Data Analysis and Integration
Project Report

Ilina Kirovska
up202301450

Francesco Barabino
up202301449

1. Subject description

The goal of this project is to design and implement a data warehouse for a retail store that supports business intelligence (BI) activities, especially analytics. The selected store dataset includes detailed records of sales transactions, customer information, as well as product and order details. Using this data, we were able to construct a data warehouse where the information is stored in a way that is easily queryable and further analysed. The store can analyse both past and current data, identify trends, and make better decisions with the support of the warehouse, which provides vital insights into its sales and operational performance.

2. Planning

To successfully design the dimensional model and later the data warehouse itself, we first need to identify its elements, such as stars and dimensions.

3. Dimensional bus matrix

Stars	**Dimensions**			
	Customer	Product	Location	Time
Order sales	X		X	X
Product sales per order	X	X	X	X
Product sales		X		X

4. Dictionary of dimensions

Dimension name	**Description**	**Version**	1.0	**Date**	10.12.2023
Customer	A customer of the store				
Attribute	**Description**	**Key**	**Type**	**Size**	**Precision**
CustomerID	Internal identifier of the customer	PK	INT		0
CustomerDB_ID	The store customer identifier	UK	Varchar	10	
Name	Name of the customer		Varchar	50	
Segment	The segment the customer belongs to		Varchar	45	

Dimension name	**Description**	**Version**	1.0	**Date**	10.12.2023
Product	A product the store sells				
Attribute	**Description**	**Key**	**Type**	**Size**	**Precision**
ProductID	Internal identifier of the product	PK	INT		0
ProductDB_ID	The store product identifier	UK	Varchar	20	
Category	Category of the product		Varchar	20	
Sub-category	Sub-category of the product		Varchar	20	
Name	Name of the product		Varchar	150	

Dimension name	Description	Version	1.0	Date	10.12.2023
Location	Location of a customer				
Attribute	**Description**	**Key**	**Type**	**Size**	**Precision**
LocationID	Internal identifier of the location	PK	INT		0
Country	Country of the location		Varchar	45	
City	City of the location		Varchar	45	
State	State of the location		Varchar	45	
Postal code	Postal code of the location		Varchar	10	
Region	Region of the location		Varchar	45	

Dimension name	Description	Version	1.0	Date	10.12.2023
Time	Date				
Attribute	**Description**	**Key**	**Type**	**Size**	**Precision**
TimeID	Internal identifier of the location	PK	INT		0
Year	The year of the date		YEAR	45	
Month	The month of the date		TINYINT	45	
Day	The day of the date		TINYINT	45	

5. Dictionary of facts

Star	Order sales	Version	1.0	Date	10.12.2023
Granularity	An order made at the store				
Dimensions					
Customer					
Location					
Time					

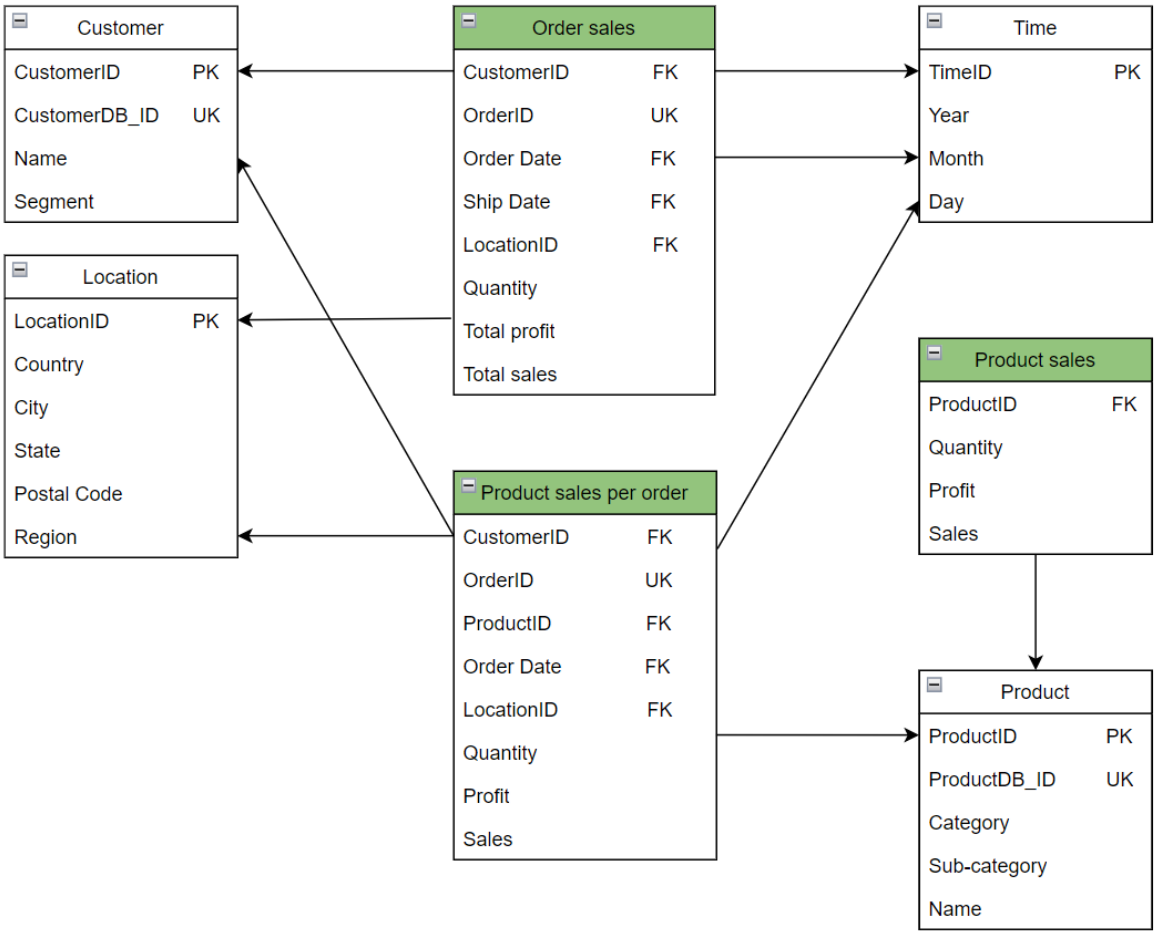
Measures	
Quantity	Total number of products on the order
Profit	Total profit made from the order
Sales	Total sales made from the order

Star	**Product sales per order**	**Version**	1.0	**Date**	10.12.2023
Granularity	A product from the order				
Dimensions					
Customer					
Product					
Location					
Time					
Measures					
Quantity	The total quantity of that product purchased on the order				
Profit	Total order profit made from that product				
Sales	Total order sales made from that product				

Star	**Product sales**	**Version**	1.0	**Date**	10.12.2023
Granularity	A product the store sells				
Dimensions					
Product					
Time					
Measures					
Quantity	The total number of products sold				
Profit	Total profit made from that product				
Sales	Total sales made from that product				

3. Dimensional data model

The previously defined dimensions and stars were combined into the dimensional data model for our data warehouse. It consists of three fact tables (marked with green in the photo), one for each star and four dimensions. The dimensional model arranges the data in a way that it is easier to retrieve information and do various different analyses.



4. Data sources selection. Extraction, transformation and loading.

Data Source Selection

The primary data source in this process is a single table, “superstore”. The table contains every order issued at the store since 2014 until 2017, retaining information on customer, location, profit, sales, date and product. The table appears to be monolithic and not suitable for high-level analysis, we thus proceed to build our model.

Data extraction and transformation

We performed extraction and transformation by using Python with Pandas library, directly operating on the “superstore.csv” file.
We first load the file as a single Pandas dataframe and then proceed to extract our model’s dimensions and fact tables through custom functions.

1. General preprocessing:
   Numeric columns like “Profit” and “Sales” are rounded to two decimal places to maintain consistency and readability.

2. Dimension Table Creation:
   - Time: Single times, as tuples of (year, month, day) are extracted from the main table, where a single mm/dd/yyyy retains dates. An incremental times ID is generated.
   - Customer: Single customers are extracted from the main table, dropping duplicates and retaining only customer name, segment and original ID, while generating an incremental customer ID.
   - Location: Single locations, as tuples of (country, city, state, postal code, region) are extracted from the main table, dropping duplicates and generating an incremental location ID.
   - Product: Single products are extracted from the main table, dropping duplicates and retaining only product name, category, sub-category and original ID, while generating an incremental product ID.

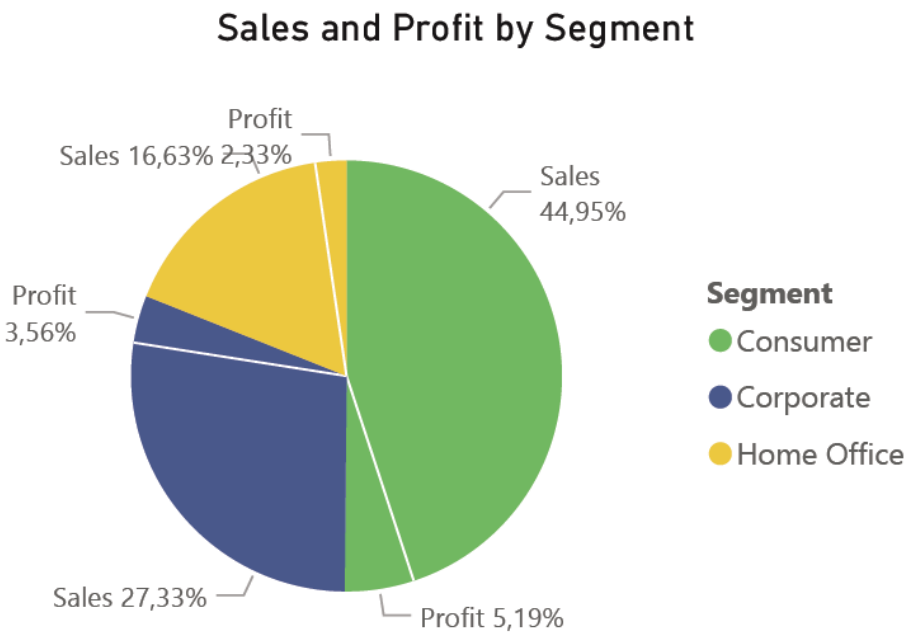
3. Fact Table Creation:
    - Product sales: Single products are grouped per product ID, summing quantity, profit and sales. The result is then connected to previously computed product dimensions.
    - Product sales per order: Single products for every order are retained, maintaining the lowest granularity. Every product entry is connected to the IDs of previously computed times, locations, products and customers dimensions.
    - Order sales: Single orders are extracted from the main table, ignoring information about single products, summing total profit, sales and quantity. Every order is connected to the IDs of previously computed times, locations and customers.

4. Data Integrity Assurance:
    Our transformation script employs assertions to validate the consistency of data through each transformation step, ensuring the reliability of the processed data.

5. Querying and data analysis

To query, analyse and visualise the data from our warehouse we used a combination of MySQL Server, MySQL Workbench and Power BI. The analyses were done on different dimensions: user, product, location and time. At the end of the report, we have attached the exported Power BI visualisations where we grouped the analyses according to these dimensions. The first page contains user demographic analyses while the second page contains product analyses, the third and fourth pages contain different trends through time and the fifth and sixth pages contain location analyses.

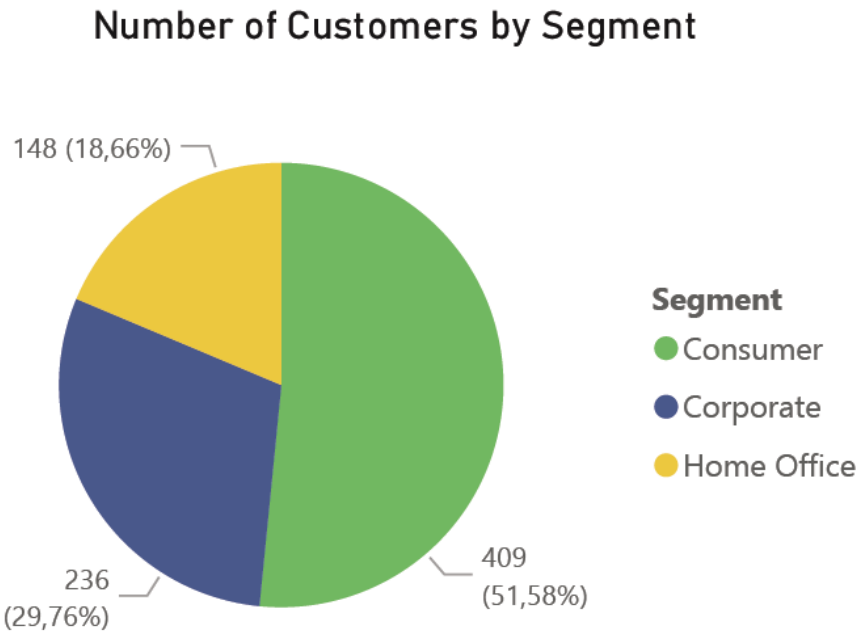
1. Sales and profit by customer segment
    SQL query:
    ```
    select Segment, SUM(Profit) as Profit, SUM(Sales) as
    Sales from sales_per_order natural join customer group
    by Segment;
    ```
    Visualisation:



2. Number of customers by segment
   SQL query:

```
select Segment, count(*) as Customers
from customer group by Segment;
```

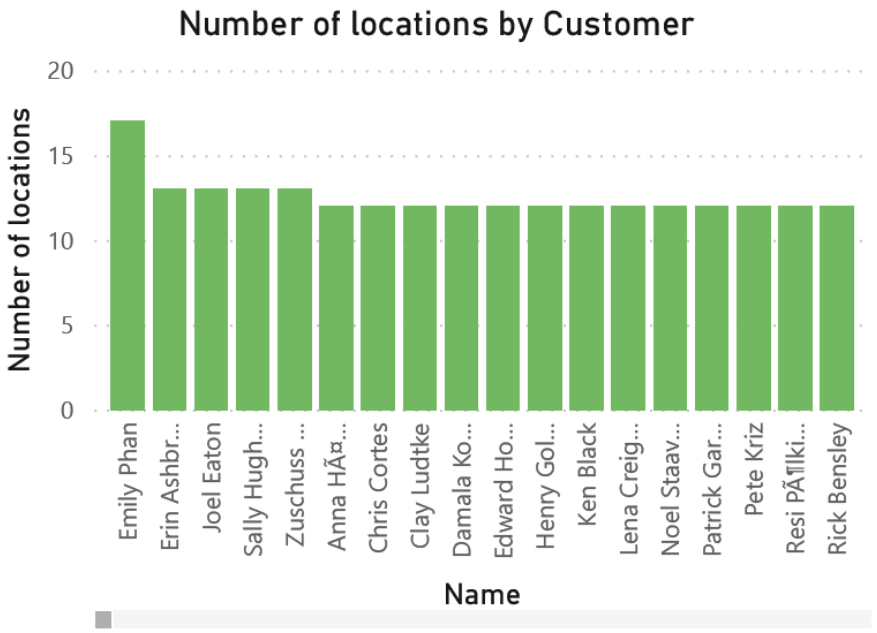
   Visualisation:



3. Number of locations by customer
   SQL query:

```
SELECT c.CustomerID, c.Name, COUNT(DISTINCT
l.LocationID) as LocationsNumber from sales_per_order
spi, customer c, location l where
spi.CustomerID=c.CustomerID
and spi.LocationID=l.LocationID group by c.CustomerID
order by LocationsNumber desc;
```

   Visualisation:

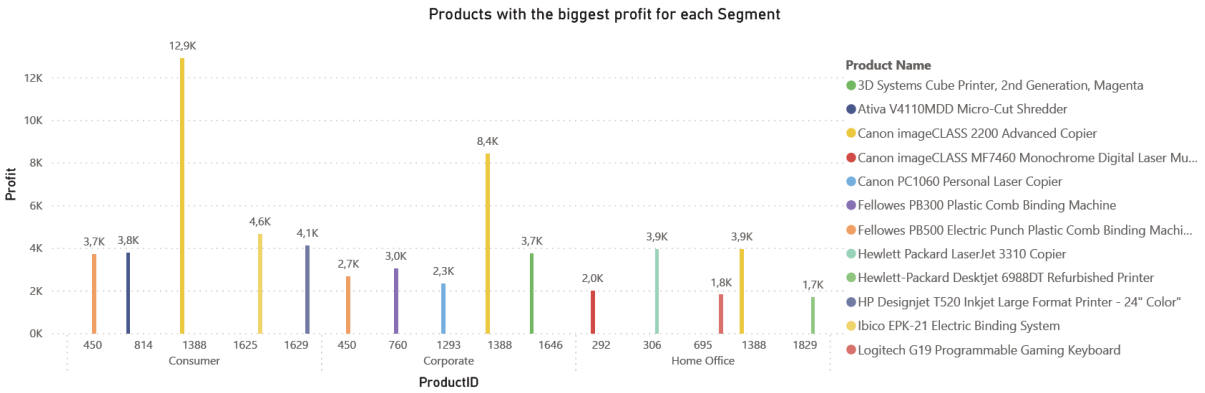


4. Products with the biggest profit for each customer segment
   SQL query:

```
SELECT
    c.Segment,
    p.ProductID,
    p.Name as ProductName,
    SUM(spi.Profit) as Profit,
    SUM(spi.Sales) as Sales
FROM
    sales_per_item spi
    JOIN customer c ON spi.CustomerID = c.CustomerID
    JOIN product p ON spi.ProductID = p.ProductID
    JOIN (
        SELECT
            spi2.ProductID,
            c2.Segment,
```

```
            RANK() OVER (PARTITION BY c2.Segment ORDER
            BY SUM(spi2.Profit) DESC) as ProfitRank
        FROM
            sales_per_item spi2
            JOIN customer c2 ON spi2.CustomerID =
            c2.CustomerID
        GROUP BY
            spi2.ProductID, c2.Segment
    ) as RankedProducts ON p.ProductID =
    RankedProducts.ProductID AND c.Segment =
    RankedProducts.Segment
WHERE
    RankedProducts.ProfitRank <= 5
GROUP BY
    c.Segment,
    p.ProductID order by c.Segment, SUM(spi.Profit)
DESC;
```

Visualisation:

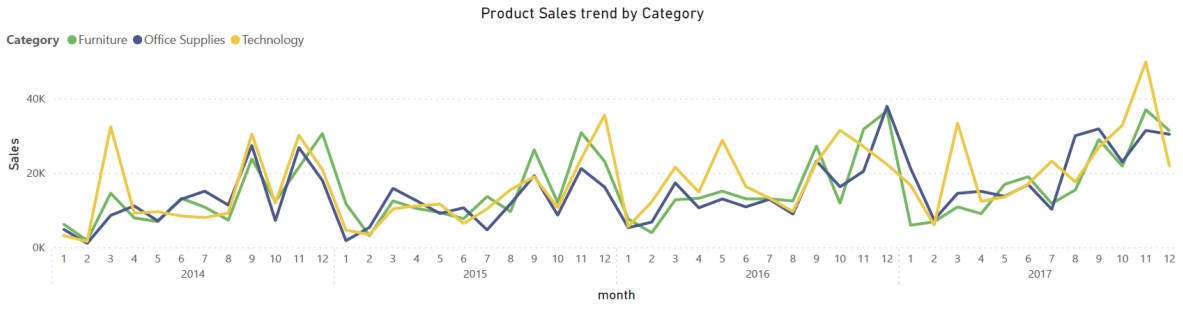


5. Product sales trend by category

SQL query:

```
SELECT p.Category, t.year, t.month, SUM(spi.Sales) as
Sales from sales_per_item spi, product p, time t
WHERE spi.ProductID=p.ProductID and
spi.OrderDateID=t.TimeID group by p.Category, t.year,
t.month order by t.year asc, t.month asc,
Profit DESC;
```

Visualisation:

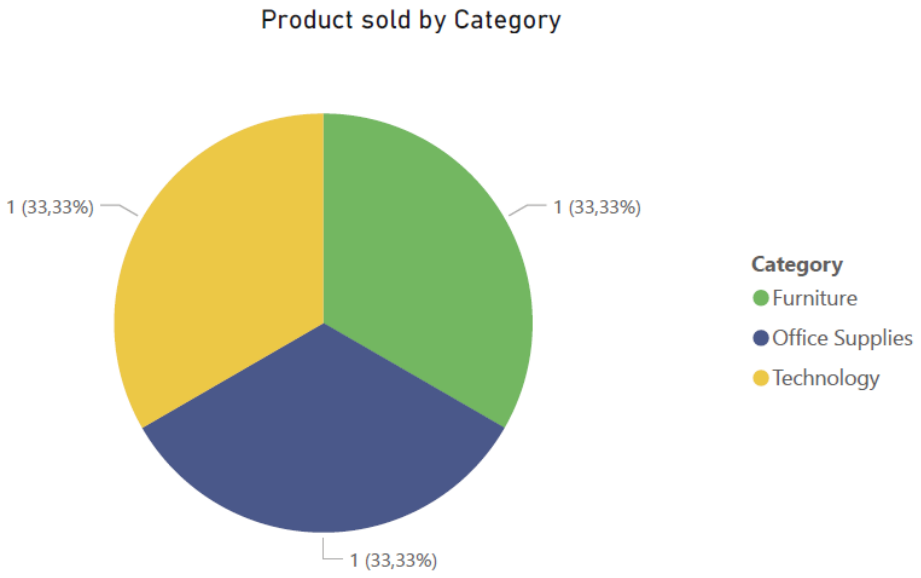


6. Products sold by category

   SQL query:

```
SELECT p.Category, SUM(i.Quantity) as Quantity
FROM store.product as p JOIN store.item_sales as i on
p.ProductID=i.ProductID group by p.Category;
```

   Visualisation:

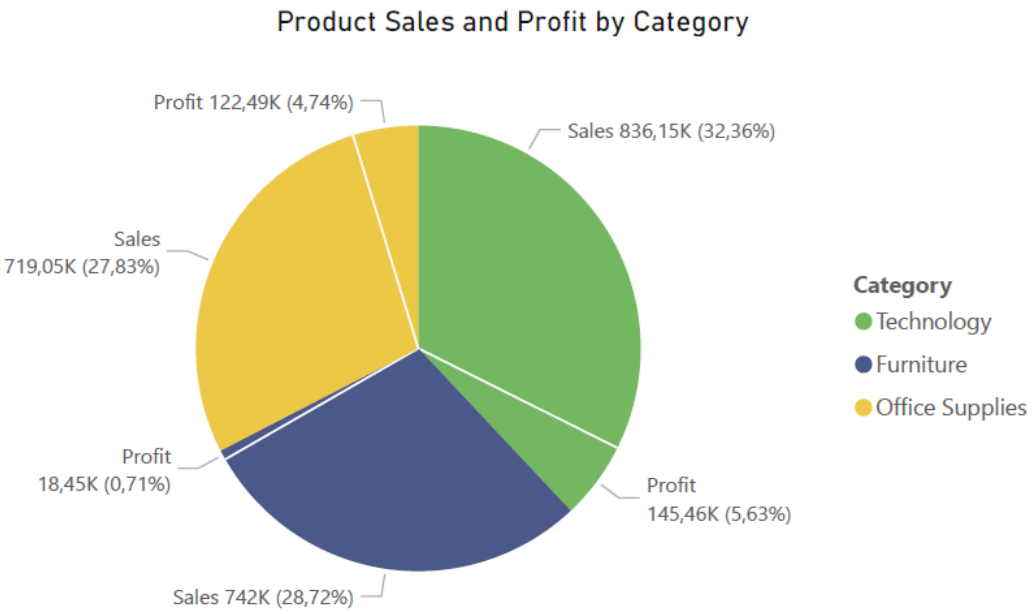


7. Product sales and profit by category

   SQL query:

```
SELECT p.Category, SUM(i.Sales) as Sales,
SUM(i.Profit) as Profit
FROM store.product as p JOIN store.item_sales as i on
p.ProductID=i.ProductID group by p.Category;
```

   Visualisation:

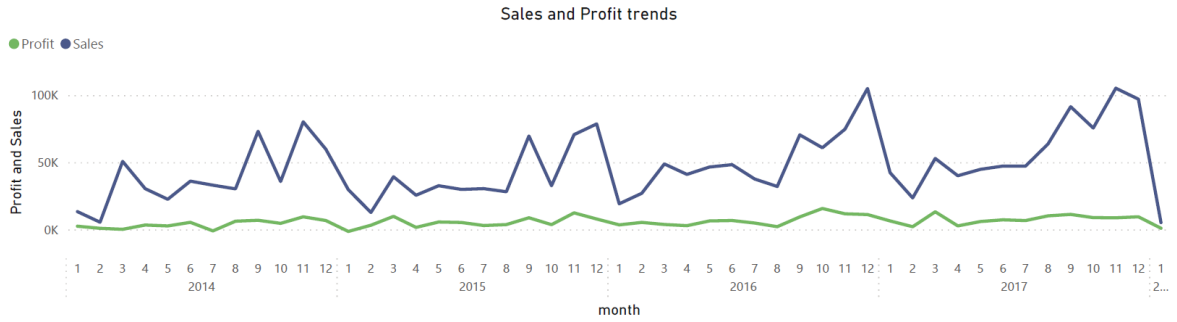


8. Sales and profit trends

   SQL query:

```
SELECT year, month, SUM(Profit) as Profit, SUM(Sales)
as Sales FROM sales_per_order spo join time t on
spo.ShipDateID=t.TimeID group by t.year, t.month order
by year, month;
```

   Visualisation:



9. Profit margin

   SQL query:

```
SELECT year, month, SUM(Profit) / SUM(Sales) * 100 as
'Profit Margin' FROM sales_per_order spo, time t where
spo.ShipDateID=t.TimeID group by t.year, t.month order
by year, month;
```

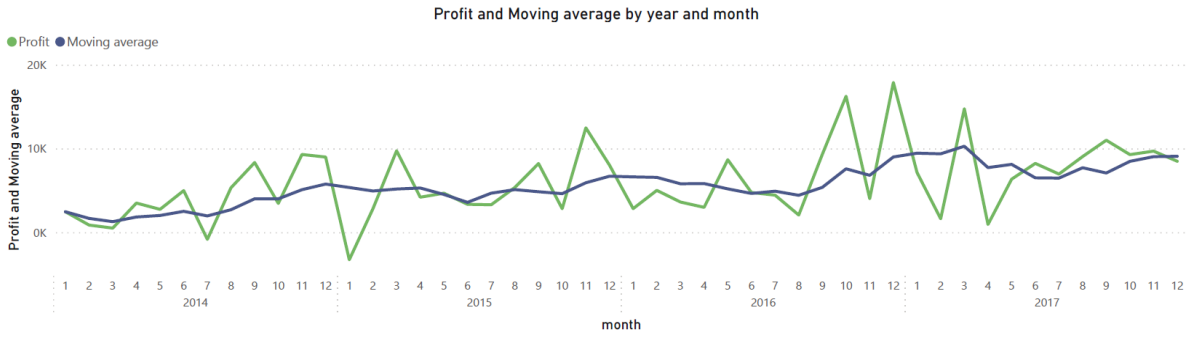
   Visualisation:

10. Profit and moving average

SQL query:

```
SELECT year, month, sum(profit) as profit,
avg(sum(profit)) over(order by t.year, t.month rows
between 5 preceding and current row) as ma_profit
FROM sales_per_item spi, time t
WHERE spi.orderdateid=t.timeid group by t.year,
t.month;
```

Visualisation:

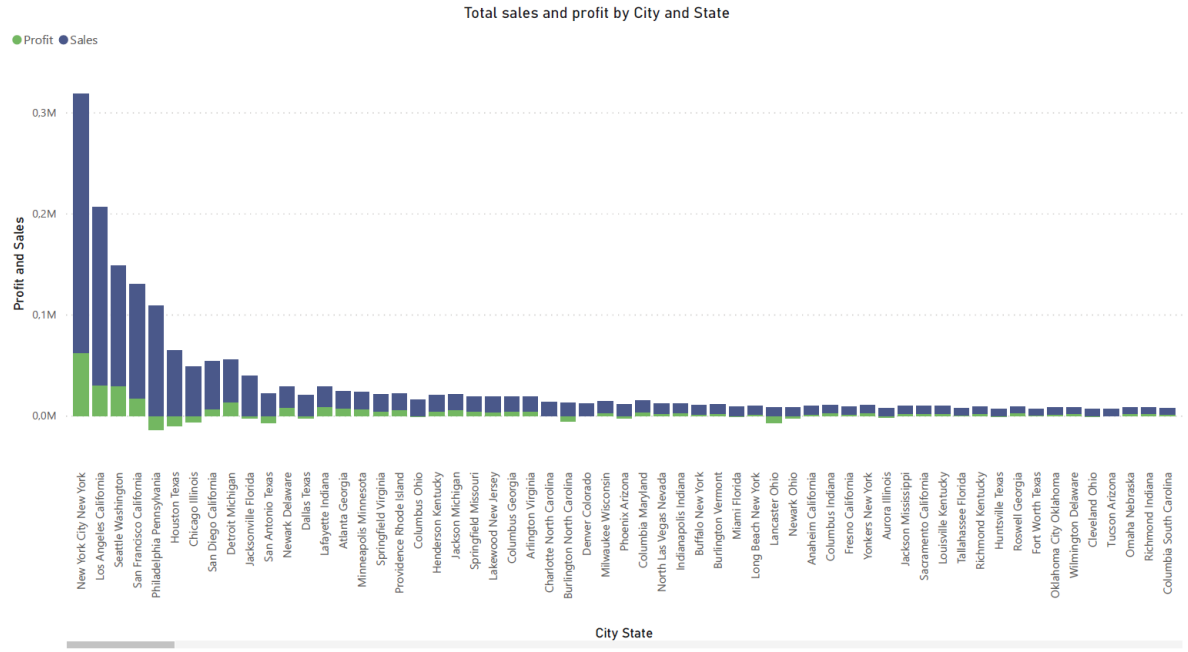


11. Sales and profit by location

SQL query:

```
SELECT l.Country, l.City, l.State, l.Region,
SUM(spo.Profit) as Profit, SUM(spo.Sales) as Sales
FROM sales_per_order spo join location l on
spo.LocationID=l.LocationID GROUP BY l.Country,
l.City, l.State, l.Region;
```

Visualisation:

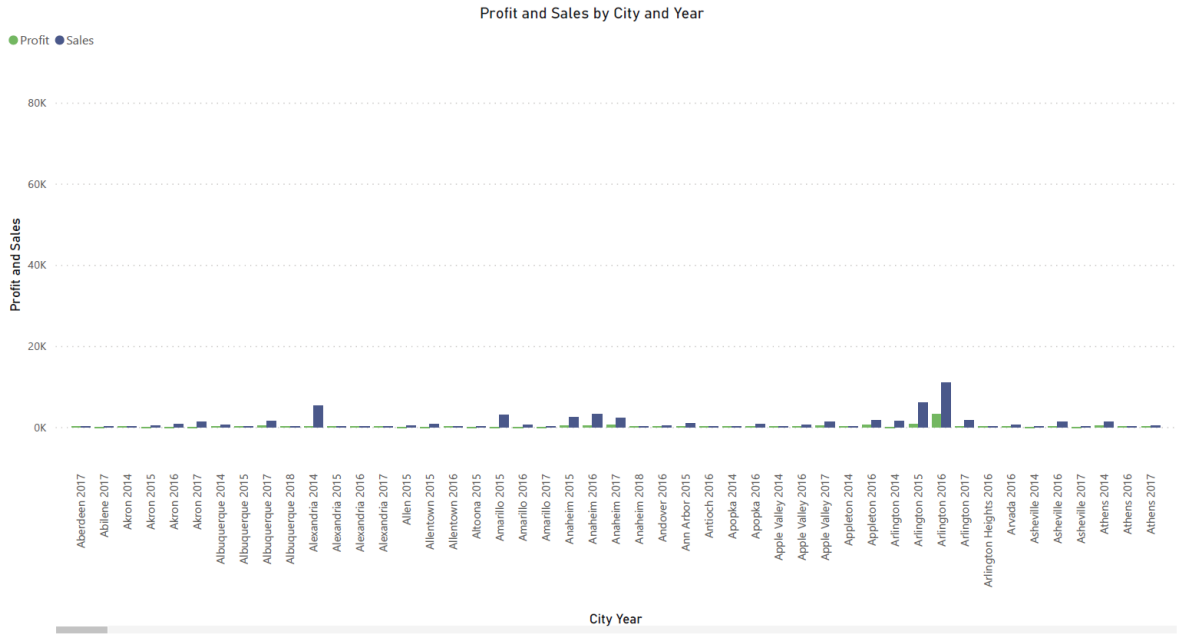


12. Sales and profit for every city through time

SQL query:

```
SELECT l.Country, l.State, l.Region, l.City, t.Year,
t.Month, SUM(spo.Profit) as Profit, SUM(spo.Sales) as
Sales
FROM location l, sales_per_order spo, time t
WHERE l.LocationID=spo.LocationID and
t.TimeID=spo.ShipDateID group by l.Country, l.State,
l.Region, l.City, t.Year, t.Month order by t.Year ASC,
t.Month ASC, Profit DESC;
```

Visualisation:



6. Advantages and shortcomings in relation to operational databases

Data warehouses are systems for data analysis and decision-making, this means that we want the data to be structured in a way that enhances the database's analytical capabilities. For this purpose data warehouses have denormalized schemas. This architecture intentionally introduces redundancy and data duplication in order to optimise the database for analytical operations. Redundancy, on the other hand, requires substantial storage resources and can lead to data inconsistency when data is updated. We are unlikely to encounter these concerns if we use an operational database with a normalised schema.

Furthermore, while operational databases specialise in transactional processing for day-to-day operations, data warehouses are more focused on historical data. To be able to perform various analyses, data warehouses have multiple data sources. This leads to complex data integration and ETL processes which can be a significant challenge. As a result of the way data is organised, when it comes to analysis queries data warehouses have a better performance than operational databases.

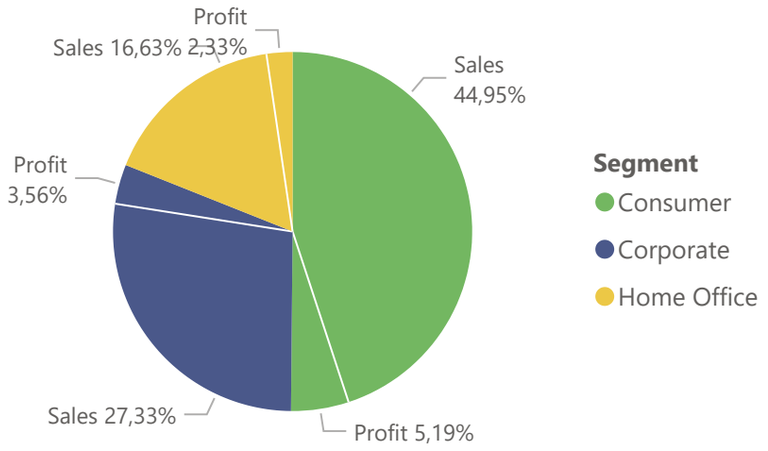
In conclusion, while operational databases focus on current, real-time transactions with their normalised schemas, data warehouses, characterised by denormalized structures, cater to analytical needs and historical data

analysis. Each of them is optimised for different scenarios and has unique advantages and disadvantages tailored to their intended use cases.

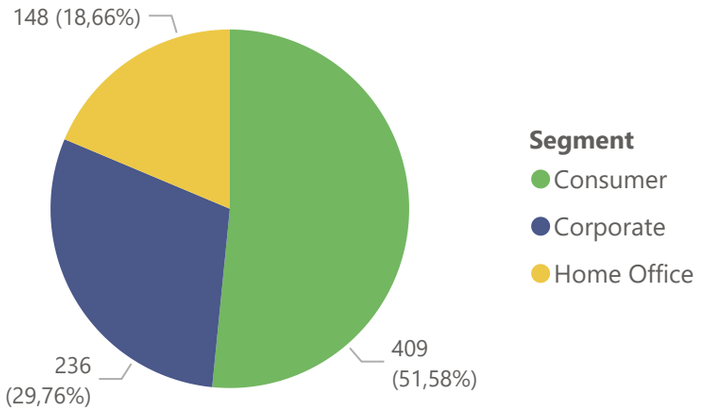
7. Conclusion

Through carefully selecting our data source and our analytical dimensions, as well as performing efficient ETL processes we successfully designed and implemented our dimensional data model and data warehouse. This enabled our in-depth data analysis spanning multiple dimensions including customer behaviours, product performance, and temporal and geographical trends. In the future, the data warehouse can be updated with new data that will allow for updates as well as new analyses that will be of great value to the store.

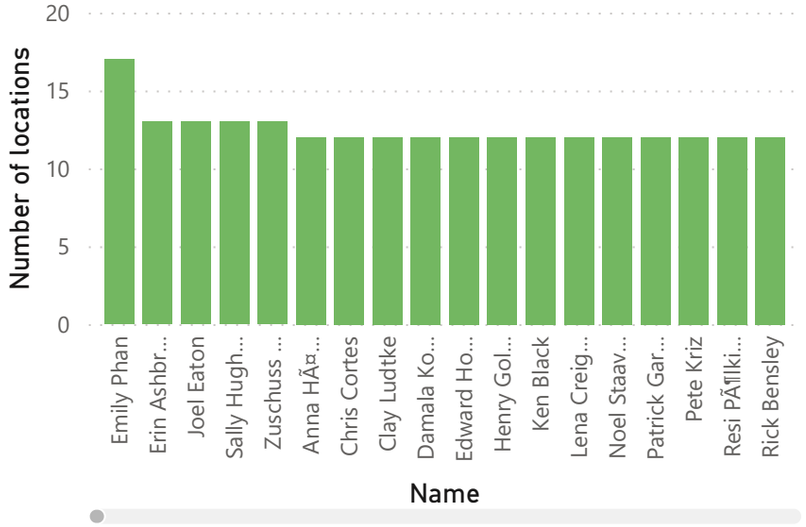
Sales and Profit by Segment



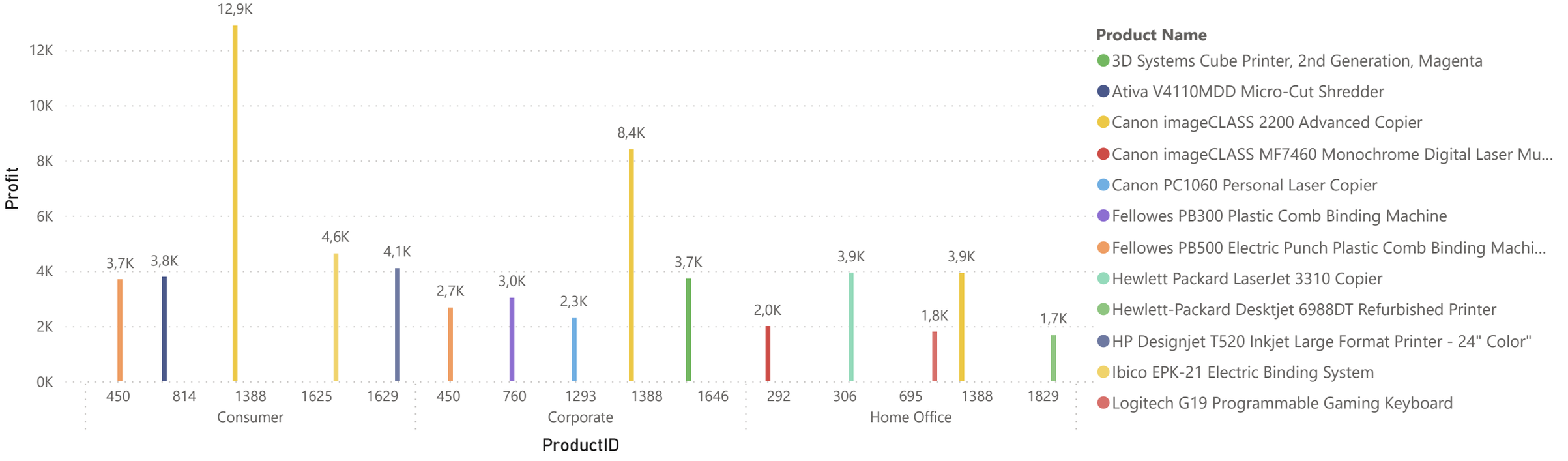
Number of Customers by Segment



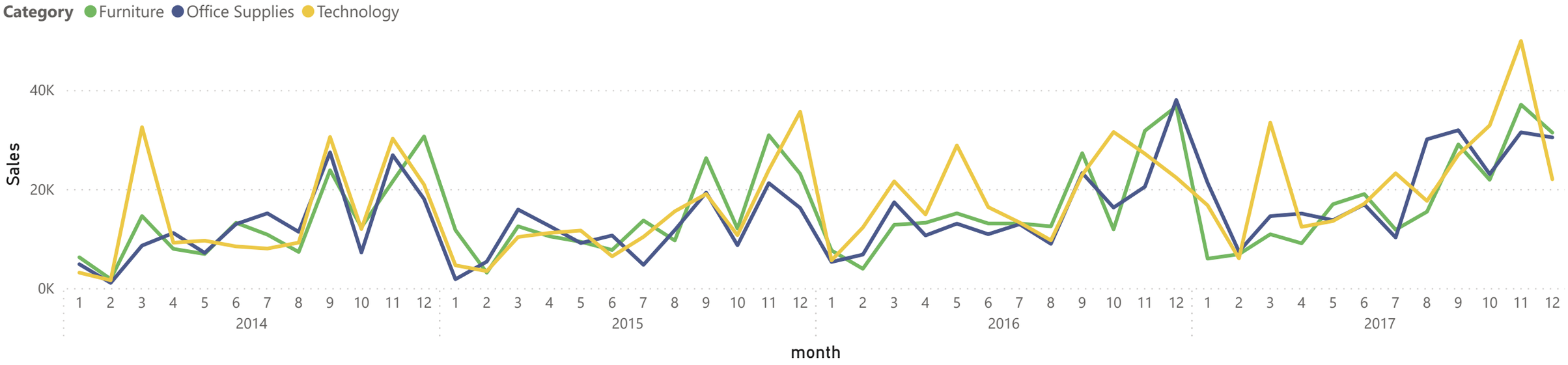
Number of locations by Customer



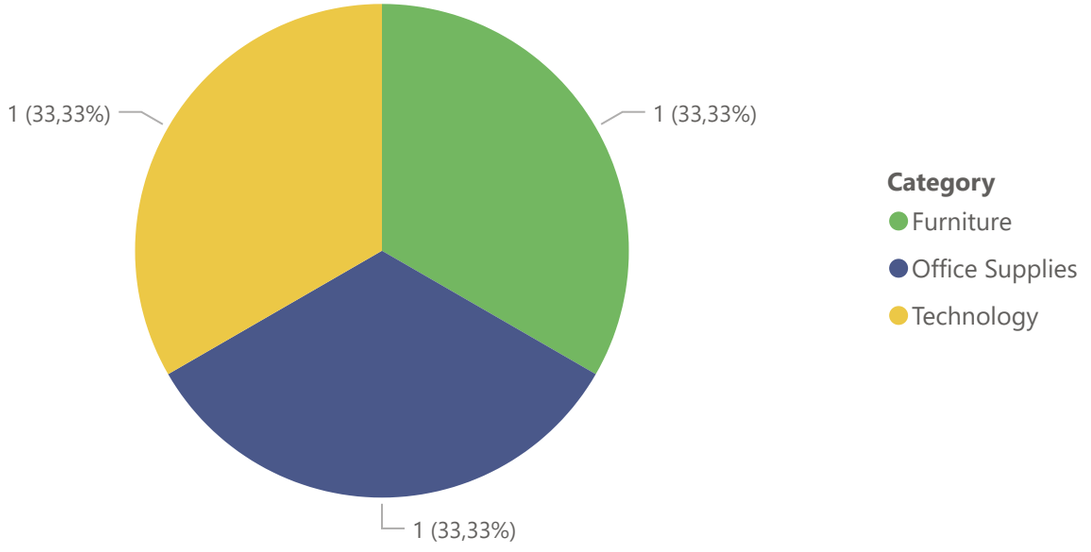
Products with the biggest profit for each Segment



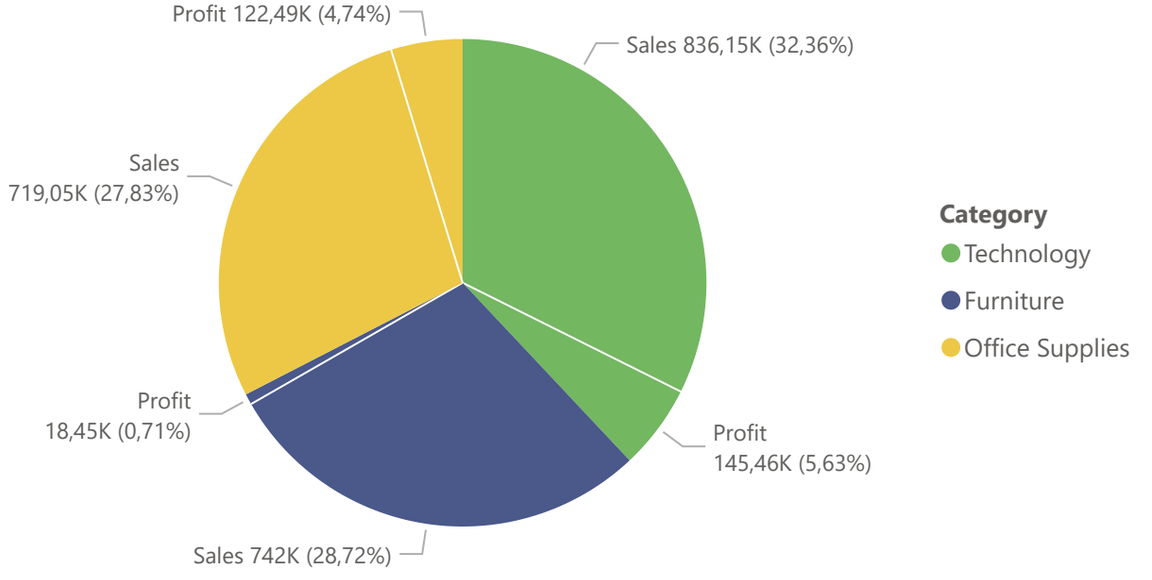
Product Sales trend by Category



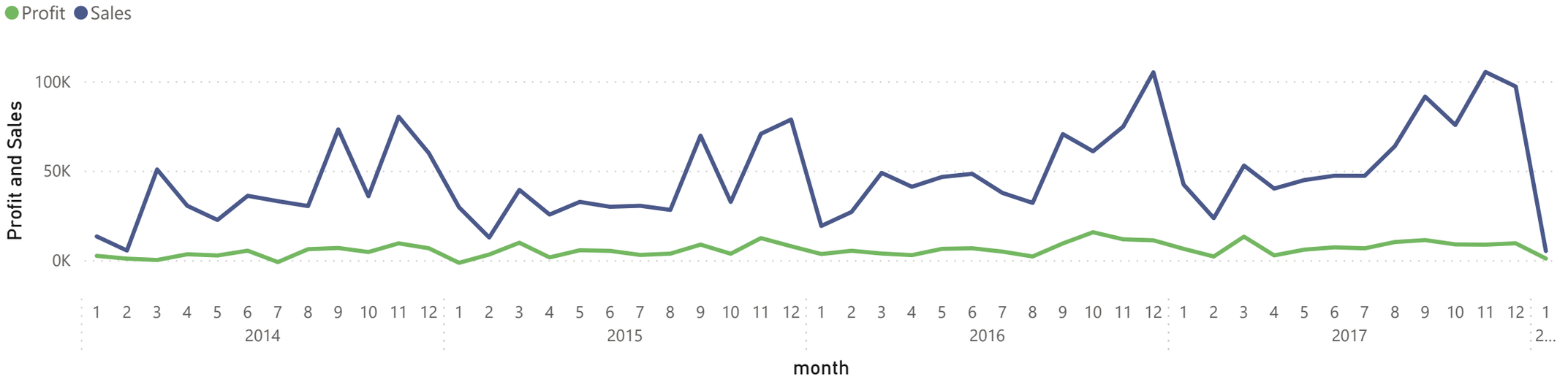
Product sold by Category



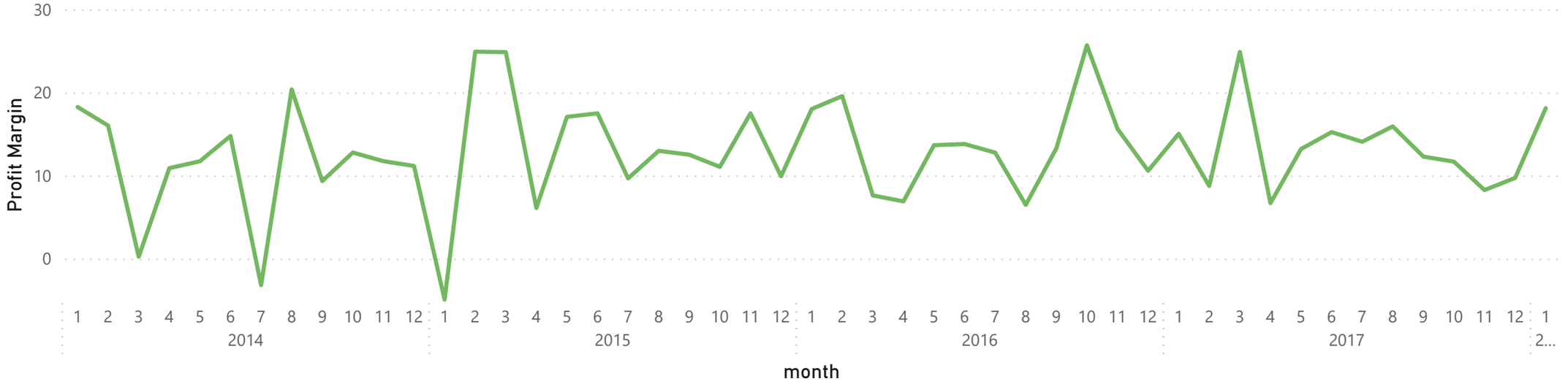
Product Sales and Profit by Category

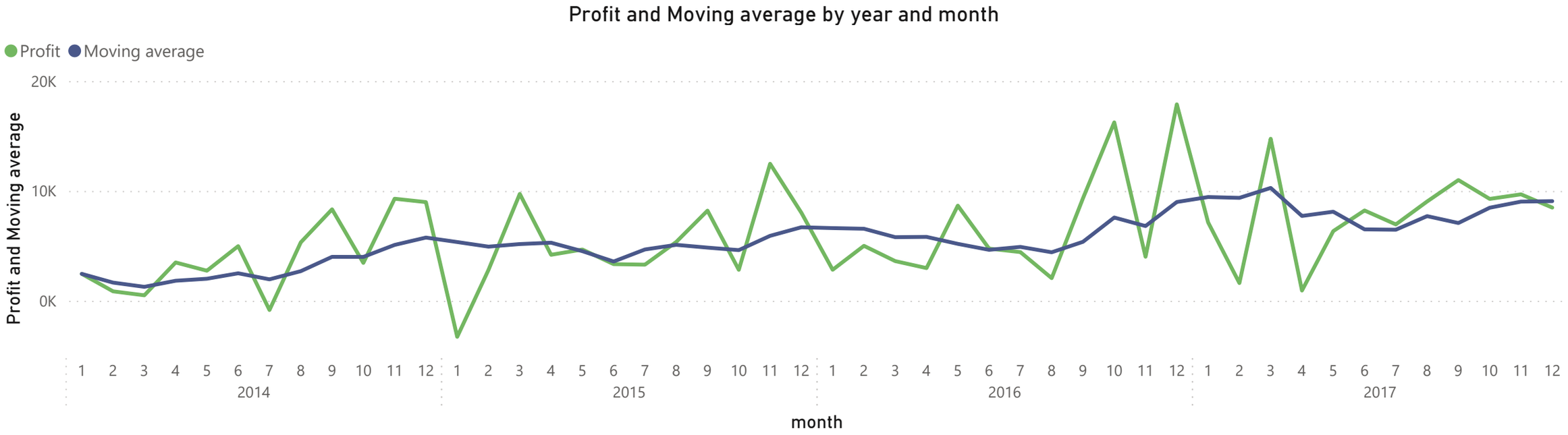


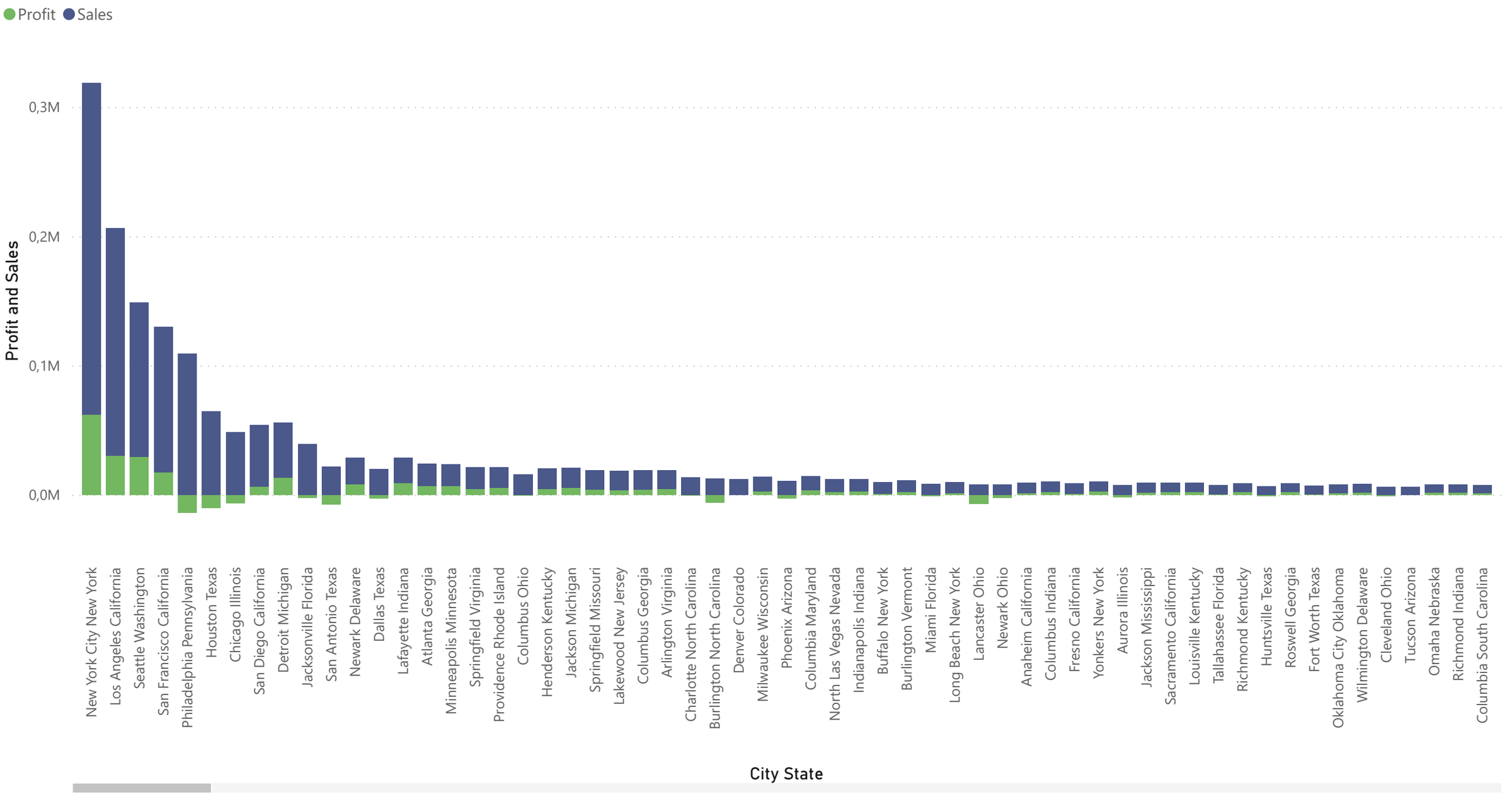
Sales and Profit trends

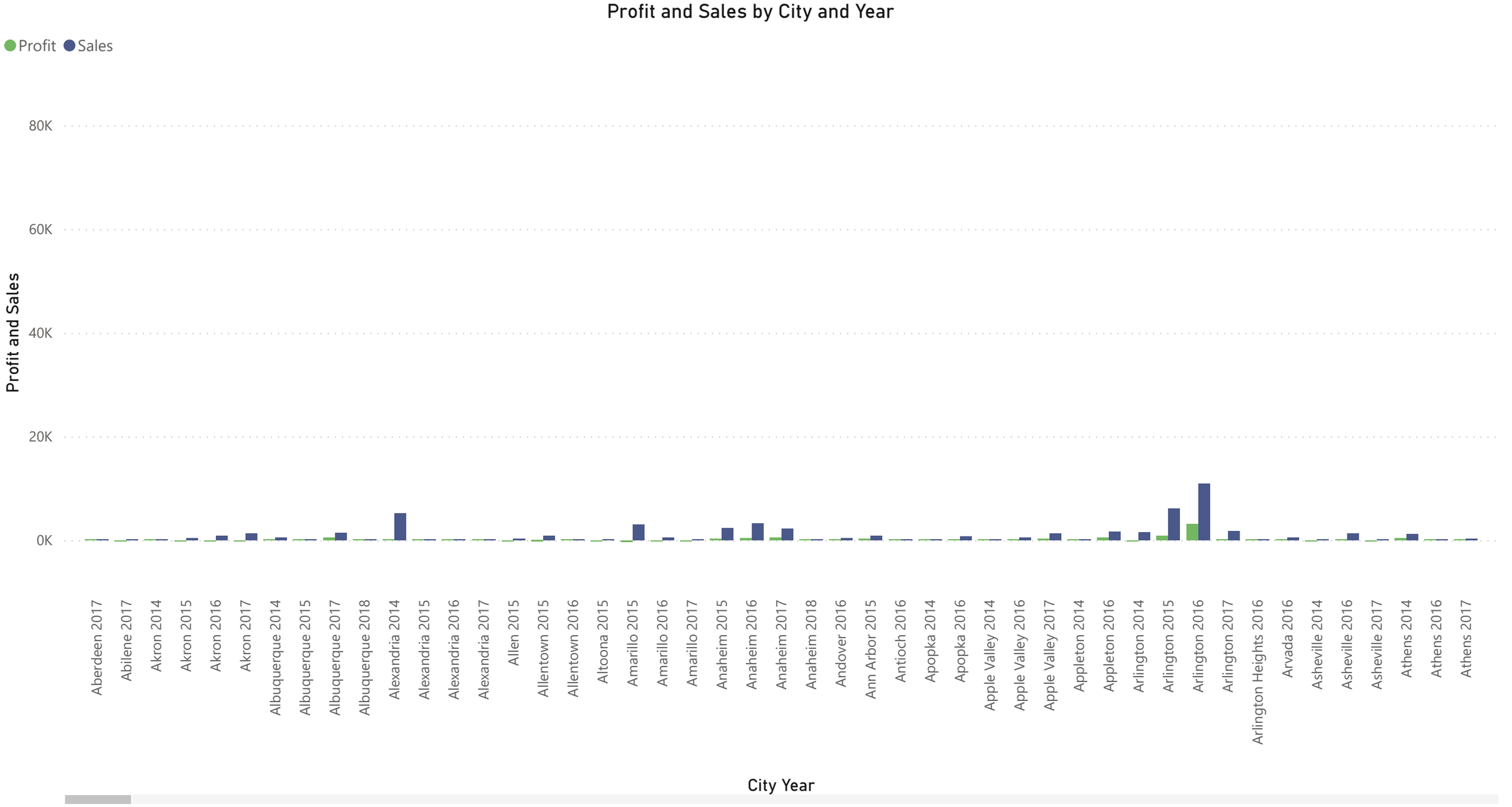


Profit Margin by year and month




Profit and Moving average by year and month
Profit
Moving average
20K
10K
0K
Profit and Moving average
1 2 3 4 5 6 7 8 9 10 11 12
2014
2015
2016
2017
month


Total sales and profit by City and State
Profit
Sales
Profit and Sales
0,3M
0,2M
0,1M
0,0M
New York City New York
Los Angeles California
Seattle Washington
San Francisco California
Philadelphia Pennsylvania
Houston Texas
Chicago Illinois
San Diego California
Detroit Michigan
Jacksonville Florida
San Antonio Texas
Newark Delaware
Dallas Texas
Lafayette Indiana
Atlanta Georgia
Minneapolis Minnesota
Springfield Virginia
Providence Rhode Island
Columbus Ohio
Henderson Kentucky
Jackson Michigan
Springfield Missouri
Lakewood New Jersey
Columbus Georgia
Arlington Virginia
Charlotte North Carolina
Burlington North Carolina
Denver Colorado
Milwaukee Wisconsin
Phoenix Arizona
Columbia Maryland
North Las Vegas Nevada
Indianapolis Indiana
Buffalo New York
Burlington Vermont
Miami Florida
Long Beach New York
Lancaster Ohio
Newark Ohio
Anaheim California
Columbus Indiana
Fresno California
Yonkers New York
Aurora Illinois
Jackson Mississippi
Sacramento California
Louisville Kentucky
Tallahassee Florida
Richmond Kentucky
Huntsville Texas
Roswell Georgia
Fort Worth Texas
Oklahoma City Oklahoma
Wilmington Delaware
Cleveland Ohio
Tucson Arizona
Omaha Nebraska
Richmond Indiana
Columbia South Carolina
City State


Profit and Sales by City and Year
Profit
Sales
80K
60K
40K
20K
0K
Profit and Sales
Aberdeen 2017
Abilene 2017
Akron 2014
Akron 2015
Akron 2016
Akron 2017
Albuquerque 2014
Albuquerque 2015
Albuquerque 2017
Albuquerque 2018
Alexandria 2014
Alexandria 2015
Alexandria 2016
Alexandria 2017
Allen 2015
Allentown 2015
Allentown 2016
Altoona 2015
Amarillo 2015
Amarillo 2016
Amarillo 2017
Anaheim 2015
Anaheim 2016
Anaheim 2017
Anaheim 2018
Andover 2016
Ann Arbor 2015
Antioch 2016
Apopka 2014
Apopka 2016
Apple Valley 2014
Apple Valley 2016
Apple Valley 2017
Appleton 2014
Appleton 2016
Arlington 2014
Arlington 2015
Arlington 2016
Arlington 2017
Arlington Heights 2016
Arvada 2016
Asheville 2014
Asheville 2016
Asheville 2017
Athens 2014
Athens 2016
Athens 2017
City Year