SUPERSTORE SALES ANALYSIS & BUSINESS INSIGHTS

A SQL Project using MySQL Workbench & Microsoft Excel

Submitted By:

Karan Rawat

Date: 30th June 2025

Email: karanrawat2701@gmail.com

LinkedIn: Karan Rawat

Project overview:

Project Goal:

“This project analyzes Superstore sales data to uncover customer patterns, product trends, and regional performance using advanced SQL techniques.”

Tools Used:

- MySQL Workbench
- MySQL Database
- Excel Power Query (for cleaning)

Data preparation and cleaning:

Data source:

The dataset used for this project was sourced from Kaggle and contains historical sales transactions for a fictional Superstore.

The raw dataset had 9200 rows and 18 columns, covering order details, customers, shipping, and product data.

Dataset link: kaggle

Data cleaning:

Cleaned the raw data using Power Query in Microsoft Excel.

Cleaning Steps:

- Removed irrelevant columns (e.g., Postal Code)
- Fixed date formats to `YYYY-MM-DD`
- Removed duplicates
- Checked for NULLs or missing values
- Removed special characters
- Checked data types for each column

Database Design & ER Diagram

After data cleaning, the dataset was structured into a relational database using the principles of database normalization.

Normalization

The data was normalized to **Third Normal Form (3NF)** to:

- Eliminate redundancy
- Ensure data consistency
- Improve query efficiency

The raw flat data was broken into logical entities:

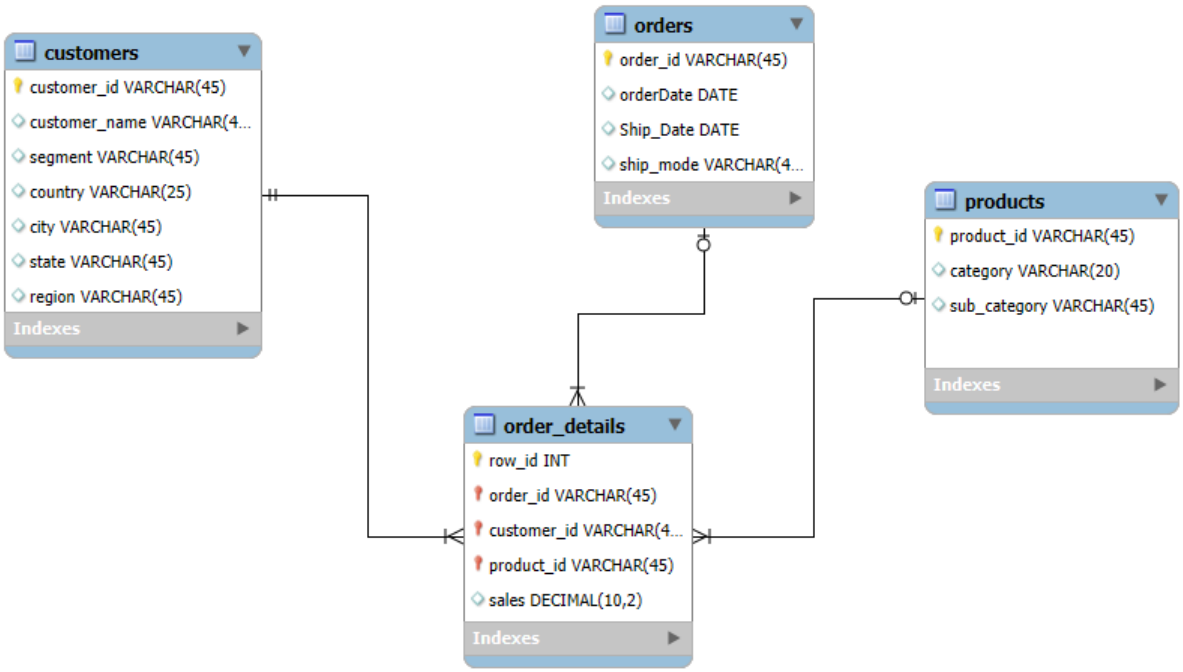
- Orders Table: Contain details about orders like order details, orderID , order date etc.
- Products Table: Contain details about products like productid, category etc.
- Customers Table: Contains customer details like customer name, country , state etc .
- Order_details table: Contains sales values and links other tables.

Primary & Foreign Keys

- order_id → primary key in orders, foreign key in order_details
- customer_id → primary key in customers, foreign key in order_details
- product_id → primary key in products, foreign key in order_details

ER Diagram

The Entity Relationship Diagram (ERD) was created using MySQL Workbench.
It visually represents the relationships between tables.



Data Loading & Transformation

To safely load the cleaned dataset into the normalized database, a staging table called staging_sales was created.

```
CREATE TABLE staging_sales (
  row_id INT,
  order_id VARCHAR(50),
  order_date DATE,
  ship_date DATE,
  ship_mode VARCHAR(50),
  customer_id VARCHAR(50),
  customer_name VARCHAR(100),
  segment VARCHAR(50),
  country VARCHAR(50),
  city VARCHAR(50),
  state VARCHAR(50),
  region VARCHAR(50),
  product_id VARCHAR(50),
  category VARCHAR(50),
  sub_category VARCHAR(50),
  sales DECIMAL(10,2);
```

The CSV file was imported using MySQL Workbench's Import Wizard, ensuring dates were converted to YYYY-MM-DD.

How Tables Were Created:

After designing the ER diagram in MySQL Workbench, the normalized tables (orders, customers, products, order_details) were **automatically created** in the MySQL database by using the **Forward Engineering** feature.

The ER diagram included all primary keys, foreign keys, and relationships, so the generated SQL script built the tables with constraints correctly.

This approach ensured that the database structure matched the design exactly without manually writing each CREATE TABLE statement.

Data was then inserted into the final tables (orders, customers, products, order_details) with appropriate keys.

For example:

```
INSERT IGNORE INTO customers (customer_id, customer_name, segment, country,city, state, region)

SELECT DISTINCT customer_id, customer_name, segment, country, city, state, region

FROM staging_sales;
```

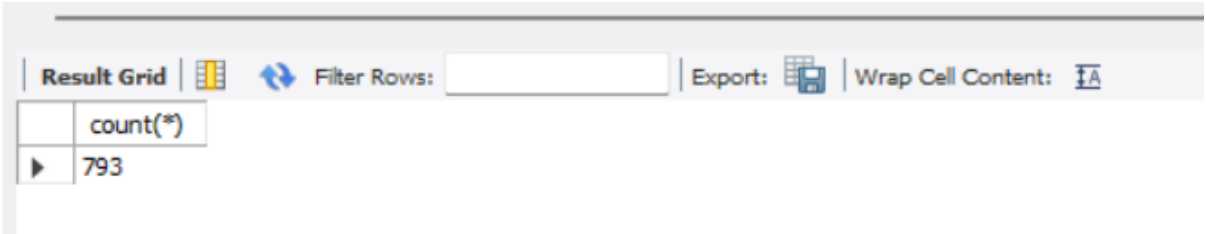
With this query the data was inserted from our staging table to our customers table.

Similarly, the data was inserted in all other tables.

```
select count(*) from customers;
```

This query was used for the row count verification of the customer table.

Output:

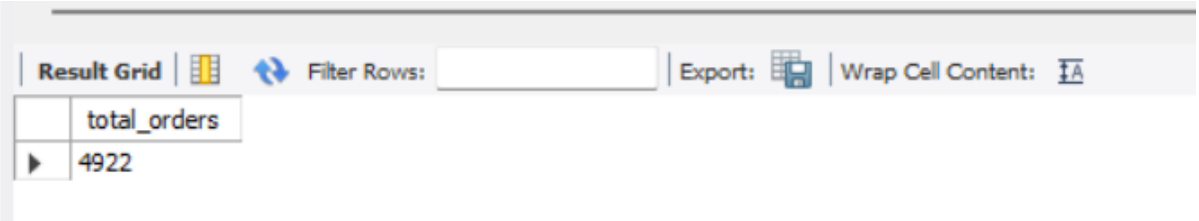


Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed to understand key trends, patterns, and anomalies in the Superstore sales dataset.
Basic queries were used to answer business questions related to sales performance, customer segments, product categories, and regions.

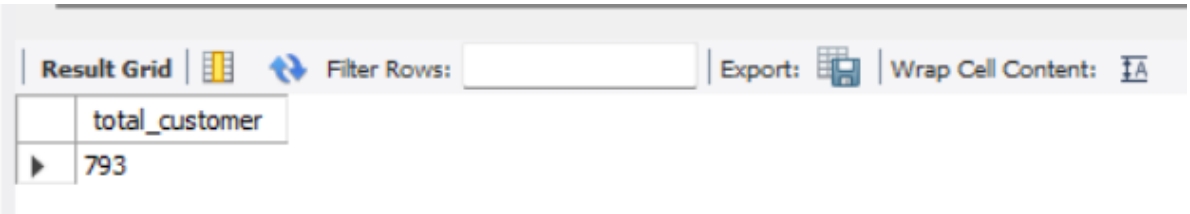
1) Total no. of orders

```
select count(distinct order_id) as total_orders from orders;
```



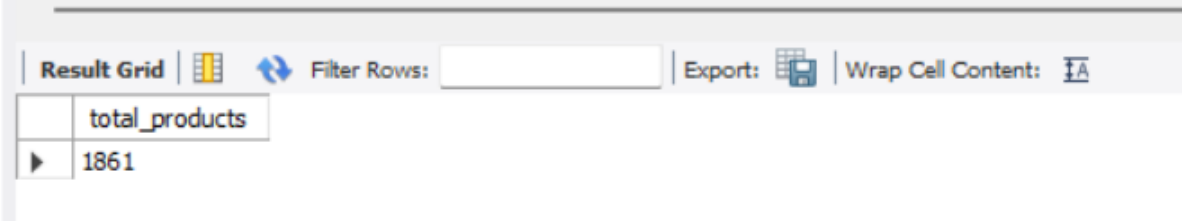
2) Total no. of customers.

```
select count(distinct customer_id) as total_customer from
customers;
```



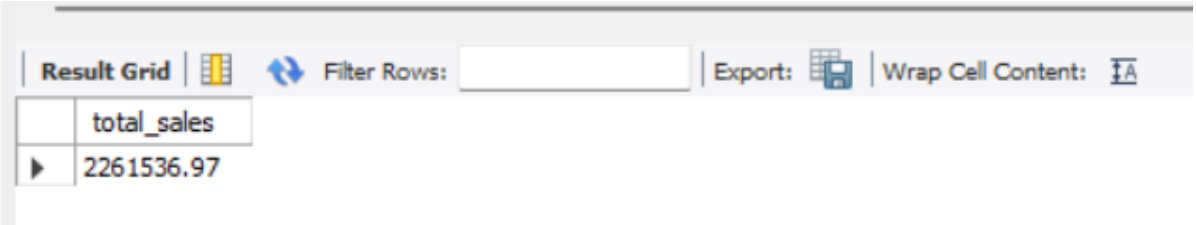
3) Total no. of products

```
select count(distinct product_id) as total_products from
products;
```



4) total sales

```
select sum(sales) as total_sales from order_details;
```

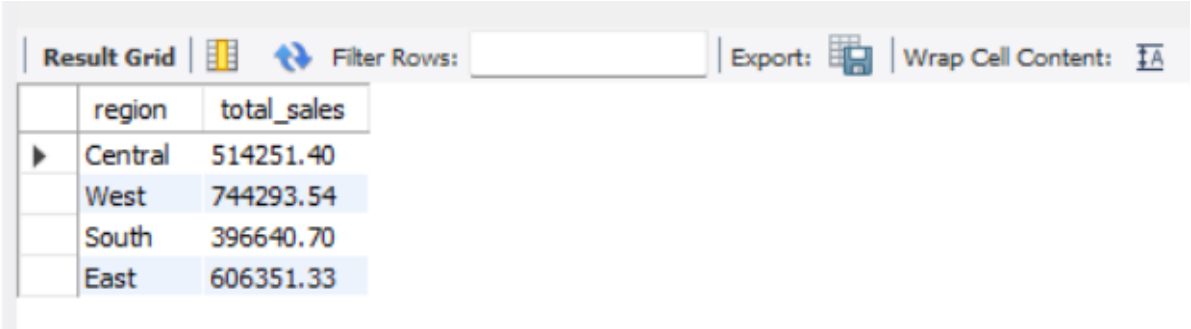


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sales
2261536.97

5) Total sales by region .

```
select c.region,sum(o.sales) as total_sales
from customers c
inner join order_details o
on c.customer_id=o.customer_id
group by region ;
```

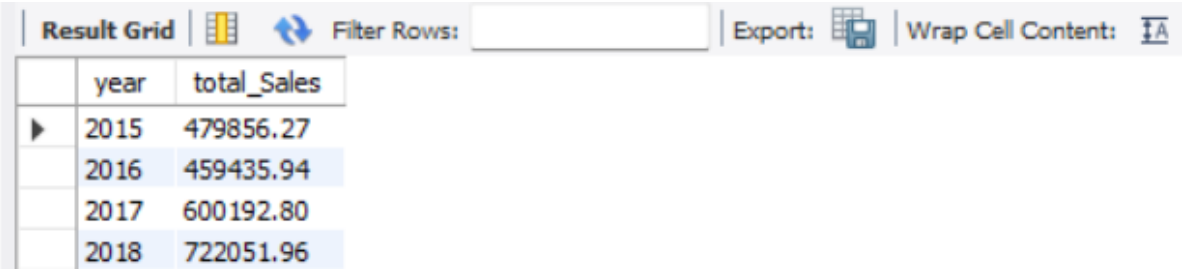


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

region	total_sales
Central	514251.40
West	744293.54
South	396640.70
East	606351.33

6) Total sales by year

```
select year(o.orderdate) as year, sum(od.sales) as total_Sales
from orders o
inner join order_details od
on o.order_id= od.order_id
group by year(orderdate);
```

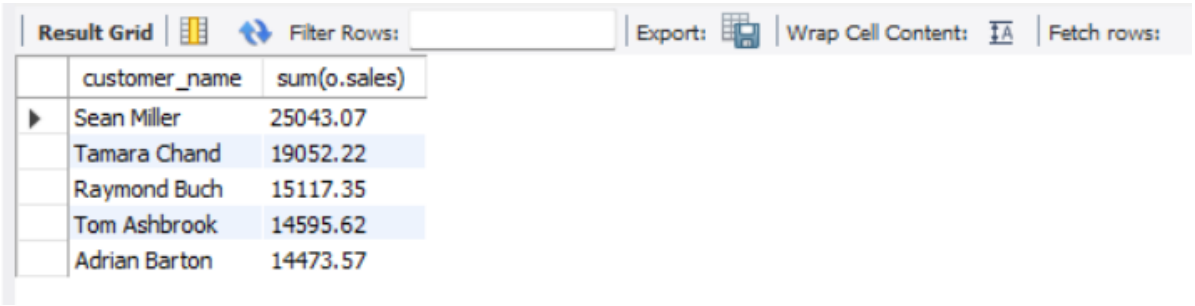


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	total_Sales
2015	479856.27
2016	459435.94
2017	600192.80
2018	722051.96

7) Top 5 customers by sales

```
select c.customer_name, sum(o.sales)
from customers c
inner join order_details o
on c.customer_id = o.customer_id
group by customer_name
order by sum(sales) desc
limit 5;
```

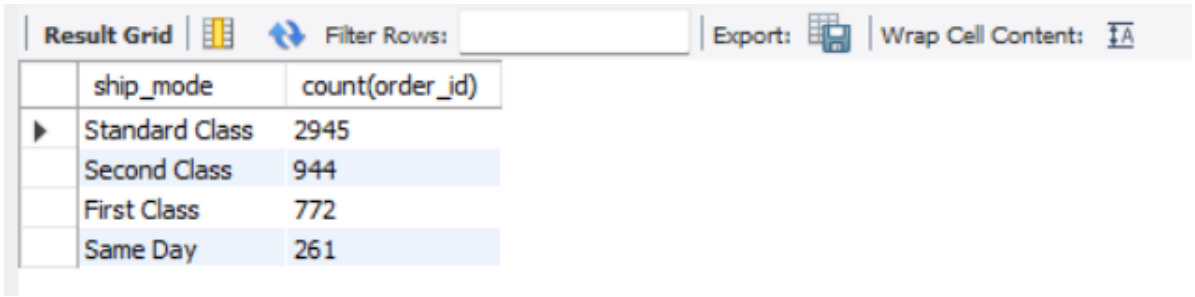


Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_name	sum(o.sales)
Sean Miller	25043.07
Tamara Chand	19052.22
Raymond Buch	15117.35
Tom Ashbrook	14595.62
Adrian Barton	14473.57

8) Orders by ship mode

```
select  ship_mode, count(order_id)
from orders
group by ship_mode;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ship_mode	count(order_id)
Standard Class	2945
Second Class	944
First Class	772
Same Day	261

9)sales by category

```
SELECT p.category, SUM(od.sales) AS total_sales
FROM order_details od
JOIN products p ON od.product_id = p.product_id
GROUP BY p.category
ORDER BY total_sales DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

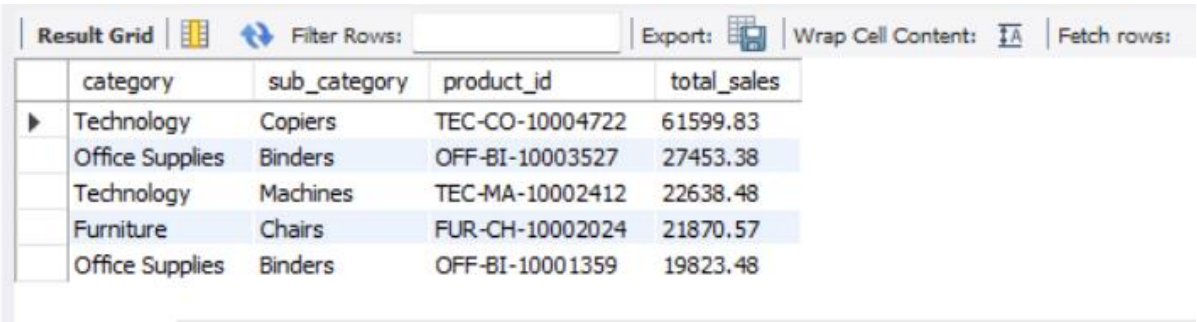
category	total_sales
Technology	827455.94
Furniture	728658.75
Office Supplies	705422.28

Advanced Business Insights

In this section, advanced SQL queries were used to answer complex business questions using techniques such as Common Table Expressions (CTEs), window functions, and subqueries. This enables deeper understanding of trends and performance.

1) Top 5 most profitable products

```
select p.category, p.sub_category, p.product_id ,sum(o.sales) as total_sales
from products p
inner join order_details o
on p.product_id= o.product_id
group by category, sub_category, product_id
order by sum(sales) desc
limit 5 ;
```



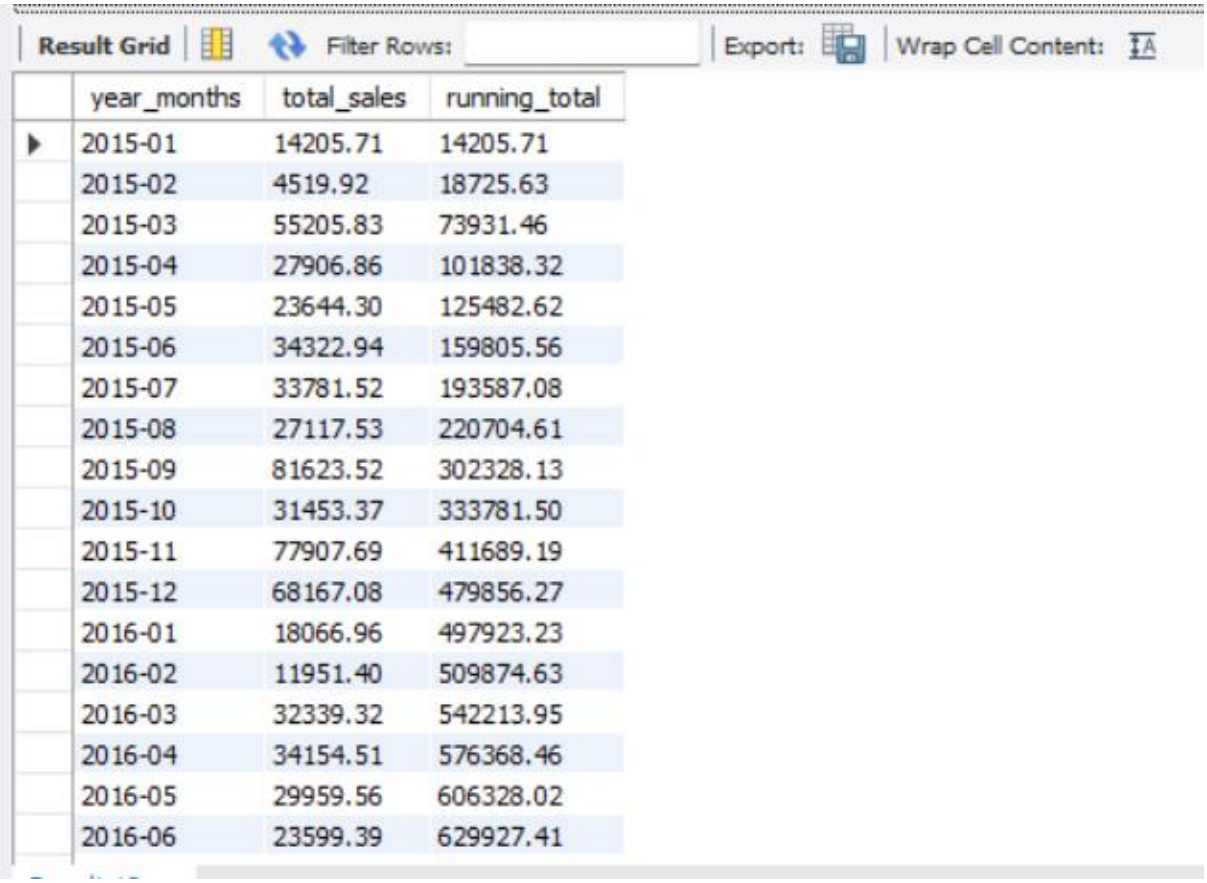
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

category	sub_category	product_id	total_sales
Technology	Copiers	TEC-CO-10004722	61599.83
Office Supplies	Binders	OFF-BI-10003527	27453.38
Technology	Machines	TEC-MA-10002412	22638.48
Furniture	Chairs	FUR-CH-10002024	21870.57
Office Supplies	Binders	OFF-BI-10001359	19823.48

2)Monthly sales and running total using window function.

Shows trends + helps forecast next months.

```
select date_format(o.orderDate, '%Y-%m') as year_months,
sum(od.sales) as total_sales, sum(sum(od.sales)) over (order by date_format(o.orderDate, '%Y-%m')) as running_total
from orders o
inner join order_details od
on o.order_id= od.order_id
group by year_months;
```



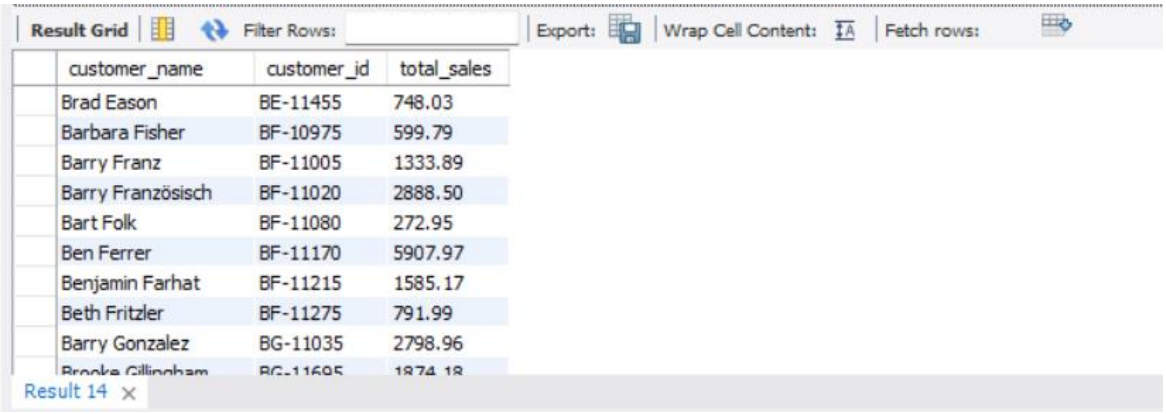
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year_months	total_sales	running_total
2015-01	14205.71	14205.71
2015-02	4519.92	18725.63
2015-03	55205.83	73931.46
2015-04	27906.86	101838.32
2015-05	23644.30	125482.62
2015-06	34322.94	159805.56
2015-07	33781.52	193587.08
2015-08	27117.53	220704.61
2015-09	81623.52	302328.13
2015-10	31453.37	333781.50
2015-11	77907.69	411689.19
2015-12	68167.08	479856.27
2016-01	18066.96	497923.23
2016-02	11951.40	509874.63
2016-03	32339.32	542213.95
2016-04	34154.51	576368.46
2016-05	29959.56	606328.02
2016-06	23599.39	629927.41

3) Customers whose total sales are above average (using subquery)

Finding high value customers for loyalty program.

```
select c.customer_name,c.customer_id, sum(o.sales) as total_sales

from customers c

inner join order_details o

on c.customer_id= o.customer_id

group by customer_name,customer_id

having total_sales>(select avg(sales) from order_details);
```



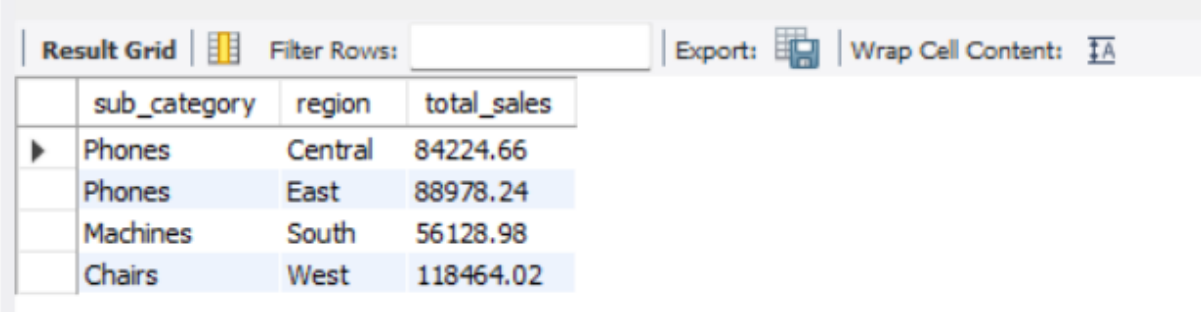
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_name	customer_id	total_sales
Brad Eason	BE-11455	748.03
Barbara Fisher	BF-10975	599.79
Barry Franz	BF-11005	1333.89
Barry Französisch	BF-11020	2888.50
Bart Folk	BF-11080	272.95
Ben Ferrer	BF-11170	5907.97
Benjamin Farhat	BF-11215	1585.17
Beth Fritzler	BF-11275	791.99
Barry Gonzalez	BG-11035	2798.96
Brooke Gillingham	BG-11695	1874.18

Result 14

4)Top sub-category by sales in each region using CTE + RANK()

```
with cte as (
select sub_category,region, sum(sales) as total_sales,
rank() over(partition by region order by sum(sales) desc) as rnk
from products p
inner join order_details o
on p.product_id=o.product_id
inner join customers c
on o.customer_id= c.customer_id
group by region, sub_category
)
select sub_category,region, total_sales
from cte
where rnk=1;
```



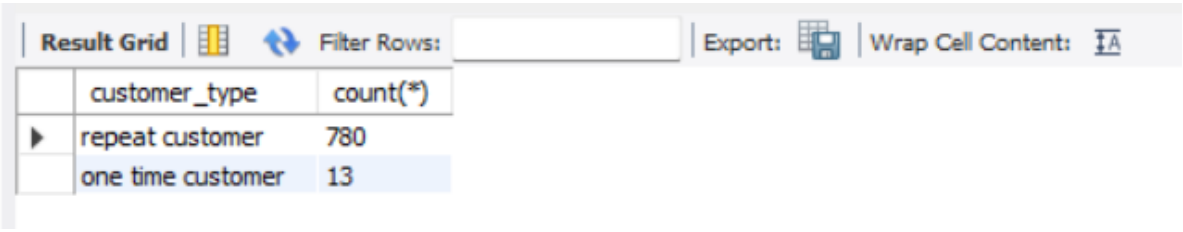
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sub_category	region	total_sales
Phones	Central	84224.66
Phones	East	88978.24
Machines	South	56128.98
Chairs	West	118464.02

5)Number of repeat and one time customer

-- shows customer loyalty and retention

```
select
case
when order_count>1 then 'repeat customer'
else  'one time customer'
end as customer_type ,
count(*) from
(select customer_id,count(distinct order_id) as order_count
 from order_details
group by customer_id )t
group by customer_type ;
```



customer_type	count(*)
repeat customer	780
one time customer	13

6) Year-over-year sales growth using window function

Shows business growth trend over year

```
with yearly_sales as (
select year(o.orderdate) as years, sum(od.sales) as total_sales
from orders o
inner join order_details od
on o.order_id= od.order_id
group by year(orderdate)
)
select years, total_sales,
lag(total_sales) over (order by years) as last_year_sales,
round((total_sales   -   lag(total_sales)   over   (order   by
years))/lag(total_sales)
over (order by years) * 100,2) as yoy_growth
```

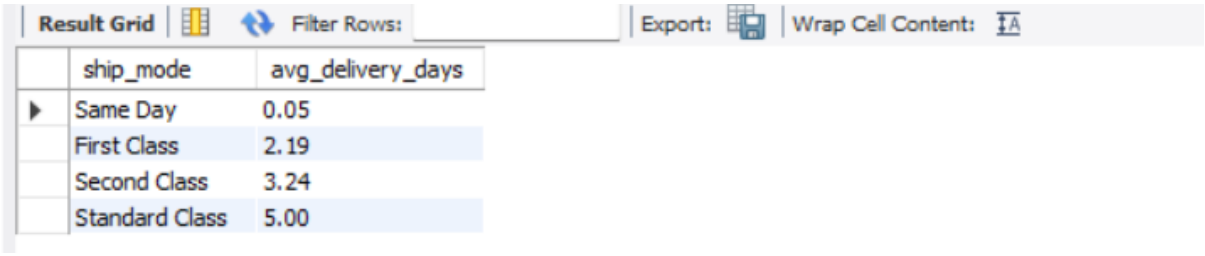
```
from  yearly_sales ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

years	total_sales	last_year_sales	yoy_growth
2015	479856.27	NULL	NULL
2016	459435.94	479856.27	-4.26
2017	600192.80	459435.94	30.64
2018	722051.96	600192.80	20.30

7) average delivery time per ship mode.

```
select ship_mode,
round(avg(datediff(ship_date,orderdate)), 2)
as avg_delivery_days
from orders
group by ship_mode
order by avg_delivery_days ;
```



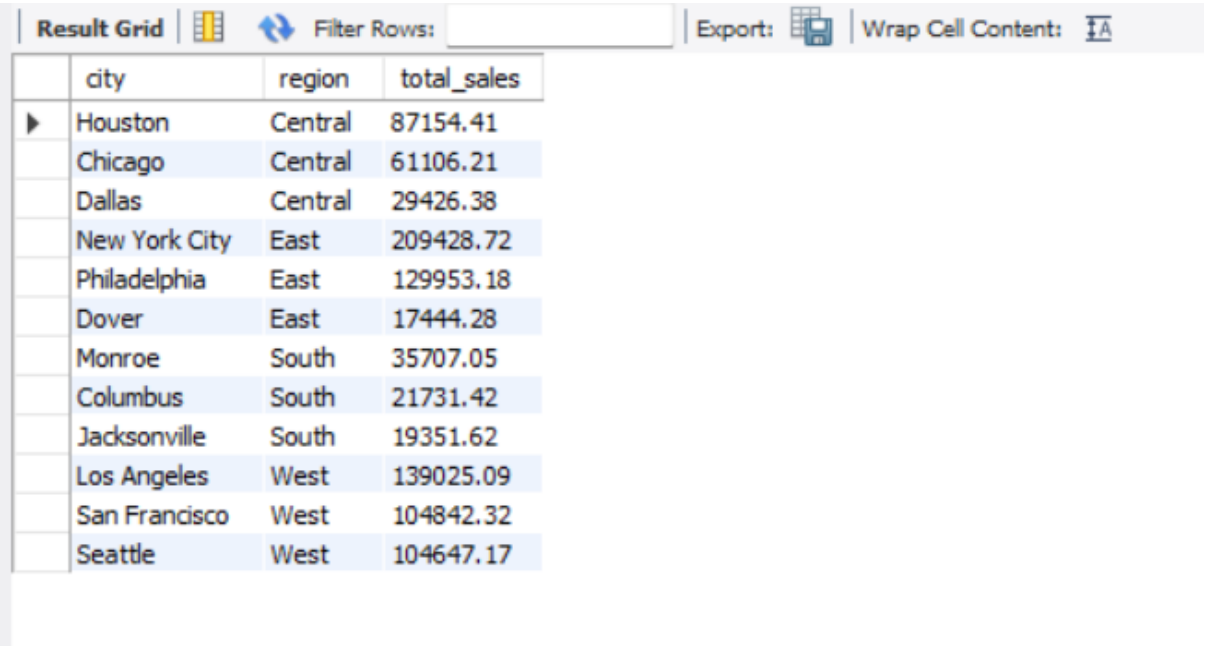
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ship_mode	avg_delivery_days
Same Day	0.05
First Class	2.19
Second Class	3.24
Standard Class	5.00

8) Top 3 city by sales in each region.(using window function and subquery)

```
select city,region, total_sales from (
select c.city,region, sum(o.sales) as total_sales ,
row_number() over (partition by c.region order by sum(o.sales)
desc) as top_city
from customers c
inner join order_details o
on c.customer_id= o.customer_id
group by city,region
) t
```

```
where top_city between 1 and 3 ;
```



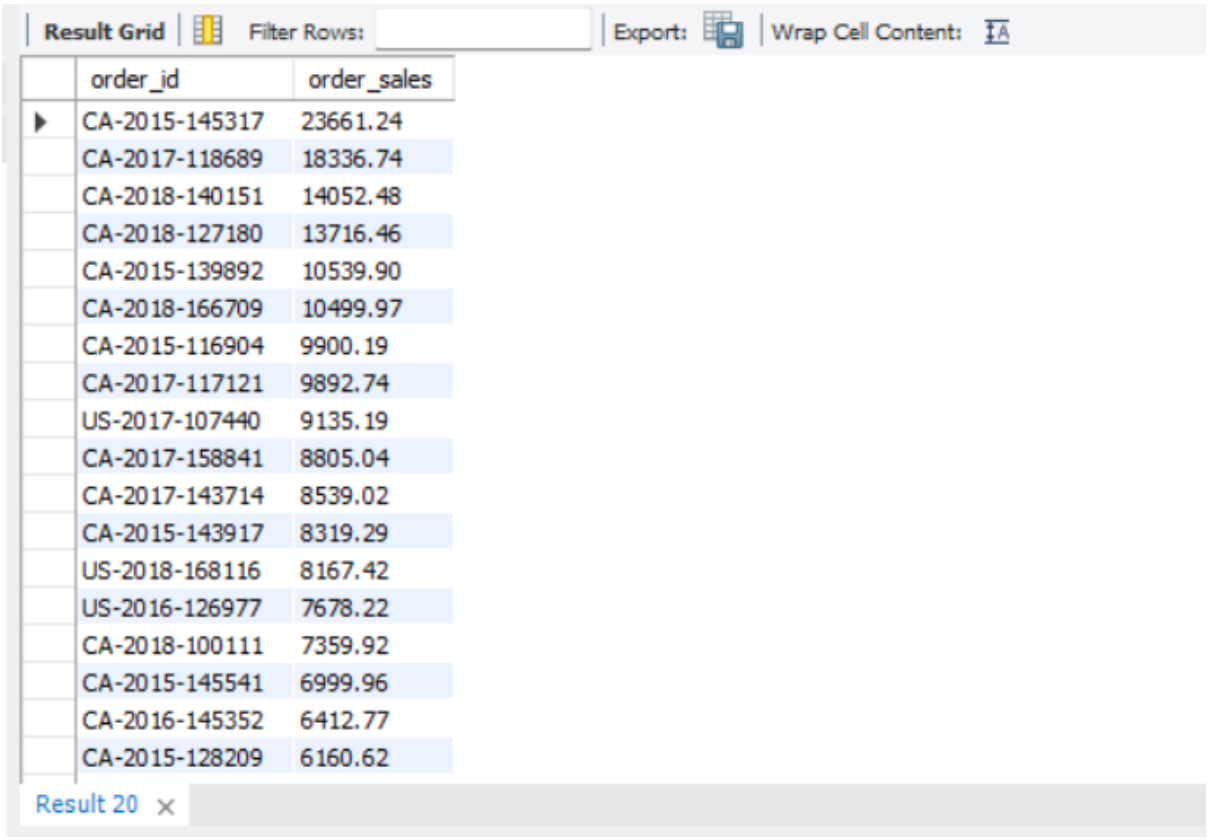
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	region	total_sales
Houston	Central	87154.41
Chicago	Central	61106.21
Dallas	Central	29426.38
New York City	East	209428.72
Philadelphia	East	129953.18
Dover	East	17444.28
Monroe	South	35707.05
Columbus	South	21731.42
Jacksonville	South	19351.62
Los Angeles	West	139025.09
San Francisco	West	104842.32
Seattle	West	104647.17

9)Find top 1% of orders by revenue using window function + CTE

```
WITH order_totals AS (
  SELECT order_id,SUM(sales) AS order_sales
  FROM order_details
  GROUP BY order_id
),
ranked_orders AS (
  SELECT
    order_id,
    order_sales,
    NTILE(100) OVER (ORDER BY order_sales DESC) AS percentile_rank
  FROM
    order_totals
)
SELECT
  order_id,
  order_sales
```

```
FROM
  ranked_orders
WHERE
  percentile_rank = 1
ORDER BY
  order_sales DESC;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_sales
CA-2015-145317	23661.24
CA-2017-118689	18336.74
CA-2018-140151	14052.48
CA-2018-127180	13716.46
CA-2015-139892	10539.90
CA-2018-166709	10499.97
CA-2015-116904	9900.19
CA-2017-117121	9892.74
US-2017-107440	9135.19
CA-2017-158841	8805.04
CA-2017-143714	8539.02
CA-2015-143917	8319.29
US-2018-168116	8167.42
US-2016-126977	7678.22
CA-2018-100111	7359.92
CA-2015-145541	6999.96
CA-2016-145352	6412.77
CA-2015-128209	6160.62

Result 20

Recommendations (Key Insights)

- **Focus on Customer Retention:** High number of repeat customers shows loyalty — maintain this with loyalty programs and personalized offers.
- **Monitor Sales Trends:** Strong YoY growth overall, but address any factors that caused past declines.
- **Optimize Shipping:** Promote faster shipping modes for better customer satisfaction; balance speed with cost.
- **Target High-Performing Regions:** Prioritize top cities and regions like NYC, Philadelphia, and LA for marketing and inventory focus.
- **Leverage High-Value Orders:** Identify products and segments driving large orders to boost upselling and cross-selling.

Conclusion

This project demonstrates the complete process of performing data analysis using MySQL, from data cleaning and database design to data loading, transformation, EDA, and advanced business insights.
By using CTEs, window functions, and subqueries, actionable insights were uncovered that can help the business make informed, data-driven decisions.
Overall, this project highlights the importance of good database design, clean data, and powerful SQL techniques in solving real-world business problems.