

**A Technical report on**  
**Historical sales of Supermarket in most populated cities**

*By:*

**Robin Roy**

*Robinroy9944@gmail.com*

**DATA ANALYTICS**

# Table of Contents

## 1. INTRODUCTION

1.2. Reasons for selecting the subject area and data

1.3. Vision and Goals

1.4. Key StakeHolders

1.5. Business requirements

## 2.SCHEMA

## 3. ETL

## 4. VISUALIZATIONS AND REPORTS

### 4.1. REPORTS

### 4.2. VISUALIZATIONS

## 5.GRAPH DATABASES

### 5.1. COMPARISON TO RELATIONAL DATABASES

## 6. CONCLUSIONS

## 7. BIBLIOGRAPHY

Appendix A – Fact table script

Appendix B – Dimension table script

Appendix C - List of tables Script

Appendix D - Neo4J Code

Appendix E - Data Source

## 1. INTRODUCTION

The dataset chosen has been taken from Kaggle that provides the info regarding the growth of supermarkets in most populated cities. The dataset holds the entries of historical sales of supermarkets. The data is collected from three different branches denoted as A,B and C. The period the data is collected is for three months duration.

Since this dataset contains a several number of entries under different columns, it is quite time consuming and intensive in regards to data handling. We use database management system (DBMS) to come over this to manage larger amount of data to meaningful conclusions.

### 1.2. Reasons for selecting the data

The dataset "Supermarket Sales" studied in this report has been selected as it contains details about the sales of the supermarket at different branches which is divided as A, B and C in the most populated cities. The data set provides all the details about the sales made including the tax paid info.

Also predict data analytics methods are easy to perform on this data.

### 1.3. Vision and Goals

- ✚ Understand vendors: - The competition of supermarkets in popular cities is very high. To increase the sales in the coming months, it is very important to study the recent high sales pattern. It will help us to understand the type or the category of goods sold. The report that will be generated gives us the idea of which products performing best and which is low. It can let the company avoid bulking less required goods and ensuring the availability of most wanted goods.
- ✚ Efficiency: - The current database of the supermarket sales at different branches and doesn't provide any security and there is possibility of unwanted access to the database. We can create a Relational Database;

the mission is to increase the security and to make updates or upgrades to the system when required.

- ✚ Planning: - With the assistance of reports and representation. These reports of buying pattern records help to envision and plan for what's to come while increasing profit margins and reduction of losses. The company can set the margins on the products as the increasing demand can be easy understood and offers can be given to the less sold products.

### 1.4. Key Stakeholders

In a corporation, a stakeholder is a member of "groups without whose support the organization would cease to exist", they are the essential partners in a regular organization are its financial specialists, workers, customers and providers.

- ✚ Customers: - Customers in the Supermarket company dataset purchase goods or commodities from the business and the supermarket reply on the customers for their source of revenue. Any business decision made would have an impact on allocation of tasks to vendors.

### 1.5. Business requirements

It is very important to develop a database model that can fit the dataset and facilitate the following:

- ✚ Securing the confidential dataset making sure they are not accessible to all.
- ✚ Making the customer related information easy to analyze and make conclusions
- ✚ Database model will enable us to analyze the data easily and effectively.

- 🚩 We can avoid the repetition or superfluity of data thus avoiding data redundancy

## 2. SCHEMA

There is a requirement for developing a suitable schema for analyzing the data warehouse. We can develop a start schema which can be developed by identifying the dimensions and the facts from the dataset.

A dimension table contains dimensions of a fact. Connected to the fact table and located at the edges of the star or snowflake schema.

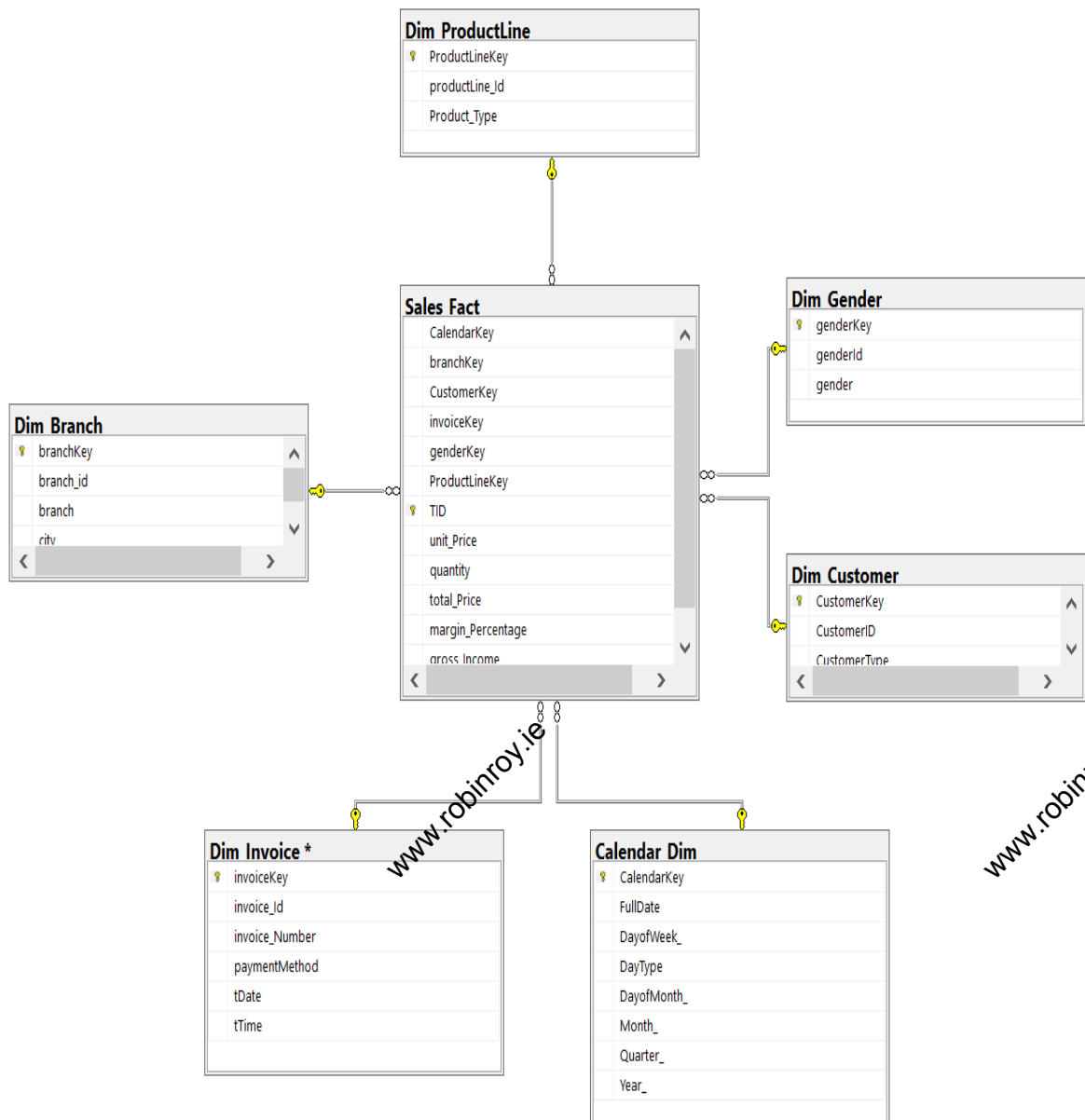
The following tables are considered as the dimension tables:

- Dim Productline
- Dim Branch
- Dim Invoice
- Calendar Dim
- Dim Customer
- Dim Gender

A fact table contains measurements, metrics or facts about a business process. Located at the centre of a star or snowflake schema and surrounded by dimensions.

The fact table “Sales fact” consists of the following facts:

- CalendarKey
- branchkey
- Customerkey
- invoiceKey
- genderKey
- Productlinekey



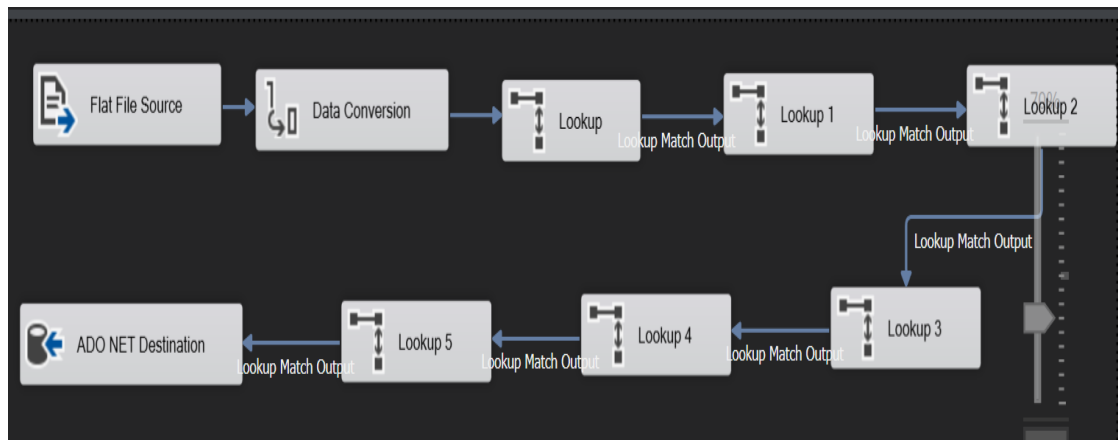
▪ Star Schema Diagram

### 3. ETL

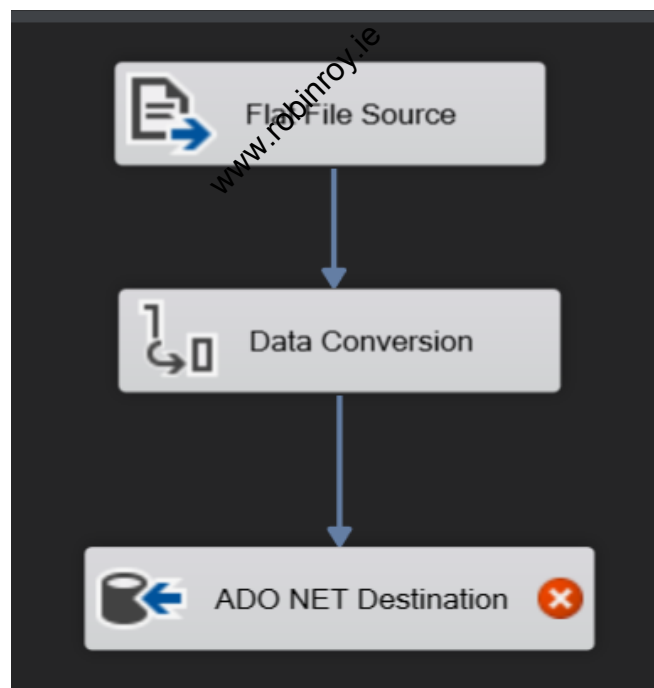
ETL is short for *extract, transform, load*, three database functions.

- 🔗 **Extract** is the process of *reading data* from a database. In this stage, the data is collected, often from multiple and different types of sources.
- 🔗 **Transform** is the process of *converting the extracted data* from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data.
- 🔗 **Load** is the process of *writing the data* into the target database

Data from one or more sources is *extracted and then copied* to the data warehouse .It uses SSIS package to load .csv file to the dimensional table.



▪ Fact table SSIS



▪ Dimension Table SSIS

## 4. VISUALIZATIONS AND REPORTS

### 4.1. REPORTS

Reports are the output delivered from the information examined. The reports are generated with SQL server reporting services (SSRS) package available in the Microsoft Visual Studio.

#### Drill down Reports:

Drill down reports helps to simplify things it also hides the complexity of the data. A drill down report is a report which permits us to explore to an alternate layer of information granularity by exploring and clicking a particular data element.

As we can see from below individual commodity purchase report, it allows to drill down details of each category of goods purchased with its unit price, quantity, tax, total amount and the date of the purchase made with day month and year.

Calendar Key	branch Key	invoice Key	unit Price	quantity	total Price	gross Income
1	1	1				
		269	70.74	4	297.108	14.148
		512	42.91	5	225.2775	10.7275
		244	62.65	4	263.13	12.53
		946	93.88	7	690.018	32.858
		1	74.69	7	548.9715	26.1415
2	2	2				
		3				
		1				
		2				
		3				
		1				
3	3	2				
		462	73.05	10	767.025	36.525
		187	94.49	8	793.716	37.796
		40	30.12	8	253.008	12.048
		3				
		1				
4	4	2				
		3				
		1				
		2				
		3				
		459	46.57	10	488.985	23.285
5	5	389	54.07	9	510.9615	24.3315
		369	14.36	10	150.78	7.18
		503	69.4	2	145.74	6.94
		592	24.24	7	178.164	8.484
		498	90.24	6	568.512	27.072
		1				
6	6	2				
		1				
		3				
		1				
		2				
		1				



## Matrix Report :

Using this matrix report we can show the report with table comparison of prices inorder to increse the sales and profit percentage This is a perfect report that can be distributed month to month/quarterly/every year to get bits of knowledge of different product buys and their all-out expense.

Sales Matrix Report - M

File Run

Design Zoom First Previous 1 of 1 Next Last Refresh Stop Back

Print Page Setup Print Layout Export Document Map Parameters Find

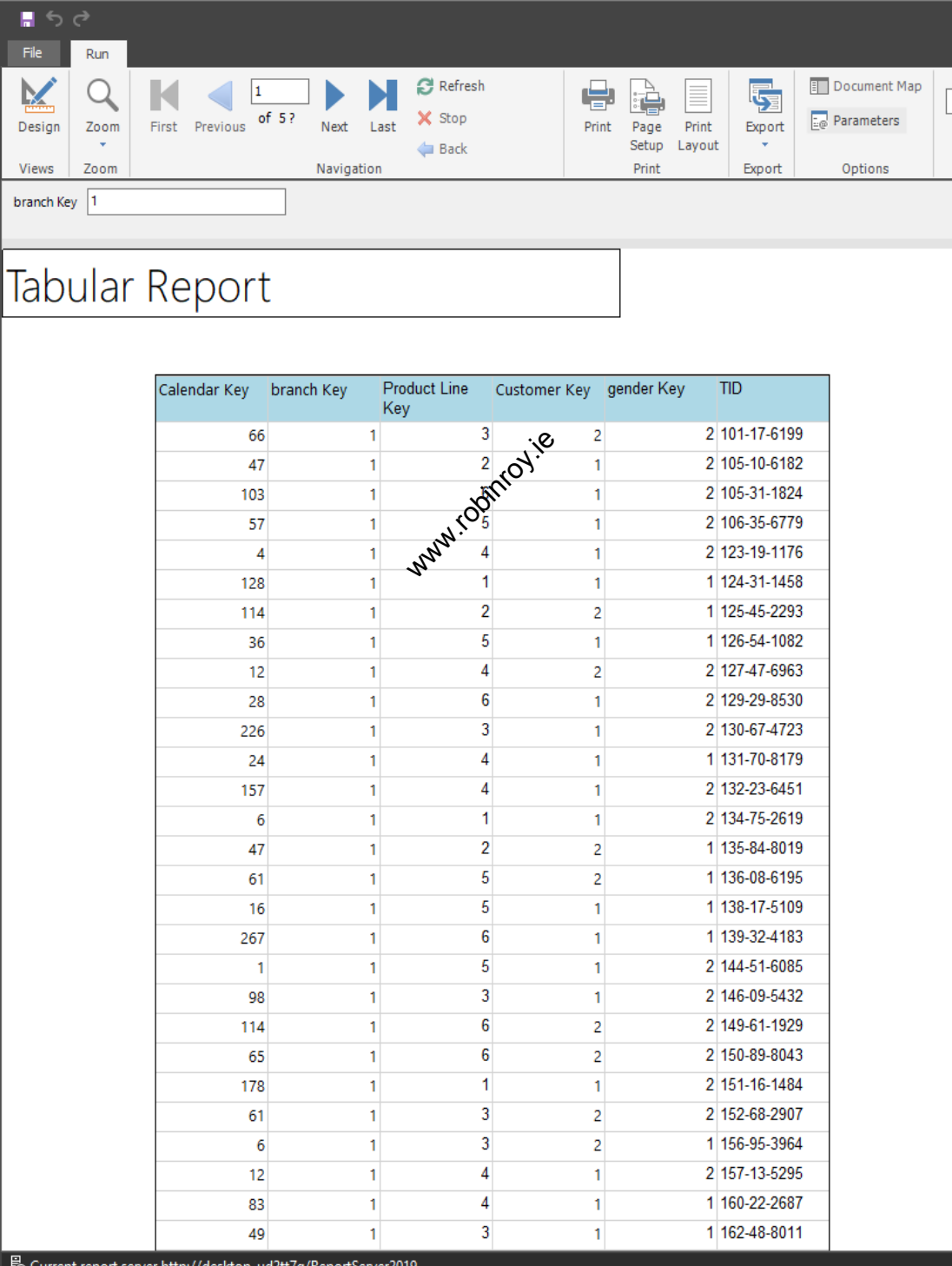
Views Zoom Navigation Export Options

Matrix Table

branch Key	Product Line Key	invoice Number	unit Price	quantity	total Price	margin Percentage
1	1	60	68.84	6	433.692	4.761904762
	2	51	87.6	2	184.107	4.761904762
	3	58	43.19	10	453.495	4.761904762
	4	47	74.69	7	548.9715	4.761904762
	5	65	46.33	7	340.5255	4.761904762
	6	59	86.31	7	634.3785	4.761904762
	Total	340				
2	1	55	25.51	4	107.142	4.761904762
	2	62	14.48	4	60.816	4.761904762
	3	50	54.84	3	172.746	4.761904762
	4	53	87.98	3	277.137	4.761904762
	5	50	40.3	2	84.63	4.761904762
	6	62	93.72	6	590.436	4.761904762
	Total	332				
3	1	55	15.28	5	80.22	4.761904762
	2	65	98.7	8	829.08	4.761904762
	3	66	99.42	4	417.564	4.761904762
	4	52	54.92	8	461.328	4.761904762
	5	45	73.56	10	772.38	4.761904762
	6	45	68.12	1	71.526	4.761904762
	Total	328				
Total		1000				

## Tabular Report :

Underneath report has been generated on examination of no. of products sold on particular date and also which gender buy more. based on this table we can able to increase the product needed more by the cusmtomers.



Calendar Key	branch Key	Product Line Key	Customer Key	gender Key	TID
66	1	3	2	2	101-17-6199
47	1	2	1	2	105-10-6182
103	1		1	2	105-31-1824
57	1	5	1	2	106-35-6779
4	1	4	1	2	123-19-1176
128	1	1	1	1	124-31-1458
114	1	2	2	1	125-45-2293
36	1	5	1	1	126-54-1082
12	1	4	2	2	127-47-6963
28	1	6	1	2	129-29-8530
226	1	3	1	2	130-67-4723
24	1	4	1	1	131-70-8179
157	1	4	1	2	132-23-6451
6	1	1	1	2	134-75-2619
47	1	2	2	1	135-84-8019
61	1	5	2	1	136-08-6195
16	1	5	1	1	138-17-5109
267	1	6	1	1	139-32-4183
1	1	5	1	2	144-51-6085
98	1	3	1	2	146-09-5432
114	1	6	2	2	149-61-1929
65	1	6	2	2	150-89-8043
178	1	1	1	2	151-16-1484
61	1	3	2	2	152-68-2907
6	1	3	2	1	156-95-3964
12	1	4	1	2	157-13-5295
83	1	4	1	1	160-22-2687
49	1	3	1	1	162-48-8011

Current report server <http://desktop-ud2tt7a/ReportServer2019>

### Parameterized reports:

A parameterized report utilizes input esteems to finish report or information handling. With a parameterized report, you can change the yield of a report dependent on values that are set when the report runs. Parmaterized reports uses input value to process the data and give report.

The underneath shows all the product sold in and particular branch based on that we can able to analyse which product is sold more in which branch.

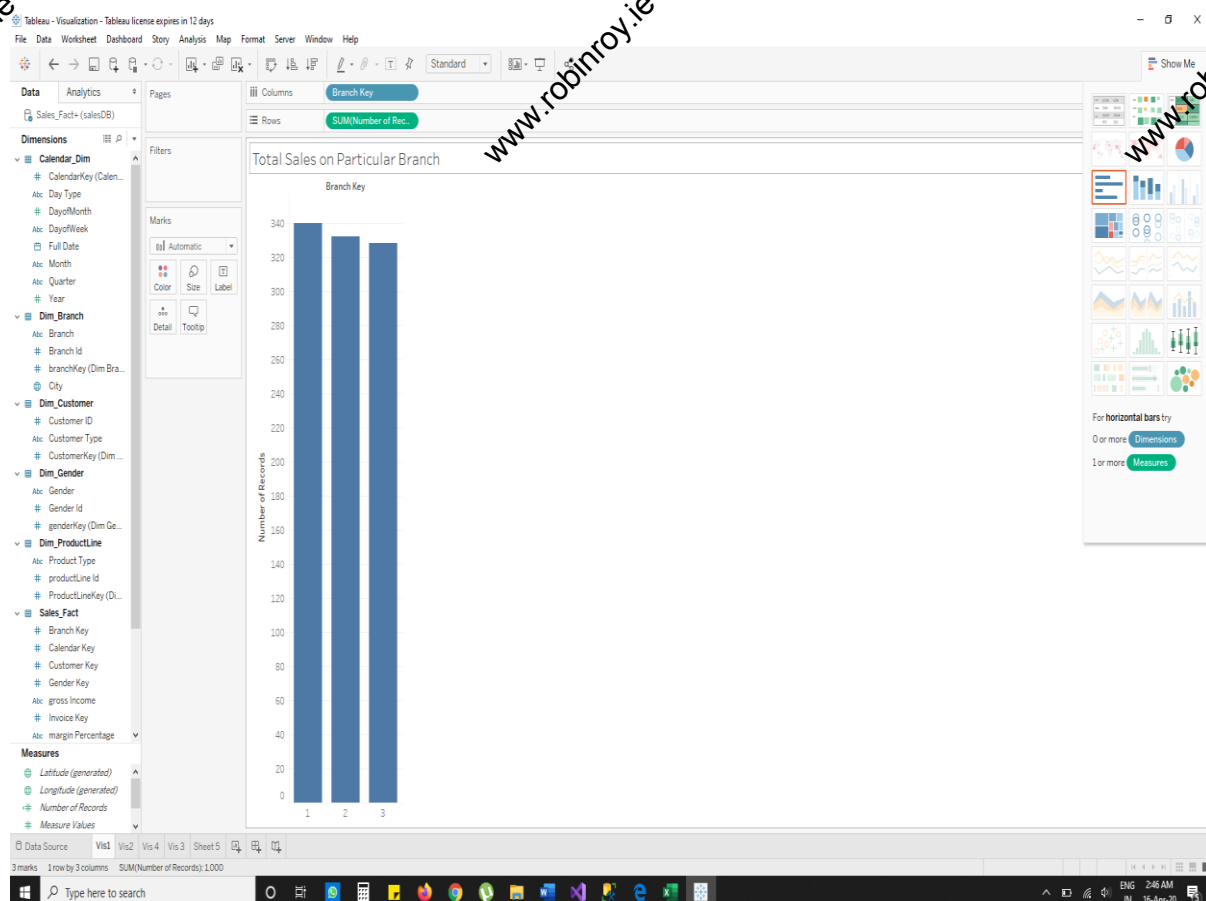
Product Line Key	branch Key	unit Price	quantity	total Price	gross Income
1	1	179.59	3	250.7085	11.9385
1	1	119.32	7	142.002	6.762
1	1	132.25	4	135.45	6.45
1	1	119.69	3	49.4235	2.3535
1	1	120.77	4	87.234	4.154
1	1	199.55	7	731.6925	34.8425
1	1	166.06	6	416.178	19.818
1	1	110.56	8	88.704	4.224
1	1	121.5	9	203.175	9.675
1	1	150.23	4	210.966	10.046
1	1	192.6	7	680.61	32.41
1	1	160.88	9	575.316	27.396
1	1	162.48	1	65.604	3.124
1	1	194.64	3	298.116	14.196
1	1	120.89	2	43.869	2.089
1	1	136.36	4	152.712	7.272
1	1	168.84	6	433.692	20.652
1	1	146.95	5	246.4875	11.7375
1	1	166.35	1	69.6675	3.3175
1	1	140.26	10	422.73	20.13
1	1	174.58	7	548.163	26.103
1	1	172.2	7	530.67	25.27
1	1	174.22	10	779.31	37.11
1	1	128.96	1	30.408	1.448
1	1	151.19	4	214.998	10.238
1	1	171.95	1	75.5475	3.5975
1	1	190.02	8	756.168	36.008
1	1	183.46	6	447.708	21.628

## 4.2. Visualizations

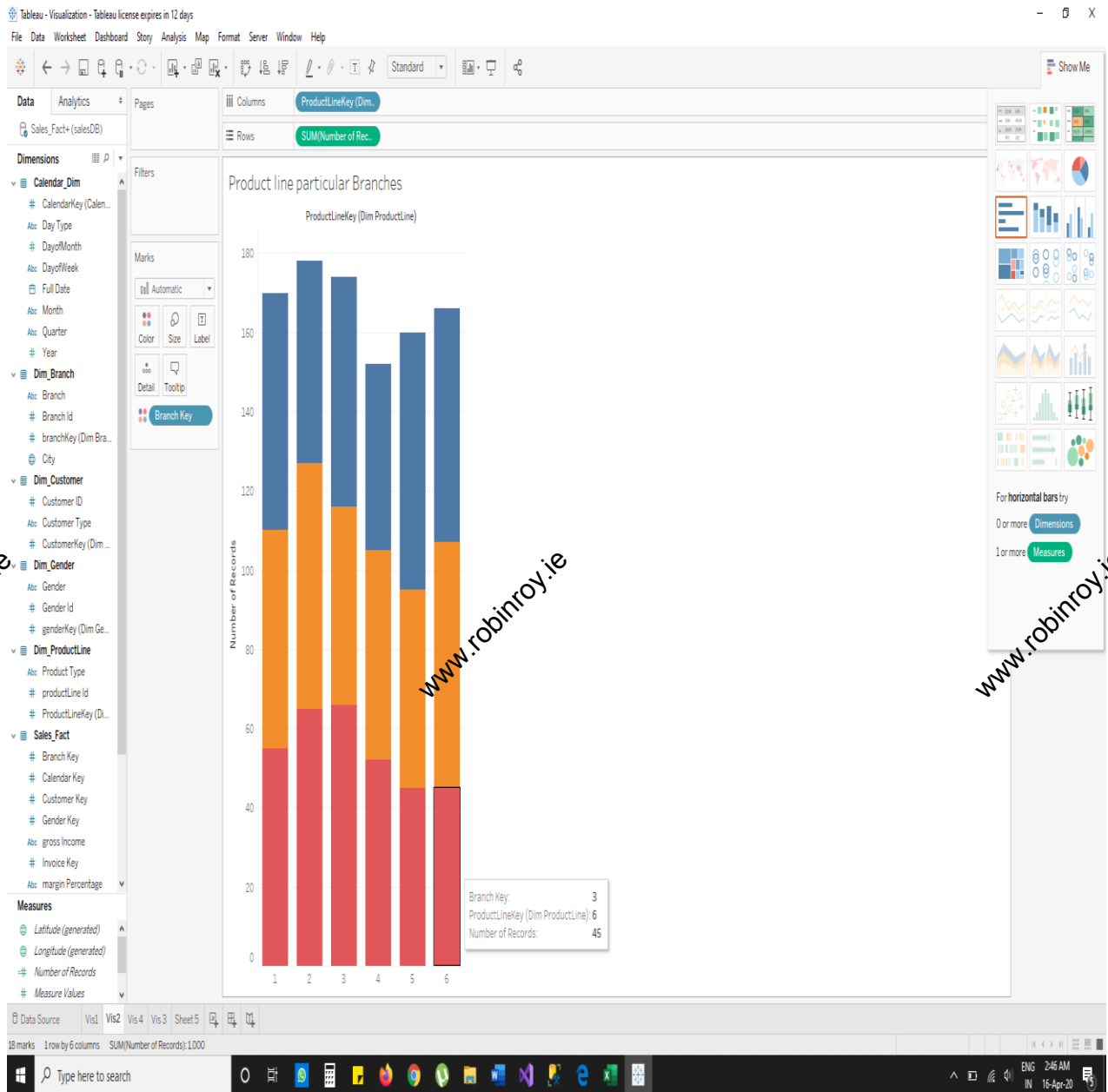
Tableau is a powerful visualization tool makes it simple for experts to impart information perceptions to your whole association. A solitary dashboard could be utilized by many partners, exponentially expanding the estimation of the dashboard.

Tableau is incredibly utilized on the grounds that information can be broke down rapidly with it. Additionally, representations are produced as dashboards and worksheets. It permits one to make dashboards that give noteworthy bits of knowledge and drives the business forward.

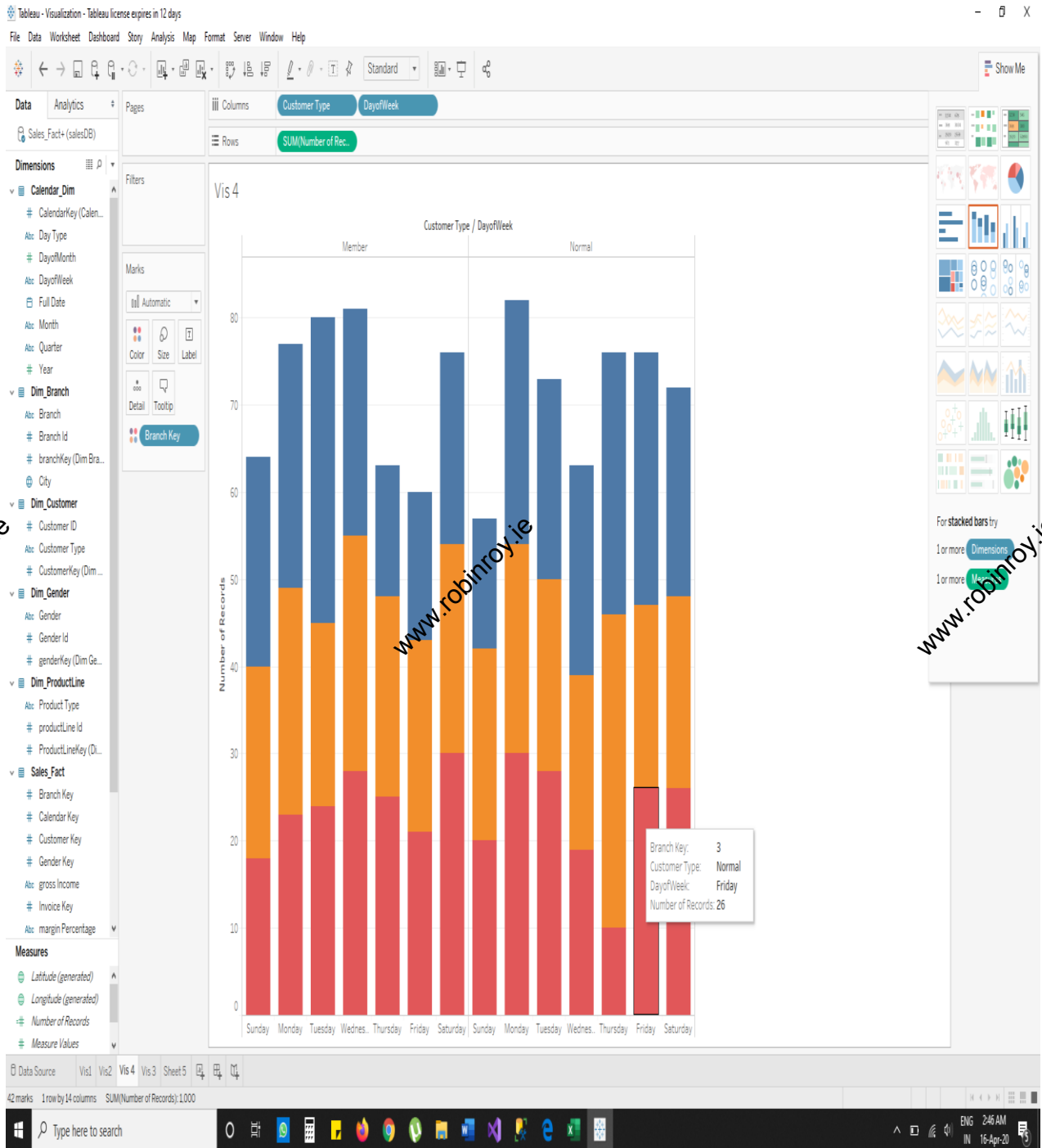
- Below is our Tableau visualization dashboard for Total sales on particular branch



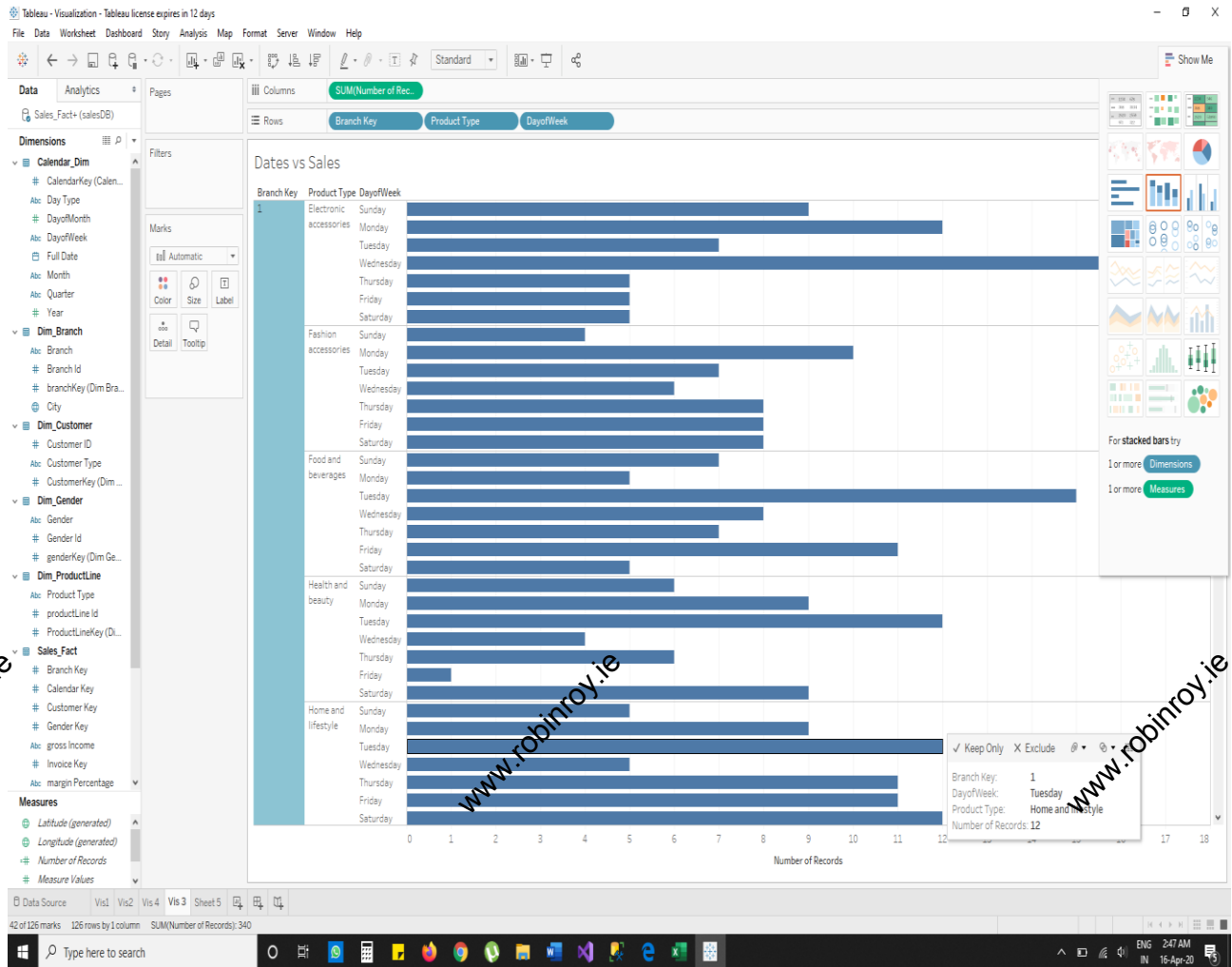
- Below is our Tableau visualization dashboard for product line in particular branches



- Below is our Tableau visualization dashboard for Customer type/ Day of week



- Below is our Tableau visualization dashboard for Dates vs Sales.



From the Reports and Visualization, we are able to understand

The store 'A' is making more no of Bills compare to B and C, But the Gross income is low compare to store 'C'.

The average spending of the customer in store A is 54, B is 55, C is 56.

Therefore, we can able to increase the Gross income in Store 'A' by organizing the store and show casing the products.

In store 'B' the customer rating is low compare to C & A.

By handling the customers in right way, we can increase the customers and Gross Income.

Count of bills is low in store 'C' compare A and B.

We can increase the customers by giving some vouchers so that they will come frequently and buy products more than usual. Therefore, the gross income increases.

## 5. GRAPH DATABASES

We are using Neo4j for the purpose of Graph Databases. We developed a graph database and evaluated the use of Neo4j, which is a native graph database

We are using CQL (cypher query language) in order to create Neo4j Graph Database.

First we loaded the .csv files into Neo4j Database using Cypher queries.

To load the csv file Branch.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:///Branch.csv" as row CREATE (b:Branch)
SET b = row RETURN b
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(b:branch) ASSERT b.branchKey IS UNIQUE
```

To load the csv file Customer.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:///Customer.csv" as row CREATE (c:Customer) SET c =
row RETURN c
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(c:Customer) ASSERT c.CustomerKey IS UNIQUE
```

To load the csv file Gender.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:///Gender.csv" as row CREATE (g:Gender) SET g = row
RETURN g
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(g:Gender) ASSERT g.genderKey IS UNIQUE
```



To load the csv file SalesFact.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:/// SalesFact.csv" as row CREATE
```

```
(i: SalesFact) SET i = row RETURN i
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(i: SalesFact) ASSERT i.TID IS UNIQUE
```

To load the csv file Invoice.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:///Invoice.csv" as row CREATE (i:Invoice) SET i = row  
RETURN i
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(i:Invoice) ASSERT i.invoiceKey IS UNIQUE
```

To load the csv file Branch.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:///Branch.csv" as row CREATE (b:Branch) SET b = row  
RETURN b
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(b:branch) ASSERT b.branchKey IS UNIQUE
```

To load the csv file ProductLine.csv by using Cypher Queries

```
LOAD CSV WITH HEADERS FROM "file:/// ProductLine.csv" as row CREATE
```

```
(p: ProductLine) SET p = row RETURN p
```

For creating a unique constraint to the table we use the query

```
CREATE CONSTRAINT ON(:ProductLine) ASSERT p. ProductLinekey IS UNIQUE
```

Now we establish a relationship between Branch Table and SalesFact Table

```
MATCH(a:Branch),(f:SalesFact) WHERE a.branchKey = f.branchKey CREATE (a)-[r:Branch]->(f)  
RETURN a,f,r
```

Now we establish a relationship between Customer Table and SalesFact Table

MATCH(a:Customer),(f:SalesFact) WHERE a.CustomerKey= f.CustomerKey CREATE (a)-[r:Customer]->(f) RETURN a,f,r

Now we establish a relationship between Gender Table and SalesFact Table

MATCH(a:Gender),(f:SalesFact) WHERE a.genderKey= f.genderKey CREATE (a)-[r:Gender]->(f) RETURN a,f,r

Now we establish a relationship between ProductLine1Table and SalesFact Table

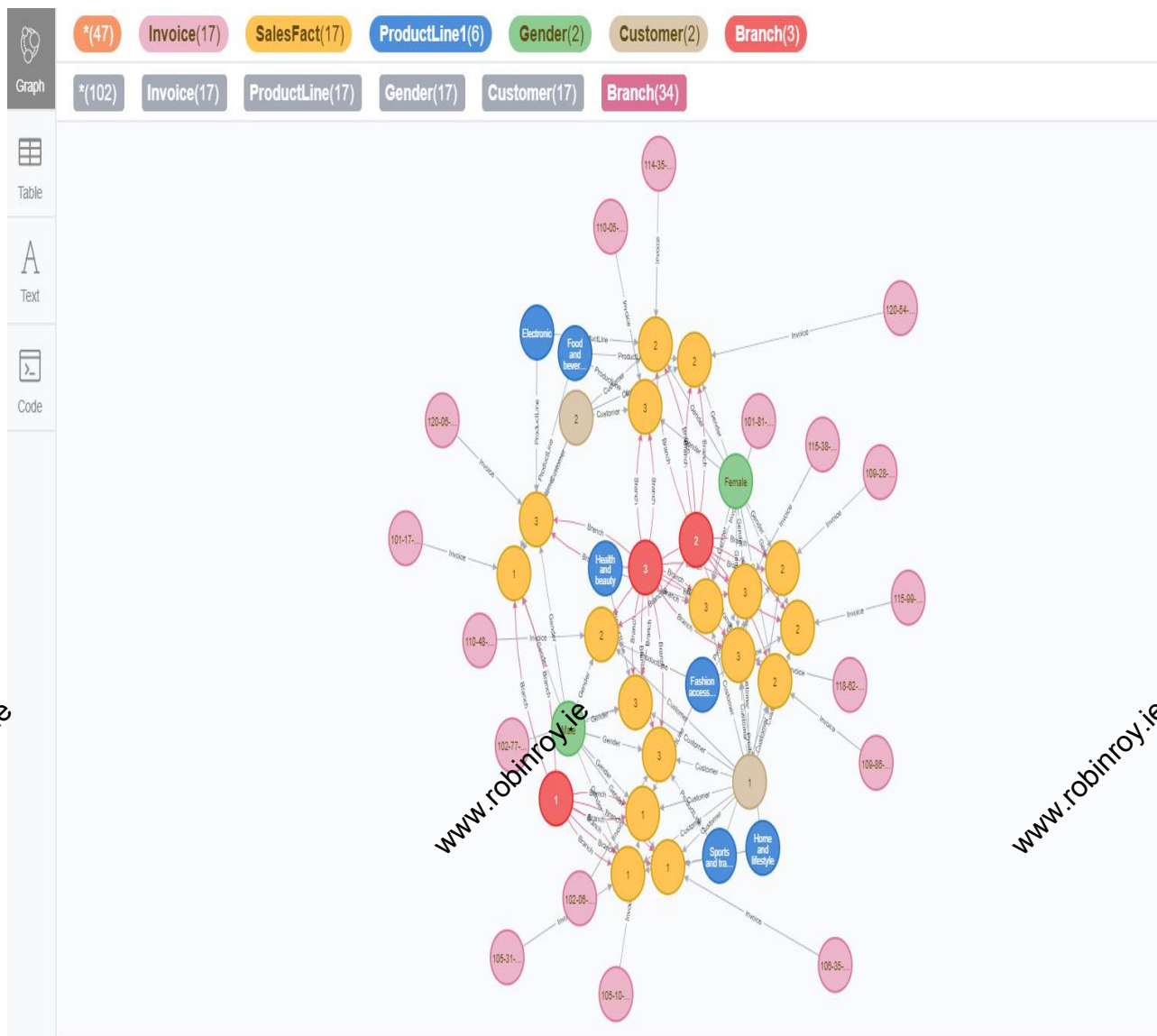
MATCH(a:ProductLine1),(f:SalesFact) WHERE a.ProductLineKey= f.ProductLineKey CREATE (a)-[r:ProductLine]->(f) RETURN a,f,r

Now we establish a relationship between Invoice Table and SalesFact Table

MATCH(a:Invoice),(f:SalesFact) WHERE a.invoiceKey= f.invoiceKey CREATE (a)-[r:Invoice]->(f) RETURN a,f,r

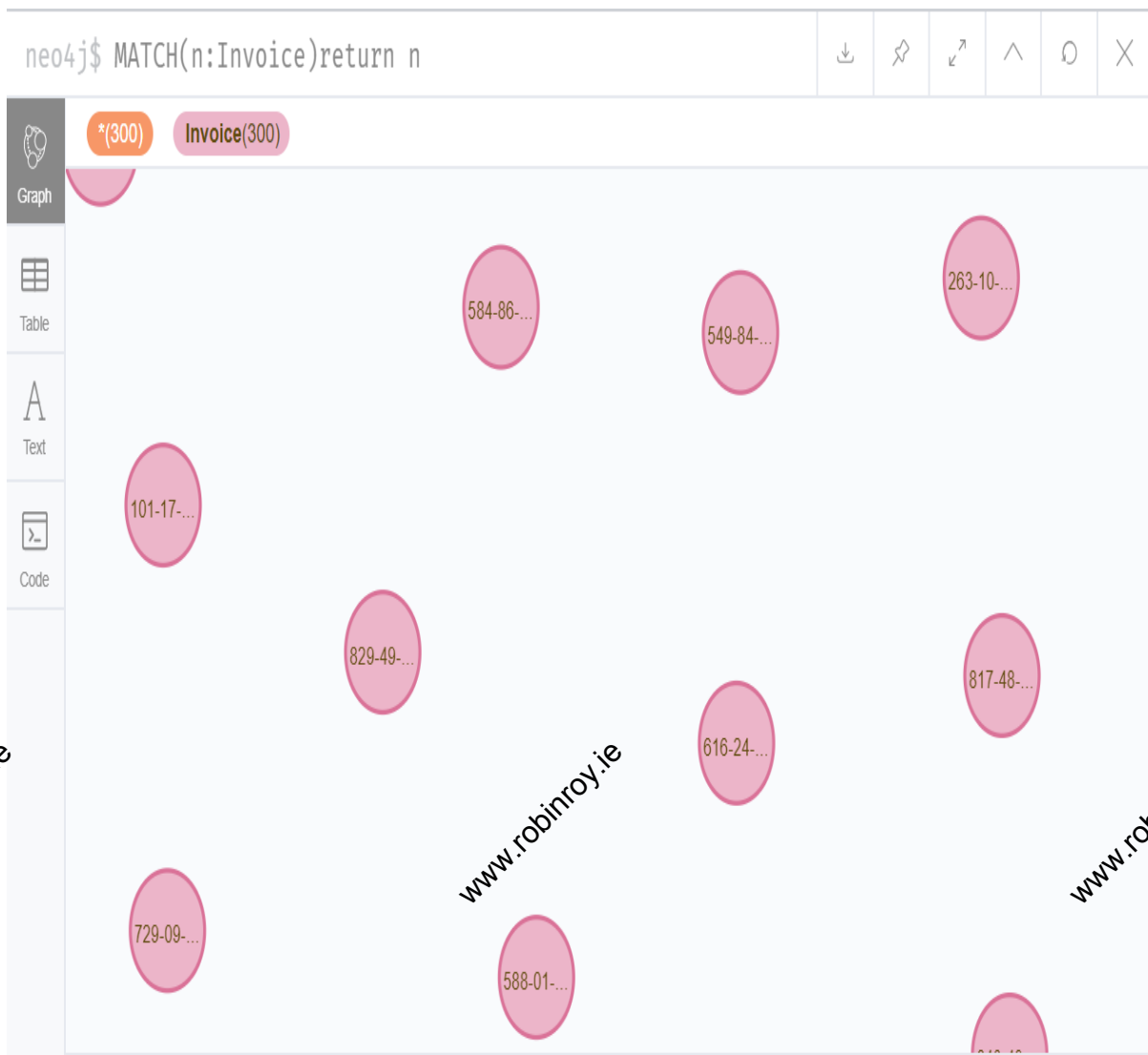
Finally we visualize all the relationships

MATCH (a)-[r]->(f) RETURN a,r,f;



#### ■ Neo4j Schema

1. To print all the elements in the table, the syntax is : match (variable:tablename) return variable



Query1 Neoj4

The corresponding SQL query is

```
SELECT * FROM [dbo].[Dim_Invoice]
```

	invoiceKey	invoice_Id	invoice_Number	paymentMethod	unit_Price	quantity	total_price	margin_Percentage	gross_income	tDate	tTime
1	1	1	750-67-8428	Ewallet	74.69	7	548.9715	4.761904762	26.1415	2019-05-01	13:08:00.0000000
2	2	2	226-31-3081	Cash	15.28	5	80.22	4.761904762	3.82	2019-08-03	10:29:00.0000000
3	3	3	631-41-3108	Credit card	46.33	7	340.5255	4.761904762	16.2155	2019-03-03	13:23:00.0000000
4	4	4	123-19-1176	Ewallet	58.22	8	489.048	4.761904762	23.288	2019-01-27	20:33:00.0000000
5	5	5	373-73-7910	Ewallet	86.31	7	634.3785	4.761904762	30.2085	2019-08-02	10:37:00.0000000
6	6	6	699-14-3026	Ewallet	85.39	7	627.6165	4.761904762	29.8865	2019-03-25	18:30:00.0000000
7	7	7	355-53-5943	Ewallet	68.84	6	433.692	4.761904762	20.652	2019-02-25	14:36:00.0000000
8	8	8	315-22-5665	Ewallet	73.56	10	772.38	4.761904762	36.78	2019-02-24	11:38:00.0000000
9	9	9	665-32-9167	Credit card	36.26	2	76.146	4.761904762	3.626	2019-10-01	17:15:00.0000000
10	10	10	692-92-5582	Credit card	54.84	3	172.746	4.761904762	8.226	2019-02-20	13:27:00.0000000
11	11	11	351-62-0822	Ewallet	14.48	4	60.816	4.761904762	2.896	2019-06-02	18:07:00.0000000
12	12	12	529-56-3974	Cash	25.51	4	107.142	4.761904762	5.102	2019-09-03	17:03:00.0000000

2.

```
neo4j$ MATCH(n:SuperMarket{productLine:"Electronic  
accessories"}) return n.productLine,n.invoice
```

```
neo4j$ MATCH(n:SuperMarket{productLine:"Electronic accessori...
```



Table



Text



Code

n.productLine

n.invoice

"Electronic accessories"

"226-31-3081"

"Electronic accessories"

"699-14-3026"

"Electronic accessories"

"355-53-5943"

"Electronic accessories"

"529-56-3974"

"Electronic accessories"

"365-64-0515"

"Electronic accessories"

"300-71-4605"

"Electronic accessories"

"636-48-8204"

"Electronic accessories"

"272-65-1806"

"Electronic accessories"

"132-32-9879"

▪ Query2 Neo4j

The corresponding SQL query is

```
SELECT productLine,invoice FROM [dbo].[supermarketSale] WHERE productLine = 'Electronic accessories'
```

	productLine	invoice
1	Electronic accessories	226-31-3081
2	Electronic accessories	699-14-3026
3	Electronic accessories	355-53-5943
4	Electronic accessories	529-56-3974
5	Electronic accessories	365-64-0515
6	Electronic accessories	300-71-4605
7	Electronic accessories	636-48-8204
8	Electronic accessories	272-65-1806
9	Electronic accessories	132-32-9879
10	Electronic accessories	669-54-1719
11	Electronic accessories	399-46-5918
12	Electronic accessories	120-06-4233

3.

```
neo4j$ MATCH(n:SuperMarket{invoice:"750-67-8428"}) SET n.gender="Male" RETURN n.invoice,n.gender
```

```
neo4j$ MATCH(n:SuperMarket{invoice:"750-67-8428"}) SET n.gen...
```

	n.invoice	n.gender
Table		
Text	"750-67-8428"	"Male"

Query3 Neo4

The corresponding SQL query

```
UPDATE [dbo].[supermarketSale] SET gender='Male' WHERE invoice='750-67-8428'
SELECT invoice,gender FROM [dbo].[supermarketSale] WHERE invoice='750-67-8428'
```

	invoice	gender
1	750-67-8428	Male

4.

```
1 match (n:SuperMarket) return n.productLine,n.quantity order by
2 n.quantity desc
```

neo4j\$ match (n:SuperMarket) return n.productLine,n.quantity o...



Table



Text



Code

n.productLine

n.quantity

"Health and beauty"

"9"

"Food and beverages"

"9"

"Sports and travel"

"9"

"Electronic accessories"

"9"

"Electronic accessories"

"9"

"Home and lifestyle"

"9"

"Health and beauty"

"9"

"Health and beauty"

"9"

▪ Query4 Neo4j

The corresponding SQL query

```
SELECT p.productLine, p.quantity FROM [dbo].[supermarketSale] as p
ORDER BY p.quantity DESC;
```

161 %

Results Messages

	productLine	quantity
3	Electronic accessories	9
4	Health and beauty	9
5	Food and beverages	9
6	Sports and travel	9
7	Home and lifestyle	9
8	Fashion accessories	9
9	Fashion accessories	9
10	Health and beauty	9
11	Fashion accessories	9
12	Electronic accessories	9
13	Food and beverages	9
14	Home and lifestyle	9

5.

```
neo4j$ CREATE(n:Customer{CustomerType:"Premium"}) RETURN n
```

```
neo4j$ CREATE(n:Customer{CustomerType:"Premium"}) RETURN n
```



Graph



Table



Text

n

```
{  
  "CustomerType": "Premium"  
}
```

Query5 NeoJ4

The corresponding SQL query

```
INSERT INTO [dbo].[Dim_Customer] VALUES(3,'Premium')
```

161 %

Results

Messages

	CustomerKey	CustomerID	CustomerType
1	1	1	Member
2	2	2	Normal
3	3	3	Premium



6.

```
neo4j$ match (n:Customer{CustomerType:"Premium"}) delete n
```

```
neo4j$ match (n:Customer{CustomerType:"Premium"}) delete n
```



Table

Deleted 1 node, completed after 9 ms.

▪ Query6 NeoJ4

The corresponding SQL query

```
DELETE FROM [dbo].[Dim_Customer] WHERE CustomerType='Premium'
```

%  
essages

(1 row affected)

Completion time: 2020-04-16T03:00:12.8864676+01:00

```
1 match (p:Invoice),(t:SalesFact) where p.invoiceKey = t.invoiceKey
2 return p.total_price,t.TID
```

```
neo4j$ match (p:Invoice),(t:SalesFact) where p.invoiceKey = t...
```



Table



Text



Code

p.total\_price

t.TID

"336.5565"

"101-17-6199"

"131.922"

"101-81-4070"

"132.5625"

"102-06-2002"

"480.0285"

"102-77-2261"

"45.108"

"105-10-6182"

"510.972"

"105-31-1824"

"93.114"

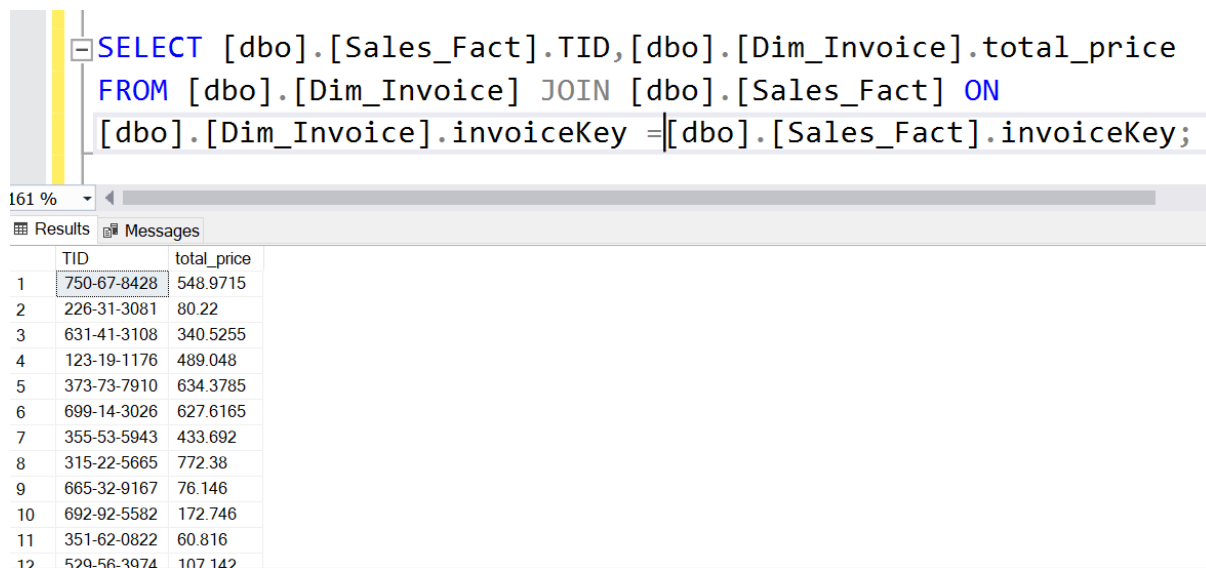
"106-35-6779"

"614.943"

"109-28-2512"

▪ Query7 NeoJ4

## The Corresponding SQL query



The screenshot shows a SQL query editor with a query window and a results pane. The query is a SELECT statement joining [dbo].[Sales\_Fact] and [dbo].[Dim\_Invoice] on invoiceKey. The results pane shows a table with two columns: TID and total\_price, containing 12 rows of data.

```
SELECT [dbo].[Sales_Fact].TID,[dbo].[Dim_Invoice].total_price
FROM [dbo].[Dim_Invoice] JOIN [dbo].[Sales_Fact] ON
[dbo].[Dim_Invoice].invoiceKey =[dbo].[Sales_Fact].invoiceKey;
```

	TID	total_price
1	750-67-8428	548.9715
2	226-31-3081	80.22
3	631-41-3108	340.5255
4	123-19-1176	489.048
5	373-73-7910	634.3785
6	699-14-3026	627.6165
7	355-53-5943	433.692
8	315-22-5665	772.38
9	665-32-9167	76.146
10	692-92-5582	172.746
11	351-62-0822	60.816
12	529-56-3974	107.142





## 6.1. GRAPH DATABASES AND RELATIONAL DATABASES

- ✚ The key contrast between a graph and relational database is that relational databases work with sets while diagram databases work with paths
- ✚ In a graph database, the relationships are stored at the individual record level. In a relational database, the structure is defined at a higher level (the table definitions).
- ✚ A relational database is a lot quicker while working on immense quantities of records. In a chart database, each record must be inspected exclusively during an inquiry so as to decide the structure of the information, while this is known early in relational databse.

## 7. CONCLUSION

All in all, we have demonstrated how information can be coordinated from various sources to a solitary storehouse called data warehouse which is utilized for conveying integrated structure to the end clients and officials. Studying the available data set and having it contrasted to the created 'Relational Database System' permitted us to see different activities in the Supermarket company branches in various cities. We understood the benefits of visualization of data to arrive at notable findings. This additionally helps us to identify drawbacks of the current process and suggest steps setting up a new secure database.

## 8. BIBLIOGRAPHY

-  Database System Concepts  
Textbook by Avi Silberschatz, Henry F. Korth, and S. Sudarshan
-  SQL Management Studio  
[https://www.tutorialspoint.com/ms\\_sql\\_server/ms\\_sql\\_server\\_management\\_studio.htm](https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_management_studio.htm)
-  Relational Databases  
<https://www.ibm.com/cloud/learn/relational-databases>
-  Tableau Tutorials  
<https://intellipaat.com/blog/what-is-tableau/>

## Appendix A – Dimension table script

```
SELECT * FROM Sales_Fact  
  
CREATE TABLE Dim_Branch  
( branchKey INT NOT NULL IDENTITY,  
  branch_id INT,  
  branch VARCHAR(50),  
  city varchar(50),  
  PRIMARY KEY (branchKey));
```

```
CREATE TABLE Dim_Customer  
(CustomerKey INT NOT NULL IDENTITY,  
  CustomerID INT,
```

CustomerType varchar(50),  
PRIMARY KEY (CustomerKey));

CREATE TABLE Dim\_Invoice  
(  
invoiceKey INT NOT NULL IDENTITY,  
invoice\_Id INT,  
invoice\_Number varchar(50),  
paymentMethod varchar(50),  
tDate DATE,  
PRIMARY KEY (invoiceKey));

CREATE TABLE Dim\_Gender  
(genderKey INT NOT NULL IDENTITY,  
genderId INT,  
gender varchar(50),  
PRIMARY KEY (genderKey));

CREATE TABLE Dim\_ProductLine  
(ProductLineKey INT NOT NULL IDENTITY,  
productLine\_Id INT,  
Product\_Type varchar(50),  
PRIMARY KEY (ProductLineKey));

CREATE TABLE Calendar\_Dim  
(  
CalendarKey INT NOT NULL IDENTITY,  
FullDate DATE,  
DayOfWeek\_ CHAR(15),  
DayType CHAR(20),  
DayOfMonth\_ INT,  
Month\_ CHAR(10),  
Quarter\_ CHAR(2),  
Year\_ INT,  
PRIMARY KEY (CalendarKey));

## Appendix B – Fact table script

```
CREATE TABLE Sales_Fact
(
  CalendarKey INT,
  branchKey INT,
  CustomerKey INT,
  invoiceKey INT,
  genderKey INT,
  ProductLineKey INT,
  TID VARCHAR(50),
  unit_Price VARCHAR(50),
  quantity VARCHAR(50),
  total_Price VARCHAR(50),
  margin_Percentage VARCHAR(50),
  gross_Income VARCHAR(50),
  PRIMARY KEY(TID),
  FOREIGN KEY (branchKey) REFERENCES Dim_Branch (branchKey),
  FOREIGN KEY (CustomerKey) REFERENCES Dim_Customer (CustomerKey),
  FOREIGN KEY (genderKey) REFERENCES Dim_Gender (genderKey),
  FOREIGN KEY (ProductLineKey) REFERENCES Dim_ProductLine(ProductLineKey),
  FOREIGN KEY (invoiceKey) REFERENCES Dim_Invoice(invoiceKey),
  FOREIGN KEY (CalendarKey) REFERENCES Calendar_Dim(CalendarKey)
);
```

## Appendix C- List of tables script

```
SELECT * FROM Sales_Fact ----Sales Fact Table-----
SELECT * FROM Dim_Branch ----Branch Dimension Table-----
SELECT * FROM Dim_Customer -----Customer Dimension Table-----
```

SELECT \* FROM Dim\_Gender -----Gender Dimension Table-----  
 SELECT \* FROM Dim\_Invoice -----Invoice Dimension Table-----  
 SELECT \* FROM Dim\_ProductLine -----ProductLine Dimension Table-----  
 SELECT \* FROM Calendar\_Dim -----Calendar Dimension Table-----

## Appendix D – Neo4J Code

LOAD CSV WITH HEADERS FROM "file:///Branch.csv" as row CREATE (b:Branch) SET b = row  
 RETURN b

CREATE CONSTRAINT ON(b:branch) ASSERT b.branchKey IS UNIQUE

LOAD CSV WITH HEADERS FROM "file:///Customer.csv" as row CREATE (c:Customer) SET c =  
 row RETURN c

CREATE CONSTRAINT ON(c:Customer) ASSERT c.CustomerKey IS UNIQUE

LOAD CSV WITH HEADERS FROM "file:///Gender.csv" as row CREATE (g:Gender) SET g = row  
 RETURN g

CREATE CONSTRAINT ON(g:Gender) ASSERT g.genderKey IS UNIQUE

LOAD CSV WITH HEADERS FROM "file:///Invoice.csv" as row CREATE (i:Invoice) SET i = row  
 RETURN i

CREATE CONSTRAINT ON(i:Invoice) ASSERT i.invoiceKey IS UNIQUE

MATCH(a:Branch),(f:SalesFact) WHERE a.branchKey = f.branchKey CREATE (a)-[r:Branch]->(f)  
 RETURN a,f,r

MATCH(a:Customer),(f:SalesFact) WHERE a.CustomerKey= f.CustomerKey CREATE (a)-  
 [r:Customer]->(f) RETURN a,f,r

MATCH(a:Gender),(f:SalesFact) WHERE a.genderKey= f.genderKey CREATE (a)-[r:Gender]->(f)  
 RETURN a,f,r

MATCH(a:ProductLine1),(f:SalesFact) WHERE a.ProductLineKey= f.ProductLineKey CREATE  
 (a)-[r:ProductLine]->(f) RETURN a,f,r

MATCH(a:Invoice),(f:SalesFact) WHERE a.invoiceKey= f.invoiceKey CREATE (a)-[r:Invoice]->(f)  
 RETURN a,f,r

MATCH (a)-[r]->(f) RETURN a,r,f;

## Appendix E – Data Source

URL : <https://www.kaggle.com/aungpyaeap/supermarket-sales/>

