Date, GETDATE()) ORDER BY A_Time DESC)
                END

            SET @onboard = (SELECT (@oldonboard + (@embark - @disembark)))
            -- calculates the new number of onboard passengers

            INSERT INTO dbo.Bus_Analytics --inputs a new analytics record
            (
                A_Service,
                A_Stop,
                A_Onboard,
                A_Embark,
                A_Disembark
            )
            VALUES
            (
                @service,
                @stop,
                @onboard,
                @embark,
                @disembark
            );
        COMMIT TRANSACTION
        END TRY
        BEGIN CATCH
            SELECT ERROR_MESSAGE() 'Error Message';
            ROLLBACK TRANSACTION
        END CATCH
    END
GO
```

5.0 Task C: Microsoft Power BI

Power BI is a business analytics tool developed by Microsoft (Microsoft, 2019). It combines several features already offered on separate products that utilise the Microsoft SQL Server such as data visualisation and report generation similar to the SQL Server Reporting Services (SSRS) while packaging them in a simple-to-use and attractive looking interface for users to create their own reports and dashboards. The application can be installed via the Microsoft Store on PCs running Windows 10.

In this project, the City Bus database created in the previous section will be imported into Power BI to demonstrate the capabilities of the application.

5.1 Data Integration

Data integration in Power BI allows the users to import data from various sources, such as databases, cloud datasets files and even data from webpages. The application contains the necessary data functions to transform data from supported sources into a model usable to generate visualisations such as charts and graphs in the application. This data can then be used for the other functions that Power BI provides.

The following screen is shown when Power BI has started up. Click on **Get Data** to start the data integration from a source 9 (Figure 23).

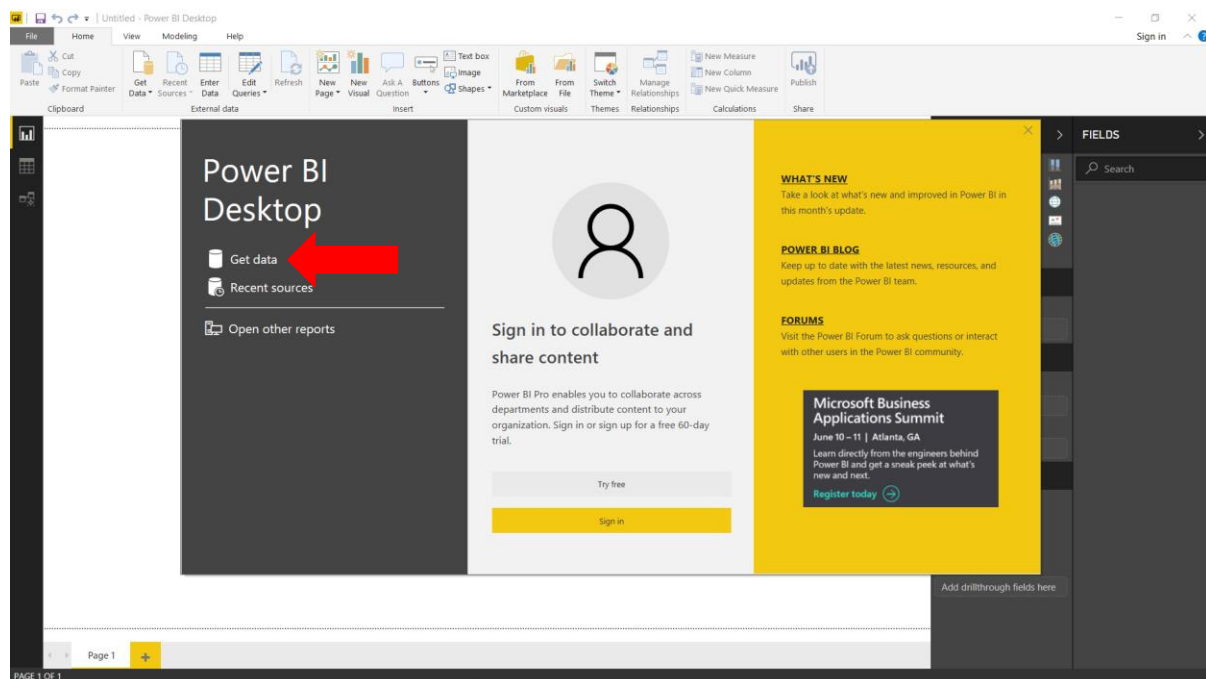


Figure 23: Power BI Start-up Screen

This will bring up the Get Data window that allows you to select from all the data sources that Power BI supports. In this case, the SQL Server Database is chosen so that the Uni_Bus database created earlier can be imported (Figure 24).

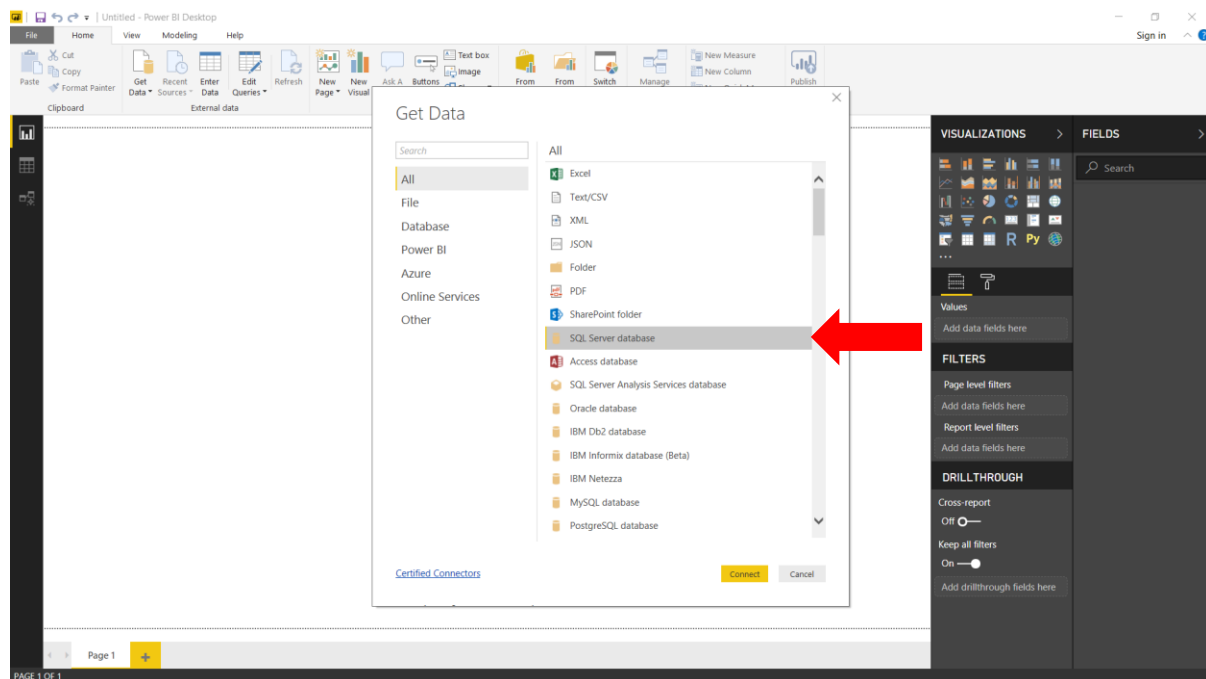


Figure 24: Power BI Get Data Screen

This brings up the window to enter the server name and the database to connect to a SQL Server database. The server name is usually formatted as “[PC name] \ [server name]”. Optionally, the server’s port number can also be entered as “[server name] \ [port number]”. The connection mode can be set to **import** to create a copy of the database to generate reports on. This has the advantage of lowering load on the running database server as the queries are done on the imported database. **DirectQuery** will allow users to generate reports with live data. The database to connect to can be specified (Figure 25).

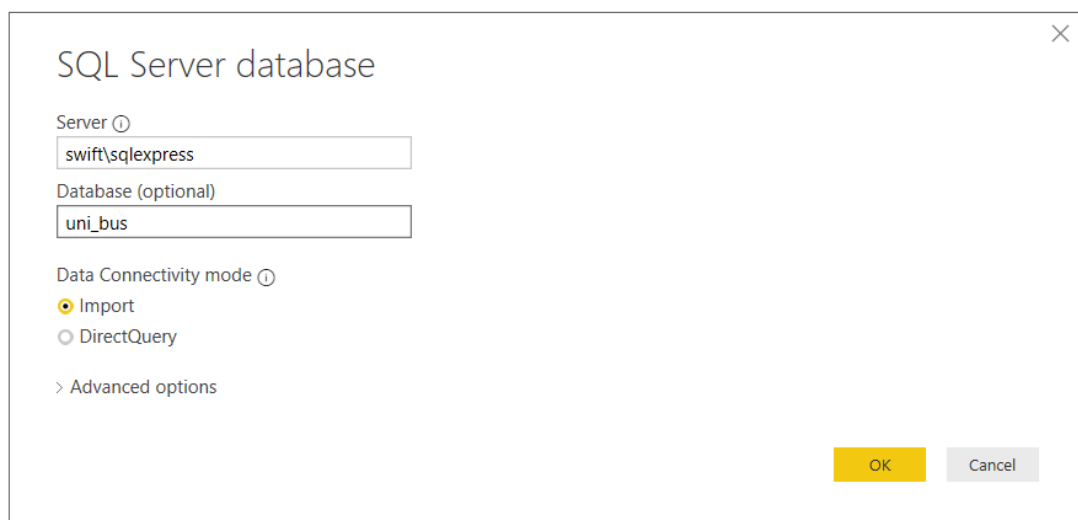


Figure 25: Power BI SQL Server Database Connection Screen

After entering a server to connect to, access credentials need to be entered for connection. On SQL Servers, this can be done using your current Windows credentials (the Windows account logged into the PC), preconfigured database username and password or using a linked Microsoft Account (Figure 26).

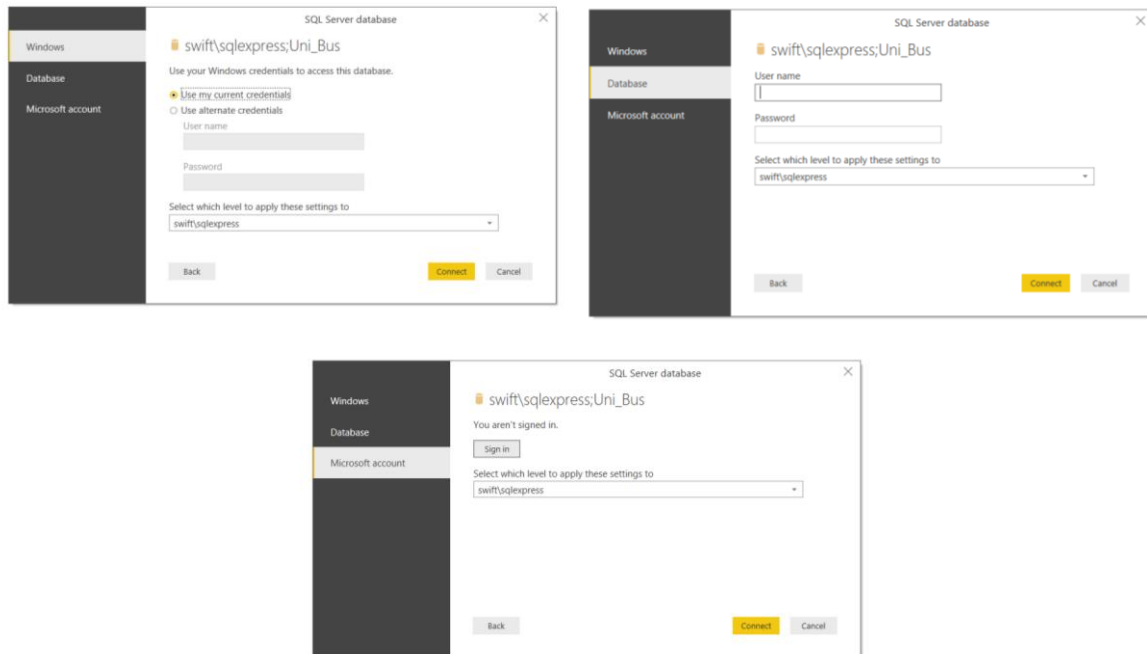


Figure 26: Power BI SQL Server Credentials Screen Options

After connection, the contents of the database can be previewed and selected to be imported into the Power BI page. After choosing and loading the desired tables from the connected database, the data integration process is completed (Figure 27).

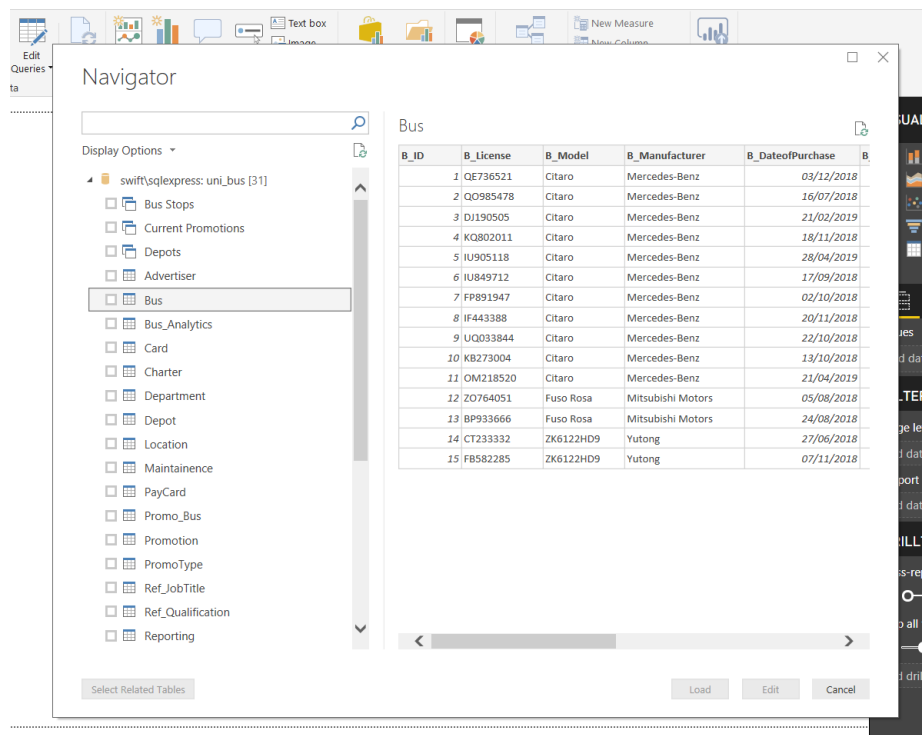


Figure 27: Power BI Data Import from SQL Server

5.2 Data Processing and Visualisation

Microsoft Power BI features powerful visualisation tools to turn imported data into easily digestible charts and graphs.

5.2.1 Query Editor

Power BI contains a query editor that allows the user to transform the data as necessary. This includes removing or adding columns, converting data types, transforming data into different types (such as extracting the month from SQL datetime format) and removing duplicates and errors. The query editor can also be used to fix errors that were entered during the automatic integration of data from non-standard sources such as tables on webpages (Figure 28).

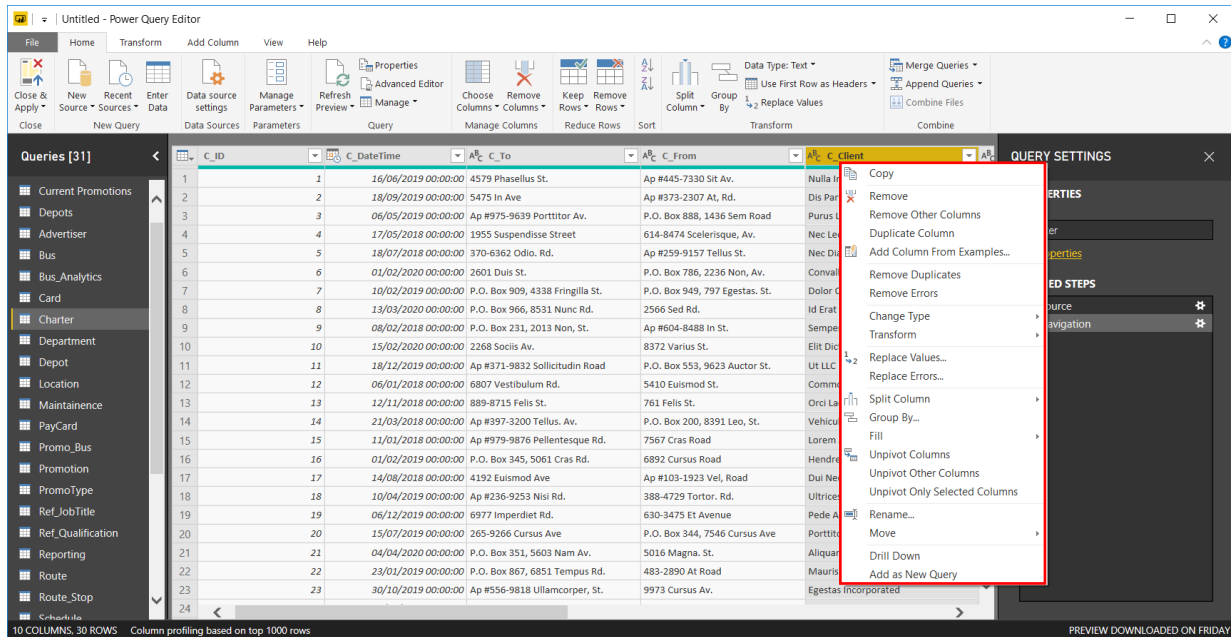


Figure 28: Power BI Query Editor and Available Functions (in red highlight)

5.2.2 Creating a Graph/Chart

5.2.2.1 Using the Visualisation Menu

Users can generate visualisations using the panel on the right. It contains the controls to choose the type of visualisation, style and formatting of visualisation, the fields to choose which columns of the chosen data act as the legend, details, values and tooltips on the visualisation and the panel to choose sets of data to be entered in the visualisation. Each type of visualisation has different fields for values.

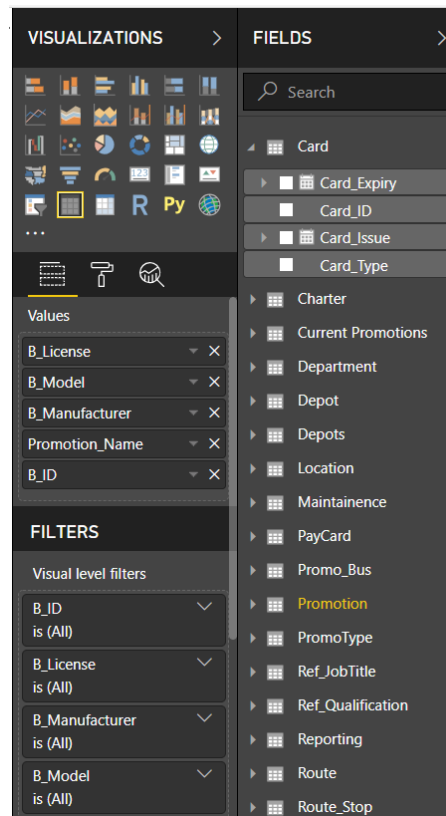


Figure 29: Visualisations Panel of Power BI

To create a visualisation, click on one of the many visualisations that are available in the panel (Figure 29). This creates an empty visualisation and the data to be represented in the horizontal and vertical axis, legend and the tooltip can be entered to populate the chart. The data entered can be filtered using various attribute values in the filter section to limit the data visualised.

5.2.2.2 Asking a Question

Power BI has the function to generate graphs by only entering a question in plain text. The application parses the question for keywords and attempts to generate a chart with relevant data from tables that it can recognise in the imported data source.

To activate this function, the user can either click on the **'Ask A Question'** button (Figure 30) on the toolbar (Figure 29) or double click on any empty space on a Power BI page.

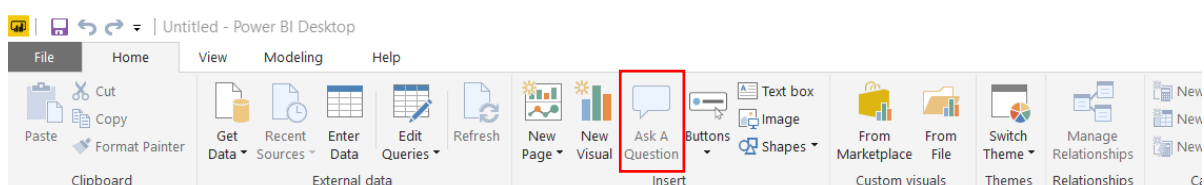


Figure 30: 'Ask A Question' Button on Power BI Toolbar

This brings up a question box that the user could enter a query in (Figure 31). The query should have identifiable keywords such as the same words used in the table and column names in the dataset or the names that the user gave to the data in the query editor.

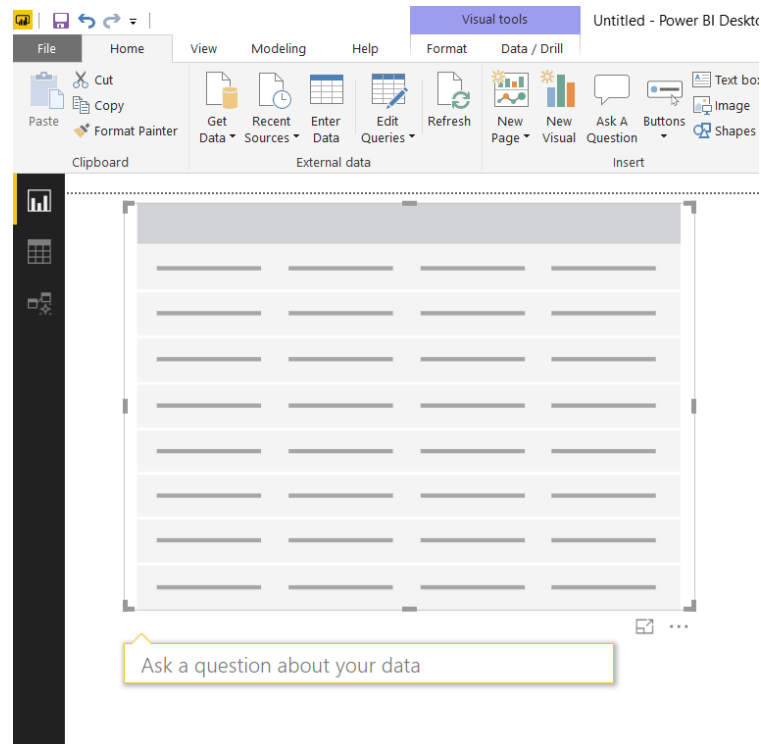


Figure 31: Query Field for Questions in Power BI

Power BI will attempt to guess what the user is looking for and generate visualisation of data requested. This can be in the form of a table (Figure 32) or a chart (Figure 33) depending on the nature of the query.

 The image shows the Power BI Desktop application window with a table visual. The table has five columns: B_License, B_Model, B_Manufacturer, Promotion_Name, and B_ID. The data is sorted by B_ID in ascending order. Below the table is a text input field with the query text 'Which were the buses with promotions sorted by B ID'.

B_License	B_Model	B_Manufacturer	Promotion_Name	B_ID
QE736521	Citro	Mercedes-Benz	Summer Promotion	1
QO985478	Citro	Mercedes-Benz	Because you cannot think of anything else to drink	2
QO985478	Citro	Mercedes-Benz	Summer Promotion	2
KQ802011	Citro	Mercedes-Benz	20GB Mobile data for \$20	4
IU905118	Citro	Mercedes-Benz	Better with Friends	5
IU905118	Citro	Mercedes-Benz	Once you pop, you cant stop	5
IU849712	Citro	Mercedes-Benz	Once you pop, you cant stop	6
IU849712	Citro	Mercedes-Benz	Summer Promotion	6
FP891947	Citro	Mercedes-Benz	Summer Promotion	7
IF443388	Citro	Mercedes-Benz	Because you cannot think of anything else to drink	8
IF443388	Citro	Mercedes-Benz	Once you pop, you cant stop	8
UQ033844	Citro	Mercedes-Benz	Better with Friends	9
KB273004	Citro	Mercedes-Benz	20GB Mobile data for \$20	10
KB273004	Citro	Mercedes-Benz	Better with Friends	10
KB273004	Citro	Mercedes-Benz	Once you pop, you cant stop	10
OM218520	Citro	Mercedes-Benz	Because you cannot think of anything else to drink	11
ZO764051	Fuso Rosa	Mitsubishi Motors	Because you cannot think of anything else to drink	12
ZO764051	Fuso Rosa	Mitsubishi Motors	Better with Friends	12
CT233332	ZK6122HD9	Yutong	20GB Mobile data for \$20	14
FB582285	ZK6122HD9	Yutong	Once you pop, you cant stop	15

Figure 32: Generated Result In Table Form From Question in Power BI

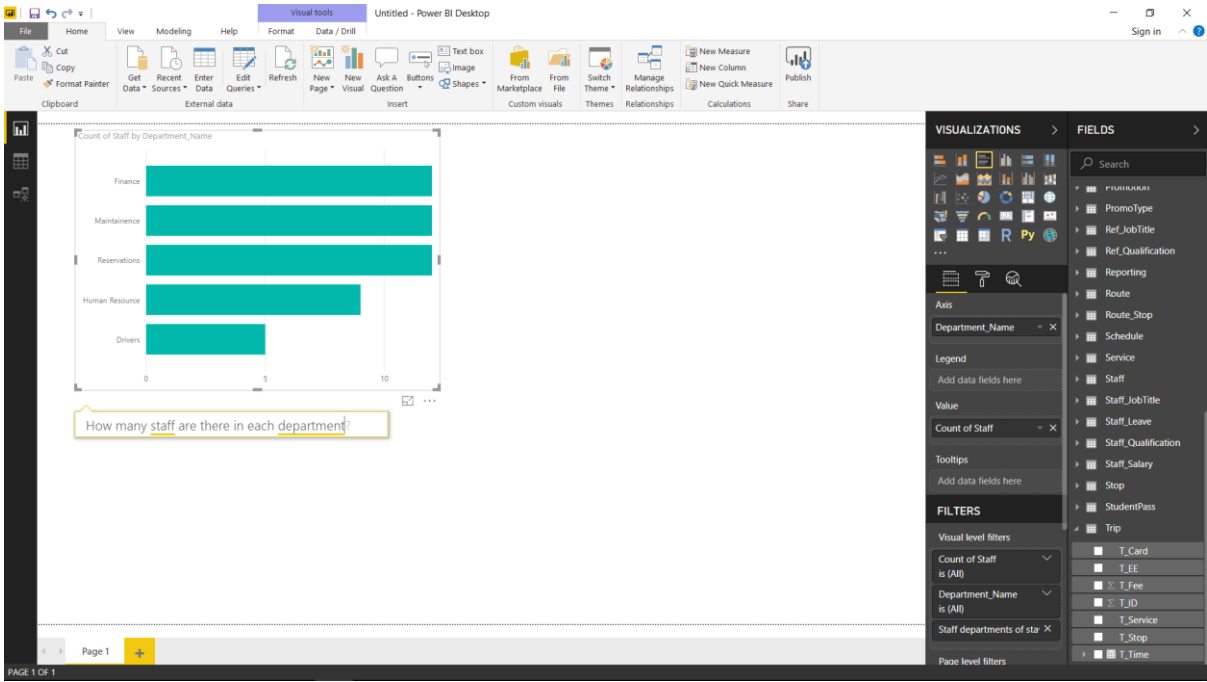


Figure 33: Generated Result in Bar Chart Form From Question in Power BI

After a chart has been generated, the same tools for creating a visualisation manually can be used to adjust the automatically generated chart's parameters such as the type of chart, data displayed, data filters, etc, to the user's requirements (Figure 34).

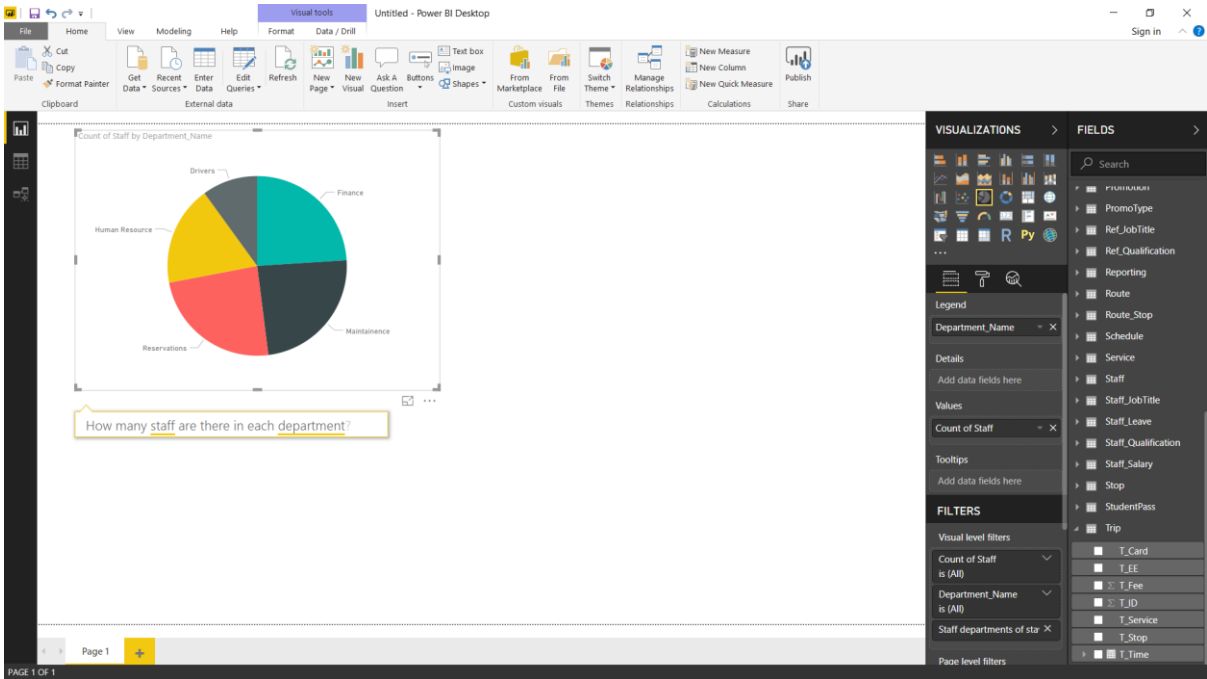


Figure 34: Previously Generated Chart Changed to Pie Chart

5.2.3 Build Report Dashboards

Multiple of these charts can be generated and placed on the Power BI page as desired by the user. Reports can consist of multiple charts that can be resized and arranged as the user desires. A report can also contain multiple of these pages of charts (Figure 35).

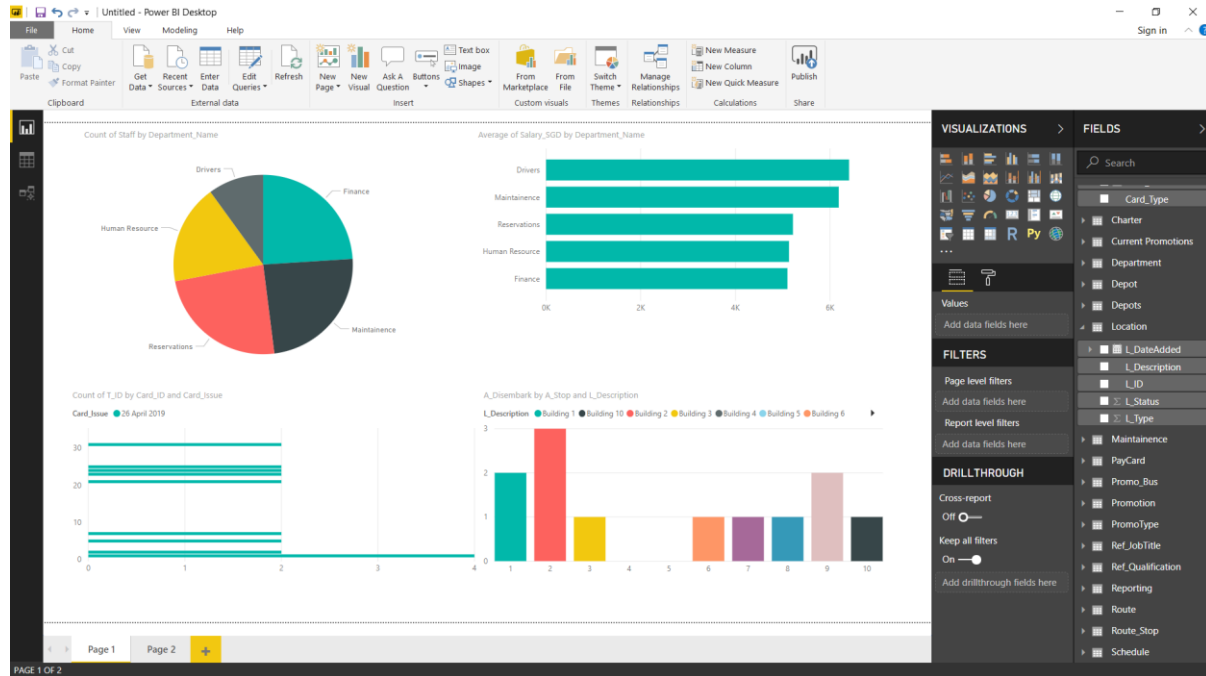


Figure 35: Power BI Report File

5.3 Sharing the report

After the report has been generated with the desired data visualisations, it can be saved and shared to various sources via multiple methods (Figure 36).

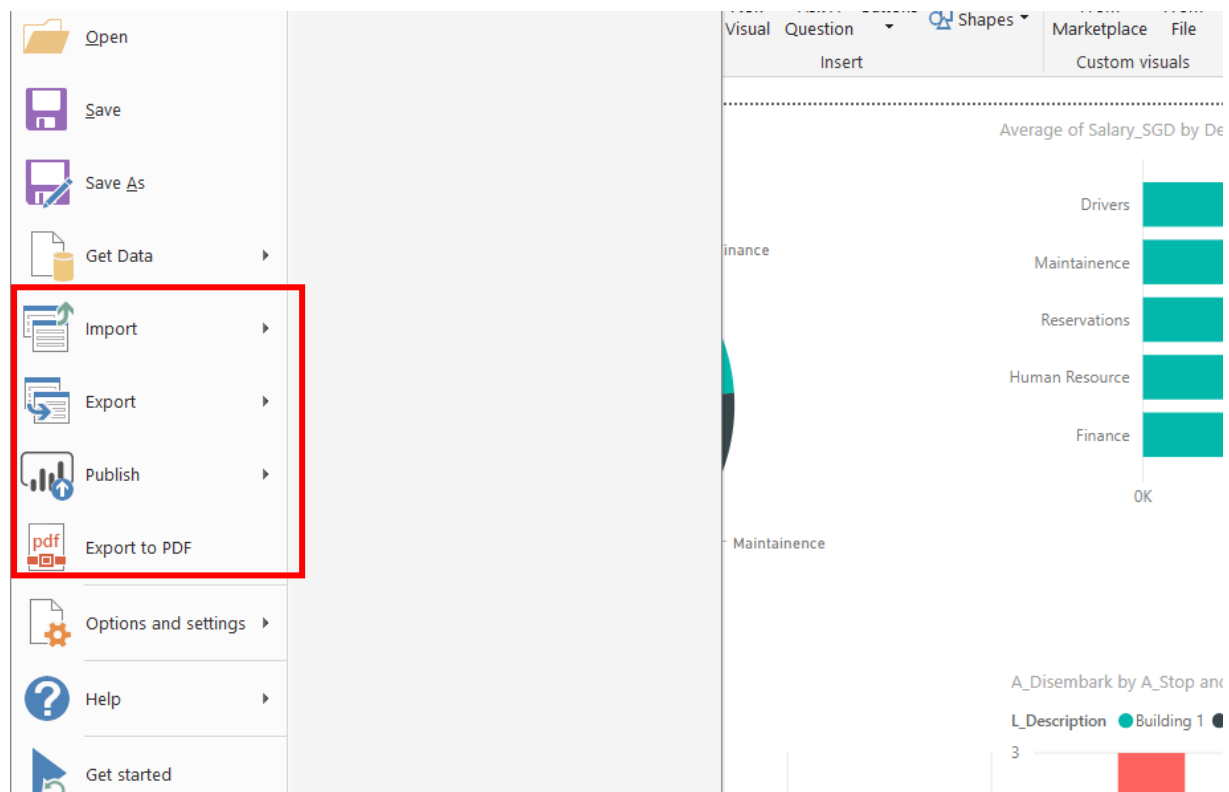


Figure 36: Exporting Options of Power BI

If the organisation has access to Power BI cloud services, the user can log in with their Power BI account and share their report on app.powerbi.com. This will allow the user to share their report with other staff in the organisation for collaboration with easy access from the web browser.

The Power BI file can also be saved or exported in multiple ways. A Power BI file (.pbix) can be saved, which can be opened by the Power BI desktop application to edit the reports and their parameters. This report can be shared like a normal file to be opened in the application, which is useful for organisations without Power BI cloud.

A PDF document can also be generated of the reports, which is useful for creating a snapshot of the report in its current form and for printing.

A template of the report can also be exported, which can be used to create similar, new report on different PCs or to create a new report of the same style.

References

Microsoft, 2019. *Power BI | Interactive Data Visualisation BI Tools*. [Online]
Available at: <https://powerbi.microsoft.com/en-us/>
[Accessed 12 April 2019].