

‘Bits and Bobs’

Data Analysis Report

Table of Content

Table of Content	2
Problem Overview	3
Executive Summary	4
Data Model/Data Load Process	5
Start schema Datamart design - Bits and Bobs' sales	5
ETL Process	6
Extract	7
Transform	9
Load	18
Output of the analysis	22
1. Net Revenue per month/year	22
2. Total transactions carried out per month/year	26
3. Net Revenue by Office	29
Summary of recent numbers and some trend analysis	30
Net income per month/ year	30
Total transactions count per year / month	31
Item count by each Item	32
Ranking of Items based on their sales count by office	33
Worst performing Items as a whole	37
Best sales person	38
Predictive analysis	42
Income forecast for year 2023 by each office	42
Sales recommendation	44
List of Figures	46
References	48

Executive Summary

Recent research conducted by BitsAndBobs revealed that some products have seen a decline in sales across the nation in recent years, and that we are currently losing money.

The company's CEO has expressed a desire to consolidate the company's Australian offices. Reduce expenses as a result to ensure the survival of the business as a whole. It was asked that rankings of offices based on analysis be provided in order to help with office decision-making. He nevertheless had to forecast sales for each office for the following 12 months. The CEO also asked for a sales item analysis. Additionally, he required a list of the top salespeople for the previous year across all branches—the "best of the best."

In order to build a datamart with the right model and address the issues the CEO raised, we will analyze recent sales data in our report in response to his request.

They may be able to increase their revenue by gaining important and useful insights into sales as a result. Along with knowing their buying habits and satisfying them with the services the company provides.

Data modeling should be the first step in the process to address the aforementioned issue. The construction of a new data warehouse is one potential fix. A specific type of database called a "data warehouse" compiles copies of transaction data from various source systems and makes them accessible for analytical use.

However, this data warehouse should only be focused on sales given the problem this company is currently facing. Therefore, using a datamart that is solely focused on sales and that enables end users to access the results immediately is effective. Therefore, we made the decision to use a star datamart design to model the supplied sales data.

The next action was taken to evaluate the effectiveness of the branches. What traits describe an office that performs the best? The answer is to be consistent. If an office consistently displays positive performance, it will be regarded as well-performed. In order to examine this consistency, we must decide which metrics are crucial.

- How much revenue was made by each office per year/month?
- How many transactions were performed by each office per month/year?
- Total revenue

The company can establish baselines of consistency for each of its salespeople by looking at the aforementioned metrics. In addition, ranking the offices according to the aforementioned metric will be the best way to assess the performance of the offices.

It will be easier to evaluate performance data and address the problems with the worst-performing office once those rankings have been established. The operations need to be restarted or revised if these offices are to become profitable businesses.

Planning for long-term growth is made simpler when a thorough understanding of sales activity over a specific time period is attained. Additionally, it gives the opportunity to compare an individual's

performance to that of the branch or business as a whole. Charts and graphs aid in the visualization of data.

They provide a quick way to compare and contrast many different pieces of information at once. The peaks and valleys show their consistency over the course of a month. In addition to the aforementioned metrics, customer and item analysis has been finished. Through this kind of analysis, the business will be able to determine which product will bring in the most money, enabling them to invest heavily in it. Prediction of revenue for the following 12 months is the most valuable insight a business can have. This allows us to identify and correct any problematic sales patterns.

Finally, the company will be able to learn insightful information from this report and receive suggestions on how to improve their sales activity. With the help of the analysis report, the company can determine monthly or yearly revenue targets that each salesperson can work toward. Consequently, each office in Australia can perform well. A business may also set sales quotas, which are transactional goals that need to be achieved by the end of a certain period of time. The salesperson typically gets a performance bonus when quotas are met. Along with item analysis, the business can provide special discounts for low-profit items to increase sales.

Data Model/Data Load Process

Start schema Datamart design - Bits and Bobs' sales

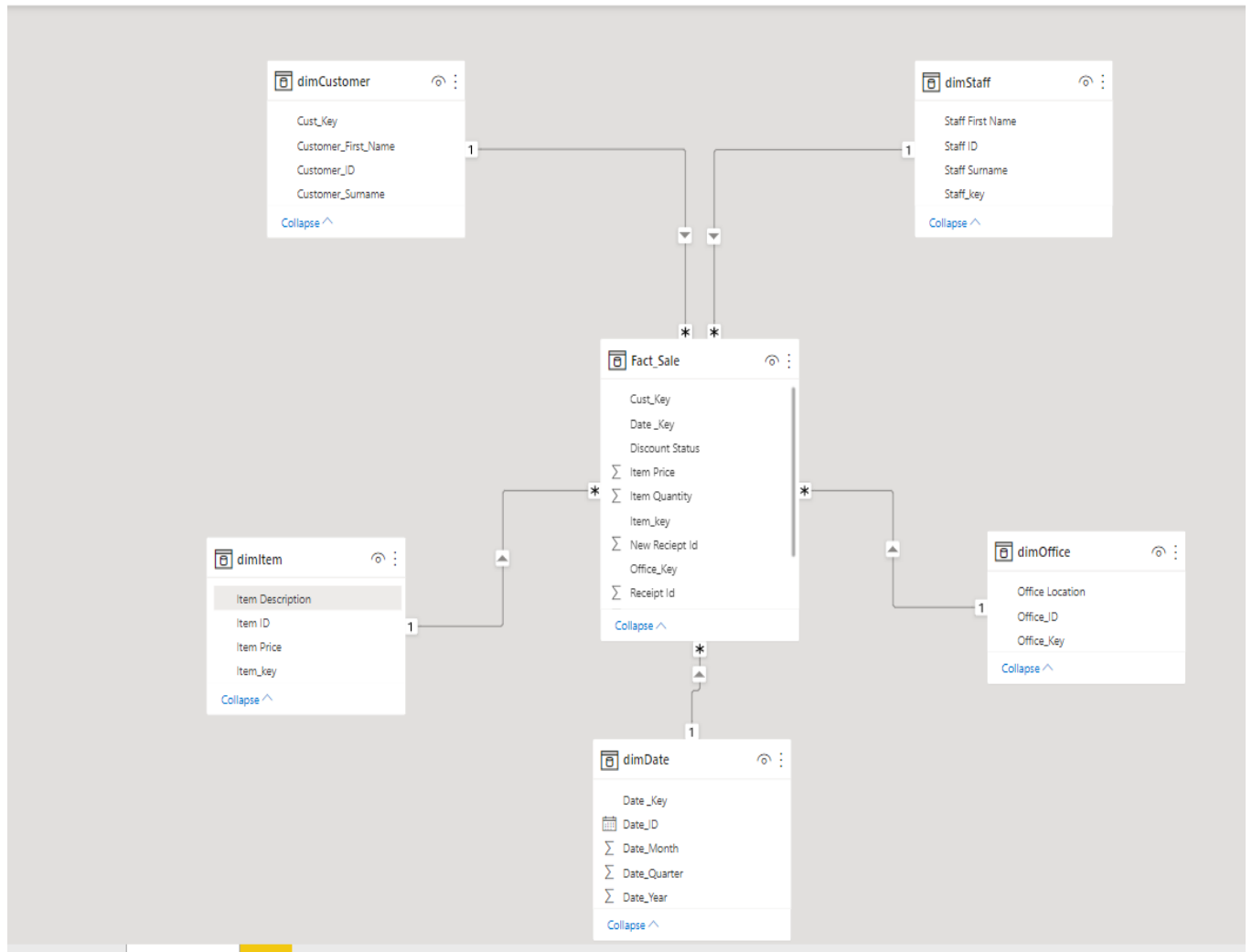


Figure 1.

The diagram above provides an overview of the data model for BitsAndBobs sales.

ETL Process

Data is combined from various data sources into a single combined consistent data store through the extract, transform, and load (ETL) process, which is then loaded into a data warehouse.

Data integration and loading (ETL) is a process used for computation and analysis. For data warehousing projects, it is the primary method of data processing.

ETL can handle more advanced analytics that might improve back-end operations or end-user experiences. It cleans and arranges data using a set of business rules to satisfy specific business intelligence requirements, like monthly reporting.

This process is used for;

- Extract data from legacy systems
- Cleanse the data to improve data quality and establish consistency
- Load data into a target database

Extract

The phase of extracting data from the current location

For the majority of my analytical work, I have used the Power BI tool. An overview of the procedure leading up to report publication is given in the following figure.

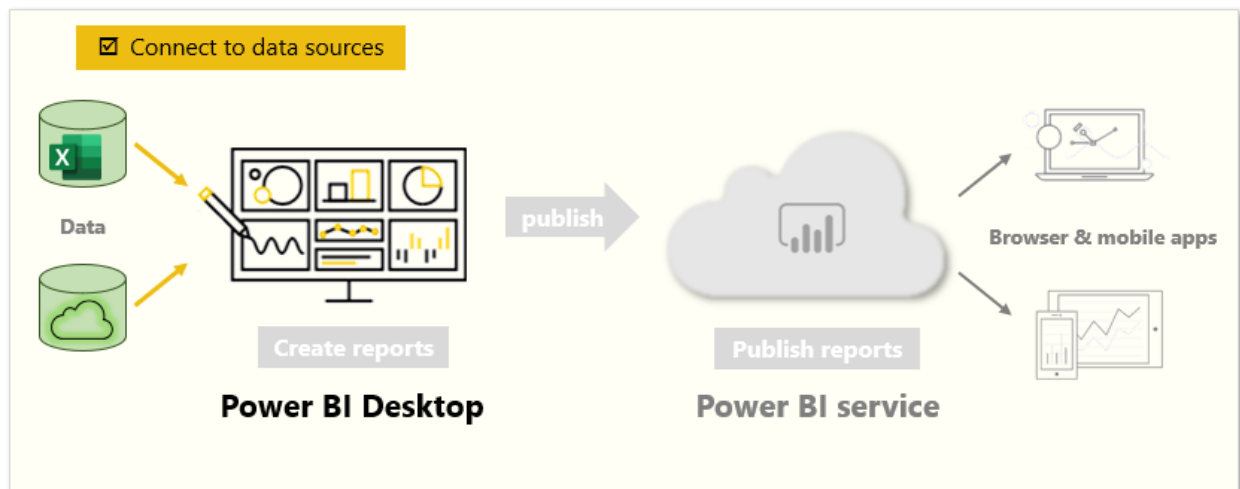


Figure 2.

Step 1: Connect to a data source.

Data source: *"AssignmentTwo2022Data.xls"*

Select Get data from the ribbon, and then select the Excel workbook option. This will allow you to connect to an Excel data source.

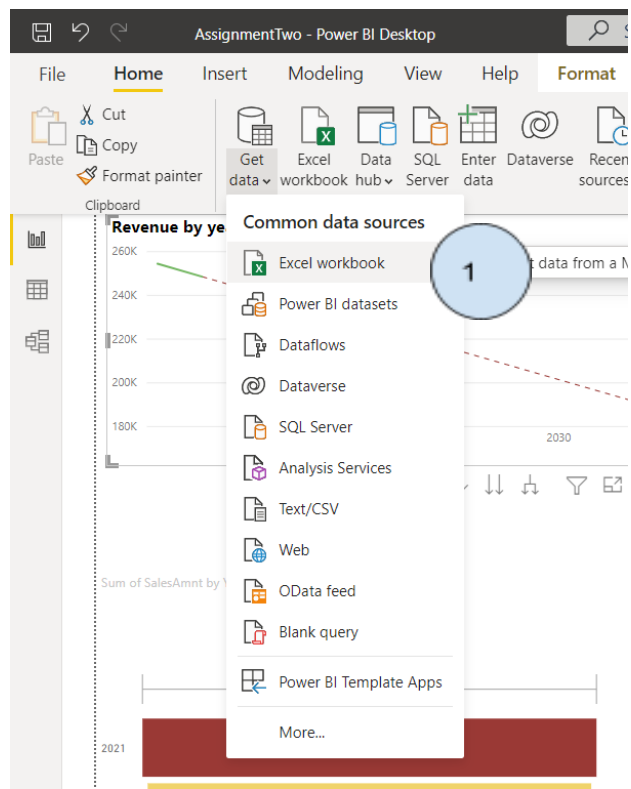


Figure 3.

Shevindi Navanjana Rodrigo
c3413510

Once you select the relevant workbook, a window will appear to choose from available sheets. Tick the relevant sheet and press the Load button as below.

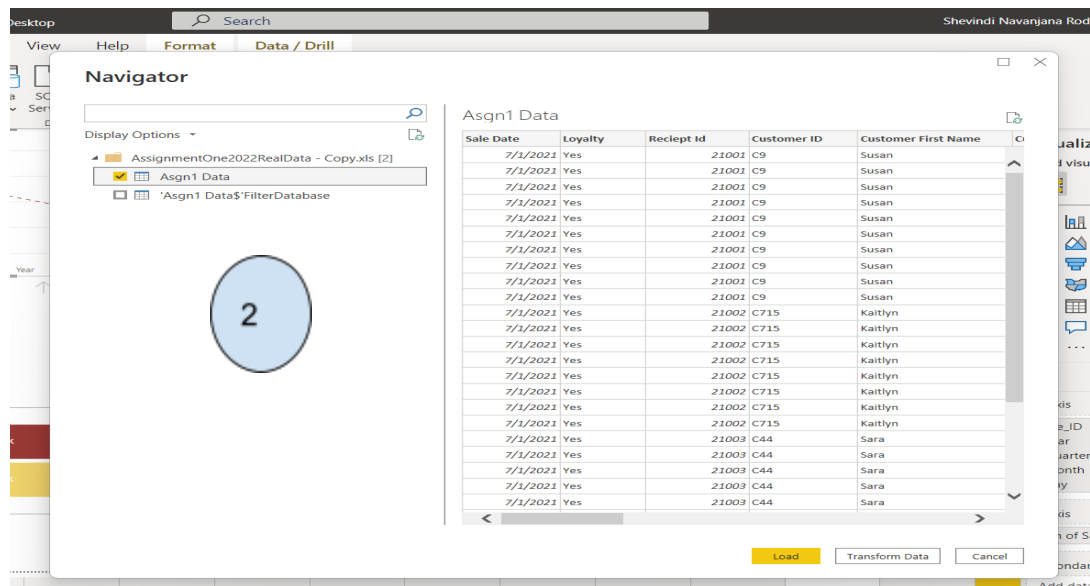


Figure 4.

Once you load data into the Power BI desktop, it will appear as follows:

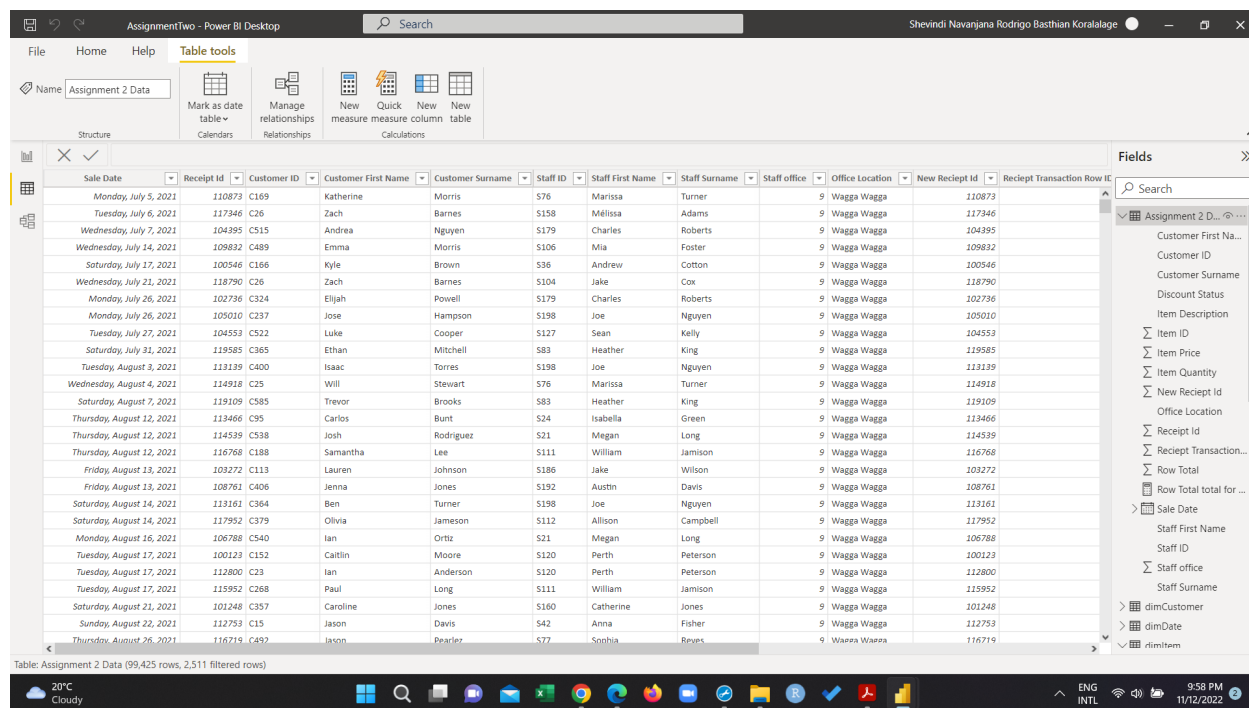


Figure 5.

Now the data is ready for the transformation. In order to get your data into the correct format to load into the datamart, use the Transform Data option from the Home tab. Power queries provide the option for the transformation of data.

Transform

This is the phase where raw data imported from a data source is converted into a workable data structure.

Aspects of the transformation might include things such as:

- Validation checks on the application of business logic.
- Data integrity checks – consistency and format of the data loading.
- Derivation/creation of new data fields to align with business requirements. E.g., including a total field, as opposed to derivation at execution time.
- Application of criteria on the final transformed table. E.g., only load the latest of the interaction with a client, not every historical interaction.
- Reworking existing data, format/structures for greater analysis. E.g., flattened tables (many columns) vs deep tables with many rows, but only 1 element of data per row.

Once our data set is loaded into power query, data type is changed into to the desired type. This will remove any formatting changes occur when the data is imported into the power tool.

1. The dataset is then checked for insertion anomalies. There were duplicated receipt IDs within the dataset. Following queries were implemented inorder find those invalid data.

- First the dataset is loaded into new query and deleted unnecessary columns as below.

```
let
    Source = Excel.Workbook(File.Contents("C:\Users\shevindi\Documents\UONT3\INFO6090\AssignmentTwo2022Data.xlsx"), null, true),
    #"Assignment 2 Data_Sheet" = Source[[Item="Assignment 2 Data",Kind="Sheet"]][Data],
    #"Promoted Headers" = Table.PromoteHeaders(#"Assignment 2 Data_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Sale Date", type date},
    {"Receipt ID", Int64.Type}, {"Customer ID", type text}, {"Customer First Name", type text}, {"Customer Surname", type text}, {"Staff ID", type text},
    {"Staff First Name", type text}, {"Staff Surname", type text}, {"Staff office", Int64.Type}, {"Office Location", type text}, {"Receipt Transaction Row ID", Int64.Type}, {"Item ID", Int64.Type},
    {"Item Description", type text}, {"Item Quantity", Int64.Type}, {"Item Price", type number}, {"Row Total", type number}}),
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Customer First Name", "Customer Surname", "Staff First Name", "Staff Surname", "Staff office", "Office Location",
    "Receipt Transaction Row ID", "Item ID", "Item Description", "Item Quantity", "Item Price", "Row Total"})
in
    #"Removed Columns"
```

Figure 6.

- Above query was appended with new query invalidData, removed duplicates, and found the count of each receipt ID entry by grouping according to receipt IDs.

```
let
    Source = Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("i44FAA==", BinaryEncoding.Base64), Compression.Deflate))),
    let _t = ((type nullable text) meta [Serialized.Text = true]) in type table [Column1 = _t],
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Column1", type text}}),
    #"Appended Query" = Table.Combine({#"Changed Type", #"Assignment 2 Data (2)"}),
    #"Reordered Columns" = Table.ReorderColumns(#"Appended Query",{"Sale Date", "Receipt Id", "Customer ID", "Staff ID"}),
    #"Removed Columns" = Table.RemoveColumns(#"Reordered Columns",{"Column1"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
    #"Grouped Rows" = Table.Group(#"Removed Duplicates", {"Receipt Id"}, {{"Count", each Table.RowCount(_), Int64.Type}}),
    #"Filtered Rows" = Table.SelectRows(#"Grouped Rows", each [Count] = 2)
in
    #"Filtered Rows"
```

Figure 7.

- Final output was taken by extracting receipt IDs with count greater than 1

	123 Receipt Id	123 Count
1	104312	2
2	118551	2

Figure 8.

In order to correct this issue, New Reciept ID column was added into the flat table. The Reciept ID and integral part of the relevant staff ID was extracted from the above two Reciept IDs and inserted into the new column. Null values of the column (New Reciept ID) were filled with their relevant Reciept ID.

Following script is provided to execute above task.

```
#"Merged Queries" = Table.NestedJoin(#"Changed Type", {"Receipt Id"}, InvalidData, {"Receipt Id"}, "InvalidData", JoinKind.RightOuter),
#"Changed Type1" = Table.TransformColumnTypes(#"Merged Queries",{{"Receipt Id", type text}}),
#"Inserted Text After Delimiter" = Table.AddColumn(#"Changed Type1", "Text After Delimiter", each Text.AfterDelimiter([Staff ID], "S"), type text),
#"Added Custom" = Table.AddColumn(#"Inserted Text After Delimiter", "New_Reciept ID", each [Receipt Id]&[Text After Delimiter]),
#"Removed Columns" = Table.RemoveColumns(#"Added Custom",{"Text After Delimiter"}),
#"Changed Type2" = Table.TransformColumnTypes(#"Removed Columns",{{"New_Reciept ID", Int64.Type}, {"Receipt Id", Int64.Type}}),
#"Appended Query" = Table.Combine({#"Changed Type2", #"Assignment 2 Data (3)"}),
#"Reordered Columns" = Table.ReorderColumns(#"Appended Query",{"Sale Date", "Receipt Id", "New_Reciept ID", "Customer ID", "Customer First Name", "Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff office", "Office Location", "Reciept Transaction Row ID", "Item ID", "Item Description", "Item Quantity", "Item Price", "Row Total"}),
#"Removed Errors" = Table.RemoveRowsWithErrors(#"Reordered Columns"),
```

Shevindi Navanjana Rodrigo
c3413510

```
#"Added Custom1" = Table.AddColumn(#"Removed Errors", "New Reciept Id", each if  
[New_Reciept ID] = null then [Receipt Id] else [New_Reciept ID]),
```

A sample of the output is shown below.

fx = Table.RemoveColumns(#"Reordered Columns1",{"New_Reciept ID"})

	Sale Date	Receipt Id	New Reciept Id	Customer ID	Customer First Name	Customer Surname	Staff ID
19	6/22/2022	104312	10431213	C148	Justin	Gray	S137
20	6/26/2022	118551	11855198	C567	Mia	Brown	S98
21	6/26/2022	118551	11855198	C567	Mia	Brown	S98
22	6/26/2022	118551	11855198	C567	Mia	Brown	S98
23	6/26/2022	118551	11855198	C567	Mia	Brown	S98
24	6/26/2022	118551	11855198	C567	Mia	Brown	S98
25	6/26/2022	118551	11855198	C567	Mia	Brown	S98
26	6/26/2022	118551	11855198	C567	Mia	Brown	S98
27	6/26/2022	118551	11855198	C567	Mia	Brown	S98
28	6/26/2022	118551	11855198	C567	Mia	Brown	S98
29	6/26/2022	118551	11855198	C567	Mia	Brown	S98
30	7/1/2021	100330	100330	C346	Lexi	Bunt	S18
31	7/1/2021	100330	100330	C346	Lexi	Bunt	S18
32	7/1/2021	100330	100330	C346	Lexi	Bunt	S18
33	7/1/2021	100330	100330	C346	Lexi	Bunt	S18
34	7/1/2021	100330	100330	C346	Lexi	Bunt	S18
35	7/1/2021	100463	100463	C53	Julia	Coles	S40
36	7/1/2021	100463	100463	C53	Julia	Coles	S40
37	7/1/2021	100593	100593	C565	Andrea	Jones	S177
38	7/1/2021	100593	100593	C565	Andrea	Jones	S177
39	7/1/2021	100593	100593	C565	Andrea	Jones	S177
40	7/1/2021	100593	100593	C565	Andrea	Jones	S177
41	7/1/2021	100593	100593	C565	Andrea	Jones	S177
42							

Figure 9.

2. According to the business rules provided by the company, one receipt row ID reserved only for one item. That is, single item can have a unique row ID within the receipt. But it was found to be there were multiple entries contradict to this rule.

The rows which contradict to this rule were identified as below

```
#"Grouped Rows" = Table.Group("#Removed Columns1", {"New Reciept Id", "Item  
ID"}, {"Count", each Table.RowCount(_), Int64.Type}, {"Item Quantity", each  
List.Sum([Item Quantity]), type nullable number})),  
#"Grouped Rows1" = Table.Group("#Grouped Rows", {"New Reciept Id"},  
{"Count", each _, type table [New Reciept Id=nullable number, Item ID=nullable  
number, Count=number, Item Quantity=nullable number]})),  
#"Added Custom2" = Table.AddColumn("#Grouped Rows1", "Reciept Transaction  
Row ID", each Table.AddIndexColumn([Count], "Index", 1)),  
#"Removed Columns2" = Table.RemoveColumns("#Added Custom2", {"Count", "New  
Reciept Id"}),
```

The rows with circled counts were some of the identified items.

	1 ² New Reciept Id	1 ² Item ID	1 ² Count	1.2 Item Quantity
1	10431243	22	1	2
2	10431243	27	1	2
3	10431243	16	1	6
4	10431243	10	2	18
5	10431243	24	1	7
6	10431243	8	1	4
7	10431243	28	1	4
8	10431243	20	1	1
9	118551157	7	2	9
10	118551157	25	1	1
11	118551157	22	1	7
12	118551157	9	1	1
13	118551157	15	1	7
14	118551157	26	1	2
15	118551157	6	1	2
16	104312137	22	1	5
17	104312137	29	1	1
18	11855198	8	1	4
19	11855198	12	1	7
20	11855198	24	1	1
21	11855198	6	1	6
22	11855198	2	2	8
23	11855198	19	1	9
24	11855198	23	1	10

Figure 10.

In order to resolve the issue Item Quantity column was reformed by adding the quantities of such item IDs. For example receipt ID 10431243 contain two entries for itemID 10. To reform it, quantities of the two entries were added and reform a new row, with its new row total and other unique details.

```

#"Expanded Reciept Transaction Row ID" = Table.ExpandTableColumn(#"Removed
Columns2", "Reciept Transaction Row ID", {"New Reciept Id", "Item ID", "Count",
"Item Quantity", "Index"}, {"Reciept Transaction Row ID.New Reciept Id",
"Reciept Transaction Row ID.Item ID", "Reciept Transaction Row ID.Count",
"Reciept Transaction Row ID.Item Quantity", "Reciept Transaction Row
ID.Index"}),
#"Renamed Columns" = Table.RenameColumns(#"Expanded Reciept Transaction Row
ID",{"Reciept Transaction Row ID.New Reciept Id", "New Reciept Id"}, {"Reciept
Transaction Row ID.Item ID", "Item ID"}, {"Reciept Transaction Row ID.Count",
"Count"}, {"Reciept Transaction Row ID.Item Quantity", "Item Quantity"},
{"Reciept Transaction Row ID.Index", "Reciept Transaction Row ID"}),
#"Changed Type4" = Table.TransformColumnTypes(#"Renamed Columns",{"New
Reciept Id", Int64.Type}),
#"Merged Queries1" = Table.NestedJoin(#"Changed Type4", {"Item ID"},
itemDesc, {"Item ID"}, "Assignment 2 Data (5)", JoinKind.LeftOuter),

```

```
#"Expanded Assignment 2 Data (5)" = Table.ExpandTableColumn(#"Merged
Queries1", "Assignment 2 Data (5)", {"Item Price"}, {"Assignment 2 Data
(5).Item Price"}),
#"Renamed Columns1" = Table.RenameColumns(#"Expanded Assignment 2 Data
(5)",{"Assignment 2 Data (5).Item Price", "Item Price"}),
#"Added Custom3" = Table.AddColumn(#"Renamed Columns1", "Row Total", each
[Item Quantity]*[Item Price]),
#"Changed Type5" = Table.TransformColumnTypes(#"Added Custom3",{"Row
Total", type number}, {"Receipt Transaction Row ID", Int64.Type}, {"Item
Quantity", Int64.Type}),
#"Removed Columns3" = Table.RemoveColumns(#"Changed Type5",{"Count"}),
#"Changed Type6" = Table.TransformColumnTypes(#"Removed Columns3",{"Item
ID", Int64.Type})
```

	1 ² New Receipt Id	1 ² Item ID	1 ² Item Quantity	1 ² Receipt Transaction Row ID	1 ² Item Price	1 ² Row Total
1	10431243	22	2	1	3.25	6.5
2	118551157	22	7	3	3.25	22.75
3	104312137	22	5	1	3.25	16.25
4	11855198	23	10	7	3.5	35
5	100330	23	3	1	3.5	10.5
6	10431243	27	2	2	4	8
7	118551157	7	9	1	15	135
8	11855198	7	10	9	15	150
9	100330	7	6	2	15	90
10	10431243	16	6	3	79.95	479.7
11	100330	3	5	3	18.95	94.75
12	10431243	10	18	4	149.9	2698.2
13	100330	4	1	4	10.95	10.95
14	10431243	24	7	5	3.5	24.5
15	11855198	24	1	3	3.5	3.5
16	10431243	28	4	7	4.45	17.8
17	100330	28	6	5	4.45	26.7
18	10431243	8	4	6	9.5	38
19	11855198	8	4	1	9.5	38
20	10431243	20	1	8	3	3
21	118551157	6	2	7	12	24
22	11855198	6	6	4	12	72
23	118551157	25	1	2	3.75	3.75
24	118551157	9	1	4	12.5	12.5
25	118551157	15	7	5	8.05	56.35

Figure 11.

The table was then reformed using the script below after the query and original staging table were combined.

A number of pointless columns were added to the staging table in order to reform. However, once we achieved the desired results, they were all cleaned.

let

```
Source =
Excel.Workbook(File.Contents("C:\Users\shevindi\Documents\UONT3\INFO6090\AssignmentTwo2022Data.xlsx"), null, true),
#"Assignment 2 Data_Sheet" = Source([Item="Assignment 2 Data",Kind="Sheet"])[Data],
#"Promoted Headers" = Table.PromoteHeaders("#Assignment 2 Data_Sheet", [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes("#Promoted Headers",{{"Sale Date", type date}, {"Receipt Id", Int64.Type}, {"Customer ID", type text}, {"Customer First Name", type text}, {"Customer Surname", type text}, {"Staff ID", type text}, {"Staff First Name", type text}, {"Staff Surname", type text}, {"Staff office", Int64.Type}, {"Office Location", type text}, {"Reciept Transaction Row ID", Int64.Type}, {"Item ID", Int64.Type}, {"Item Description", type text}, {"Item Quantity", Int64.Type}, {"Item Price", type number}, {"Row Total", type number}})},
#"Merged Queries" = Table.NestedJoin("#Changed Type", {"Receipt Id"}, InvalidData, {"Receipt Id"}, "InvalidData", JoinKind.RightOuter),
#"Changed Type1" = Table.TransformColumnTypes("#Merged Queries",{{"Receipt Id", type text}}),
#"Inserted Text After Delimiter" = Table.AddColumn("#Changed Type1", "Text After Delimiter", each Text.AfterDelimiter([Staff ID], "S"), type text),
#"Added Custom" = Table.AddColumn("#Inserted Text After Delimiter", "New_Reciept ID", each [Receipt Id]&[Text After Delimiter]),
#"Removed Columns" = Table.RemoveColumns("#Added Custom",{"Text After Delimiter"}),
#"Changed Type2" = Table.TransformColumnTypes("#Removed Columns",{{"New_Reciept ID", Int64.Type}, {"Receipt Id", Int64.Type}}),
#"Appended Query" = Table.Combine({#"Changed Type2", #"Assignment 2 Data (3)"}),
#"Reordered Columns" = Table.ReorderColumns("#Appended Query",{"Sale Date", "Receipt Id", "New_Reciept ID", "Customer ID", "Customer First Name", "Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff office", "Office Location", "Reciept Transaction Row ID", "Item ID", "Item Description", "Item Quantity", "Item Price", "Row Total"}),
#"Removed Errors" = Table.RemoveRowsWithErrors("#Reordered Columns"),
#"Added Custom1" = Table.AddColumn("#Removed Errors", "New Reciept Id", each if [New Reciept ID] = null then [Receipt Id] else [New_Reciept ID]),
#"Changed Type3" = Table.TransformColumnTypes("#Added Custom1",{{"New Reciept Id", Int64.Type}}),
#"Reordered Columns1" = Table.ReorderColumns("#Changed Type3",{"Sale Date", "Receipt Id", "New_Reciept ID", "New Reciept Id", "Customer ID", "Customer First Name", "Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff office", "Office Location", "Reciept Transaction Row ID", "Item ID", "Item Description", "Item Quantity", "Item Price", "InvalidData", "Row Total"}),
#"Removed Columns1" = Table.RemoveColumns("#Reordered Columns1",{"New_Reciept ID"}),
#"Merged Queries1" = Table.NestedJoin("#Removed Columns1", {"New Reciept Id"}, dataCleaning1, {"New Reciept Id"}, "dataCleaning1", JoinKind.Inner),
#"Expanded dataCleaning1" = Table.ExpandTableColumn("#Merged Queries1", "dataCleaning1", {"New Reciept Id", "Item ID", "Item Quantity", "Reciept Transaction Row ID", "Item Price", "Row Total"}, {"dataCleaning1.New Reciept Id", "dataCleaning1.Item ID", "dataCleaning1.Item Quantity", "dataCleaning1.Reciept Transaction Row ID", "dataCleaning1.Item Price", "dataCleaning1.Row Total"}),
```

```
#"Removed Duplicates" = Table.Distinct(#"Expanded dataCleaning1",
{"dataCleaning1.New Reciept Id", "dataCleaning1.Item ID", "dataCleaning1.Item
Quantity", "dataCleaning1.Reciept Transaction Row ID", "dataCleaning1.Item
Price", "dataCleaning1.Row Total"}),
#"Removed Columns2" = Table.RemoveColumns(#"Removed Duplicates",{"Reciept
Transaction Row ID", "Item ID", "Item Quantity", "Item Price", "Row Total"}),
#"Reordered Columns2" = Table.ReorderColumns(#"Removed Columns2",{"Sale
Date", "Receipt Id", "New Reciept Id", "Customer ID", "Customer First Name",
"Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff
office", "Office Location", "dataCleaning1.Reciept Transaction Row ID",
"dataCleaning1.Item ID", "Item Description", "dataCleaning1.Item Quantity",
"dataCleaning1.Item Price", "dataCleaning1.Row Total", "InvalidData",
"dataCleaning1.New Reciept Id"}),
#"Removed Columns3" = Table.RemoveColumns(#"Reordered
Columns2",{"dataCleaning1.New Reciept Id"}),
#"Renamed Columns" = Table.RenameColumns(#"Removed
Columns3",{{"dataCleaning1.Item Quantity", "Item Quantity"},
{"dataCleaning1.Item Price", "Item Price"}, {"dataCleaning1.Row Total", "Row
Total"}, {"dataCleaning1.Item ID", "Item ID"}}),
#"Merged Queries2" = Table.NestedJoin(#"Renamed Columns", {"Item ID"},
itemDesc2, {"Item ID"}, "itemDesc2", JoinKind.LeftOuter),
#"Expanded itemDesc2" = Table.ExpandTableColumn(#"Merged Queries2",
"itemDesc2", {"Item Description"}, {"itemDesc2.Item Description"}),
#"Renamed Columns1" = Table.RenameColumns(#"Expanded
itemDesc2",{{"dataCleaning1.Reciept Transaction Row ID", "Reciept Transaction
Row ID"}}),
#"Removed Columns4" = Table.RemoveColumns(#"Renamed Columns1",{"Item
Description"}),
#"Reordered Columns3" = Table.ReorderColumns(#"Removed Columns4",{"Sale
Date", "Receipt Id", "New Reciept Id", "Customer ID", "Customer First Name",
"Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff
office", "Office Location", "Reciept Transaction Row ID", "Item ID",
"itemDesc2.Item Description", "Item Quantity", "Item Price", "Row Total",
"InvalidData"}),
#"Renamed Columns2" = Table.RenameColumns(#"Reordered
Columns3",{{"itemDesc2.Item Description", "Item Description"}}),
#"Merged Queries3" = Table.NestedJoin(#"Renamed Columns2", {"New Reciept
Id"}, DiscountStatus, {"New Reciept Id"}, "Assignment 2 Data (4)",
JoinKind.LeftOuter),
#"Expanded Assignment 2 Data (4)" = Table.ExpandTableColumn(#"Merged
Queries3", "Assignment 2 Data (4)", {"New Reciept Id", "Discount Status"},
{"Assignment 2 Data (4).New Reciept Id", "Assignment 2 Data (4).Discount
Status"}),
#"Removed Columns5" = Table.RemoveColumns(#"Expanded Assignment 2 Data
(4)",{"Assignment 2 Data (4).New Reciept Id"}),
#"Reordered Columns4" = Table.ReorderColumns(#"Removed Columns5",{"Sale
Date", "Receipt Id", "New Reciept Id", "Customer ID", "Customer First Name",
"Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff
office", "Office Location", "Reciept Transaction Row ID", "Item ID", "Item
Description", "Item Quantity", "Item Price", "Row Total", "Assignment 2 Data
(4).Discount Status", "InvalidData"}),
#"Renamed Columns3" = Table.RenameColumns(#"Reordered
Columns4",{{"Assignment 2 Data (4).Discount Status", "Discount Status"}})
in
#"Renamed Columns3"
```


The table was checked for any duplicates, which were then removed.

```
#"Removed Duplicates" = Table.Distinct("#Expanded dataCleaning1",
{"dataCleaning1.New Receipt Id", "dataCleaning1.Item ID", "dataCleaning1.Item
Quantity", "dataCleaning1.Receipt Transaction Row ID", "dataCleaning1.Item
Price", "dataCleaning1.Row Total"}),
```

By using the column quality tool in the Power Query Editor, each column was examined for null values and validity.

Column	Valid	Error	Empty
Sale Date	100%	0%	0%
Receipt Id	100%	0%	0%
New Receipt Id	100%	0%	0%
Customer ID	100%	0%	0%
Customer First Name	100%	0%	0%
Customer Surname	100%	0%	0%
Staff ID	100%	0%	0%
Staff First Name	100%	0%	0%

Figure 12.

Column	Valid	Error	Empty
Staff office	100%	0%	0%
Office Location	100%	0%	0%
Receipt Transaction Row ID	100%	0%	0%
Item ID	100%	0%	0%
Item Description	100%	0%	0%
Item Quantity	100%	0%	0%
Item Price	100%	0%	0%
Row Total	100%	0%	0%

Figure 13.

Finally, a new column was added to specify the eligibility for the 5% discount for receipts of more than 5 items.

Checked for item count in each receipt

```
= Table.AddColumn("#Grouped Rows", "Discount Status", each if [Count]>=5 then
"Eligible" else "Non Eligible")
```

Merged with original staging table and eligibility status was added to newly added column, Discount status

Load

In this last step, the transformed data is moved from the staging area into a targeted datamart.

Six reference tables were created from the staging table, as shown below.

dimdate

```
let
    Source = #"Assignment 2 Data",
    #"Removed Other Columns" = Table.SelectColumns(Source,{"Sale Date"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Other Columns"),
    #"Added Custom" = Table.AddColumn(#"Removed Duplicates", "Date_Month", each
Number.ToText(Date.Month([Sale Date]), "00")),
    #"Changed Type" = Table.TransformColumnTypes(#"Added Custom",{{"Date_Month",
Int64.Type}}),
    #"Added Custom1" = Table.AddColumn(#"Changed Type", "Date_Quarter", each
Number.ToText(Date.QuarterOfYear([Sale Date]), "00")),
    #"Changed Type1" = Table.TransformColumnTypes(#"Added Custom1",{{"Date_Quarter",
Int64.Type}}),
    #"Added Custom2" = Table.AddColumn(#"Changed Type1", "Date_Year", each
Number.ToText(Date.Year([Sale Date]), "00")),
    #"Renamed Columns" = Table.RenameColumns(#"Added Custom2",{{"Sale Date",
"Date_ID"}}),
    #"Changed Type2" = Table.TransformColumnTypes(#"Renamed Columns",{{"Date_Year",
Int64.Type}}),
    #"Added Index" = Table.AddIndexColumn(#"Changed Type2", "Index", 1, 1,
Int64.Type),
    #"Renamed Columns1" = Table.RenameColumns(#"Added Index",{{"Index", "Date _Key"}})
in
    #"Renamed Columns1"
```

Surrogate Key - Date_Key

Natural key - Date ID

dimCustomer

```
let
    Source = #"Assignment 2 Data",
    #"Removed Columns" = Table.RemoveColumns(Source,{"Sale Date", "Receipt Id", "New
Reciept Id", "Staff ID", "Staff First Name", "Staff Surname", "Staff office", "Office
Location", "Reciept Transaction Row ID", "Item ID", "Item Description", "Item
Quantity", "Item Price", "Row Total", "Discount Status", "InvalidData"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
    #"Sorted Rows" = Table.Sort(#"Removed Duplicates",{{"Customer ID",
Order.Ascending}}),
    #"Renamed Columns" = Table.RenameColumns(#"Sorted Rows",{{"Customer ID",
"Customer_ID"}, {"Customer First Name", "Customer_First_Name"}, {"Customer Surname",
"Customer_Surname"}}),
    #"Added Index" = Table.AddIndexColumn(#"Renamed Columns", "Index", 1, 1,
Int64.Type),
    #"Renamed Columns1" = Table.RenameColumns(#"Added Index",{{"Index", "Cust_Key"}})
in
    #"Renamed Columns1"
```

Shevindi Navanjana Rodrigo
c3413510

Surrogate Key - Cust_Key

Natural key - Customer ID

dimStaff

```
let
    Source = #"Assignment 2 Data",
    #"Removed Columns" = Table.RemoveColumns(Source,{"Sale Date", "Receipt Id", "New
Reciept Id", "Customer ID", "Customer First Name", "Customer Surname", "Staff office",
"Office Location", "Reciept Transaction Row ID", "Item ID", "Item Description", "Item
Quantity", "Item Price", "Row Total", "Discount Status", "InvalidData"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
    #"Sorted Rows" = Table.Sort(#"Removed Duplicates",{"Staff ID",
Order.Ascending})),
    #"Added Index" = Table.AddIndexColumn(#"Sorted Rows", "Index", 1, 1, Int64.Type),
    #"Renamed Columns" = Table.RenameColumns(#"Added Index",{"Index", "Staff_key"})
in
    #"Renamed Columns"
```

Surrogate Key - Staff_key

Natural key - Staff ID

dimItem

```
let
    Source = #"Assignment 2 Data",
    #"Removed Columns" = Table.RemoveColumns(Source,{"Sale Date", "Receipt Id", "New
Reciept Id", "Customer ID", "Customer First Name", "Customer Surname", "Staff ID",
"Staff First Name", "Staff Surname", "Staff office", "Office Location", "Reciept
Transaction Row ID", "Item Quantity", "Row Total", "Discount Status", "InvalidData"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
    #"Added Index" = Table.AddIndexColumn(#"Removed Duplicates", "Index", 1, 1,
Int64.Type),
    #"Renamed Columns" = Table.RenameColumns(#"Added Index",{"Index", "Item_key"})
in
    #"Renamed Columns"
```

Surrogate Key - Item_key

Natural key - Item ID

dimOffice

```
Source = #"Assignment 2 Data",
    #"Removed Columns" = Table.RemoveColumns(Source,{"Sale Date", "Receipt Id", "New
Reciept Id", "Customer ID", "Customer First Name", "Customer Surname", "Staff ID",
"Staff First Name", "Staff Surname", "Reciept Transaction Row ID", "Item ID", "Item
Description", "Item Quantity", "Item Price", "Row Total", "Discount Status",
"InvalidData"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
```

```
#"Sorted Rows" = Table.Sort(#"Removed Duplicates",{{"Staff office",
Order.Ascending}}),
#"Renamed Columns" = Table.RenameColumns(#"Sorted Rows",{{"Staff office",
"Office_ID"}}),
#"Added Index" = Table.AddIndexColumn(#"Renamed Columns", "Index", 1, 1,
Int64.Type),
#"Renamed Columns1" = Table.RenameColumns(#"Added Index",{{"Index",
"Office_Key"}})
in
#"Renamed Columns1"
```

Surrogate Key - Office_Key

Natural key - Office ID

To create the fact table, a data set was loaded and merged with all dimension tables. Unnecessary columns were deleted to get the final output.

FactSale

```
let
Source = #"Assignment 2 Data",
#"Merged Queries" = Table.NestedJoin(Source, {"Sale Date"}, dimDate, {"Date_ID"},
"dimDate", JoinKind.LeftOuter),
#"Expanded dimDate" = Table.ExpandTableColumn(#"Merged Queries", "dimDate",
{"Date_Month", "Date_Quarter", "Date_Year", "Date_Key"}, {"dimDate.Date_Month",
"dimDate.Date_Quarter", "dimDate.Date_Year", "dimDate.Date_Key"}),
#"Removed Columns" = Table.RemoveColumns(#"Expanded dimDate",{"InvalidData",
"dimDate.Date_Month", "dimDate.Date_Quarter", "dimDate.Date_Year"}),
#"Reordered Columns" = Table.ReorderColumns(#"Removed Columns",{"Sale Date",
"dimDate.Date_Key", "Receipt Id", "New Reciept Id", "Customer ID", "Customer First
Name", "Customer Surname", "Staff ID", "Staff First Name", "Staff Surname", "Staff
office", "Office Location", "Reciept Transaction Row ID", "Item ID", "Item
Description", "Item Quantity", "Item Price", "Row Total", "Discount Status"}),
#"Removed Columns1" = Table.RemoveColumns(#"Reordered Columns",{"Sale Date",
"Customer First Name", "Customer Surname", "Staff First Name", "Staff Surname",
"Office Location", "Item Description"}),
#"Merged Queries1" = Table.NestedJoin(#"Removed Columns1", {"Staff office"},
dimOffice, {"Office_ID"}, "dimOffice", JoinKind.LeftOuter),
#"Merged Queries2" = Table.NestedJoin(#"Merged Queries1", {"Customer ID"},
dimCustomer, {"Customer_ID"}, "dimCustomer", JoinKind.LeftOuter),
#"Merged Queries3" = Table.NestedJoin(#"Merged Queries2", {"Staff ID"}, dimStaff,
{"Staff ID"}, "dimStaff", JoinKind.LeftOuter),
#"Merged Queries4" = Table.NestedJoin(#"Merged Queries3", {"Item ID"}, dimItem,
{"Item ID"}, "dimItem", JoinKind.LeftOuter),
#"Expanded dimOffice" = Table.ExpandTableColumn(#"Merged Queries4", "dimOffice",
{"Office_Key"}, {"dimOffice.Office_Key"}),
#"Removed Columns2" = Table.RemoveColumns(#"Expanded dimOffice",{"Staff office"}),
#"Expanded dimCustomer" = Table.ExpandTableColumn(#"Removed Columns2",
"dimCustomer", {"Cust_Key"}, {"dimCustomer.Cust_Key"}),
#"Expanded dimStaff" = Table.ExpandTableColumn(#"Expanded dimCustomer",
"dimStaff", {"Staff_key"}, {"dimStaff.Staff_key"}),
```

Shevindi Navanjana Rodrigo
c3413510

```
#"Expanded dimItem" = Table.ExpandTableColumn("#Expanded dimStaff", "dimItem",
{"Item_key"}, {"dimItem.Item_key"}),
#"Removed Columns3" = Table.RemoveColumns("#Expanded dimItem",{"Staff ID",
"Customer ID", "Item ID"}),
#"Reordered Columns1" = Table.ReorderColumns("#Removed Columns3",{"dimDate.Date
_Key", "Receipt Id", "New Reciept Id", "Reciept Transaction Row ID",
"dimCustomer.Cust_Key", "dimStaff.Staff_key", "dimOffice.Office_Key",
"dimItem.Item_key", "Item Quantity", "Item Price", "Row Total", "Discount Status"}),
#"Renamed Columns" = Table.RenameColumns("#Reordered
Columns1",{{"dimCustomer.Cust_Key", "Cust_Key"}, {"dimStaff.Staff_key", "Staff_key"},
{"dimOffice.Office_Key", "Office_Key"}, {"dimItem.Item_key", "Item_key"},
{"dimDate.Date_Key", "Date_Key"}})
in
#"Renamed Columns"
```

Foreign Keys - Date_key, Cust_key, Staff_key, Office_key, Item_key

Output of the analysis

The following studies were completed in order to address the business issue brought up by the CEO of BitsAndBobs.

He needed to consolidate his Australian offices, as the CEO had already mentioned. Australia has ten offices, and each office's performance must be taken into account.

Reports on successful branch performance can be used to determine which procedures are effective. Nevertheless, recognizing harmful behaviors is just as important. It's critical to comprehend how each company branch operates.

Following metrics were evaluated on each of the branch in order to get an idea of the performance

- Net Revenue per month/year
- Total transactions carried out per month/year
- Total Income

1. Net Revenue per month/year

A new table was created in order to summarise revenues by month and year for each office

Table_Revenue =

```
SUMMARIZE (Fact_Sale, dimDate[Date_Month], dimDate[Date_Year], dimOffice[Office Location], "Revenue", SUM(Fact_Sale[Row Total]))
```

Date_Month	Date_Year	Revenue	Office Location
7	2021	99858.65	Newcastle
8	2021	97832.4	Newcastle
9	2021	100204.75	Newcastle
10	2021	95938.45	Newcastle
11	2021	101610.55	Newcastle
12	2021	92549.5	Newcastle
1	2022	87236.79999999999	Newcastle
2	2022	105109.1	Newcastle
3	2022	89374.45000000001	Newcastle
6	2022	95832.5	Newcastle
4	2022	94919.7	Newcastle
5	2022	107249.75	Newcastle
7	2021	105909.9	Sydney
8	2021	117228.6	Sydney
9	2021	100887.25	Sydney
10	2021	113392.3	Sydney
11	2021	113110.3	Sydney
12	2021	108033.6	Sydney
1	2022	116819.9	Sydney
2	2022	104099.15	Sydney
3	2022	110150.4	Sydney
6	2022	108817.5	Sydney
4	2022	103751.2	Sydney
5	2022	107022.45	Sydney
7	2021	101128.85	Hobart
8	2021	86162.5	Hobart
9	2021	89198.55	Hobart

Figure 14.

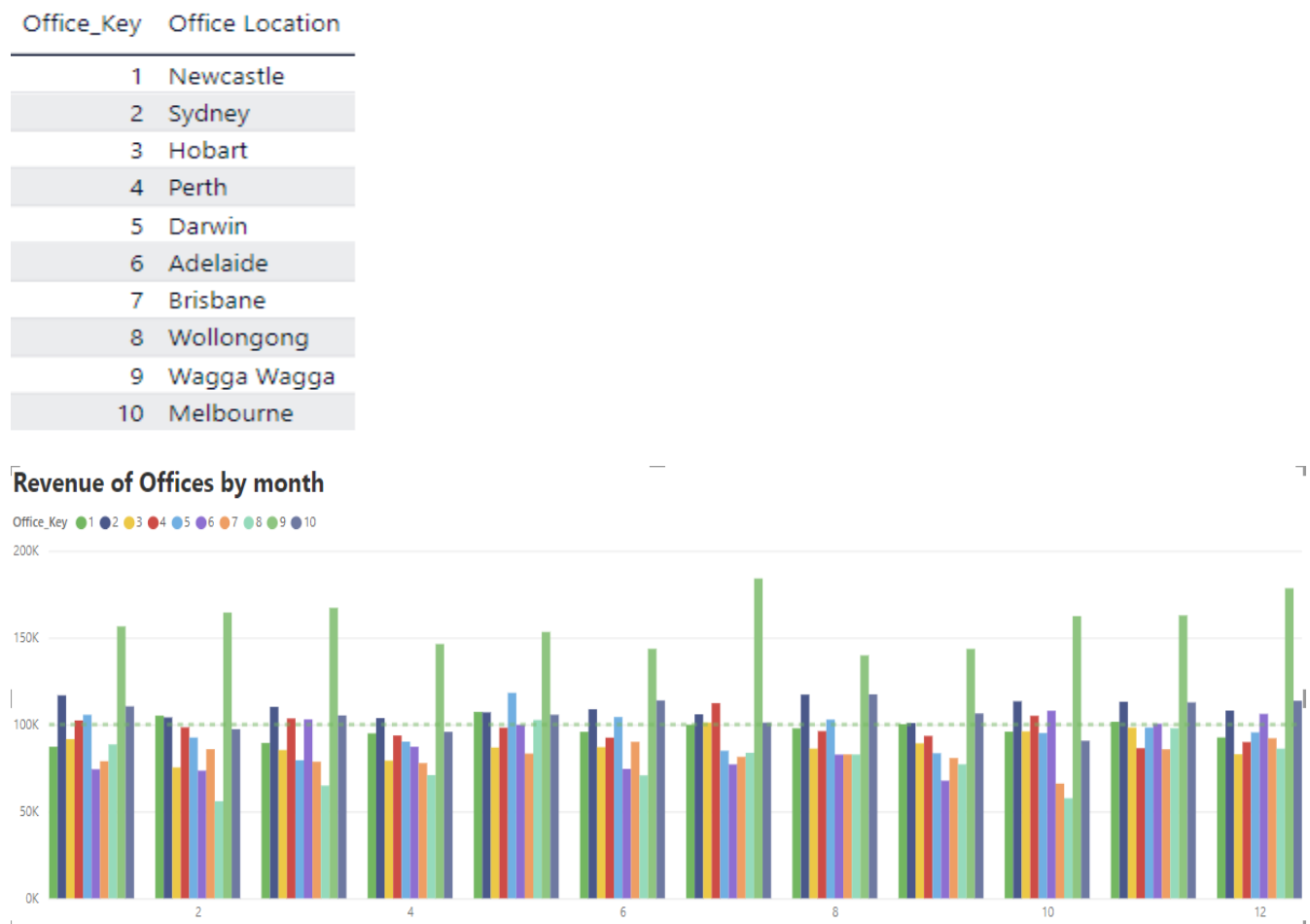


Figure 16.

According to the above visualization, it can be seen that **Office key 9 - Wagga wagga** has highest revenue in each month.

Date_Month	1	2	3	4	5	6	7	8	9	10
1	87,236.80	116,819.90	91,591.15	102,326.05	105,611.20	74,267.85	78,868.80	88,567.10	156,456.35	110,455.10
2	105,109.10	104,099.15	75,348.15	98,412.45	92,509.35	73,455.00	85,859.40	55,842.90	164,428.65	97,303.90
3	89,374.45	110,150.40	85,360.95	103,538.45	79,430.85	102,955.90	78,600.10	64,890.10	167,142.40	105,238.20
4	94,919.70	103,751.20	79,239.65	93,714.40	90,159.30	87,218.60	77,844.80	70,887.65	146,326.85	95,826.25
5	107,249.75	107,022.45	86,784.30	98,207.20	118,225.60	99,774.35	83,316.10	102,617.30	153,296.60	105,634.50
6	95,832.50	108,817.50	86,992.75	92,448.25	104,322.90	74,474.50	90,049.35	70,750.30	143,568.25	113,885.70
7	99,858.65	105,909.90	101,128.85	112,275.75	84,926.35	77,067.70	81,389.40	83,739.85	184,055.05	101,103.20
8	97,832.40	117,228.60	86,162.50	96,220.90	102,869.55	82,738.65	82,863.35	82,785.30	139,808.95	117,361.90
9	100,204.75	100,887.25	89,198.55	93,465.45	83,498.15	67,698.95	80,744.85	77,116.60	143,521.55	106,411.70
10	95,938.45	113,392.30	96,101.95	105,032.75	95,102.65	107,942.75	66,022.35	57,615.10	162,385.65	90,668.55
11	101,610.55	113,110.30	98,284.60	86,380.65	98,299.35	100,472.50	85,785.80	97,857.90	162,826.40	112,711.85
12	92,549.50	108,033.60	82,948.25	89,907.60	95,492.00	106,145.15	92,095.15	86,142.20	178,454.15	113,695.75

Figure 17.



Figure 18.

According to the above visualization, it can be seen that **Office key 9 - Wagga wagga** has highest revenue in each year.

Offices were ranked year-wise as below.

```
Column = var a = SUMMARIZE('Net_Rev', 'Net_Rev'[Date_Year], 'Net_Rev'[Office Location], "Revenue", SUM('Net_Rev'[Row Total])) var b = ADDCOLUMNS(a, "_rank", RANKX(FILTER(a, 'Net_Rev'[Date_Year]=EARLIER('Net_Rev'[Date_Year])), [Revenue])) return SUMX(FILTER(b, 'Net_Rev'[Date_Year]=EARLIER('Net_Rev'[Date_Year]) && 'Net_Rev'[Office Location]=EARLIER('Net_Rev'[Office Location])), [_rank])
```

2021



Figure 19.

The first column of the table's table gives the ranking by the year 2021. The bar graph clearly visualises the Highest revenue of branch Wagga Wagga by 2021

2022

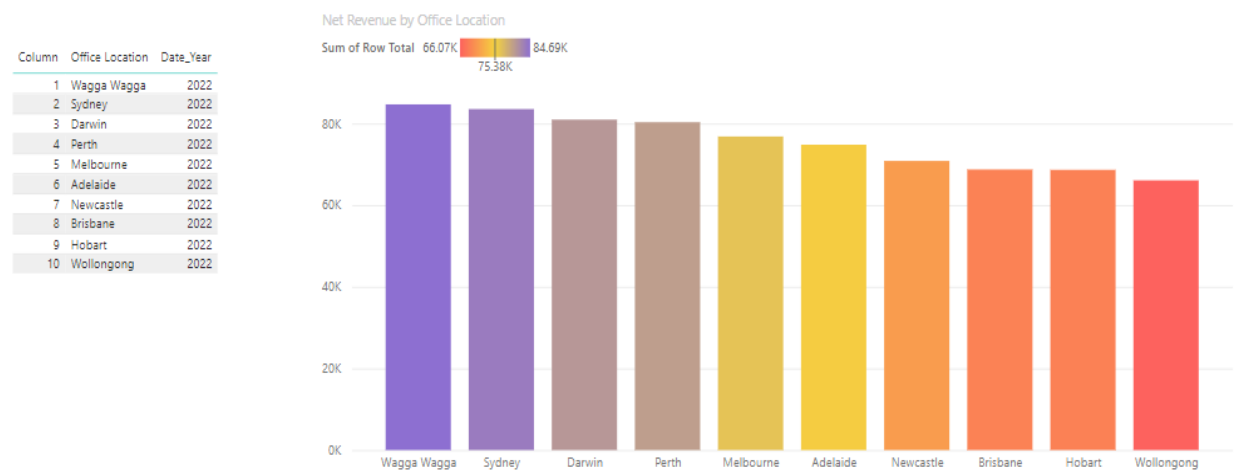


Figure 20.

The first column of the table's table gives the ranking by the year 2022. The bar graph clearly visualises the Highest revenue of branch Wagga Wagga by 2022

2. Total transactions carried out per month/year

A new measure was created in order to summarise total transactions by month and year for each office

Net Sales count = `DISTINCTCOUNT(Fact_Sale[New Reciept Id])`

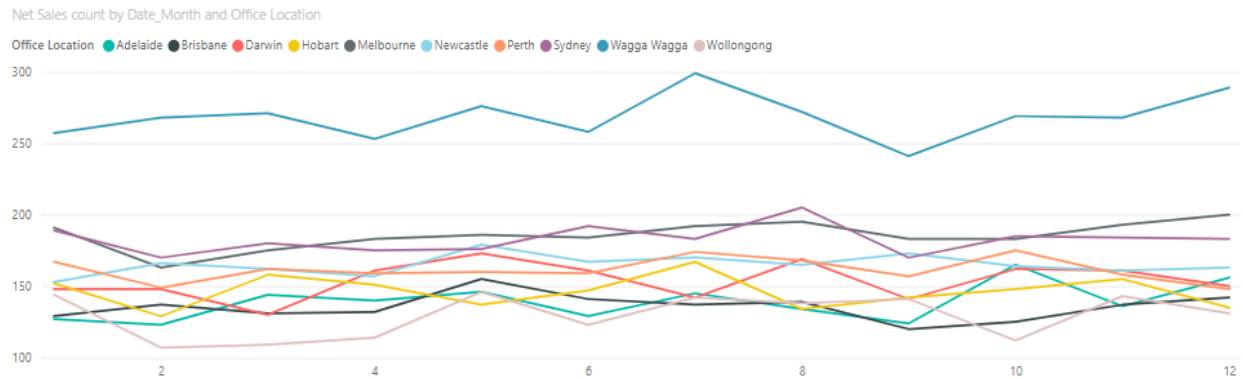


Figure 21.

Date_Month	Adelaide	Brisbane	Darwin	Hobart	Melbourne	Newcastle	Perth	Sydney	Wagga Wagga	Wollongong
1	127	129	148	152	191	153	167	189	257	144
2	123	137	148	129	163	166	149	170	268	107
3	144	131	130	158	175	162	162	180	271	109
4	140	132	161	151	183	157	159	175	253	114
5	146	155	173	137	186	179	160	176	276	146
6	129	141	161	147	184	167	159	192	258	123
7	145	137	142	167	192	170	174	183	299	142
8	134	139	169	134	195	165	168	205	272	138
9	124	120	141	142	183	173	157	170	241	141
10	165	125	162	148	183	164	175	185	269	112
11	136	137	161	155	193	161	158	184	268	143
12	156	142	150	135	200	163	148	183	289	131

Figure 22.

According to the above visualization, it can be seen that **Office - Wagga wagga** has highest transaction count in each month.

Transaction count by Year

Office Location ● Adelaide ● Brisbane ● Darwin ● Hobart ● Melbourne ● Newcastle ● Perth ● Sydney ● Wagga Wagga ● Wollongong

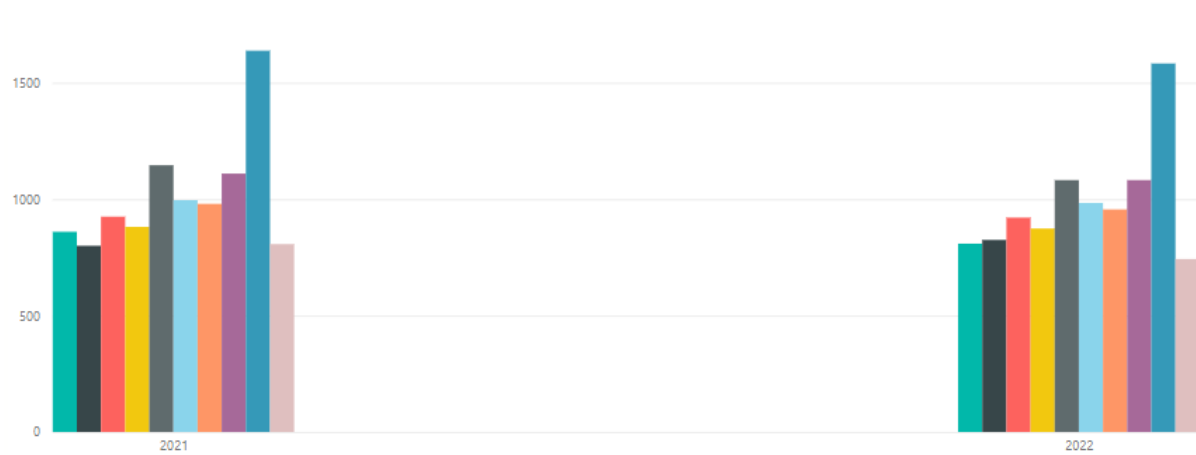


Figure 23.

According to the above visualization, it can be seen that **Office - Wagga wagga** has highest transaction count in each year.

3. Net Revenue by Office

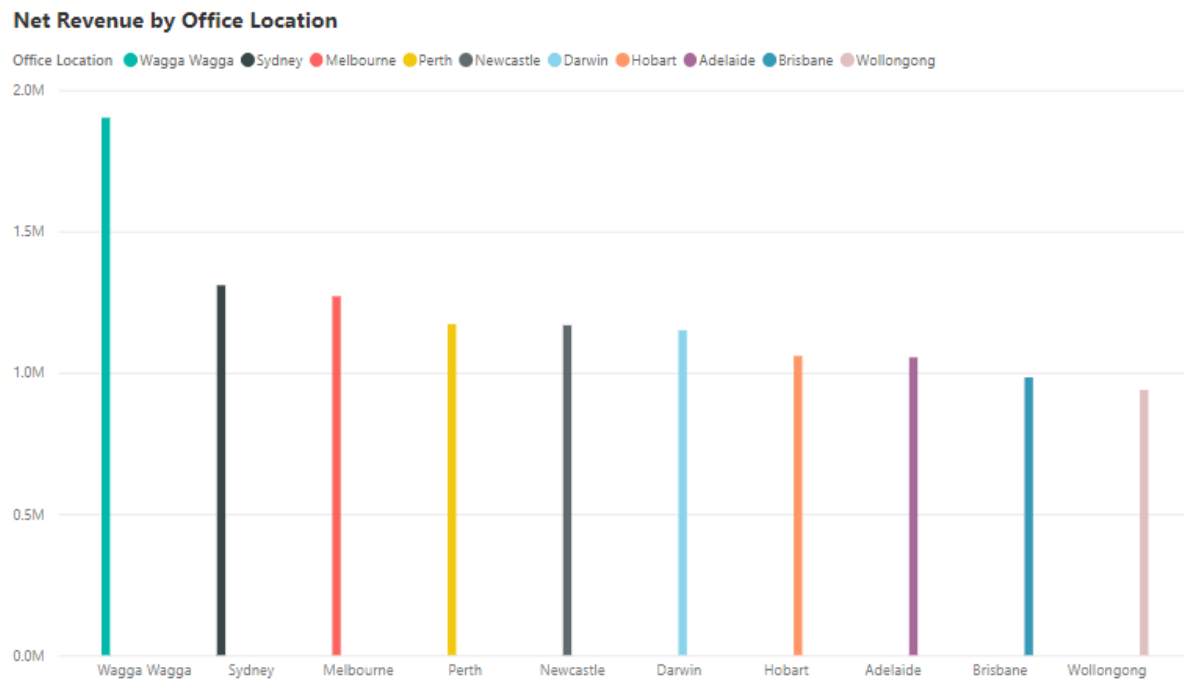


Figure 24.

Based on the above metrics we can rank each Offices as below

```
Net Sales Revenue = SUMMARIZE(Fact_Sale,dimOffice[Office Location],"Net  
Revenue",SUM(Fact_Sale[Row Total]))  
Rank = RANKX('Net Sales Revenue','Net Sales Revenue'[Net Revenue])
```

The output was as below

Rank	Office Location	Sum of Net Revenue
8	Adelaide	1,054,211.90
9	Brisbane	983,439.45
6	Darwin	1,150,447.25
7	Hobart	1,059,141.65
3	Melbourne	1,270,296.60
5	Newcastle	1,167,716.60
4	Perth	1,171,929.90
2	Sydney	1,309,222.55
1	Wagga Wagga	1,902,270.85
10	Wollongong	938,812.30
Total		12,007,489.05

Figure 25.

Based on the rankings and visualisation it can be suggested that branch Wagga Wagga has the highest performance, while Wollongong has least performance.

Summary of recent numbers and some trend analysis

Net income per month/ year

Sales Revenue = `SUMMARIZE (Fact_Sale,dimDate[Date_ID],dimItem[Item ID],dimStaff[Staff ID],dimOffice[Office Location],"SalesAmnt",SUM(Fact_Sale[Row Total]))`

Revenue by year



Figure 26.

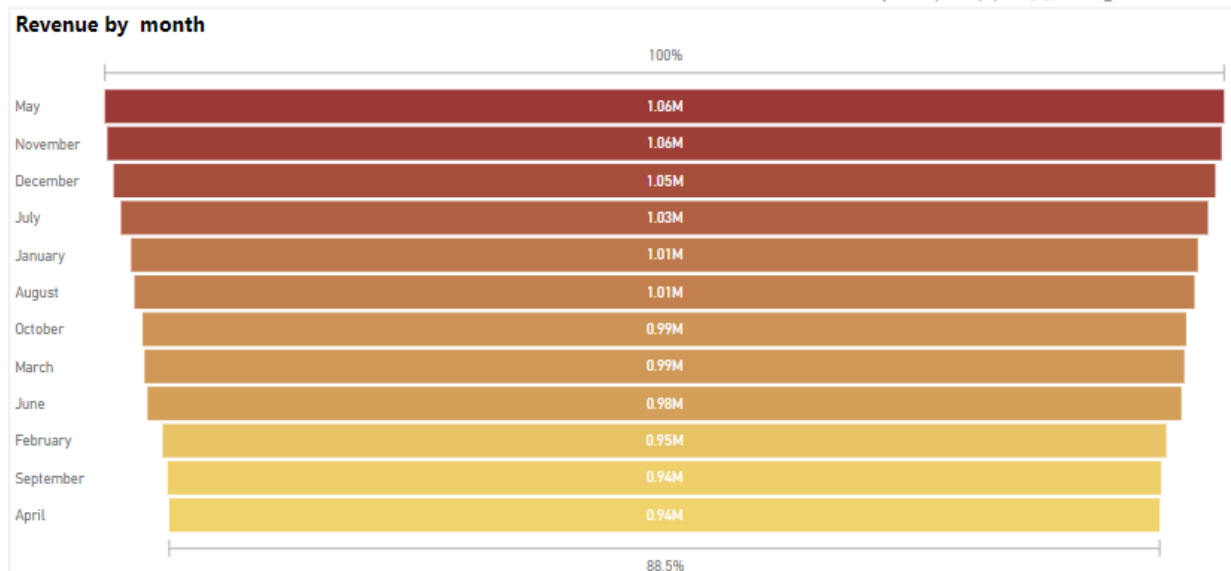


Figure 27.

Total transactions count per year / month

Sales count =

```
SUMMARIZE(Fact_Sale, dimDate[Date_ID], "SalesAmnt", DISTINCTCOUNT(Fact_Sale[New Reciept Id]))
```

Transaction count by Month

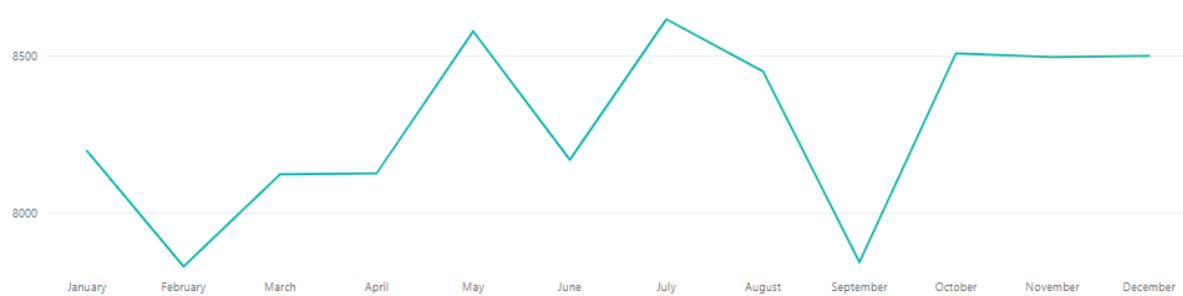


Figure 28.



Figure 29.

Item count by each Item

The new table was created to calculate total item count grouped by each item

```
ItemByOffice = SUMMARIZE(Fact_Sale,dimItem[Item
Description],dimDate[Date_Year],dimOffice[Office Location],"Total",SUM(Fact_Sale[Item
Quantity]))
```

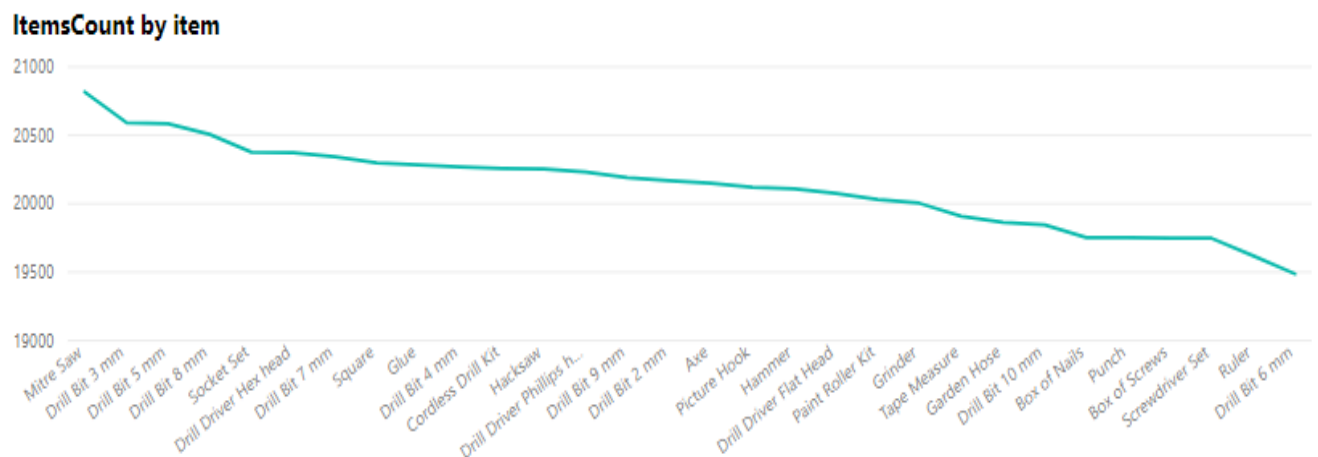


Figure 30.

The above table indicates that the **Mitre Saw** is the most well-known item.

Ranking of Items based on their sales count by office

The following process was executed to rank items by Office

```
Table 10 = SUMMARIZE(Fact_Sale,dimOffice[Office Location],dimItem[Item  
Description],"ItemQtyTot",SUM(Fact_Sale[Item Quantity]))  
rank = RANKX(FILTER('Table 10','Table 10'[Office Location]=EARLIER('Table 10'[Office  
Location])), 'Table 10'[ItemQtyTot])
```


Office Location	Item Description	ItemQtyTot	rank
Newcastle	Drill Bit 5 mm	2034	12
Sydney	Drill Bit 5 mm	2039	27
Hobart	Drill Bit 5 mm	2144	1
Perth	Drill Bit 5 mm	1944	18
Darwin	Drill Bit 5 mm	1798	24
Adelaide	Drill Bit 5 mm	1761	11
Brisbane	Drill Bit 5 mm	1676	11
Wollongong	Drill Bit 5 mm	1495	22
Wagga Wagga	Drill Bit 5 mm	3302	11
Melbourne	Drill Bit 5 mm	2384	2
Newcastle	Drill Bit 10 mm	1996	16
Sydney	Drill Bit 10 mm	1947	30
Hobart	Drill Bit 10 mm	1561	28
Perth	Drill Bit 10 mm	1933	21
Darwin	Drill Bit 10 mm	2042	2
Adelaide	Drill Bit 10 mm	1816	7
Brisbane	Drill Bit 10 mm	1590	19
Wollongong	Drill Bit 10 mm	1508	19
Wagga Wagga	Drill Bit 10 mm	3253	15
Melbourne	Drill Bit 10 mm	2193	16
Newcastle	Socket Set	2071	7
Sydney	Socket Set	2201	18
Hobart	Socket Set	1698	21
Perth	Socket Set	2151	3
Darwin	Socket Set	2003	7
Adelaide	Socket Set	1856	2
Brisbane	Socket Set	1560	22

Table 10 (300 rows) Column: rank (30 distinct values)

Figure 31.

Following are the top 3 items of each Office

Office Location	rank	Item Description
Adelaide	1	Screwdriver Set
Adelaide	2	Socket Set
Adelaide	3	Drill Bit 8 mm
Brisbane	1	Drill Bit 7 mm
Brisbane	2	Drill Driver Hex head
Brisbane	3	Drill Bit 3 mm
Brisbane	3	Paint Roller Kit
Darwin	1	Cordless Drill Kit
Darwin	2	Drill Bit 10 mm
Darwin	3	Drill Bit 3 mm
Hobart	1	Drill Bit 5 mm
Hobart	2	Garden Hose
Hobart	3	Cordless Drill Kit
Melbourne	1	Hacksaw
Melbourne	2	Drill Bit 5 mm
Melbourne	3	Square
Newcastle	1	Drill Bit 9 mm
Newcastle	2	Ruler
Newcastle	3	Drill Bit 7 mm
Perth	1	Box of Nails
Perth	2	Drill Bit 3 mm
Perth	3	Socket Set
Sydney	1	Tape Measure
Sydney	2	Drill Driver Phillips head
Sydney	3	Mitre Saw
Wagga Wagga	1	Square
Wagga Wagga	2	Drill Bit 4 mm
Wagga Wagga	3	Drill Bit 8 mm
Wollongong	1	Socket Set
Wollongong	2	Hacksaw
Wollongong	3	Tape Measure

Figure 32.

Following are the least 3 items of each Office

Office Location	rank	Item Description
Adelaide	28	Punch
Adelaide	29	Drill Bit 3 mm
Adelaide	30	Drill Driver Flat Head
Brisbane	28	Screwdriver Set
Brisbane	29	Box of Nails
Brisbane	30	Drill Bit 8 mm
Darwin	28	Paint Roller Kit
Darwin	29	Hacksaw
Darwin	30	Hammer
Hobart	28	Drill Bit 10 mm
Hobart	29	Hacksaw
Hobart	30	Tape Measure
Melbourne	29	Grinder
Melbourne	30	Tape Measure
Newcastle	28	Punch
Newcastle	29	Screwdriver Set
Newcastle	30	Grinder
Perth	28	Square
Perth	29	Ruler
Perth	30	Drill Bit 6 mm
Sydney	28	Screwdriver Set
Sydney	29	Box of Screws
Sydney	30	Drill Bit 10 mm
Wagga Wagga	28	Drill Driver Phillips head
Wagga Wagga	29	Drill Bit 6 mm
Wagga Wagga	30	Socket Set
Wollongong	28	Drill Bit 7 mm
Wollongong	29	Punch
Wollongong	30	Square

Figure 33.

Worst performing Items as a whole

Worst performing Items

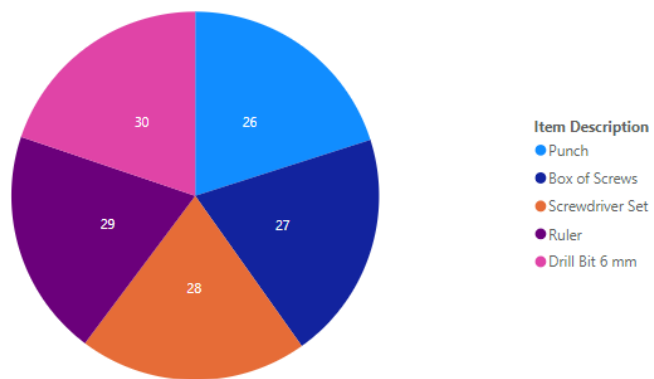


Figure 34.

Best sales person

Ranking of Salesperson on their average monthly earning

```
Table 3 = SUMMARIZE(Fact_Sale,dimStaff[Staff  
ID],dimDate[Date_Month],"Earning",AVERAGE(Fact_Sale[Row Total]))
```

```
rank = RANKX(FILTER('Table 3','Table 3'[Date_Month]=EARLIER('Table  
3'[Date_YMonth])), 'Table 3'[Earning])
```

Staff ID	rank	Date_Month
S1	77	8
S1	78	12
S1	81	4
S1	97	9
S1	128	6
S1	149	7
S10	18	7
S10	27	3
S10	43	5
S10	50	8
S10	62	4
S10	111	6
S10	128	11
S10	149	9
S10	154	2
S10	159	12
S10	175	1
S10	177	10
S100	13	11
S100	65	1
S100	84	3
S100	84	8
S100	85	5
S100	104	10
S100	109	6
S100	122	9

Figure 35.

Sort staff IDs into categories, treating those with ranks of less than 10 as 1 and all others as 0.

```
Column = IF('Table'[rank1] < 10, 1,0)
```

For each staff ID, a count with category 1 was calculated.

```
CALCULATE(COUNTA('Table'[Staff ID]), 'Table'[Column] IN { 1 })
```

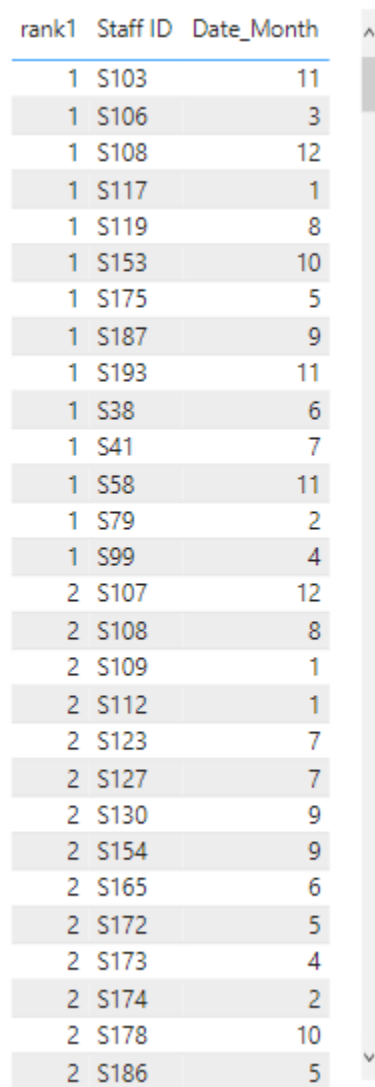
Count with 2 or more were filtered out and the output is as follows

Measure 2	Staff ID
3	S10
3	S106
2	S108
2	S109
5	S119
2	S12
2	S120
2	S127
2	S129
4	S136
2	S153
2	S165
2	S167
2	S174
2	S175
3	S178
2	S186
2	S187
2	S193
2	S196
2	S2
2	S3
4	S45
2	S6
2	S60
2	S62
3	S72
3	S83
2	S85
70	

Figure 36.

Ranking of Salesperson on monthly total transaction

```
Table = SUMMARIZE(Fact_Sale,dimStaff[Staff  
ID],dimDate[Date_Month],"Tot_Transactions",DISTINCTCOUNT(Fact_Sale[New Reciept Id]))  
rank1 =  
RANKX(FILTER('Table','Table'[Date_Month]=EARLIER('Table'[Date_Month])), 'Table'[Tot_Tra  
nsactions])
```



rank1	Staff ID	Date_Month
1	S103	11
1	S106	3
1	S108	12
1	S117	1
1	S119	8
1	S153	10
1	S175	5
1	S187	9
1	S193	11
1	S38	6
1	S41	7
1	S58	11
1	S79	2
1	S99	4
2	S107	12
2	S108	8
2	S109	1
2	S112	1
2	S123	7
2	S127	7
2	S130	9
2	S154	9
2	S165	6
2	S172	5
2	S173	4
2	S174	2
2	S178	10
2	S186	5

Figure 37.

Sort staff IDs into categories, treating those with ranks of less than 10 as 1 and all others as 0.

```
Column = IF('Table 3'[rank] < 10, 1,0)
```

Shevindi Navanjana Rodrigo
c3413510

For each staff ID, a count with category 1 was calculated.

```
CALCULATE(COUNTA('Table 3'[Staff ID]), 'Table 3'[Column] IN { 1 })
```

Count with 2 or more were filtered out and the output is as follows

Count of Staff ID for 1	Staff ID
2	S105
2	S125
2	S126
2	S127
2	S147
2	S157
2	S16
2	S170
2	S173
2	S184
4	S186
2	S19
2	S26
2	S33
2	S51
2	S52
2	S63
2	S66
2	S67
2	S71
2	S74
2	S76
2	S89
48	

Figure 38.

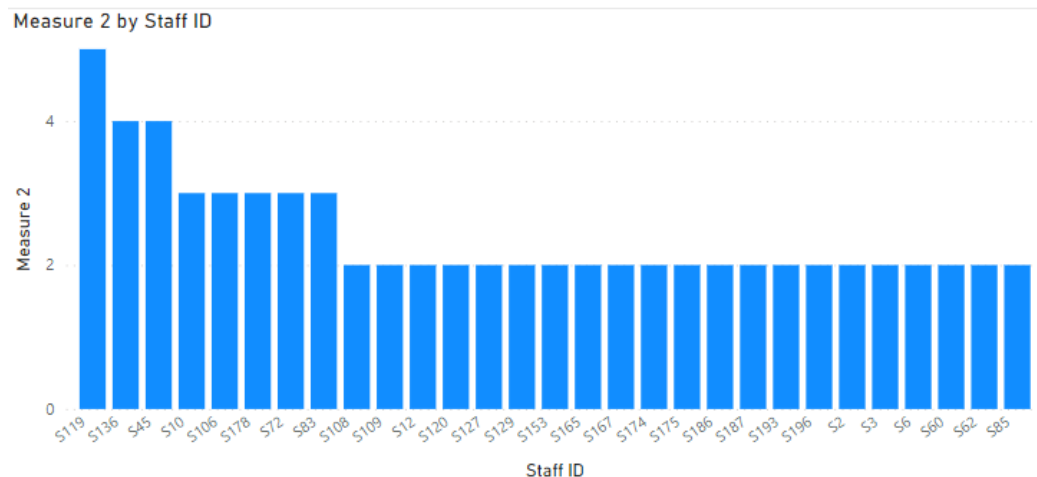


Figure 39.

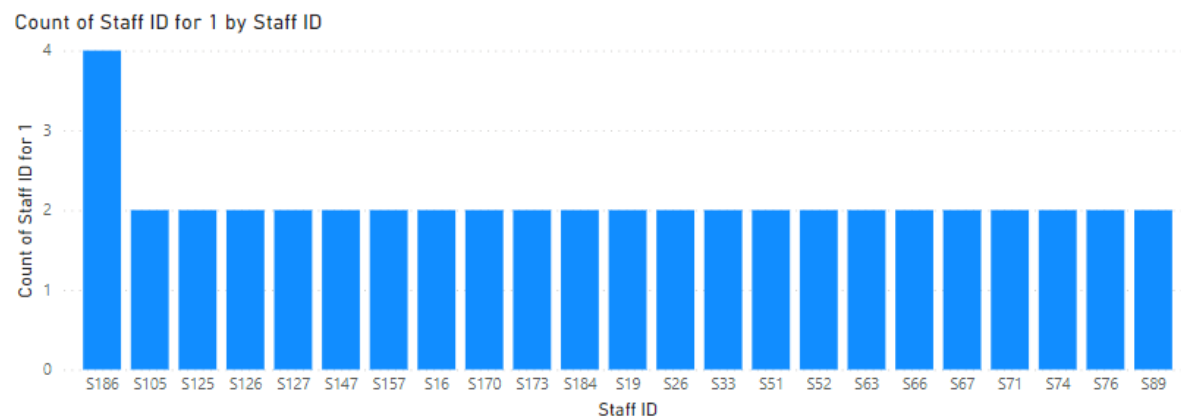


Figure 40.

When you consider transaction count S186 has occurred in highest ranks for 4 times and S127 is only 2
 When you consider earning S186 has occurred in highest ranks for 2 times and S127 is only 2
By these two metrics we can conclude that S186 has highest performance out of other salespersons

Predictive analysis

Income forecast for year 2023 by each office

Change percentage of income between previous and current year was calculated according to following formula

Change_percentage = (Current year - Previous year)/ previous year *100

According to the percentage value, income for each office for year 2023 was predicted as below.

Revenue for 2021 was extracted

Rev 2021 = CALCULATETABLE('Table 8',FILTER('Table 8','Table 8'[Date_Year]=2021))

Revenue for 2022 was extracted

Shevindi Navanjana Rodrigo
c3413510

```
Rev 2022 = CALCULATETABLE('Table 8',FILTER('Table 8','Table 8'[Date_Year]=2022))
```

Change between the years 2022 and 2021 calculated

```
Change = 'Rev 2022'[Revenue] - LOOKUPVALUE('Rev 2021'[Revenue], 'Rev 2021'[Office  
Location], 'Rev 2022'[Office Location], 0)
```

Percentage calculation

```
Change_percent = ('Rev 2022'[Change]/LOOKUPVALUE('Rev 2021'[Revenue], 'Rev 2021'[Office  
Location], 'Rev 2022'[Office Location], 0)) * 100
```

Estimate the change for 2023

```
EstimationChange = ('Rev 2022'[Revenue] * 'Rev 2022'[Change_percent]) / 100
```

Predicted the income for 2023

```
Forecast_2023 = 'Rev 2022'[Revenue] + 'Rev 2022'[EstimationChange]
```

Prediction revenue for 2023

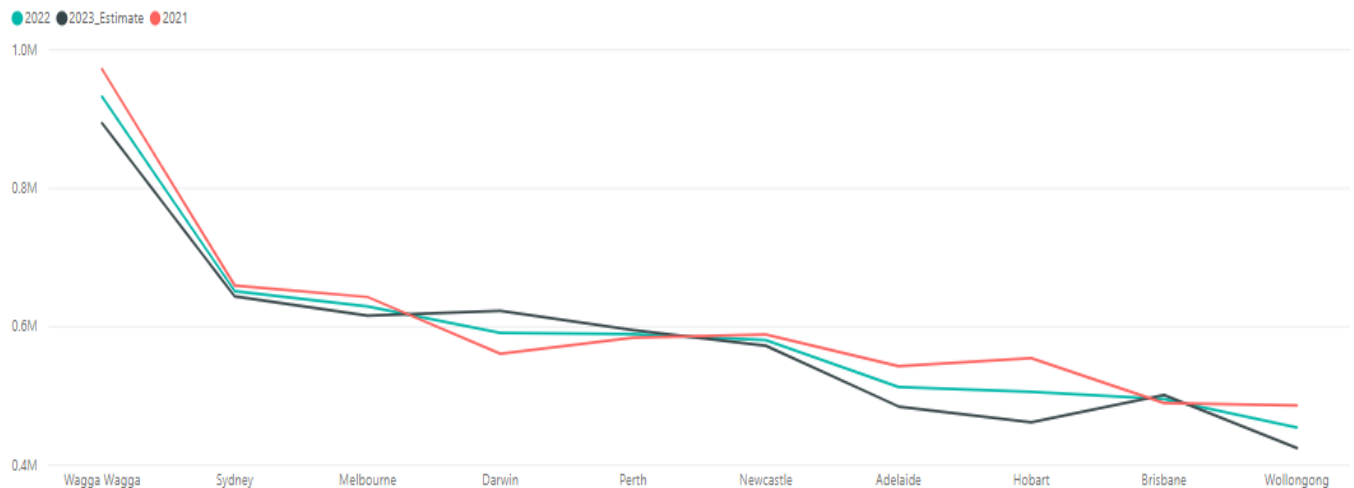


Figure 41.

Sales recommendation

The examination of particular developments or factors that have an impact on a company's sales activities is generally understood to be the definition of sales analysis. With regard to the positive and negative aspects, the overall sales strategy, revenue, and other factors, each analysis gives you a unique perspective on the performance of the sales team and the performance of the sales branches. Planning for long-term growth becomes simpler when a thorough understanding of sales activity over a specific time period is attained.

As a result, we were able to learn some unique and important things about the sales strategy using the provided sales data. We can use the analysis to generate important recommendations for the long-term expansion of the BitsAndBobs company.

It was established through the company's branch performance analysis that Wollongong's overall performance is at the lowest level. The performance was shown to decline with the anticipated metric for the year. Therefore, rather than choosing to consolidate, it is advised to improve performance by raising staff performance.

Performance of the sales staff is a factor in this case.

The table below illustrates how each employee performed based on their compensation.

Sum of SalesAmnt by Staff ID																											
Staff ID	S187	S45	S178	S122	S71	S193	S106	S190	S101	S104	S44	S68	S129	S138	S33	S196	S36	S132	S165	S156	S154	S186	S111	S58	S173	S21	
S187	S101	S36	S173	S27	S12	S137	S116	S199	S54	S31	S134	S88	S118	S107	S177	S191	S128	S...	S...	S97	S46	S64	S26	S55	S23		
S45	S104	S132	S21	S108	S6	S102	S139	S172	S67	S1	S78	S2	S163	S70	S20	S159	S110	S117									
S178	S44	S165	S48	S56	S62	S114	S52	S35	S182	S81	S121	S150	S63	S131	S19	S174	S47	S180									
S122	S68	S156	S5	S167	S119	S151	S95	S140	S85	S152	S10	S147	S53	S103	S175	S198	S123	S28	S124	S4							
S71	S129	S154	S188	S157	S127	S7	S143	S155	S94	S109	S59	S13	S72	S22	S197	S49	S30	S25	S184	S9							
S193	S138	S186	S91	S142	S14	S79	S169	S149	S50	S92	S15	S32	S24	S160	S112	S18	S189	S57	S185	S144							
S106	S33	S111	S29	S40	S83	S34	S74	S136	S195	S162	S89	S3	S65	S126	S96	S77	S99	S93	S183	S194	S164						
S190	S196	S58	S113	S179	S153	S170	S60	S200	S41	S75	S90	S87	S38	S145	S66	S86	S61	S105	S17	S37	S11						
												S181	S146	S171	S42	S16	S141	S148	S73	S176	S168						

Nevertheless, once these performances are analysed based on their branches , we could easily determine underperforming salespersons. officesoffice.

Average revenues per each office is given below.

```
Row Total average per Office Location =  
AVERAGEX (  
    KEEPFILTERS (VALUES ('dimOffice'[Office Location])),  
    CALCULATE (SUM ('Fact_Sale'[Row Total]))  
)
```

Row Total average per Office Location	Office Location
1,902,270.85	Wagga Wagga
1,309,222.55	Sydney
1,270,296.60	Melbourne
1,171,929.90	Perth
1,167,716.60	Newcastle
1,150,447.25	Darwin
1,059,141.65	Hobart
1,054,211.90	Adelaide
983,439.45	Brisbane
938,812.30	Wollongong
1,200,748.91	

These benchmark values can be used to help you propose a revenue target. You can define expectations and benchmarks for your sales cycle, for instance, by setting a \$1,500,000 yearly target for Melbourne. Assume the sales team for the specific branch has 10 members. Once you give your 10-person sales team this objective, they should be able to achieve annual earnings of close to \$150,000. This sum of money should serve as the basis for the salesperson's annual quota. A salesperson's bonus and commission structure ought to be directly related to their quota. This will motivate each salesperson to accomplish their objective.

Thus, the branch's performance will automatically improve as a result.

List of Figures

List Of Images

- [Figure 1.](#)
- [Figure 2.](#)
- [Figure 3.](#)
- [Figure 4.](#)
- [Figure 5.](#)
- [Figure 6.](#)
- [Figure 7.](#)
- [Figure 8.](#)
- [Figure 9.](#)
- [Figure 10.](#)
- [Figure 11.](#)
- [Figure 12.](#)
- [Figure 13.](#)
- [Figure 14.](#)
- [Figure 16.](#)
- [Figure 16.](#)
- [Figure 17.](#)
- [Figure 18.](#)
- [Figure 19.](#)
- [Figure 20.](#)
- [Figure 21.](#)
- [Figure 22.](#)
- [Figure 23.](#)
- [Figure 24.](#)
- [Figure 25.](#)
- [Figure 26.](#)
- [Figure 27.](#)
- [Figure 28.](#)
- [Figure 29.](#)
- [Figure 30.](#)
- [Figure 31.](#)
- [Figure 32.](#)
- [Figure 33.](#)
- [Figure 34.](#)
- [Figure 35.](#)
- [Figure 36.](#)
- [Figure 37.](#)
- [Figure 38.](#)

- [Figure 39.](#)
- [Figure 40.](#)
- [Figure 41.](#)

Dashboard

https://app.powerbi.com/groups/me/dashboards/50e3223d-6beb-40bf-8a0c-59d24e72791b?ctid=ee903dcb-2b77-4da5-be02-c45235783dad&pbi_source=linkShare

References

Evaluating Branch Performance: Maintaining the Momentum for Improved Branch Performance.

(2019, January 29). Syntellis Performance Solutions.

<https://www.syntellis.com/resources/article/evaluating-branch-performance-maintaining-momentum-improved-branch-performance#:~:text=Efficiency%20ratio%2C%20expense%20ratio%20and>

heyrob. (n.d.). *Get data from Excel - Training*. Learn.microsoft.com. Retrieved November 13, 2022, from <https://learn.microsoft.com/en-us/training/modules/get-data-power-bi/3b-data-from-excel>