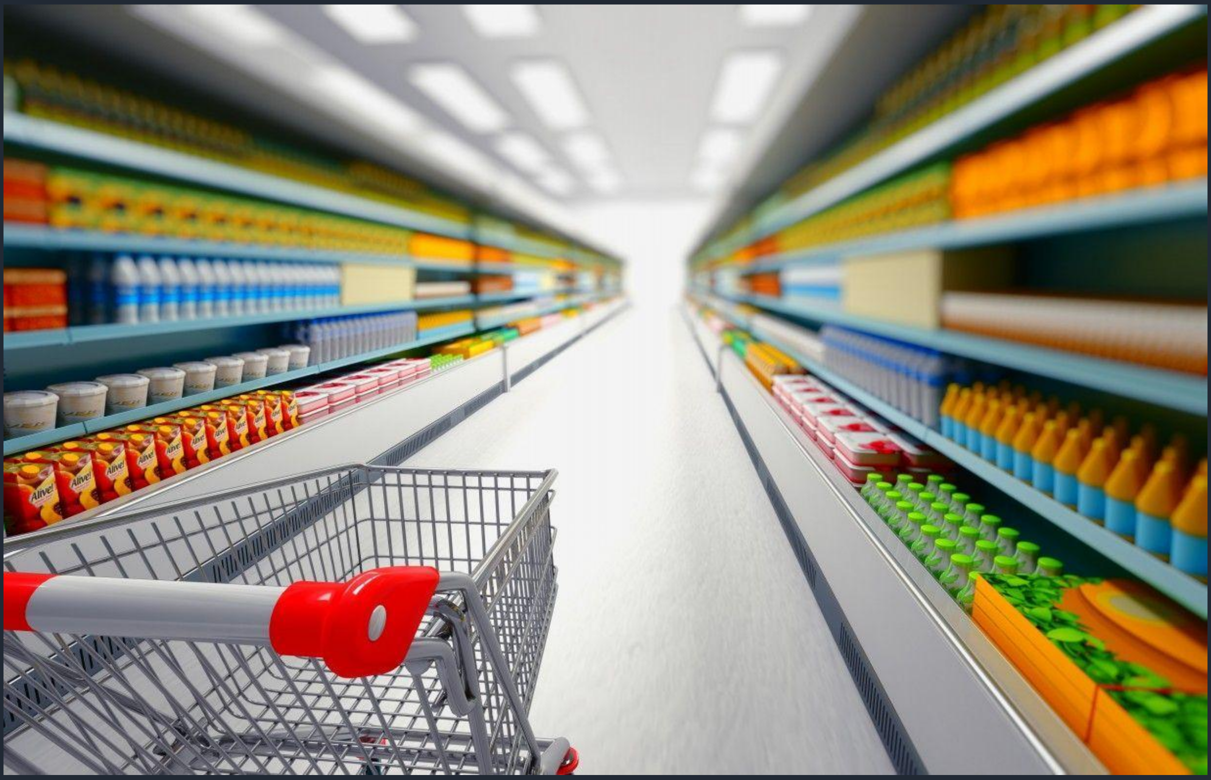
# Sample Superstore



## About Dataset:

The dataset contains information about products, sales, profits, region and so on that you can use to identify key areas for improvement within this fictitious company.

The dataset has three tables:

- Orders: It contains info related orders like customer name, sales, profit and many more.
- Returns: It contains the data related the returned order.
- Location: It contains the address of the order.

## Metadata:

1. Row ID => Unique ID for each row.
2. Order ID => Unique Order ID for each Customer.
3. Order Date => Order Date of the product.
4. Ship Date => Shipping Date of the Product.
5. Ship Mode=> Shipping Mode specified by the Customer.
6. Customer ID => Unique ID to identify each Customer.
7. Customer Name => Name of the Customer.
8. Segment => The segment where the Customer belongs.
9. Country => Country of residence of the Customer.
10. City => City of residence of the Customer.
11. State => State of residence of the Customer.
12. Postal Code => Postal Code of every Customer.
13. Region => Region where the Customer belong.
14. Product ID => Unique ID of the Product.
15. Category => Category of the product ordered.
16. Sub-Category => Sub-Category of the product ordered.
17. Product Name => Name of the Product
18. Sales => Sales of the Product.
19. Quantity => Quantity of the Product.
20. Discount => Discount provided.
21. Profit => Profit/Loss incurred.

## Tech Used:

### MySQL Workbench 8.0 CE

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

- MySQL is released under an open-source license. So, you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.

The most widely used open-source database worldwide is MySQL. MySQL is the second-most popular database, after Oracle Database, according to DB-Engines. Numerous of the most popular apps, including as Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com, are powered by MySQL.

Because MySQL is open source, it has many features that have been created over more than 25 years in close collaboration with users. Therefore, it is extremely possible that MySQL Database supports your preferred application or programming language.

# Insights:

## 1) Which is the most loss-making category in the East region?

The most loss-making category in east region is Technology with a loss amount of **-6599.978**

Query:

```
WITH most_loss AS
(
SELECT category, profit,
        RANK() OVER(ORDER BY profit) AS RNK
FROM orders
WHERE region = 'East'
)
SELECT category, profit as max_lost
FROM most_loss
WHERE RNK = 1;
```

Result:

| Category | max_lost |
|---|---|
| Technology | -6599.978 |

## 2) Give me the top 3 product ids by most returns?

The top 3 product ids by most returns are OFF-PA-10001970, OFF-BI-10002215, FUR-CH-10003968 and many more are there.

Query:

```
WITH most_return AS
(
SELECT ret.orderID, ord.`Product ID` AS ProductID
FROM returns AS ret
INNER JOIN orders AS ord
ON ret.orderID = ord.`Order ID`
ORDER BY 2
)
SELECT ProductID, max_return
```

```
FROM
(
SELECT ProductID, COUNT(ProductID) AS max_return,
            DENSE_RANK() OVER(ORDER BY COUNT(ProductID) DESC) AS RNK
FROM most_return
GROUP BY 1
ORDER BY 2 DESC
) T1
WHERE RNK BETWEEN 1 AND 3
```

## 3) In which city the greatest number of returns are being recorded?

The greatest number of returns are being recorded in Seattle city.

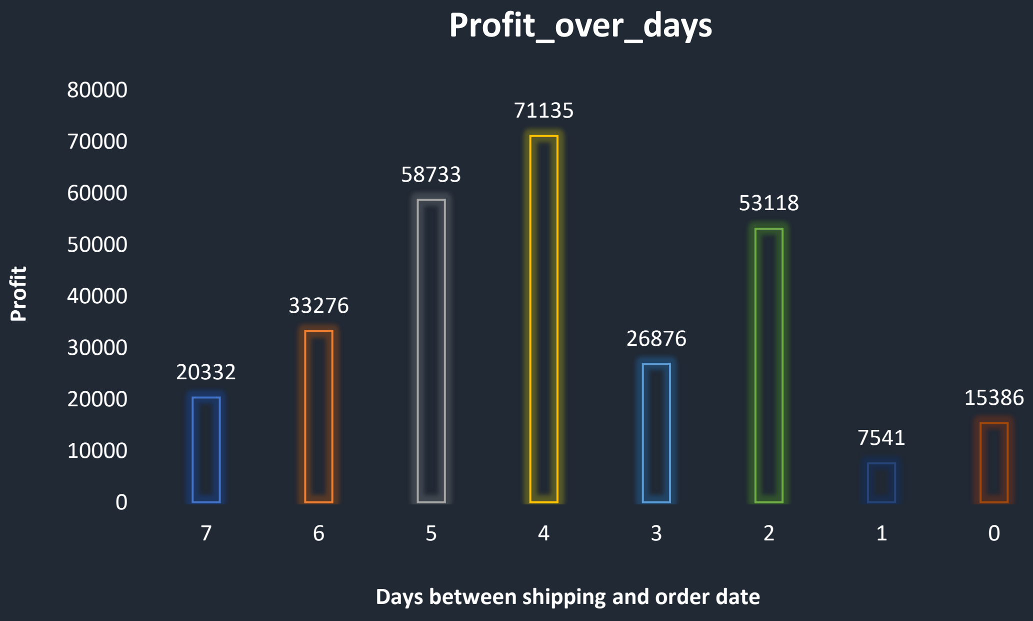
## Query:

```
WITH postal_code AS
(
SELECT distinct ret.orderID, ord.`Postal Code` as PostalCode
FROM returns AS ret
INNER JOIN orders AS ord
ON ret.orderID = ord.`Order ID`
),
max_return as
(
SELECT PostalCode, max_post
FROM
(
SELECT PostalCode, COUNT(PostalCode) AS max_post,
            DENSE_RANK() OVER(ORDER BY COUNT(PostalCode) DESC) AS RNK
FROM postal_code
GROUP BY 1
ORDER BY 2 DESC
) T1
WHERE RNK = 1
)
SELECT City
FROM
(
SELECT mr.PostalCode AS PinCode, loc.City AS City
FROM max_return AS mr
INNER JOIN location AS loc
ON mr.PostalCode = loc.`Postal Code`
) T2;
```

Result:

| City |
|---|
| Seattle |

4) Find the relationship between days between order date, ship date and profit?



The profit is high when the order is delivered within 4 days of the ordering date.

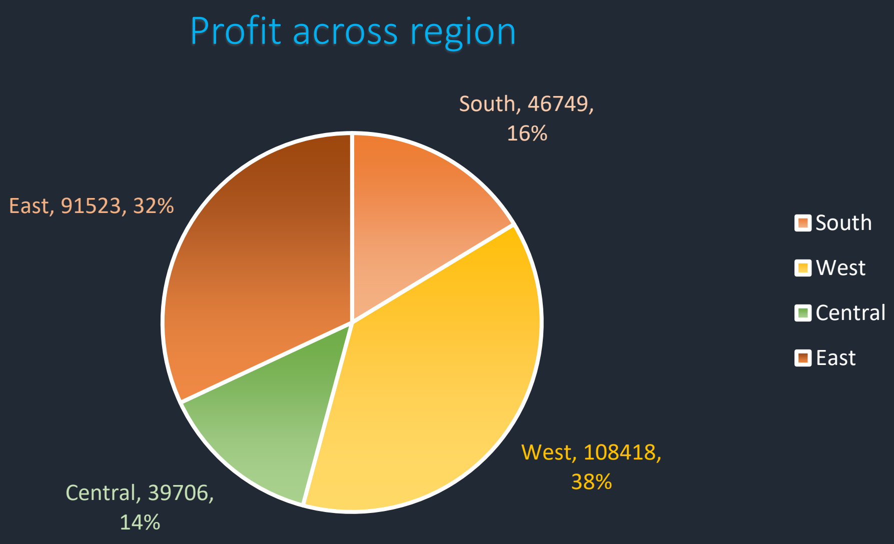
## Query:

```
WITH profit_wrt_days AS

(SELECT `Order Date`, `Ship Date`, DATEDIFF( `Ship Date`,`Order Date`) AS
days_bet_order_and_shipping, Profit,

            ROUND(SUM(Profit) OVER(PARTITION BY DATEDIFF( `Ship Date`,`Order Date`)
ORDER BY DATEDIFF( `Ship Date`,`Order Date`) DESC ),2) AS Profit_over_days

FROM Orders

ORDER BY 3 DESC

)

SELECT DISTINCT days_bet_order_and_shipping, Profit_over_days

FROM profit_wrt_days
```

## Result:

| days_bet_order_and_shipping | Profit_over_days |
|---|---|
| 7 | 20332 |
| 6 | 33276 |
| 5 | 58733 |
| 4 | 71135 |
| 3 | 26876 |
| 2 | 53118 |
| 1 | 7541 |
| 0 | 15386 |

5) Find the region wise profits for all the regions and give the output of the most profitable region.



The most profitable region is west with profit amount of 108418

Query:

A) Region wise profit:

```
SELECT Region, ROUND(SUM(Profit),2) AS Profit

FROM Orders

GROUP BY 1;
```

Result:

| Region | Profit |
|---|---|
| South | 46749.43 |
| West | 108418.45 |
| Central | 39706.36 |
| East | 91522.78 |

B) Region wise maximum profit:

```
WITH MAX_PROFIT AS
(
SELECT Region, ROUND(SUM(Profit),2) AS Profit,
        RANK() OVER(ORDER BY ROUND(SUM(Profit),2) DESC) AS RNK
FROM Orders
GROUP BY 1
)
SELECT REGION
FROM MAX_PROFIT
WHERE RNK = 1;
```

Result:

| Region |
|---|
| West |

## 6) Which month observe the highest number of orders placed and return placed for each year?

In the month of February of 2014, 46 orders were placed, which was the most.

In the month of December of 2017, 18 orders were returned, which was the most.

Query:

A) Highest number of orders placed:

```
WITH max_order AS
```

```
(

SELECT YEAR(`ORDER DATE`) AS ORDER_YEAR, MONTH(`ORDER DATE`) AS
ORDER_MONTH, COUNT(`ORDER DATE`) AS ORDER_PLACED

FROM Orders

GROUP BY 1,2

ORDER BY 1

)

SELECT ORDER_YEAR, ORDER_MONTH, ORDER_PLACED

FROM

(

SELECT ORDER_YEAR, ORDER_MONTH, ORDER_PLACED,

            RANK() OVER(ORDER BY ORDER_PLACED) AS MAX_ORDER_OF_THE_YEAR

FROM max_order

)T4

WHERE MAX_ORDER_OF_THE_YEAR = 1;
```

Result:

| ORDER_YEAR | ORDER_MONTH | ORDER_PLACED |
|---|---|---|
| 2014 | 2 | 46 |

A) Highest number of orders returned:

```
WITH return_order AS

(

SELECT distinct `ORDER ID`, YEAR(`ORDER DATE`) AS RETURN_YEAR, MONTH(`ORDER
DATE`) AS RETURN_MONTH

FROM Orders AS ord

INNER JOIN returns AS ret

ON ord.`ORDER ID` = ret.OrderID

ORDER BY 2 ASC, 3 ASC

)
```

```
SELECT RETURN_MONTH, RETURN_YEAR, RETURN_ORDER_PER_MONTH AS
MAX_RETURN_ORDER

FROM

(

SELECT RETURN_MONTH, RETURN_YEAR, COUNT(RETURN_MONTH) AS
RETURN_ORDER_PER_MONTH,

    RANK() OVER(ORDER BY COUNT(RETURN_MONTH) DESC) AS RNK

FROM return_order

GROUP BY 1,2

) T3

WHERE RNK = 1
```

Result:

| RETURN_MONTH | RETURN_YEAR | MAX_RETURN_ORDER |
|---|---|---|
| 12 | 2017 | 18 |

7) Calculate percentage change in sales for the entire dataset?

X axis should be year_month, Y axis percent change

Find out if any sales pattern exists for all the region?

Query:

```
WITH pct_change AS

(

SELECT YEAR(`Order Date`) * 100 + MONTH(`Order Date`) as YearMonth, YEAR(`Order
Date`) AS 'YEAR', Region, ROUND(SUM(Sales), 2) AS sales,

    LEAD(ROUND(SUM(Sales), 2)) OVER(ORDER BY ROUND(SUM(Sales), 2)) AS
LED

FROM orders

GROUP BY 1,2,3

ORDER BY 3,1)
```

```
SELECT YearMonth, region, sales , Round(((LEAD(Sales) OVER(ORDER BY region) -
sales)/ sales) * 100, 2) AS PercentageChange

FROM pct_change;
```

## Result:

| YearMonth | Region | Sales | PercentageChange |
|---|---|---|---|
| 201401 | Central | 1539.91 | -19.92 |
| 201402 | Central | 1233.17 | 372.57 |
| 201403 | Central | 5827.6 | -36.3 |
| 201404 | Central | 3712.34 | 9.06 |
| 201405 | Central | 4048.51 | 138.27 |
| 201406 | Central | 9646.3 | -30.12 |
| 201407 | Central | 6740.57 | -55.16 |
| 201408 | Central | 3022.18 | 1038.54 |
| 201409 | Central | 34408.69 | -73.94 |
| ... | .... | ... | .... |
| ... | .... | ... | .... |

## 8) Top and bottom selling product for each region

The top selling products of each region is:

| Region | Product Name | Top_selling_product |
|---|---|---|
| Central | Easy-staple paper | 13 |
| Central | Staples | 13 |
| Central | Staple envelope | 13 |
| East | Staple envelope | 17 |
| South | Staples | 9 |
| South | Easy-staple paper | 9 |
| West | Staples | 13 |

## Query:

```
SELECT Region, `Product Name`, ProductSales AS Top_selling_product

FROM (
SELECT Region, `Product Name`, COUNT(`Product Name`) AS ProductSales,

        RANK() OVER(PARTITION BY Region ORDER BY COUNT(`Product Name`) DESC)
AS RNK

FROM orders

GROUP BY 1,2

) T5

WHERE RNK = 1;
```

The least selling products of each region is:

## Query:

```
SELECT Region, `Product Name`, ProductSales AS less_selling_product

FROM (

SELECT Region, `Product Name`, COUNT(`Product Name`) AS ProductSales,

        RANK() OVER(PARTITION BY Region ORDER BY COUNT(`Product Name`) ASC)
AS RNK

FROM orders

GROUP BY 1,2

) T5

WHERE RNK = 1

ORDER BY 2;
```

## Result:

| Region | Product Name | less_selling_product |
|---|---|---|
| West | "While you Were Out" Message Book, One Form per Page | 1 |
| Central | "While you Were Out" Message Book, One Form per Page | 1 |
| South | "While you Were Out" Message Book, One Form per Page | 1 |
| East | #10 Self-Seal White Envelopes | 1 |
| .. | ....... | ..... |

9) Why are returns initiated? Are there any specific characteristics for all the returns?  Hint: Find return across all categories to observe any pattern

The most orders that returns are under office supplies category. Across all category the order that returns most is fall under office supplies category.

Query:

```
WITH returned_order AS(
SELECT distinct `Order ID` as Returned_Order, Region, Category
FROM orders AS ord
INNER JOIN returns AS ret
ON ord.`Order ID` = ret.OrderID)

SELECT CATEGORY, COUNT(CATEGORY)
FROM returned_order
GROUP BY 1
ORDER BY 2 DESC
```

Result :

| CATEGORY | COUNT(CATEGORY) |
|---|---|
| Office Supplies | 234 |
| Furniture | 136 |
| Technology | 123 |

10) Create a table having two columns (date and sales), Date should start with the min date of data and end at max date - in between we need all the dates.

If date is available show sales for that date else show date and NA as sales.

Query:

```
with date_calendar as

(

select * from

(select adddate('1970-01-01',t4*10000 + t3*1000 + t2*100 + t1*10 + t0) gen_date from

 (select 0 t0 union select 1 union select 2 union select 3 union select 4 union select 5 union select 6 union select 7 union select 8 union select 9) t0,

 (select 0 t1 union select 1 union select 2 union select 3 union select 4 union select 5 union select 6 union select 7 union select 8 union select 9) t1,

 (select 0 t2 union select 1 union select 2 union select 3 union select 4 union select 5 union select 6 union select 7 union select 8 union select 9) t2,

 (select 0 t3 union select 1 union select 2 union select 3 union select 4 union select 5 union select 6 union select 7 union select 8 union select 9) t3,

 (select 0 t4 union select 1 union select 2 union select 3 union select 4 union select 5 union select 6 union select 7 union select 8 union select 9) t4) v

where gen_date between '2014-01-03' and '2017-12-30'

order by gen_date asc )

, common_date as

(

select distinct dc.gen_date as gen_date, date(ord.`Order Date`) as orderdate

from date_calendar as dc

left join orders as ord

on dc.gen_date = date(ord.`Order Date`)

), sales_date AS

(

select distinct cd.gen_date AS G_DATE, cd.orderdate as order_date, round(sum(ord.Sales), 2) as sum_of_sales

from common_date as cd

left join orders as ord
on cd.gen_date = ord.`Order date`

group by 1,2

order by 1

)

SELECT G_DATE, IF(sum_of_sales is not null, sum_of_sales, "NA") as sales

FROM sales_date
```

Result:

| G_DATE | sales |
|---|---|
| 2014-01-21 | 25.25 |
| 2014-01-22 | NA |
| 2014-01-23 | 46.02 |
| 2014-01-24 | NA |
| 2014-01-25 | NA |
| 2014-01-26 | 1097.25 |
| 2014-01-27 | 426.67 |
| 2014-01-28 | 3.93 |
| … | … |
| … | … |

- GitHub link: https://github.com/SAROJNILESH10/SAMPLE_SUPERSTORE

Thank You