Designing and Implementing a Data Warehouse

Author: Mahmudul Fakrul

Part 1: Problem Domain Understanding

Task 1

Given the nature of the business you have been asked to conduct a comparative analysis of on-premises, cloud-based, and hybrid data engineering solutions e.g., ETL and ELT.

The evaluation criteria should consider the following:

- *Security*
- *Compliance*
- *Scalability*
- *Efficiency*
- *Reliability*
- *Fidelity*
- *Flexibility*
- *Portability*

	On-premises	**Cloud-based**	**Hybrid**
Security	- Own responsibility to create a security team, maintain the team and the physical security of all the systems. - Own responsibility in creating a secure network infrastructure and access. - Data is stored on company drives therefore, the company has only access to the data hence reduced risk of unwanted access to data.	- Access to the servers maintained by cloud service provider. -Physical security cloud vendors responsibility. - Highly secure as they have secure network infrastructures. - Security responsibility varies on the type of service receiving (PaaS/SaaS/IaaS) - Ensure the tightest security controls with certifications such as ISO 270001 and SOC 2	- Combination of on-premises and Could. - the on premises resources security at companies hand. - The cloud vendors are responsible for the part which the client is using the service for. - shares benefits and disadvantages of both on-premises and cloud solutions
Compliance	- Responsibility of the companies to maintain all the regulations and compliances are met and know where the data is at all times. - easier to keep up to date as the company knows exactly what they need to comply by - some compliance regulations prohibit certain data types to be hosted in	- Responsibility of the cloud provider to maintain all compliances and the keeping up with new compliances. However, it is also duty of the client to ensure that cloud vendor is meeting regulatory mandates. - some compliance regulations prohibit certain data types to be hosted in the cloud	- Mixed responsibility depending what resources are in house and what is on cloud. - data types that are prohibited to be stored on the cloud can be stored on premises - data that can be stored on the cloud can be stored in the cloud. - In both cases the organisation needs to be

	the cloud. Therefore on premises would be a better option for some companies.		aware of the regulations and compliances to be always compliant.
Scalability	- Hard to scale if initially didn't plan of future expansion of the area. - physical machines and servers are expensive and take long time to purchase and install	- Easily scalable. It can be scaled up within matter of minutes to meet increased demand. - Can also easily reduce resources if demand is lower therefore also lowering costs.	- Can easily scale depending on demand. - On-premises resources can be used for the predictable workload. - Can request more resources from the public cloud when the unpredicted workload come in.
Efficiency	- Nearly impossible for on-premises to compete on price or efficiency as it would take a lot more time, and money for companies to manually build solutions.	- highly efficient. Cloud vendors offer massive economies of scale, scale elastically and run fully automated.	- Companies can take the advantage of the existing architecture in a data centre and leverage it to get the benefits of the cloud solutions.
Reliability	- In order to assure as good reliability as cloud based services, the company will have to spend a lot of millions of dollars on building and maintaining the infrastructure. - a lot of computer hardware is just prone to failure. Availability of the data might be a major issue if not enough resources planned out.	Highly reliable. Some cloud vendors provide 99.9% availability. The data can be replicated across different regions, even in the event of failure makes cloud service highly available and reliable.	- Companies can get the best of both worlds. - the service can be distributed across multiple data centres to increase reliability
Fidelity	- Since there's no need of internet connection when dealing with data within the organisation, it is more faithful. -Migrating data can be done more faithfully within the organisation with less risks - More transparent as the company has full control over hardware resources.	- No control over the hardware resources hence out of the loop. - not possible to know what actions cloud vendors take regarding security - Faith dependent: once the agreement is signed, the client assumes that cloud vendors cost structure will not chance considerably and service will not weakening in the future	- the organisation has full control on what to rely on the cloud provider for and what not.
Flexibility	- Hardly flexible. Might need to plan to get the latest innovations and wait	-Highly flexible, as service can be provided on demand within matter of minutes.	- Flexible, based on the amount of resources on premises and on cloud.

	until budget allows upgrading the systems - hard to adopt new infrastructure solutions - flexibility comes at a higher cost compared to cloud.	- access to the latest innovations without spending a penny. - can affordably test new solutions without having to waste money in buying equipment.	- Organisations can use existing infrastructure for tasks that won't require equipment upgrade. - Can benefit from cloud-based IaaS flexibility by companies just paying for the infrastructures needed.
Portability	- Not so portable - Depending on the size of the company, there may be only one or two locations for data centres.	- Cloud providers have locations all over the world. Within each region they will have at least two data centres.	- Depends whether there's a backup on the cloud for the organisations data.

Task 2

Review the case study and identify the business requirements from a reporting perspective and identify additional relevant business questions for reporting purposes.

Questions:

Requirement 1: Which are the top 10 selling items across all the outlets?
Requirement 2: Which are the items that produce most revenue?
Requirement 3: Which are the most popular product groups?
Requirement 4: Which product group provides the most revenue?
Requirement 5: Construct a "league table" of the total weekly and monthly sales of the various outlets to monitor their sales performance.
Requirement 6: Has the revenue increased/decreased over the years?

Part 2: Data Understanding

Task 1

Perform an assessment of the initial dataset to:

- *Understand the data items and how they relate together.*

Once the source data Galleria.txt is opened up in Power BI we can explore it to understand it more and identify problems. The data available consist of 12 different columns which are **SalesDate, TicketNo, Outlet, Total, OrderQty, Stock_Code, Name, Description, Price, Product_Group, Group_name, CardType.**

	SaleDate	TicketNo	Outlet	Total	OrderQty	Stock_Code	Name	Description	Price	Product_Group	Group_name	CardType
1	21/08/2018	92208	Birmingham	5	1	BK0101	Eggs Florentine	NULL		B	Breakfast	Visa
2	21/08/2018	92209	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach and our special tomato sa...		C	Crepes and Galettes	Avis
3	21/08/2018	92211	Middlesborough	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Visa
4	21/08/2018	92217	Birmingham	5	1	BK0101	Eggs Florentine	NULL		B	Breakfast	Avis
5	21/08/2018	92217	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach and our special tomato sa...		C	Crepes and Galettes	Avis
6	21/08/2018	92219	Weymouth	5	1	BK0110	Belgian waffle	NULL		B	Breakfast	Avis
7	21/08/2018	92220	Poole	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Maestercard
8	21/08/2018	92222	Weymouth	5	1	BK0101	Eggs Florentine	NULL		B	Breakfast	Debit Card
9	21/08/2018	92229	Middlesborough	5	1	BK0101	Eggs Florentine	NULL		B	Breakfast	Debit Card
10	21/08/2018	92232	Birmingham	10	1	PZ0020	SAPORITA	Tomato base, Mozzarella, mushrooms, ham, green pepper & red onion		P	Pizza	Debit Card
11	21/08/2018	92233	Birmingham	10	1	PZ0018	4FORMAGGI	Tomato base, Mozzarella, Ricotta, Gorgonzola & Parmesan shavings		P	Pizza	Visa
12	21/08/2018	92235	Middlesborough	9.5	1	PZ0005	CAPRICCIOSA	Tomato base, Mozzarella, ham & mushrooms		P	Pizza	Debit Card
13	21/08/2018	92235	Middlesborough	5	1	BK0110	Belgian waffle	NULL		B	Breakfast	Debit Card
14	21/08/2018	92237	Middlesborough	5	1	BK0110	Belgian waffle	NULL		B	Breakfast	Debit Card
15	21/08/2018	92242	Weymouth	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Debit Card
16	22/08/2018	92247	Peterborough	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Debit Card
17	22/08/2018	92252	Middlesborough	6.5	1	CP0256	Chicken and Vegetable crepe	Chicken, peppers, sweetcorn, cheese, mustard dressing.		C	Crepes and Galettes	Debit Card
18	22/08/2018	92254	Middlesborough	5	1	BK0110	Belgian waffle	NULL		B	Breakfast	Debit Card
19	22/08/2018	92255	Edinburgh	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Avis
20	22/08/2018	92257	Poole	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach and our special tomato sa...		C	Crepes and Galettes	Avis
21	22/08/2018	92260	Poole	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad		C	Crepes and Galettes	Avis

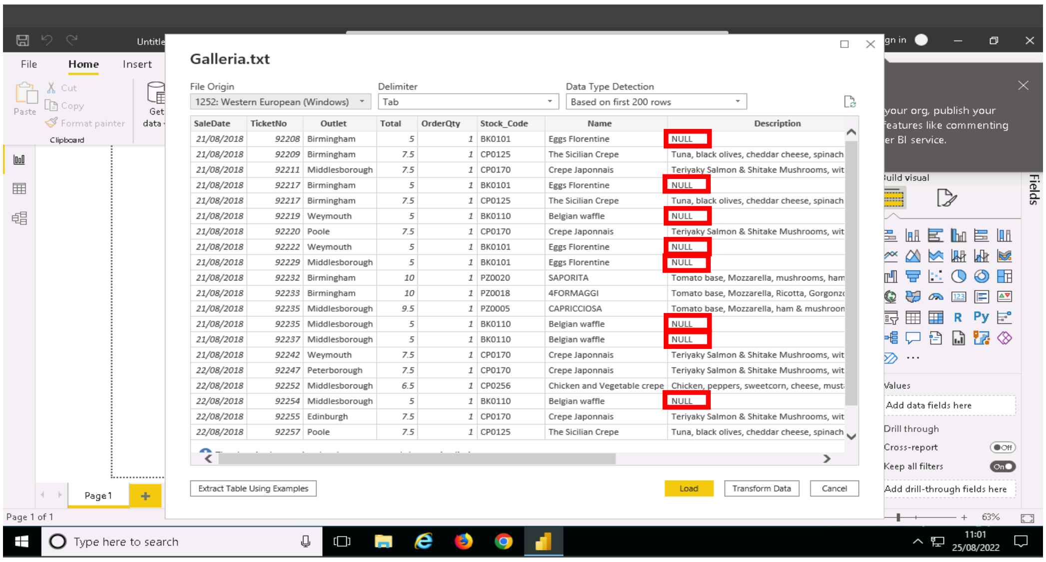
The table below shows the relationship between columns and the details:

Column name	Data Type	Data information
SalesDate	Date	Date of purchase
TicketNo	Whole Number	Order number from customers. Several product names under same ticket number
Outlet	Text	Geographical location of the Outlet
Total	Decimal Number	Total income from order obtained from unit Price multiplied by OrderQty.
OrderQty	Whole Number	Total number of product units ordered
Stock_Code	Text	Code reference to the product name
Name	Text	Product name
Description	Text	Brief description of the product
Price	Decimal Number	Price per unit of the product
Product_Group	Text	Code used to reference the Group_name of product
Group_name	Text	Product group name, used to categorise the product
CardType	Text	Card type used to make the payment by customers

- *Identify any data quality issues which could affect your implementation.*

Instantly after loading up the data in Power BI in the data preview window it's possible to notice null values in the description column which will need taking care of. A discussion with further issues are listed below.

- Missing data, such as NULL values in the Description Column. This should be fixed to 'Unspecified' or removed.



SaleDate	TicketNo	Outlet	Total	OrderQty	Stock_Code	Name	Description
21/08/2018	92208	Birmingham	5	1	BK0101	Eggs Florentine	NULL
21/08/2018	92209	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach
21/08/2018	92211	Middlesborough	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, wit
21/08/2018	92217	Birmingham	5	1	BK0101	Eggs Florentine	NULL
21/08/2018	92217	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach
21/08/2018	92219	Weymouth	5	1	BK0110	Belgian waffle	NULL
21/08/2018	92220	Poole	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, wit
21/08/2018	92222	Weymouth	5	1	BK0101	Eggs Florentine	NULL
21/08/2018	92229	Middlesborough	5	1	BK0101	Eggs Florentine	NULL
21/08/2018	92232	Birmingham	10	1	PZ0020	SAPORITA	Tomato base, Mozzarella, mushrooms, ham
21/08/2018	92233	Birmingham	10	1	PZ0018	4FORMAGGI	Tomato base, Mozzarella, Ricotta, Gorgonz
21/08/2018	92235	Middlesborough	9.5	1	PZ0005	CAPRICCIOSA	Tomato base, Mozzarella, ham & mushroon
21/08/2018	92235	Middlesborough	5	1	BK0110	Belgian waffle	NULL
21/08/2018	92237	Middlesborough	5	1	BK0110	Belgian waffle	NULL
21/08/2018	92242	Weymouth	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, wit
22/08/2018	92247	Peterborough	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, wit
22/08/2018	92252	Middlesborough	6.5	1	CP0256	Chicken and Vegetable crepe	Chicken, peppers, sweetcorn, cheese, must
22/08/2018	92254	Middlesborough	5	1	BK0110	Belgian waffle	NULL
22/08/2018	92255	Edinburgh	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, wit
22/08/2018	92257	Poole	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach

- The column headers are worded inconsistently with mixed types such as being capitalised to split the two words or underscore separated. This should be rectified by reformatting into a consistent format for ease of reference.

Product_Group	Group_name	CardType
B	Breakfast	Visa
C	Crepes and Galettes	Avis
C	Crepes and Galettes	Visa
B	Breakfast	Avis
C	Crepes and Galettes	Avis
B	Breakfast	Avis
C	Crepes and Galettes	Maestercard
B	Breakfast	Debit Card
B	Breakfast	Debit Card
P	Pizza	Debit Card
P	Pizza	Visa
P	Pizza	Debit Card
B	Breakfast	Debit Card
B	Breakfast	Debit Card
C	Crepes and Galettes	Debit Card
C	Crepes and Galettes	Debit Card
C	Crepes and Galettes	Debit Card
B	Breakfast	Debit Card
C	Crepes and Galettes	Avis
C	Crepes and Galettes	Avis
C	Crepes and Galettes	Avis

- **TicketNo** column, are not unique values. Therefore, cannot be used as primary key as a different items can have the same ticket number. Need to delete duplicates and assign a key.

- The **Description** column is not very relevant and could be removed.

Part 3: OLAP Schema Design

Task 1

Identify and justify your OLAP schema design.

In order to create the database, it is required to create a single fact table and different dimensions tables to collate the data.

1. **dimProduct**

The product table contains all the information relevant to each product. Therefore the **ProductID StockCode**, **Name**, **Price, ProductGroup, ProductGroupID.**

2. **dimProductGroup**

This table contains different groups which the products can be loosely categorised as for example Side dishes, Drinks & Beverages etc. It needs the **GroupName** and the **ProductGroupID** columns.

3. **dimPayment**

This table contains the card payment types. Only the **CardType** column will be needed from the source data.

4. **dimOutlet**

This table contains the list of outlets located in the UK therefore it only needs the **Outlet** column from the source data.

5. **dimTicket**

This table contains the ticket numbers generated when the products were sold. Only the **TicketNo** column will be needed from the source data.

6. **dimDate**

This table contains the date in different formats. Only the SaleDate column is needed to create the dimDate table.

7. **factSale**

This table contains all the information related to the sales such as total income from each order item quantity. Also it contains all the primary keys of the dimension tables (which are foreign keys in the fact table).

Task 2

Design and implement the schema in Power BI to satisfy the requirements.

Solution to Part 3, Task 2 is discussed under Part 4, Task 2.

Part 4: Data deliverables

Task 1

Produce a document (Source to Target Mapping) that contains the mapping of source system fields to the fields of the target system.

Source		**Target**		**Transformation**
File	**Column**	**Table**	**Column**	
Galleria.txt	ProductGroup, GroupName	dimProductGroup	ProductGroupID	None
Galleria.txt	StockCode, Name, Price, ProductGroup	dimProduct	ProductID	None
Galleria.txt	CardType	dimPayment	PaymentID	None
Galleria.txt	Outlet	dimOutlet	OutletID	None
Galleria.txt	TicketNo	dimTicket	TicketID	None
Galleria.txt	SaleDate	dimDate	DateID	None
Galleria.txt	Total, OrderQty	factSales	factSalesID	None

Task 2 & Task 3

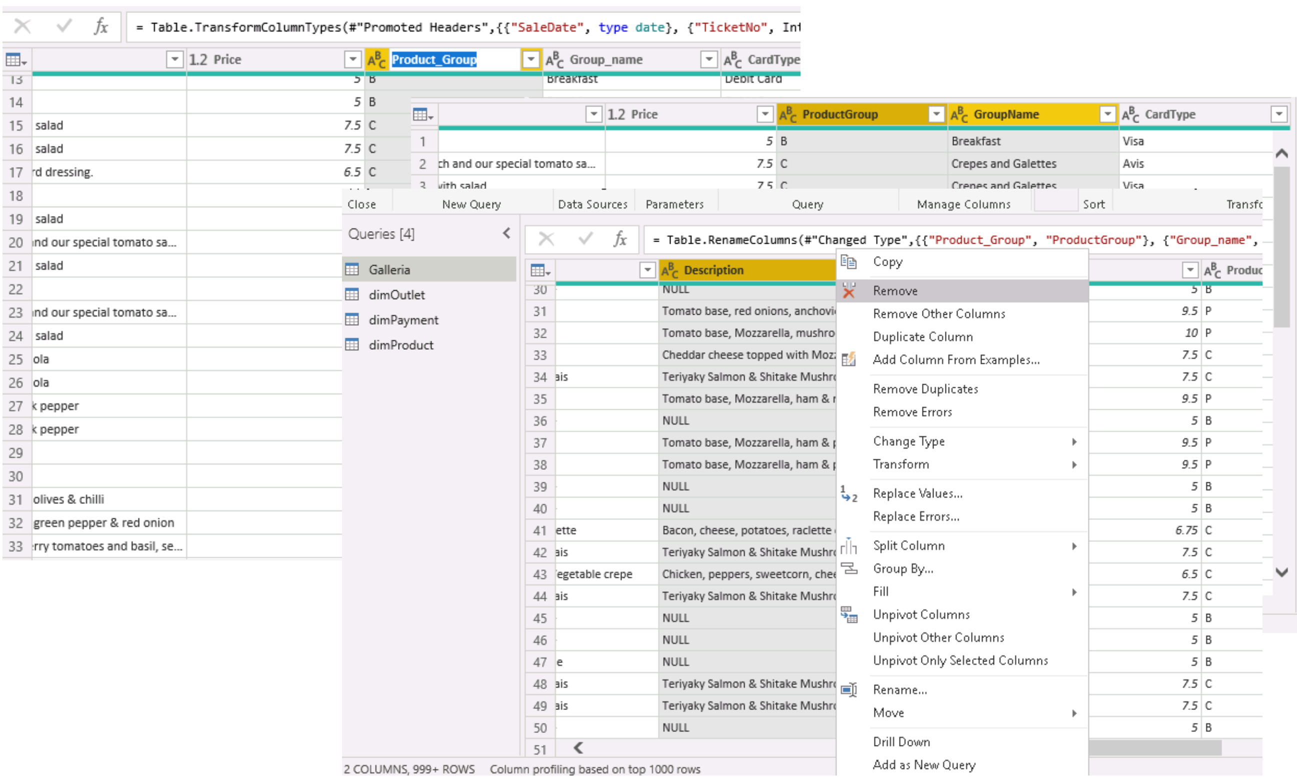
Create the tables in Power BI.

Extract Transform and Load the data using Power BI

Here I talk about the process of creating the tables using Power BI. In the process I also perform ELT processes with the data.

Data cleaning

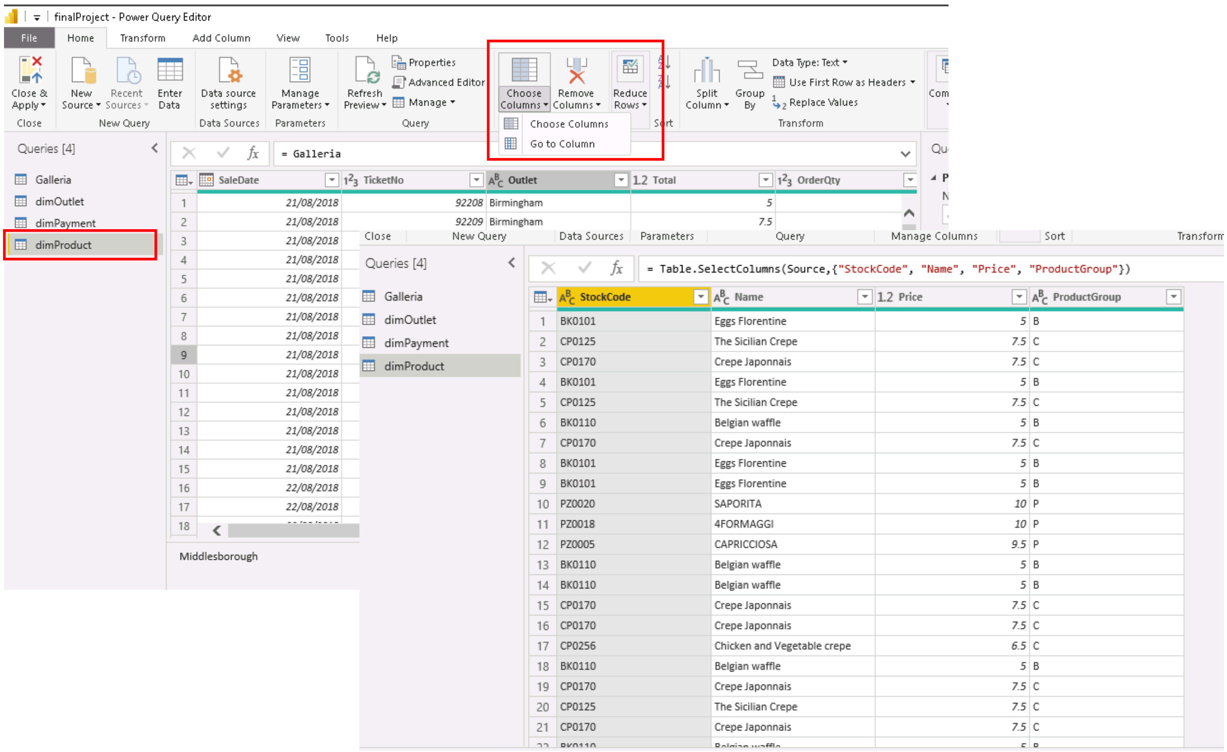
While creating the tables I am also dealing with some data quality issues mentioned in Part 2, Task 1, such as reformatting column headers into a consistent format. Here I change the **Product_Group, Group_name** and **Stock_Code** columns to **ProductGroup**, **GroupName, StockCode** by double clicking on the columns to change the text.



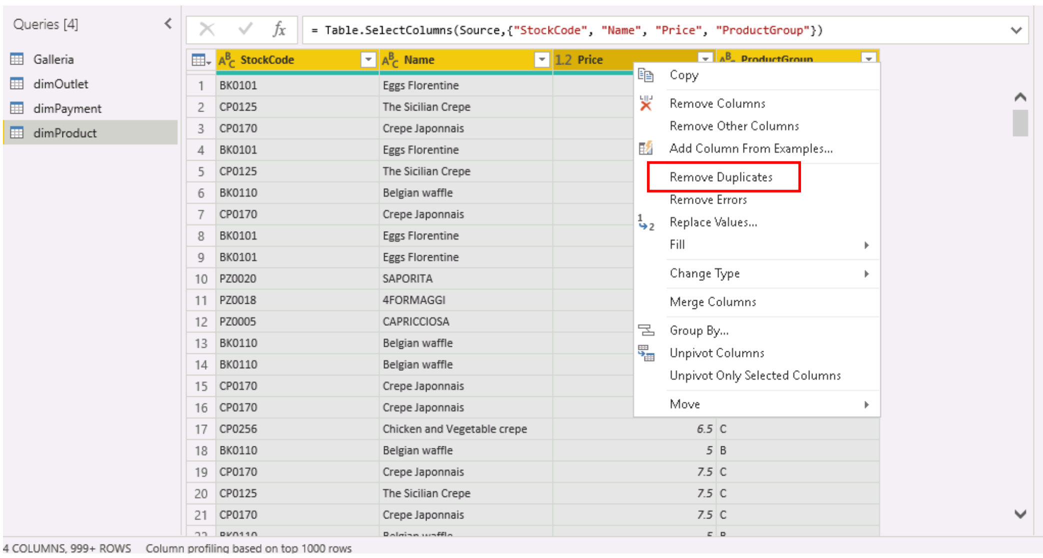
As also mentioned earlier, the **Description** column is not relevant therefore I deleted it by right clicking the relevant header and clicking Remove from the list.

1. dimProduct

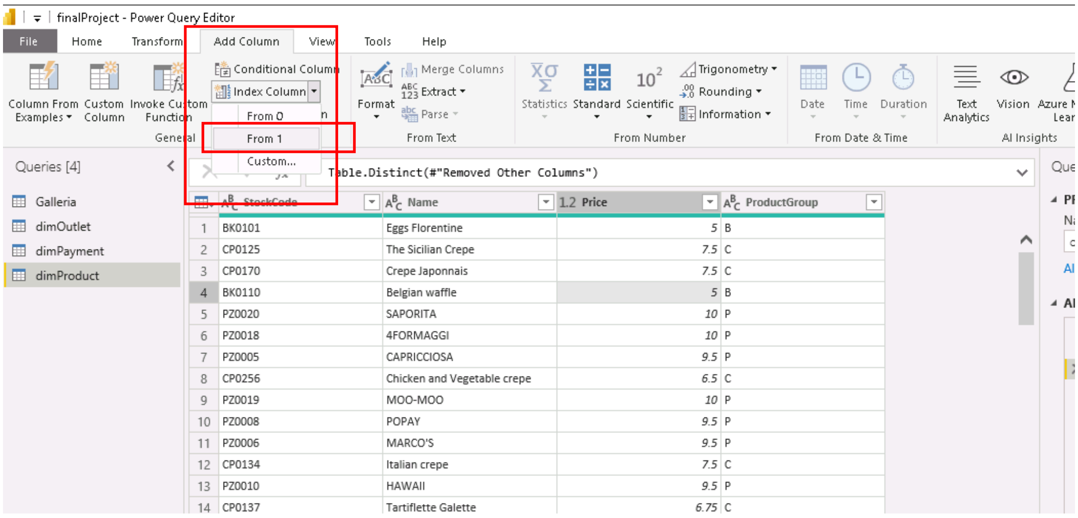
In order to create the dimProduct table, I firstly referenced the source data **Galleria** then renamed the table to dimProduct. I then removed all the columns that were not relevant to the products by going into Choose columns, under Manage Columns.



I then removed all the duplicates by selecting all the columns, then right clicking on any of the column headers to get the popup menu and finally clicking on **Remove Duplicates.**



Then in order to create a ProductID, I did the following steps: **Add Column** > **Index Column > From 1**



I also added another ProductGroupID column to link the dimProduct to the dimProductGroup. The final dimProduct table is shown below.

	ProductID	StockCode	Name	Price	ProductGroup	ProductGroupID
1	1	BK0101	Eggs Florentine	5	B	1
2	4	BK0110	Belgian waffle	5	B	1
3	2	CP0125	The Sicilian Crepe	7.5	C	2
4	3	CP0170	Crepe Japonnais	7.5	C	2
5	8	CP0256	Chicken and Vegetable crepe	6.5	C	2
6	5	PZ0020	SAPORITA	10	P	3
7	6	PZ0018	4FORMAGGI	10	P	3
8	7	PZ0005	CAPRICCIOSA	9.5	P	3
9	9	PZ0019	MOO-MOO	10	P	3
10	10	PZ0008	POPAY	9.5	P	3
11	11	PZ0006	MARCO'S	9.5	P	3
12	12	CP0134	Italian crepe	7.5	C	2
13	13	PZ0010	HAWAII	9.5	P	3
14	14	CP0137	Tartiflette Galette	6.75	C	2
15	15	PZ0009	CALZONE	10	P	3
16	16	PZ0007	MEDITERRANEA	9.5	P	3
17	17	PZ0021	THE DON	10	P	3
18	18	CP0113	Coq au vin	13.5	C	2
19	19	CP0118	Bananas or Strawberries	7.5	D	4
20	20	CP0107	La Vegetarienne	11.5	C	2
21	21	CP0117	Chocolate or Nutella	7.5	D	4

I repeated the steps above to create the following dimension tables listed below, showing the screenshots of the final view.

2. dimProductGroup

	ProductGroup	GroupName	ProductGroupID
1	B	Breakfast	1
2	C	Crepes and Galettes	2
3	P	Pizza	3
4	D	Desserts	4
5	S	Salads	5
6	G	Drinks & Beverages	6
7	K	Side Dishes	7
8	A	Starters/Appetisers	8
9	F	Fish & Seafood	9

3. dimPayment

Queries [5]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet

```
= Table.RenameColumns(#"Reordered Columns",{{"Index", '
```

	PaymentID	CardType
1	1	Visa
2	2	Avis
3	3	Maestercard
4	4	Debit Card

4. dimOutlet

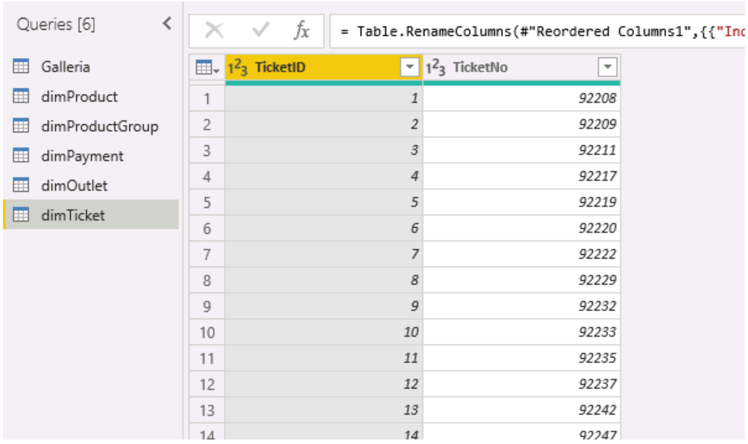
Queries [5]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet

```
= Table.RenameColumns(#"Reordered Columns",{{"Index", "OutletID"}})
```

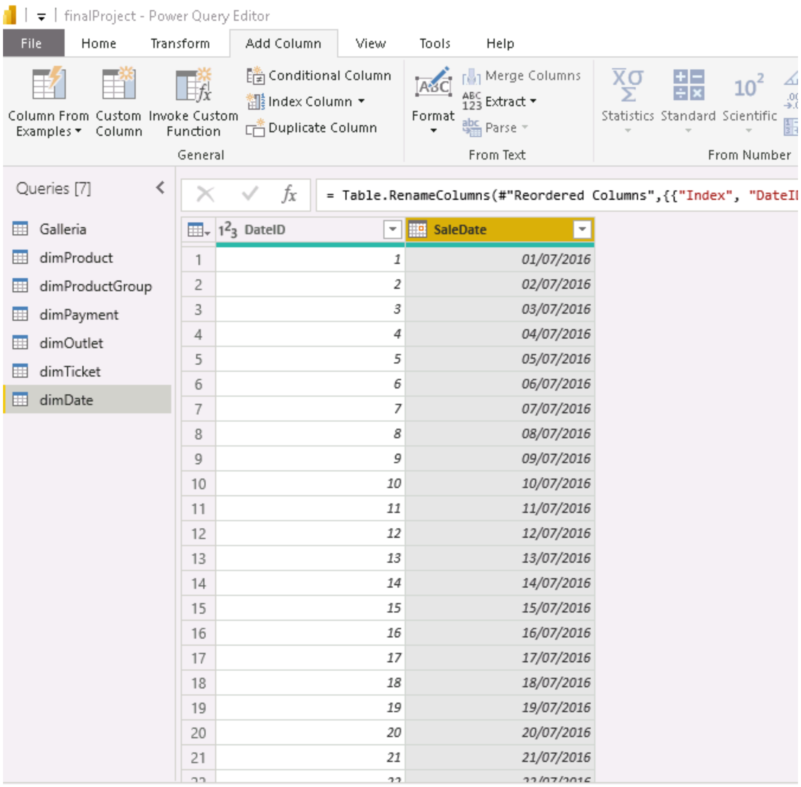
	OutletID	Outlet
1	1	Birmingham
2	2	Middlesborough
3	3	Weymouth
4	4	Poole
5	5	Peterborough
6	6	Edinburgh
7	7	Ipswich
8	8	London
9	9	Cardiff
10	10	Worthing

5. dimTicket



6. dimDate

This table contains the date in different formats. As the steps above, I deleted the duplicate values and created a DateID.



In order to create different time formats, I clicked on **Add column** while selecting the SaleDate column,

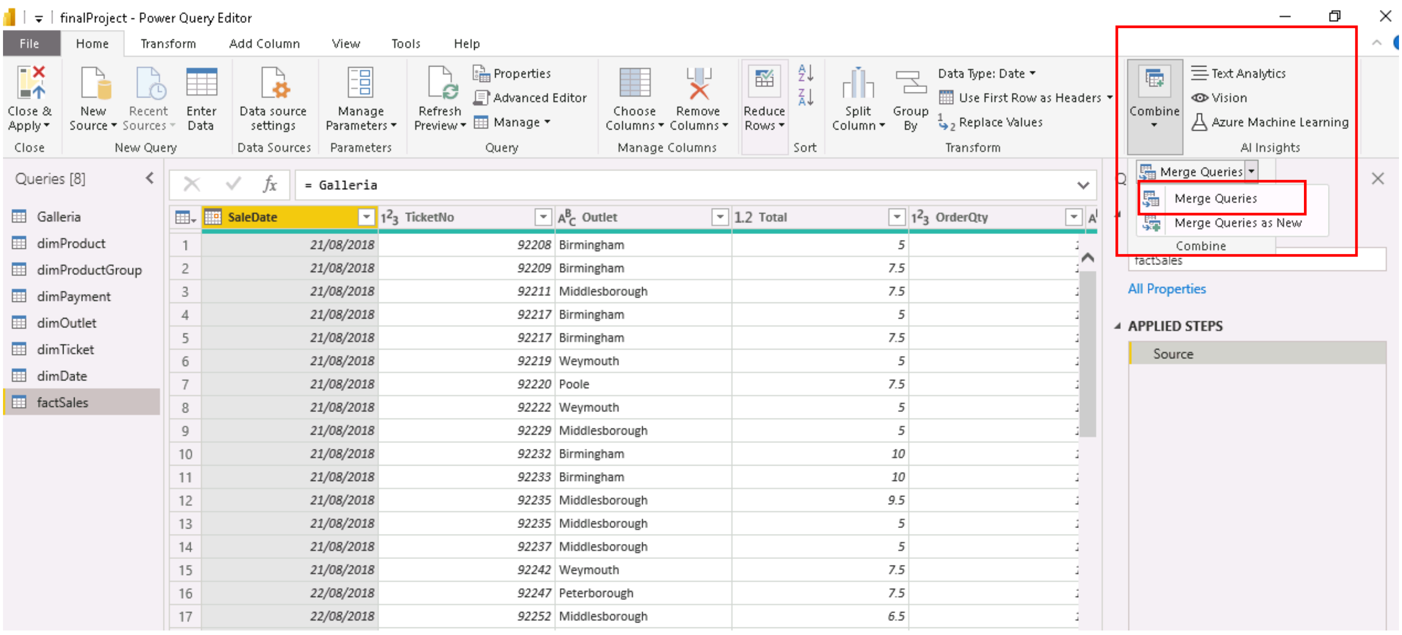
factSales table

The final table that needs to be created is the fact table which contains the quantitative information for the analysis. To create this table we need to join it to other tables and essentially read in all the keys from the different dimension tables.

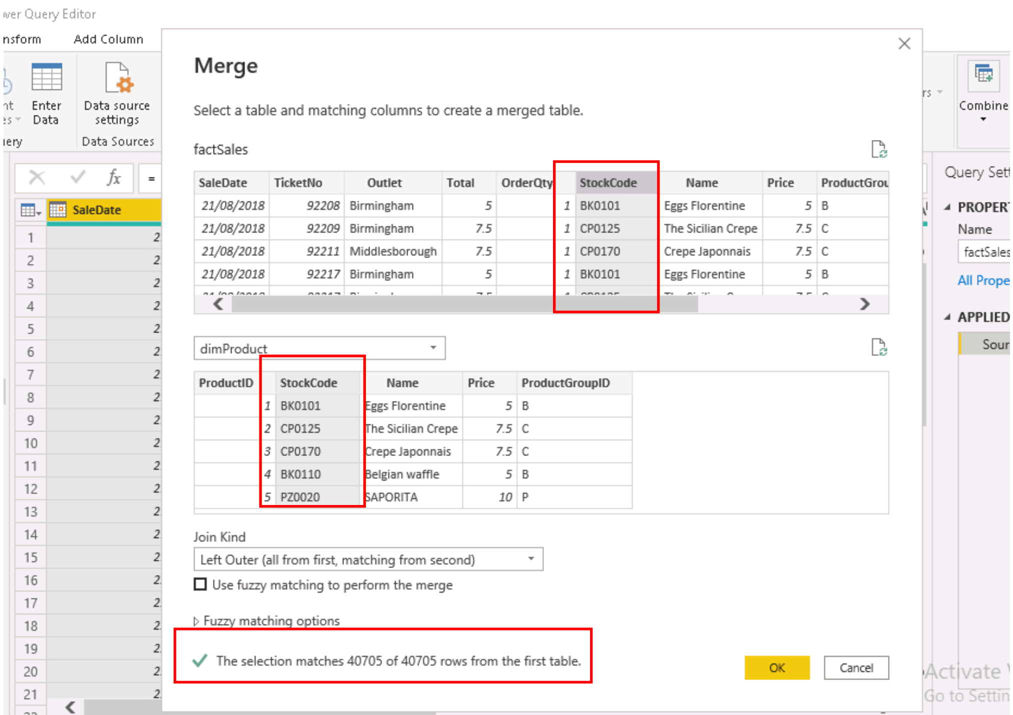
As previously, added a reference to the source data and then named the table **factSales.** This time however, I waited to remove columns as I needed them to perform merging tasks below.

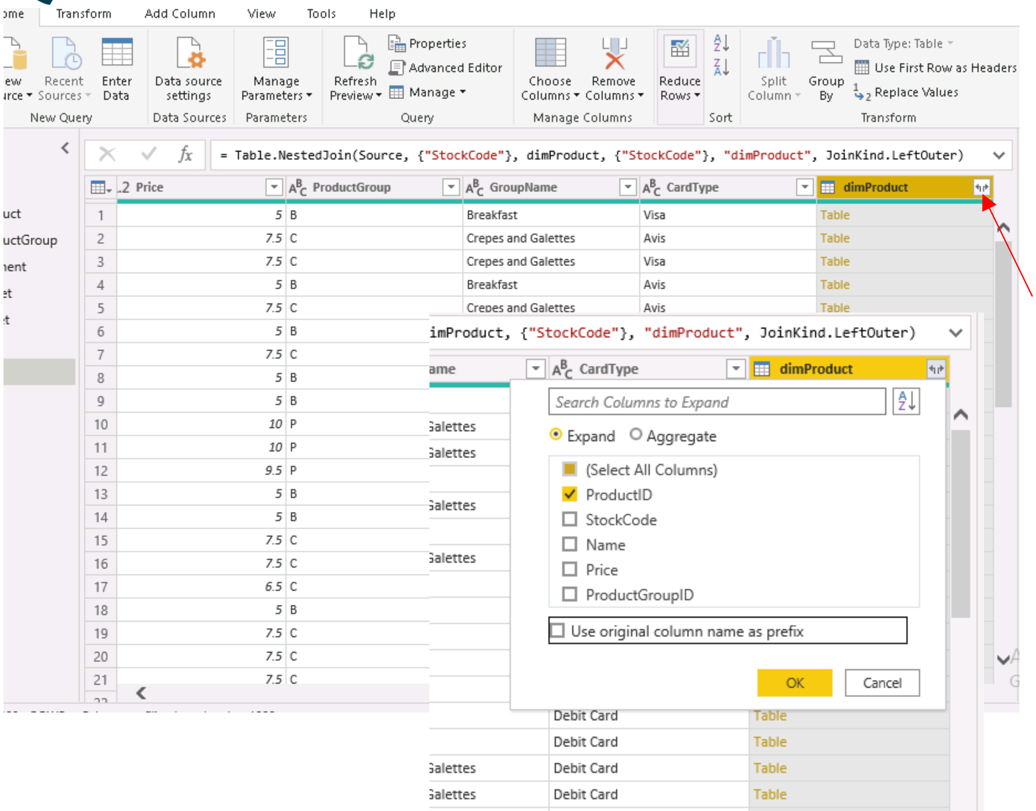
To join all the tables I selected the **factSales query** I went on the Home ribbon, under Combine > Merge Queries > Merge Queries

dimProduct (key: ProductID)



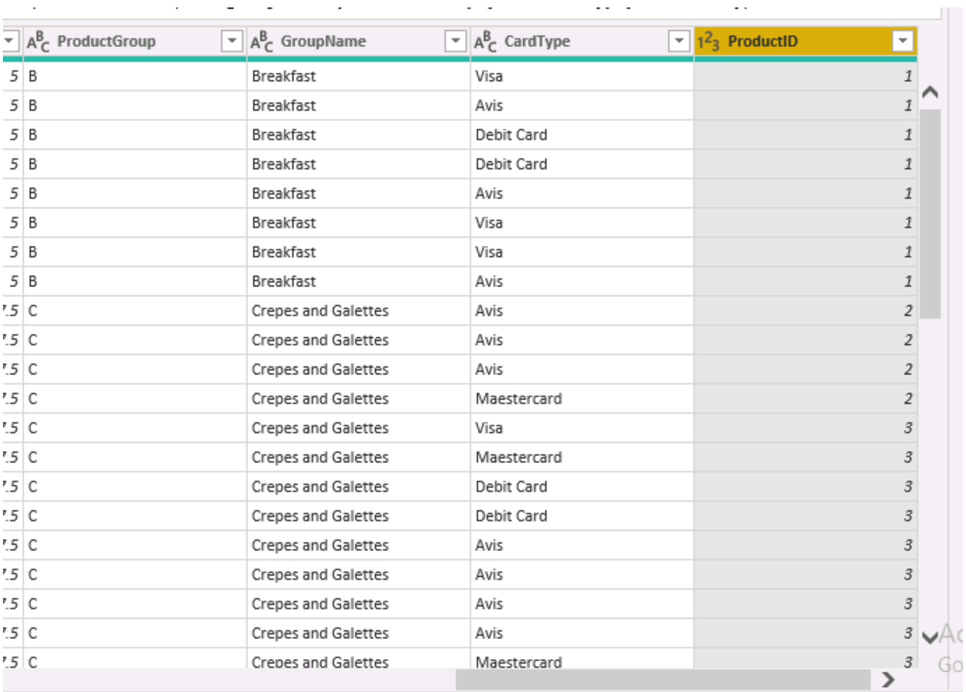
On the Merge popup menu we need to specify the common column where the tables will join. For example, for the dimProduct it is the StockCode column. Always need to check the rows numbers are matching at the bottom. Then I pressed OK.





A new column with dimProduct appears with "Table" as column values. By clicking on the outgoing arrows on the top right corner of the dimProduct column, lets me pick the column I want to return back. In this particular case I ticked ProductID and then OK.

This resulted on adding the primary keys of the ProductID to the fact table as foreign keys.



	ProductGroup	GroupName	CardType	ProductID
5	B	Breakfast	Visa	1
5	B	Breakfast	Avis	1
5	B	Breakfast	Debit Card	1
5	B	Breakfast	Debit Card	1
5	B	Breakfast	Avis	1
5	B	Breakfast	Visa	1
5	B	Breakfast	Visa	1
5	B	Breakfast	Avis	1
'.5	C	Crepes and Galettes	Avis	2
'.5	C	Crepes and Galettes	Avis	2
'.5	C	Crepes and Galettes	Avis	2
'.5	C	Crepes and Galettes	Maestercard	2
'.5	C	Crepes and Galettes	Visa	3
'.5	C	Crepes and Galettes	Maestercard	3
'.5	C	Crepes and Galettes	Debit Card	3
'.5	C	Crepes and Galettes	Debit Card	3
'.5	C	Crepes and Galettes	Avis	3
'.5	C	Crepes and Galettes	Avis	3
'.5	C	Crepes and Galettes	Avis	3
'.5	C	Crepes and Galettes	Avis	3
'.5	C	Crepes and Galettes	Maestercard	3

To link the rest of the dimension tables to the fact table I needed to do similar steps as above.

dimDate (key: DateID)

```
= Table.ExpandTableColumn(#"Merged Queries1", "dimDate", {"DateID"}, {"DateID"})
```

	ProductGroup	GroupName	CardType	ProductID	DateID
1	3	Breakfast	Visa	1	779
2	3	Breakfast	Avis	1	779
3	3	Breakfast	Debit Card	1	779
4	3	Breakfast	Debit Card	1	779
5	:	Crepes and Galettes	Avis	2	779
6	:	Crepes and Galettes	Avis	2	779
7	:	Crepes and Galettes	Visa	3	779
8	:	Crepes and Galettes	Maestercard	3	779
9	:	Crepes and Galettes	Debit Card	3	779
10	3	Breakfast	Avis	4	779
11	3	Breakfast	Debit Card	4	779
12	3	Breakfast	Debit Card	4	779
13	'	Pizza	Debit Card	5	779
14	'	Pizza	Visa	6	779
15	'	Pizza	Debit Card	7	779
16	3	Breakfast	Avis	1	781
17	:	Crepes and Galettes	Avis	3	781
18	:	Crepes and Galettes	Maestercard	3	781
19	:	Crepes and Galettes	Maestercard	3	781
20	:	Crepes and Galettes	Visa	3	781
21	:	Crepes and Galettes	Maestercard	3	781

dimOutlet(key: OutletID)

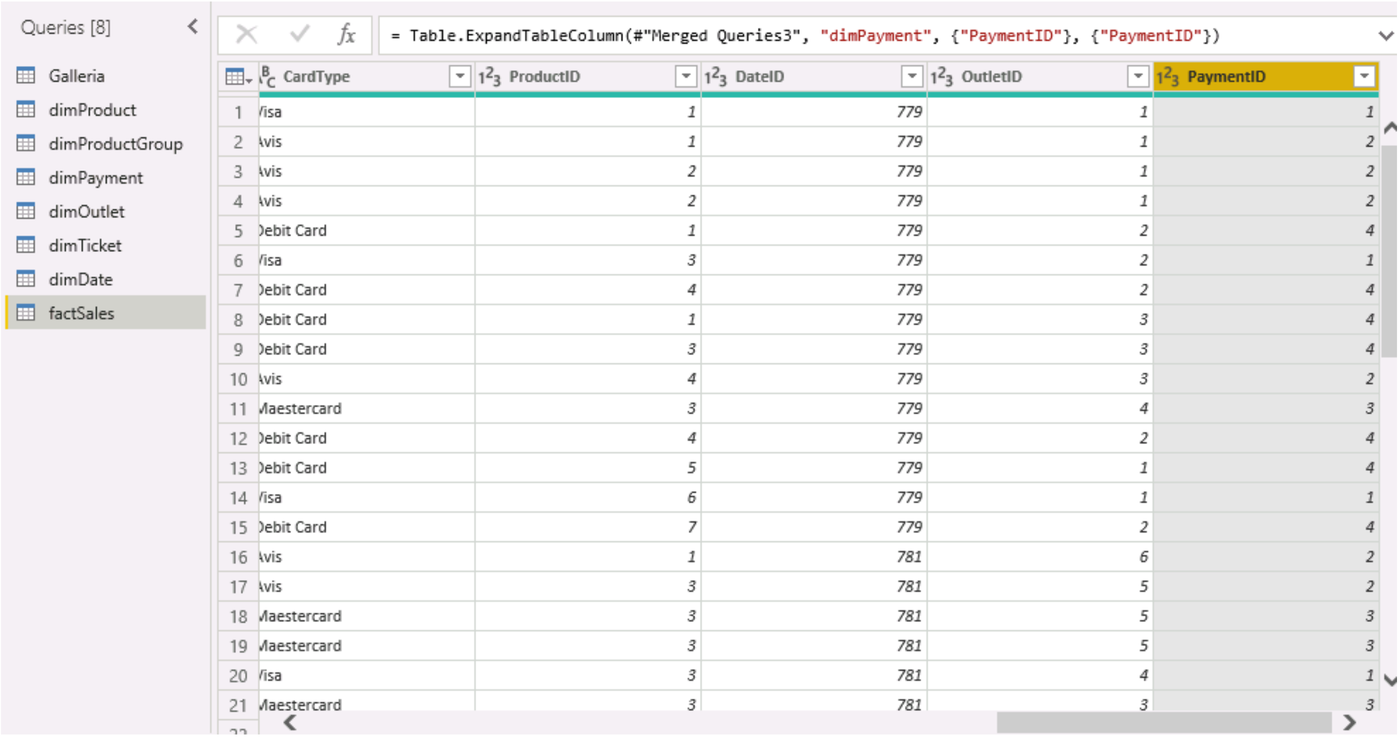
Queries [8]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet
- dimTicket
- dimDate
- factSales

```
= Table.ExpandTableColumn(#"Merged Queries3", "dimPayment", {"PaymentID"}, {"PaymentID"})
```

	GroupName	CardType	ProductID	DateID	OutletID
1	reakfast	Visa	1	779	1
2	reakfast	Avis	1	779	1
3	repes and Galettes	Avis	2	779	1
4	repes and Galettes	Avis	2	779	1
5	reakfast	Debit Card	1	779	2
6	repes and Galettes	Visa	3	779	2
7	reakfast	Debit Card	4	779	2
8	reakfast	Debit Card	1	779	3
9	repes and Galettes	Debit Card	3	779	3
10	reakfast	Avis	4	779	3
11	repes and Galettes	Maestercard	3	779	4
12	reakfast	Debit Card	4	779	2
13	izza	Debit Card	5	779	1
14	izza	Visa	6	779	1
15	izza	Debit Card	7	779	2
16	reakfast	Avis	1	781	6
17	repes and Galettes	Avis	3	781	5
18	repes and Galettes	Maestercard	3	781	5
19	repes and Galettes	Maestercard	3	781	5
20	repes and Galettes	Visa	3	781	4
21	repes and Galettes	Maestercard	3	781	3

dimPayment(key: PaymentID)



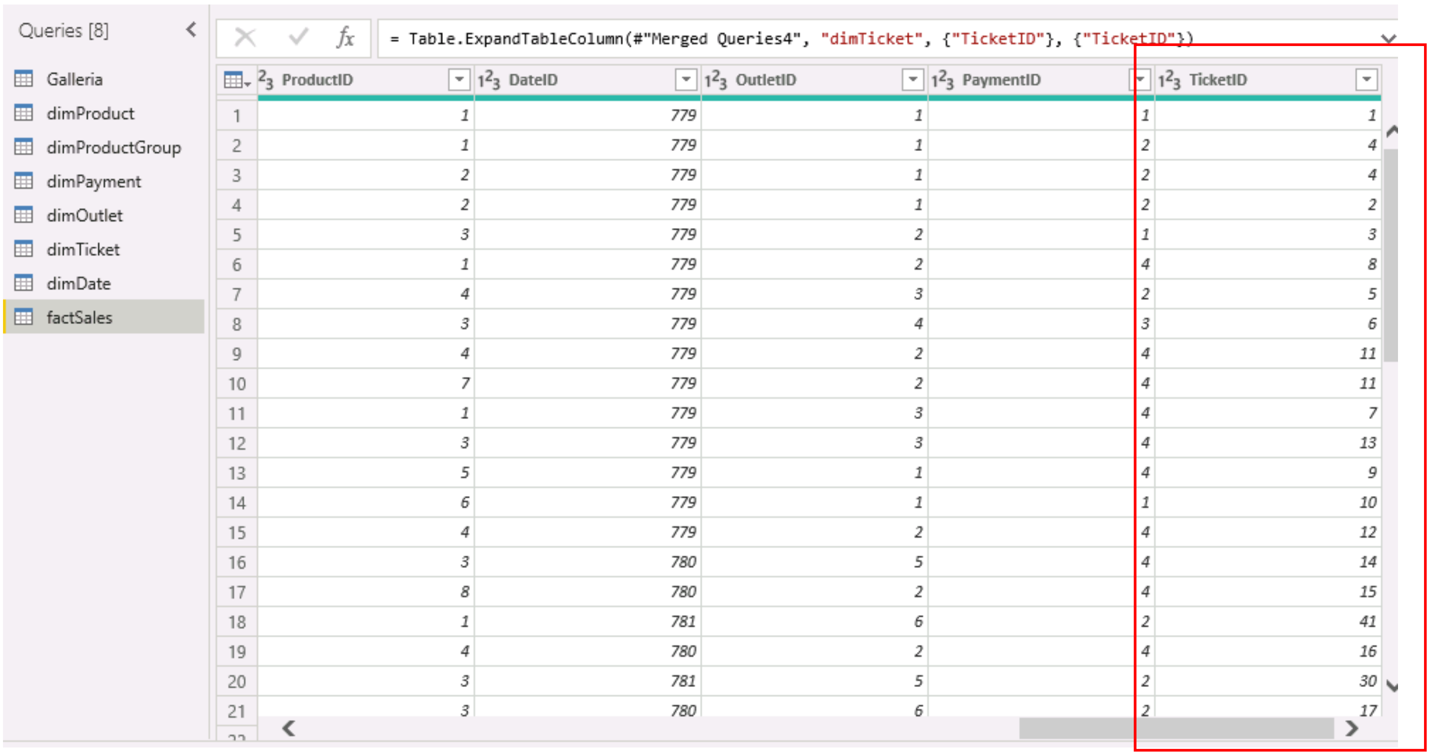
Queries [8]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet
- dimTicket
- dimDate
- factSales

```
= Table.ExpandTableColumn(#"Merged Queries3", "dimPayment", {"PaymentID"}, {"PaymentID"})
```

	CardType	ProductID	DateID	OutletID	PaymentID
1	Visa	1	779	1	1
2	Avis	1	779	1	2
3	Avis	2	779	1	2
4	Avis	2	779	1	2
5	Debit Card	1	779	2	4
6	Visa	3	779	2	1
7	Debit Card	4	779	2	4
8	Debit Card	1	779	3	4
9	Debit Card	3	779	3	4
10	Avis	4	779	3	2
11	Maestercard	3	779	4	3
12	Debit Card	4	779	2	4
13	Debit Card	5	779	1	4
14	Visa	6	779	1	1
15	Debit Card	7	779	2	4
16	Avis	1	781	6	2
17	Avis	3	781	5	2
18	Maestercard	3	781	5	3
19	Maestercard	3	781	5	3
20	Visa	3	781	4	1
21	Maestercard	3	781	3	3

dimTicket (key: TicketID)



Queries [8]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet
- dimTicket
- dimDate
- factSales

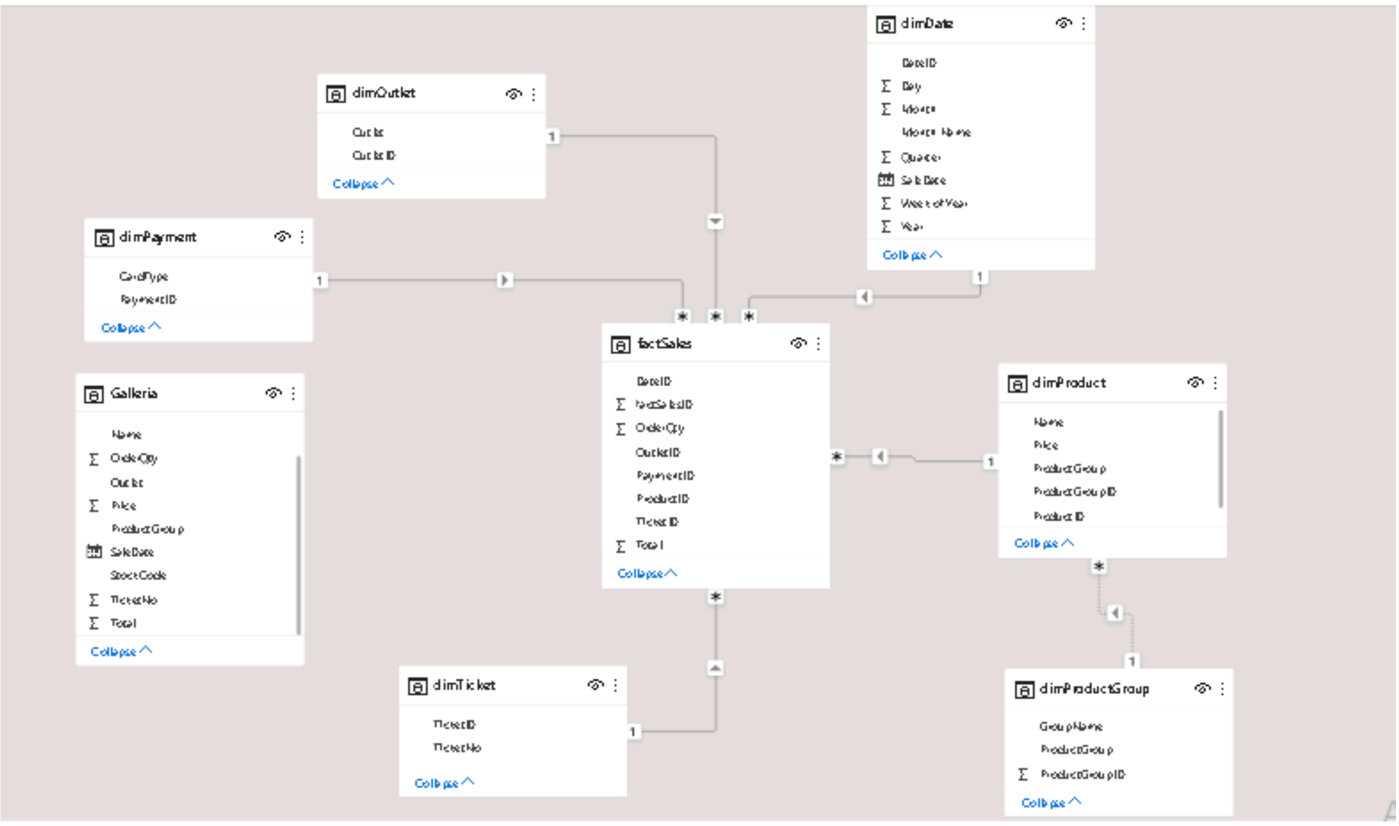
```
= Table.ExpandTableColumn(#"Merged Queries4", "dimTicket", {"TicketID"}, {"TicketID"})
```

	ProductID	DateID	OutletID	PaymentID	TicketID
1	1	779	1	1	1
2	1	779	1	2	4
3	2	779	1	2	4
4	2	779	1	2	2
5	3	779	2	1	3
6	1	779	2	4	8
7	4	779	3	2	5
8	3	779	4	3	6
9	4	779	2	4	11
10	7	779	2	4	11
11	1	779	3	4	7
12	3	779	3	4	13
13	5	779	1	4	9
14	6	779	1	1	10
15	4	779	2	4	12
16	3	780	5	4	14
17	8	780	2	4	15
18	1	781	6	2	41
19	4	780	2	4	16
20	3	781	5	2	30
21	3	780	6	2	17

Finally I removed all the columns from the factSales table that were not needed anymore resulting in the final factSales table as below.

	Total	OrderQty	ProductID	DateID	OutletID	PaymentID	TicketID	factSalesID
1	5	1	1	779	1	1	1	1
2	5	1	1	779	1	2	4	2
3	7.5	1	2	779	1	2	4	3
4	7.5	1	2	779	1	2	2	4
5	7.5	1	3	779	2	1	3	5
6	5	1	4	779	3	2	5	6
7	5	1	1	779	2	4	8	7
8	7.5	1	3	779	4	3	6	8
9	5	1	1	779	3	4	7	9
10	5	1	4	779	2	4	11	10
11	9.5	1	7	779	2	4	11	11
12	10	1	5	779	1	4	9	12
13	10	1	6	779	1	1	10	13
14	5	1	4	779	2	4	12	14
15	7.5	1	3	779	3	4	13	15
16	7.5	1	3	780	5	4	14	16
17	6.5	1	8	780	2	4	15	17
18	5	1	4	780	2	4	16	18
19	7.5	1	3	780	6	2	17	19
20	5	1	1	781	6	2	41	20
21	7.5	1	2	780	4	2	18	21

I then clicked on Close & Apply to apply. This results in the following model.



Part 5: Test approaches

Task 1

Perform some tests to check the four different levels of data testing.

1. ***Entity Level:*** *Review the reporting requirements that were listed as part of the initial data warehouse project brief. Can your design support the requirements?*

Requirement 1: Which are the top 10 selling items across all the outlets?

In order to report on this requirement, I will need the **Name** field from **dimProduct** table and **OrderQty** field in the **factSales** table. Furthermore, adding a filter of type “Top N” will allow me to answer this question

Requirement 2: Which are the items that produce most revenue?

In order to report on this requirement, I will need the **Name** field from **dimProduct** table and the **Total** field in the **factSales** table. These will allow me to answer this question.

Requirement 3: Which are the most popular product groups?

In order to report on this requirement, I will need the order quantity of each item and then group it by the groups of products. I will need the **GroupName** field from the **dimProductGroup** and the **OrderQty** field from the **factSales** table to answer this question.

Requirement 4: Which product group provides the most revenue?

In order to report on this requirement, I will need the total income from each order and then group it by the groups of products. I will need the **GroupName** field from the **dimProductGroup** and the **Total** field from the **factSales** table to answer this question.

Requirement 5: Construct a “league table” of the total weekly and monthly sales of the various outlets to monitor their sales performance.

In order to report on this requirement, I will need the information of the outlets and their total income. Then I will need to filter it by week and month using a slicer. I will need the **Outlet** field from the **dimOutlet** table, **Total** field from the **factSales** table. Furthermore, in order to filter by year, month and week I need the **Year, Month Name, Week of the Year** fields from the **dimDate** table.

Requirement 6: Has the revenue increased/decreased over the years?

In order to report in this I will need information of the total revenue per each year. I will need the **Total** field from **factSales** and the **Year** field from the **dimDate** table.

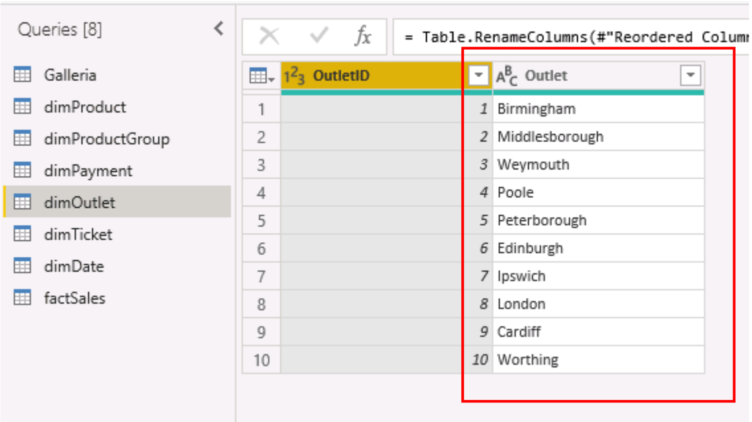
2. *Record Level:*

Run some queries to get the counts for each of your populated tables.
Look back at the original source data and analyse what each entity count should be.
Do your counts align? If not, consider what needs to be fixed.

I am using Pythons pandas library in the Jupyter notebook to run queries for the data testing part using the original source data Galleria.txt.

Outlet column

The target dimension table dimOutlet records 10 entity counts.



Below is the Python output for the Outlet column from the source data.

In [15]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("Galleria.txt", delimiter='\t')

df.head()
```

Out[15]:

	SaleDate	TicketNo	Outlet	Total	OrderQty	Stock_Code	Name	Description	Price	Product_Group	Group_name	CardType
0	2018-08-21	92208	Birmingham	5.0	1	BK0101	Eggs Florentine	NaN	5.0	B	Breakfast	Visa
1	2018-08-21	92209	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach an...	7.5	C	Crepes and Galettes	Avis
2	2018-08-21	92211	Middlesborough	7.5	1	CP0170	Crepe Japonnais	Teriyaky Salmon & Shitake Mushrooms, with salad	7.5	C	Crepes and Galettes	Visa
3	2018-08-21	92217	Birmingham	5.0	1	BK0101	Eggs Florentine	NaN	5.0	B	Breakfast	Avis
4	2018-08-21	92217	Birmingham	7.5	1	CP0125	The Sicilian Crepe	Tuna, black olives, cheddar cheese, spinach an...	7.5	C	Crepes and Galettes	Avis

Number of different outlet locations

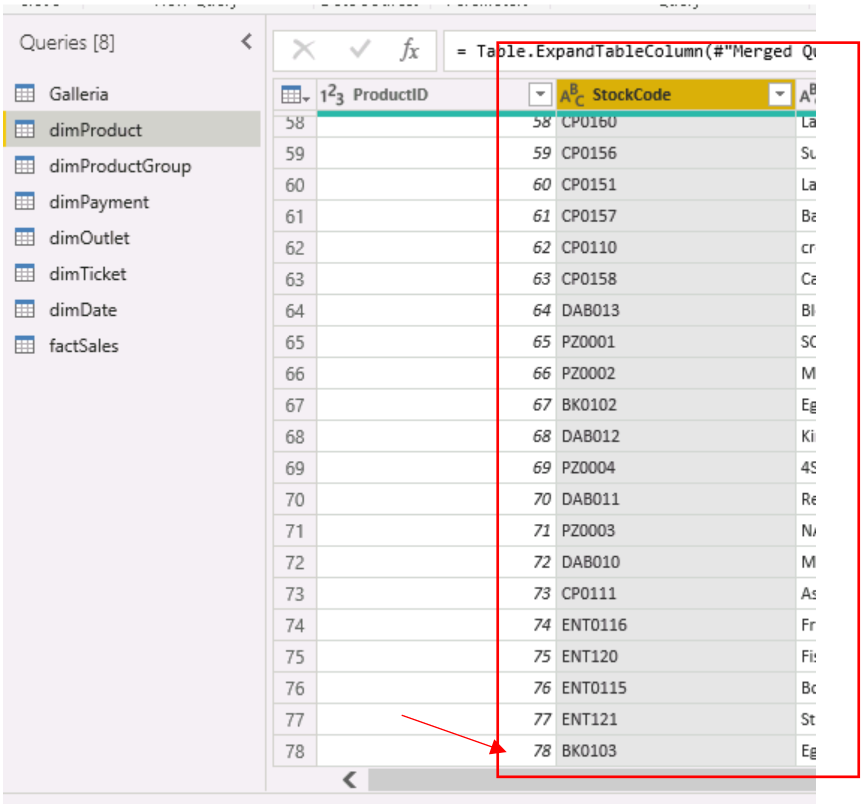
In [26]:
```
outlets = df['Outlet'].unique()
print(outlets)
```

```
['Birmingham' 'Middlesborough' 'Weymouth' 'Poole' 'Peterborough'
 'Edinburgh' 'Ipswich' 'London' 'Cardiff' 'Worthing']
```

Upon analysing the original source data, both outputs produce the same number of outlets.

StockCode

The target dimension table dimProduct, records total of 78 entity counts.



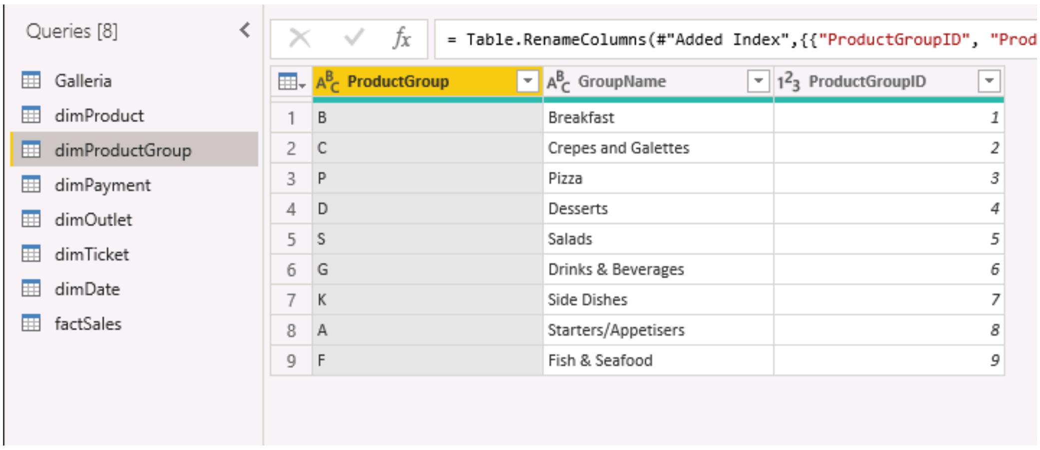
Below is the Python output for the Stock_Code column from the source data.

```
In [34]: stocks = df['Stock_Code'].unique()
         stocksCount = df['Stock_Code'].nunique()

         print(stocks)
         print('Stock Codes count: ',stocksCount)

         ['BK0101' 'CP0125' 'CP0170' 'BK0110' 'PZ0020' 'PZ0018' 'PZ0005' 'CP0256'
          'PZ0019' 'PZ0008' 'PZ0006' 'CP0134' 'PZ0010' 'CP0137' 'PZ0009' 'PZ0007'
          'PZ0021' 'CP0113' 'CP0118' 'CP0107' 'CP0117' 'ICE002' 'CP0102' 'CP0101'
          'ICE004' 'SAL100' 'CP0105' 'CP0112' 'CP0108' 'PZ0011' 'PZ0013' 'PZ0014'
          'PZ0012' 'PZ0017' 'DAB001' 'SAL102' 'DAB008' 'DAB005' 'PZ0016' 'SD0001'
          'SUP121' 'SAL103' 'APP002' 'DAB003' 'DAB009' 'DAB007' 'PZ0015' 'APP001'
          'DAB004' 'DAB006' 'DAB002' 'CP0163' 'BK0162' 'BK0161' 'CP0119' 'CP0159'
          'BK0160' 'CP0160' 'CP0156' 'CP0151' 'CP0157' 'CP0110' 'CP0158' 'DAB013'
          'PZ0001' 'PZ0002' 'BK0102' 'DAB012' 'PZ0004' 'DAB011' 'PZ0003' 'DAB010'
          'CP0111' 'ENT0116' 'ENT120' 'ENT0115' 'ENT121' 'BK0103']
         Stock Codes count:  78
```

Upon analysing the original source data, both outputs produce the same number of Stock_Code.

Product_Group

The target dimension table **dimProductGroup**, records total of 9 entity counts.



	ProductGroup	GroupName	ProductGroupID
1	B	Breakfast	1
2	C	Crepes and Galettes	2
3	P	Pizza	3
4	D	Desserts	4
5	S	Salads	5
6	G	Drinks & Beverages	6
7	K	Side Dishes	7
8	A	Starters/Appetisers	8
9	F	Fish & Seafood	9

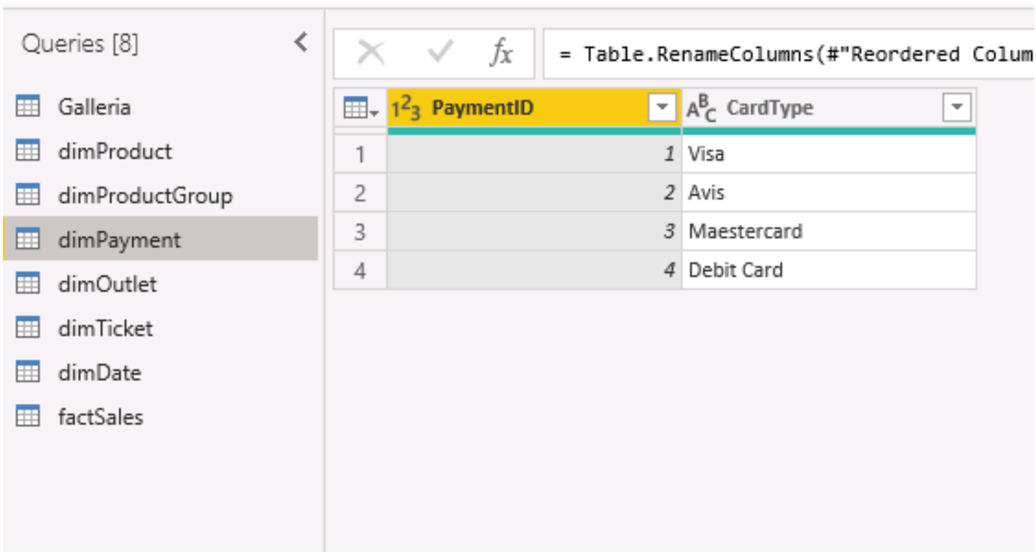
Below is the Python output for the Product_Group column from the source data.

```
In [36]: productGroups = df['Product_Group'].unique()
         productGroupCount = df['Product_Group'].nunique()

         print(productGroups)
         print('Product Group count: ',productGroupCount)

         ['B    ' 'C' 'P    ' 'D    ' 'S    ' 'G    ' 'K    ' 'A    ' 'F    ']
         Product Group count:  9
```

Upon analysing the original source data, both outputs produce the same number of Stock_Code.

CardPayment

The target dimension table **dimPayment**, records total of 4 entity counts.



	PaymentID	CardType
1	1	Visa
2	2	Avis
3	3	Maestercard
4	4	Debit Card

Below is the Python output for the CardType column from the source data.

```
In [37]: cards = df['CardType'].unique()
         cardsCount = df['CardType'].nunique()

         print(cards)
         print('Card types: ',cardsCount)

         ['Visa' 'Avis' 'Maestercard' 'Debit Card']
         Card types:  4
```

Upon analysing the original source data, both outputs produce the same number of CardType.

SaleDate

The target dimension table **dimDate**, records total of 1123 days count.

Queries [8]

- Galleria
- dimProduct
- dimProductGroup
- dimPayment
- dimOutlet
- dimTicket
- dimDate
- factSales

= Table.ReorderColumns(#"Inserted Day",{"DateID", "Day", "SaleDate", "Week of Year", "Month", "Month

	DateID	Day	SaleDate	Week of Year	Month
1103	1103	11	11/07/2019	28	
1104	1104	12	12/07/2019	28	
1105	1105	13	13/07/2019	28	
1106	1106	14	14/07/2019	28	
1107	1107	15	15/07/2019	29	
1108	1108	16	16/07/2019	29	
1109	1109	17	17/07/2019	29	
1110	1110	18	18/07/2019	29	
1111	1111	19	19/07/2019	29	
1112	1112	20	20/07/2019	29	
1113	1113	21	21/07/2019	29	
1114	1114	22	22/07/2019	30	
1115	1115	23	23/07/2019	30	
1116	1116	24	24/07/2019	30	
1117	1117	25	25/07/2019	30	
1118	1118	26	26/07/2019	30	
1119	1119	27	27/07/2019	30	
1120	1120	28	28/07/2019	30	
1121	1121	29	29/07/2019	31	
1122	1122	30	30/07/2019	31	
1123	1123	31	31/07/2019	31	

Below is the Python output for the SaleDate column from the source data.

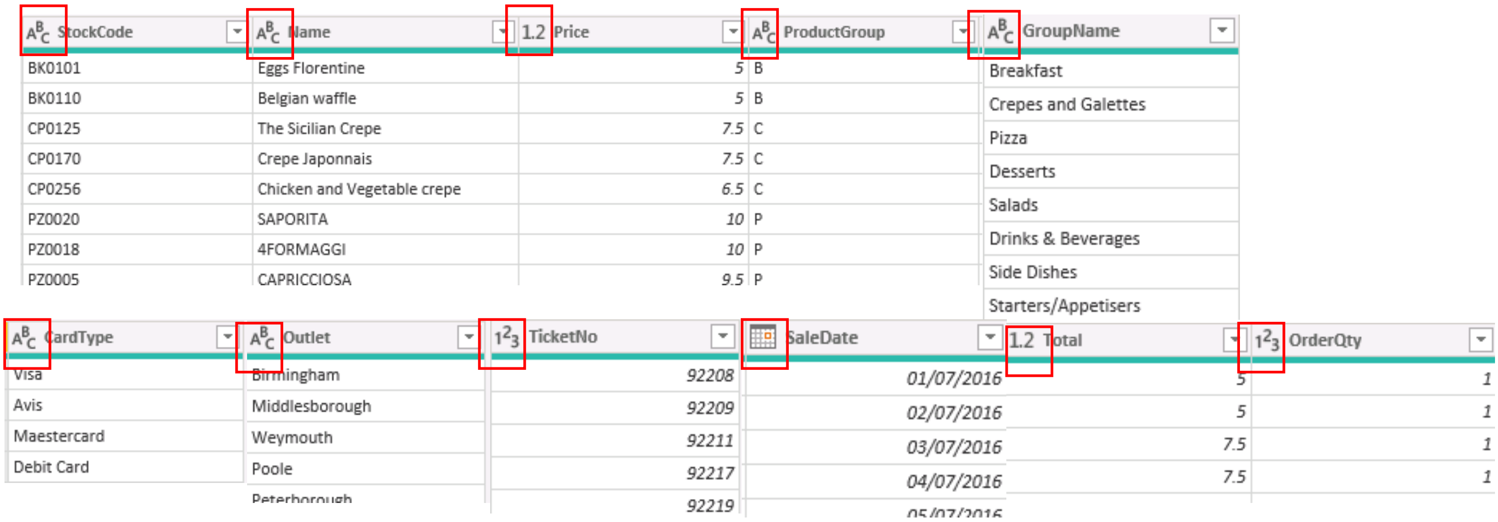
```
In [39]: dates = df['SaleDate'].unique()
         datesCount = df['SaleDate'].nunique()

         print(dates)
         print('Days Count: ',datesCount)

         ['2018-08-21' '2018-08-22' '2018-08-23' ... '2018-08-18' '2018-08-19'
          '2018-08-20']
         Days Count:  1123
```

Upon analysing the original source data, both outputs produce the same number of days of sales.

3. ***Column Level:***

Check your table structures against the original source data. Have you accounted for all the data items?

When we are talking about table structure, we are referring to the data types. Below are the columns visuals in Power BI. On the left of each column header, there is a logo which represents the data type.



Using the legend in **Fig1** below from Power BI I determined what each logo represents in terms of data types and tabulated it in **Table 1** below. Then I used pandas **dtypes** function to verify the column types of the source data; the results are in **Fig 2** below.

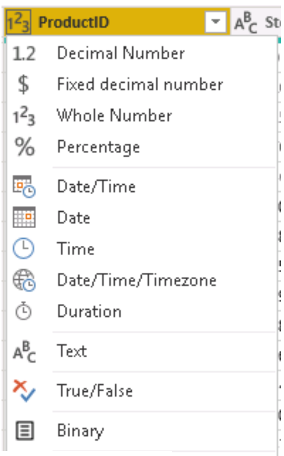


Fig 1

Column name	Data Type
SalesDate	Date
TicketNo	Whole Number
Outlet	Text
Total	Decimal Number
OrderQty	Whole Number
Stock_Code	Text
Name	Text
Description	Text
Price	Decimal Number
Product_Group	Text
Group_name	Text
CardType	Text

Table 1

DATA TYPES

```
In [8]: df.dtypes
Out[8]: SaleDate         object
        TicketNo          int64
        Outlet           object
        Total           float64
        OrderQty          int64
        Stock_Code       object
        Name             object
        Description      object
        Price           float64
        Product_Group    object
        Group_name       object
        CardType         object
        dtype: object
```

Fig 2

Upon analysing the outputs the data types match for the Price, Total , TicketNo and Outlet columns. The SaleDate, Outlet, Stock_Code, Name, Description, Product_Group, Group_name, CardType are listed as “object” types from the pandas output. However, in pandas “object” data type can mean either strings (equivalent to text) and mixed numeric or non-numeric values. So if using pandas for further analysis, the date column should be handled with additional techniques to be treated as dates.

4. Column Value Level:

Run some queries to get frequency counts of your columns. As a minimum, pick one column per table and do a frequency count of that column.
Do the same in your source data. Do the values match?

For the column value level of testing, using Power BI, I ran some queries on the dimension tables and created tables indicating the total order quantity for some of the columns. I then ran queries using pandas using the source data to verify whether the values match. Below are the outputs of the two methods I used for testing.

Outlet – from dimOutlet

Outlet	OrderQty
Birmingham	12803
Cardiff	8359
Edinburgh	19153
Ipswich	5830
London	8626
Middlesborough	23413
Peterborough	4652
Poole	5644
Weymouth	3094
Worthing	8673
Total	**100247**

```
In [79]: df.groupby('Outlet', as_index=False).agg({"OrderQty": "sum"})
Out[79]:
```

	Outlet	OrderQty
0	Birmingham	12803
1	Cardiff	8359
2	Edinburgh	19153
3	Ipswich	5830
4	London	8626
5	Middlesborough	23413
6	Peterborough	4652
7	Poole	5644
8	Weymouth	3094
9	Worthing	8673

SaleDate – from dimDate (per year)

Year	OrderQty
2016	11484
2017	36861
2018	36573
2019	15329
Total	**100247**

```
In [77]: df.groupby('year', as_index=False).agg({"OrderQty": "sum"})
Out[77]:
```

	year	OrderQty
0	2016	11484
1	2017	36861
2	2018	36573
3	2019	15329

StockCode - from dimProduct

As the table was too large, I am showing only the first 20 and last 20 items from the Power BI table. For pandas I am showing the head and tail of the table.

Name	StockCode	OrderQty
Assiette de pate	APP001	996
Assiette de fromages	APP002	697
Eggs Florentine	BK0101	6434
Eggs Benedict	BK0102	1107
Eggs Royale	BK0103	90
Belgian waffle	BK0110	6636
chicken 'n cheese omelette	BK0160	156
Scrambled eggs	BK0161	710
Ham 'n Cheese Omelette	BK0162	613
La Superbe Banane Quebec	CP0101	486
Hazelnut Chocolate Cream	CP0102	109
Epinards creme/creamed spinach	CP0105	238
La Vegetarienne	CP0107	1581
Poulet et fromage de chevre	CP0108	347
creme caramel	CP0110	419
Asperges et Champignons Mornay	CP0111	173
Poulet au curry	CP0112	649
Coq au vin	CP0113	1173
Chocolate or Nutella	CP0117	659
Bananas or Strawberries	CP0118	1200
Total		**100247**

Name	StockCode	OrderQty
MEDITERRANEA	PZ0007	2183
POPAY	PZ0008	2947
CALZONE	PZ0009	2623
HAWAII	PZ0010	2072
SALAME	PZ0011	676
COLAZIONE	PZ0012	768
GIARDINO	PZ0013	736
TONNO CIPPOLA	PZ0014	732
TRENTINA	PZ0015	889
DIAVOLA	PZ0016	805
FIORENTINA	PZ0017	632
4FORMAGGI	PZ0018	935
MOO-MOO	PZ0019	1631
SAPORITA	PZ0020	1482
THE DON	PZ0021	1234
Salade Parisienne	SAL100	485
Salade Nicoise	SAL102	991
Salade Maison	SAL103	671
French fries	SD0001	1893
Soup a l'oignon gratinee	SUP121	100
Total		**100247**

In [78]:
```
df.groupby('Stock_Code', as_index=False).agg({"OrderQty": "sum"})
```

Out[78]:

	Stock_Code	OrderQty
0	APP001	996
1	APP002	697
2	BK0101	6434
3	BK0102	1107
4	BK0103	90
...	...	...
73	SAL100	485
74	SAL102	991
75	SAL103	671
76	SD0001	1893
77	SUP121	100

78 rows × 2 columns

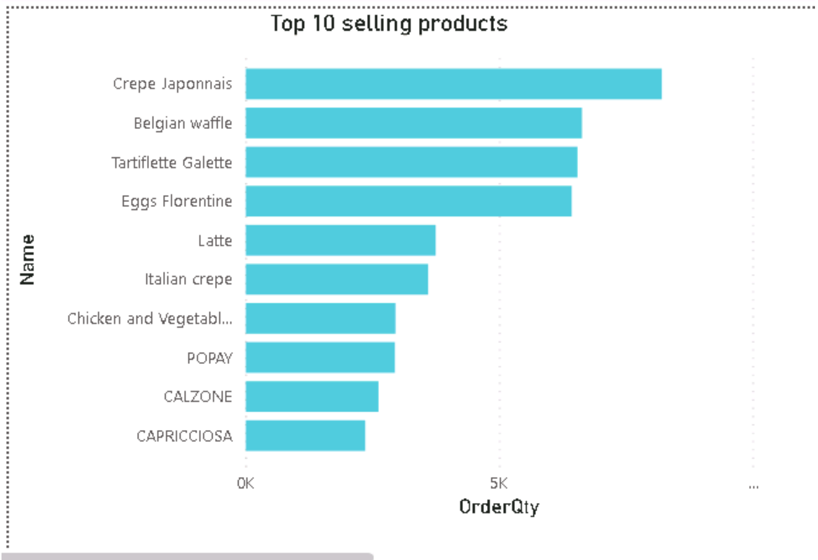
Part 6: Measures and Visualisations

Task 1

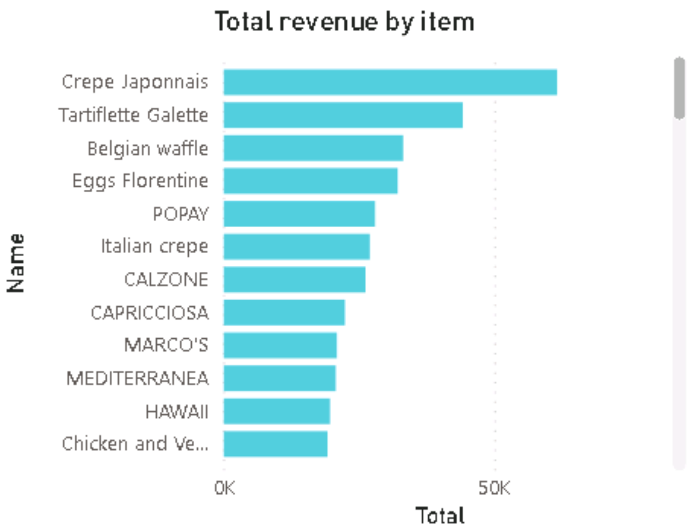
Create visualisations to meet the original reporting requirements. Include any additional visualisations to support the extra business questions you identified in Part 1.

Requirement 1: Which are the top 10 selling items across all the outlets?

By plotting a clustered bar chart of product name vs order quantity in Power BI I can get a ranking visual. I added a Top N filter type, in the Name field to only show me the top 10 products across all the regions.



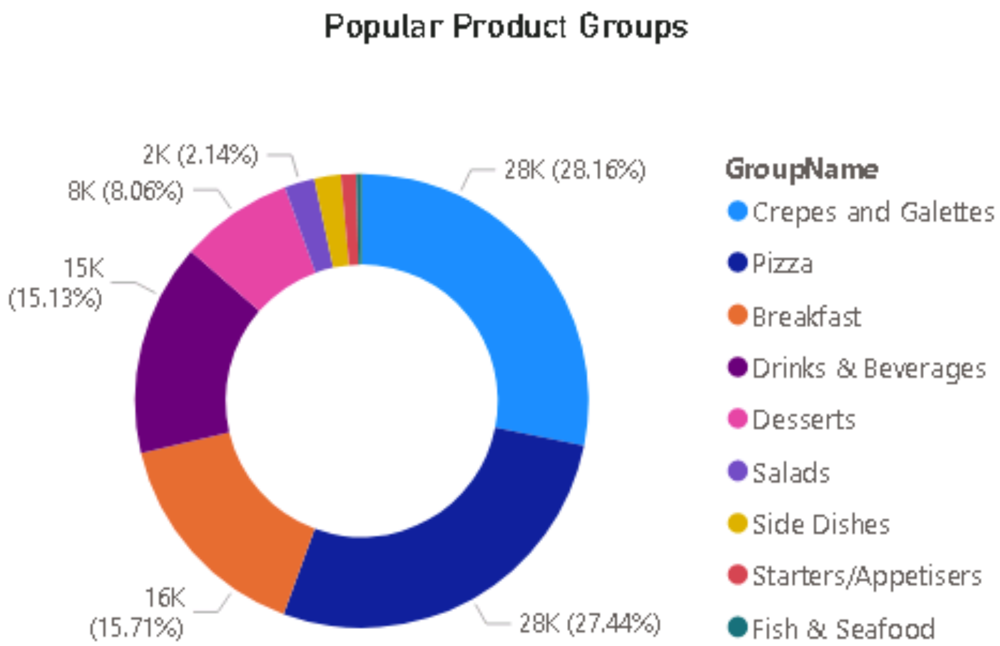
Requirement 2: Which are the items that produce most revenue?



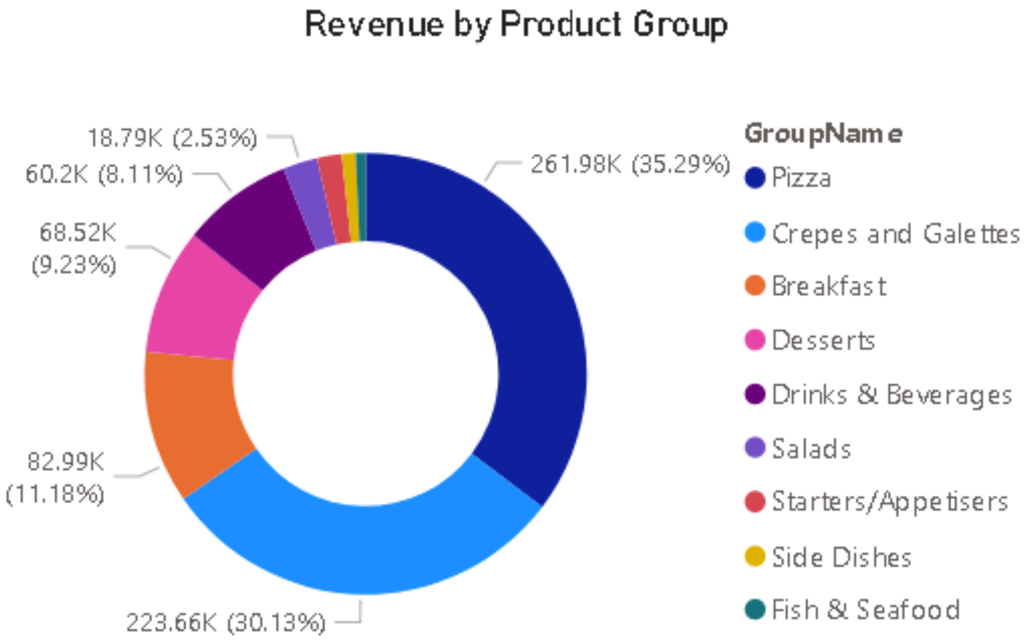
Similarly to the previous visualisation, I plotted a clustered bar chart of product name vs total amount sold in GBP. By scrolling down in the Power BI dashboard it’s possible to see the least favourites.

Requirement 3: Which are the most popular product groups?

To report on this requirement, I choose to use a donut chart. As we can see from the chart the most popular product groups are Crepes and Gallettes, Pizza, Breakfast, Drinks & Bevarages. Fish & Seafood were the least popular product group.



Requirement 4: Which product group provides the most revenue?



Requirement 5: Construct a "league table" of the total weekly and monthly sales of the various outlets to monitor their sales performance.

Sales League Table

Outlet	Total
Middlesborough	171,995.75
Edinburgh	140,184.25
Birmingham	99,457.75
London	63,510.75
Worthing	62,498.75
Cardiff	60,655.50
Ipswich	44,163.50
Poole	41,820.25
Peterborough	35,930.00
Weymouth	22,149.75
Total	**742,366.25**

Time-based filter

- ☐ 2016
- ☐ 2017
- ☐ 2018
- ☐ 2019

Sales League Table

Outlet	Total
Edinburgh	2,687.75
Middlesborough	2,487.00
Birmingham	1,719.50
Ipswich	1,281.25
Cardiff	1,112.00
Worthing	1,057.00
London	1,040.75
Weymouth	867.25
Poole	684.50
Peterborough	411.75
Total	**13,348.75**

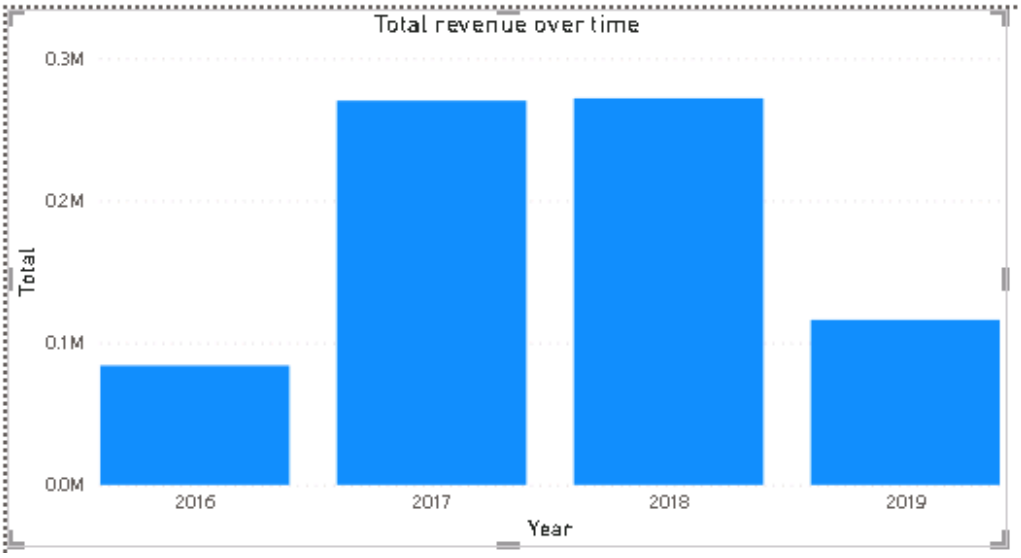
Time-based filter

- ☐ 2018
- ▣ 2019
  - ▣ April
    - ■ 14
    - ☐ 15

I created a table with each outlet and their respective overall total sales. I then added a Slicer visual at the bottom to filter the sales by Year, Month and Week of Year and monitor the sales performance of the different outlets. For example, if I wanted to see the sales performance for each outlet of the first week of April of the year 2019 I can do so as shown in the figure on the right.

Requirement 6: Has the revenue increased/decreased over the years?

I created a bar chart with total sales per year to answer this question. After analysing the data, it is possible to see that the revenue increased from 2016 till 2018. However, it dropped by more than 50% from 2018 to 2019.



Task 2

Identify and create measures in Power BI for non-additive and semi-additive facts, so they can be used in visualisations and reports.

We don't really have non-additive and semi-additive facts within the Galleria source data, hence we cannot create any measures in Power BI for this.