SuperMarketAnalysis

SQL Queries

1. **To create and use a database:**

   SQL Query:

   ```
   CREATE DATABASE SalesAnalysis;
   USE SalesAnalysis;
   ```

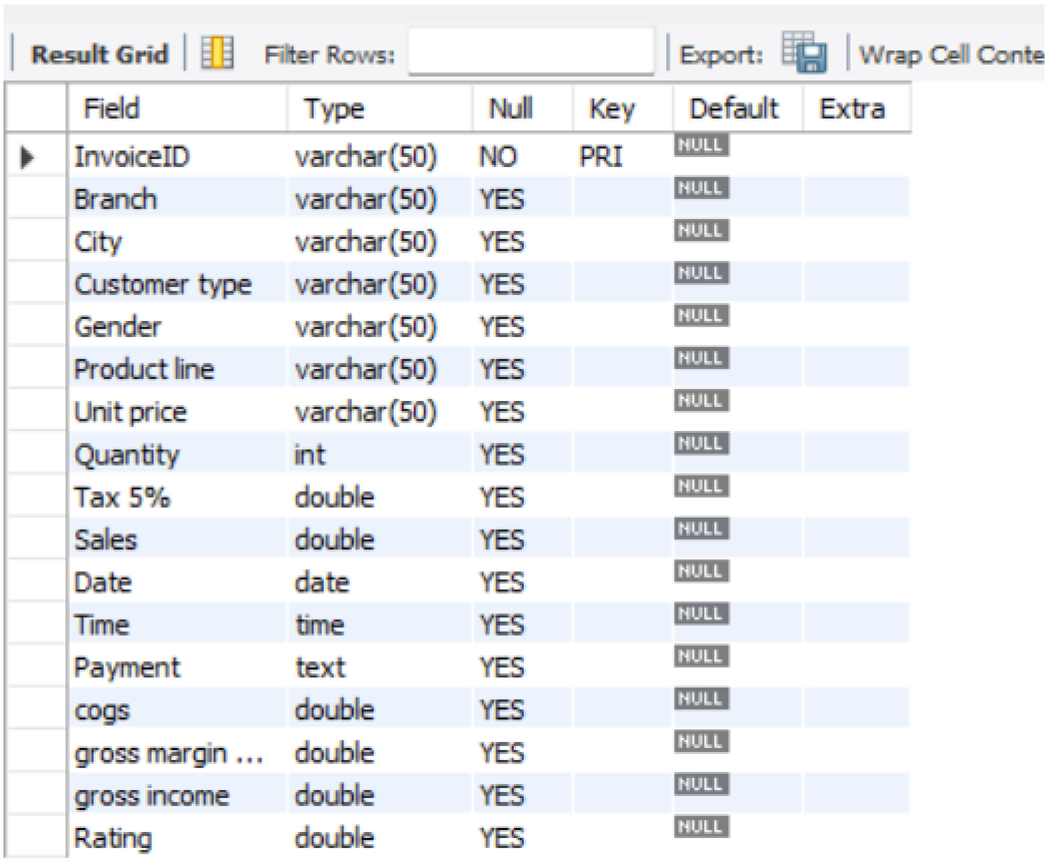
2. **Describe the schema:**

   Describe command is used to display the structure of a table. It provides metadata about the table.

   SQL Query:

   ```
   DESCRIBE salesanalysis.supermarketanalysis;
   ```

   Output:



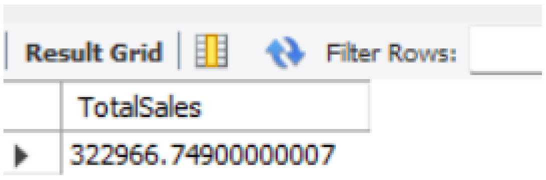
Field	Type	Null	Key	Default	Extra
InvoiceID	varchar(50)	NO	PRI	NULL	
Branch	varchar(50)	YES		NULL	
City	varchar(50)	YES		NULL	
Customer type	varchar(50)	YES		NULL	
Gender	varchar(50)	YES		NULL	
Product line	varchar(50)	YES		NULL	
Unit price	varchar(50)	YES		NULL	
Quantity	int	YES		NULL	
Tax 5%	double	YES		NULL	
Sales	double	YES		NULL	
Date	date	YES		NULL	
Time	time	YES		NULL	
Payment	text	YES		NULL	
cogs	double	YES		NULL	
gross margin ...	double	YES		NULL	
gross income	double	YES		NULL	
Rating	double	YES		NULL	

3. Total Sales:

SQL Query:

```
SELECT SUM(Sales) TotalSales FROM supermarketanalysis;
```

Output:



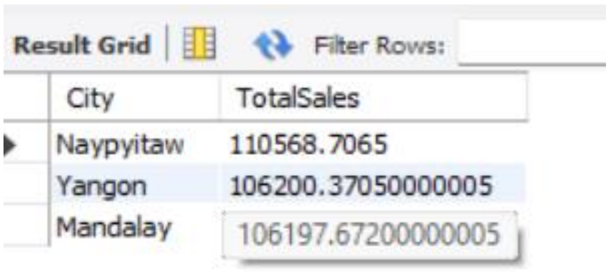
```
TotalSales
322966.74900000007
```

4. Total Sales by City:

SQL Query:

```
SELECT City, SUM(Sales) TotalSales
 FROM  supermarketanalysis
GROUP BY City
 ORDER BY TotalSales DESC;
```

Output:



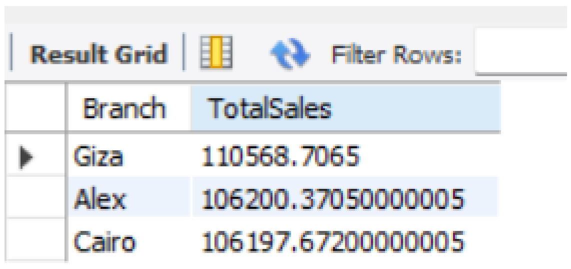
```
City        TotalSales
Naypyitaw   110568.7065
Yangon      106200.37050000005
Mandalay    106197.67200000005
```

5. Calculate total sales for each branch:

SQL Query:

```
 SELECT Branch, SUM(Sales) TotalSales
 FROM  supermarketanalysis
GROUP BY Branch
ORDER BY TotalSales DESC;
```

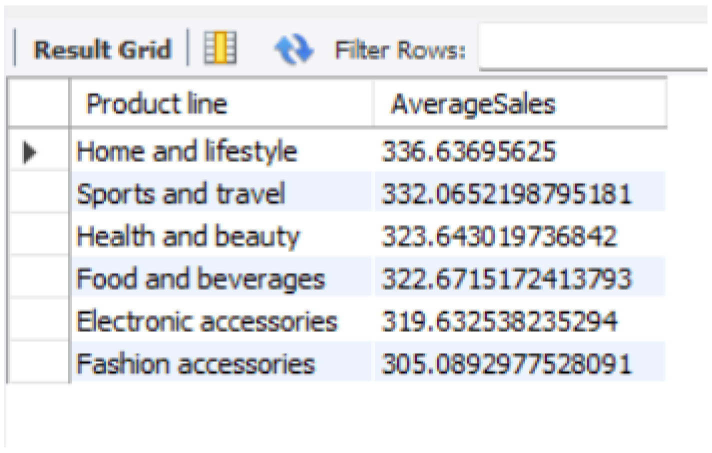
Output:



Result Grid | Filter Rows:

Branch	TotalSales
Giza	110568.7065
Alex	106200.37050000005
Cairo	106197.67200000005

6. **Get the average customer rating for each product line.**
   SQL Query:
   ```
   SELECT `Product line`, avg(sales) AverageSales
   FROM supermarketanalysis
   GROUP BY `Product line`
   ORDER BY AverageSales DESC;
   ```
   Output:

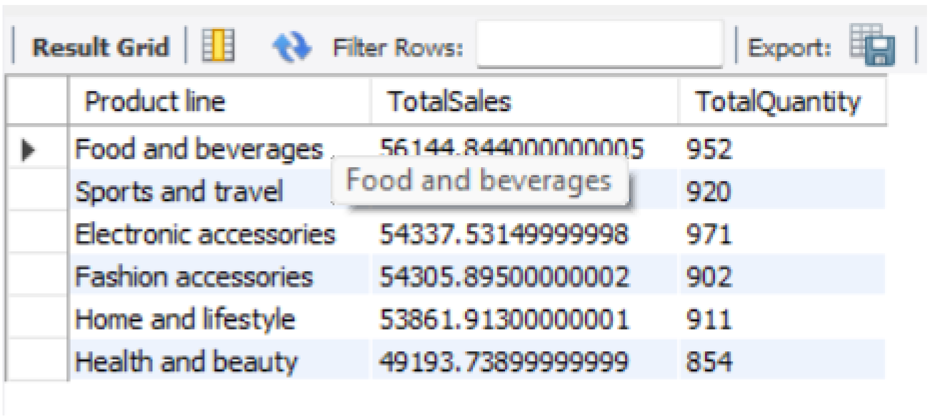


Result Grid | Filter Rows:

Product line	AverageSales
Home and lifestyle	336.63695625
Sports and travel	332.0652198795181
Health and beauty	323.643019736842
Food and beverages	322.6715172413793
Electronic accessories	319.632538235294
Fashion accessories	305.0892977528091

7. **Calculate the total sales and total quantity sold for each product line.**
   SQL Query:
   ```
   SELECT `Product line`, SUM(sales) TotalSales, SUM(quantity) TotalQuantity
   FROM supermarketanalysis
   GROUP BY `Product line`
   ORDER BY TotalSales DESC;
   ```

Output:



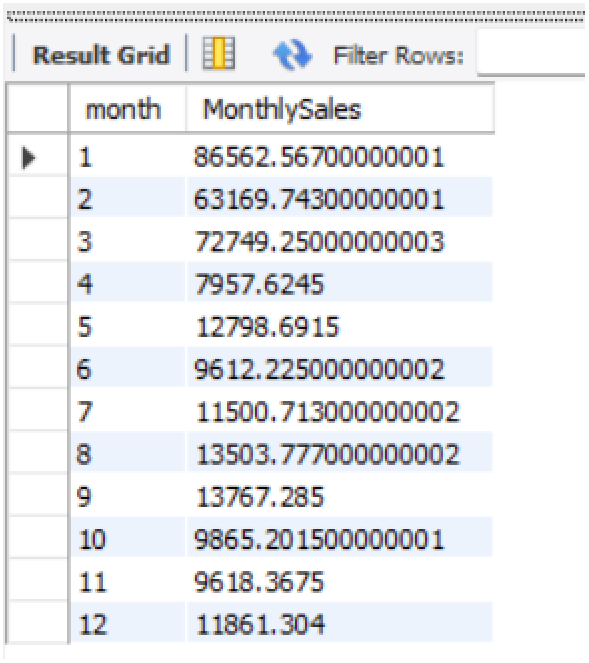
	Product line	TotalSales	TotalQuantity
▶	Food and beverages	56144.844000000005	952
	Sports and travel	[illegible]	920
	Electronic accessories	54337.53149999998	971
	Fashion accessories	54305.89500000002	902
	Home and lifestyle	53861.91300000001	911
	Health and beauty	49193.73899999999	854

8. **Show sales trend over month.**

SQL Query:

```
 SELECT MONTH(Date) month, SUM(Sales) MonthlySales
 FROM supermarketanalysis
GROUP BY MONTH
ORDER BY month;
```

Output:



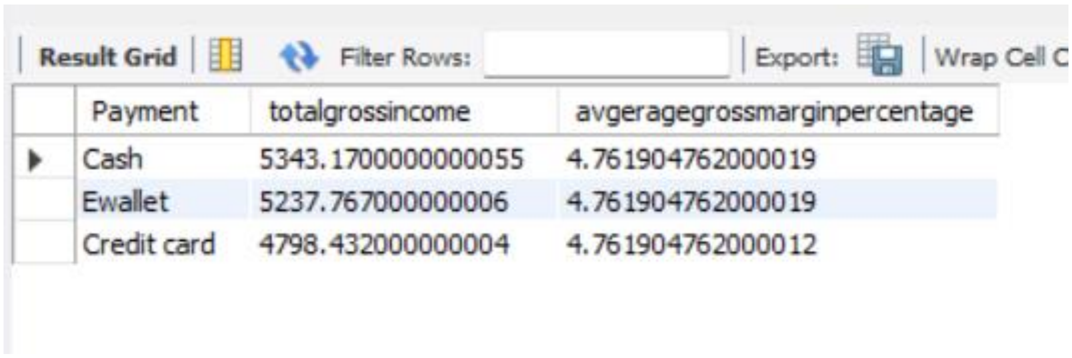
	month	MonthlySales
▶	1	86562.56700000001
	2	63169.74300000001
	3	72749.25000000003
	4	7957.6245
	5	12798.6915
	6	9612.225000000002
	7	11500.713000000002
	8	13503.777000000002
	9	13767.285
	10	9865.201500000001
	11	9618.3675
	12	11861.304

9. **Calculate gross income and gross margin percentage for each payment type.**

SQL Query:

```
SELECT `Payment`,
   SUM(`gross income`) AS totalgrossincome,
   AVG(`gross margin percentage`) AS avgeragegrossmarginpercentage
FROM supermarketanalysis
GROUP BY `Payment`
ORDER BY totalgrossincome DESC;
```

Output:



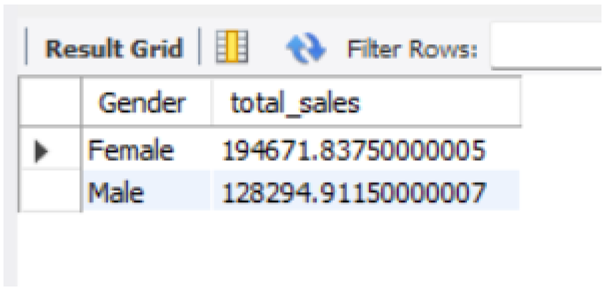
Payment	totalgrossincome	avgeragegrossmarginpercentage
Cash	5343.1700000000055	4.761904762000019
Ewallet	5237.767000000006	4.761904762000019
Credit card	4798.432000000004	4.761904762000012

10. **Analyze total sales by gender.**

SQL Query:

```
 SELECT Gender, SUM(`Sales`) AS total_sales
 FROM supermarketanalysis
 GROUP BY Gender
ORDER BY total_sales DESC;
```

Output:



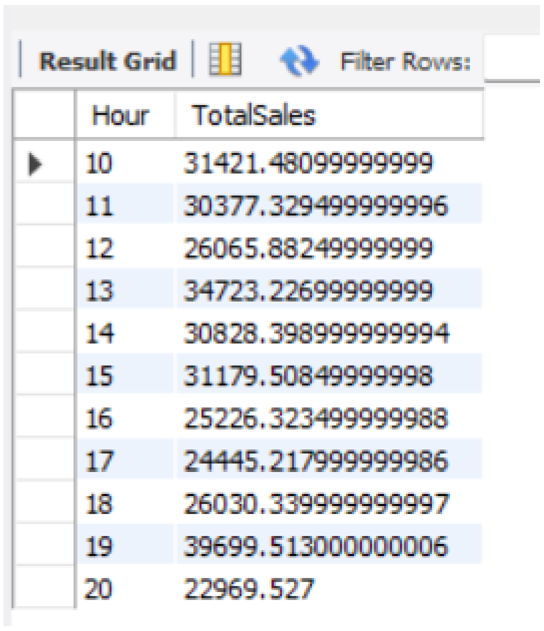
Gender	total_sales
Female	194671.83750000005
Male	128294.91150000007

11. Analyze sales by time of day.

SQL Query:

```
SELECT HOUR(time) Hour, SUM(sales) TotalSales
FROM supermarketanalysis
GROUP BY Hour
ORDER BY Hour;
```

Output:


Result Grid | Filter Rows:

Hour	TotalSales
10	31421.48099999999
11	30377.329499999996
12	26065.88249999999
13	34723.22699999999
14	30828.398999999994
15	31179.50849999998
16	25226.323499999988
17	24445.217999999986
18	26030.339999999997
19	39699.513000000006
20	22969.527

12. Analyze gross income based on customer type.

SQL Query:

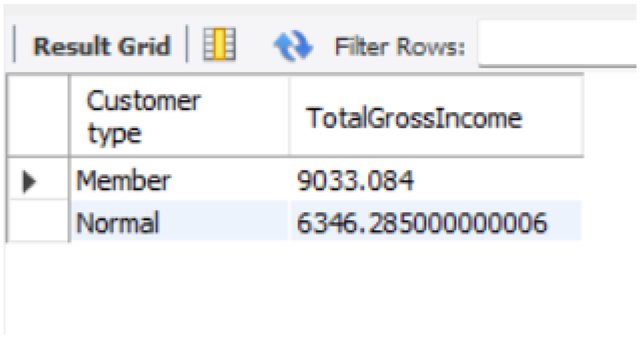
SELECT `Customer type`, SUM(`gross income`) TotalGrossIncome

FROM supermarketanalysis

GROUP BY `Customer type`

ORDER BY TotalGrossIncome DESC;

Output:


Result Grid | Filter Rows:

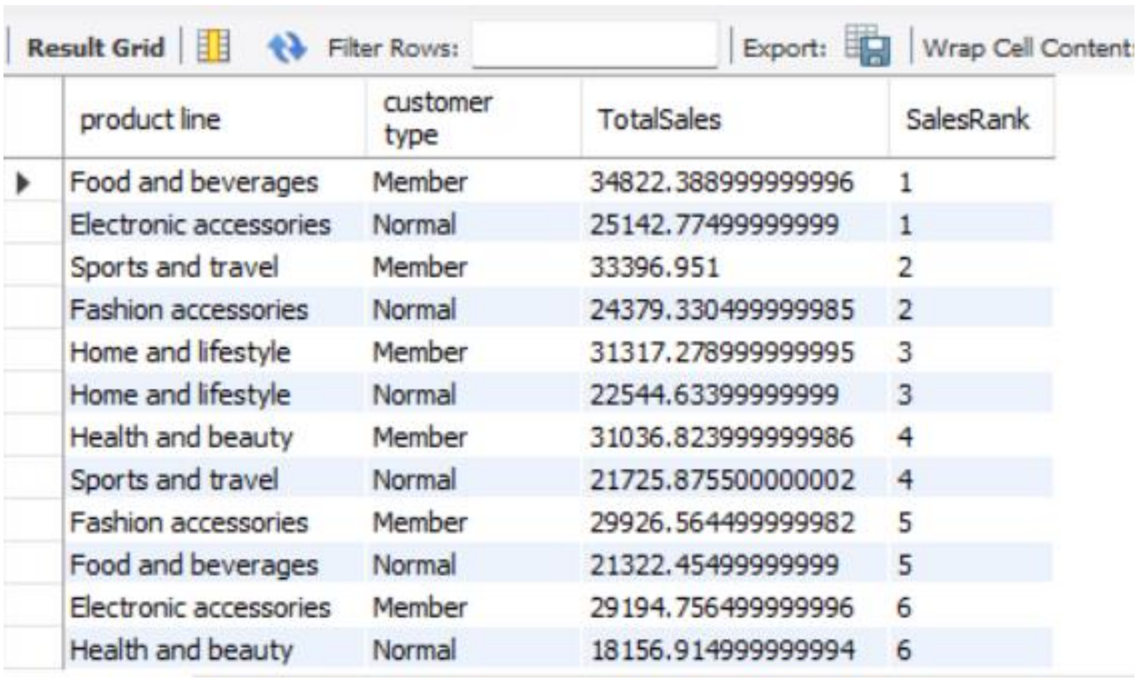
Customer type	TotalGrossIncome
Member	9033.084
Normal	6346.285000000006

13. The top 3 total sales of each product line within each customer type.

Sql Query:

SELECT `product line`, `customer type`, sum(sales) TotalSales,
RANK() OVER(PARTITION BY `customer type` ORDER BY sum(sales) DESC) SalesRank
FROM supermarketanalysis
GROUP BY `product line`, `customer type`
ORDER BY SalesRank;

Output:


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

product line	customer type	TotalSales	SalesRank
Food and beverages	Member	34822.388999999996	1
Electronic accessories	Normal	25142.77499999999	1
Sports and travel	Member	33396.951	2
Fashion accessories	Normal	24379.330499999985	2
Home and lifestyle	Member	31317.278999999995	3
Home and lifestyle	Normal	22544.63399999999	3
Health and beauty	Member	31036.823999999986	4
Sports and travel	Normal	21725.875500000002	4
Fashion accessories	Member	29926.564499999982	5
Food and beverages	Normal	21322.45499999999	5
Electronic accessories	Member	29194.756499999996	6
Health and beauty	Normal	18156.914999999994	6

14. Calculates monthly sales and the percentage change from the previous month.

Sql Query:

WITH MonthlySales as (

SELECT DATE_FORMAT(Date,'%Y-%m') Month, sum(sales) TotalSales

FROM supermarketanalysis

GROUP BY Month),

PreviousMonthSales as(

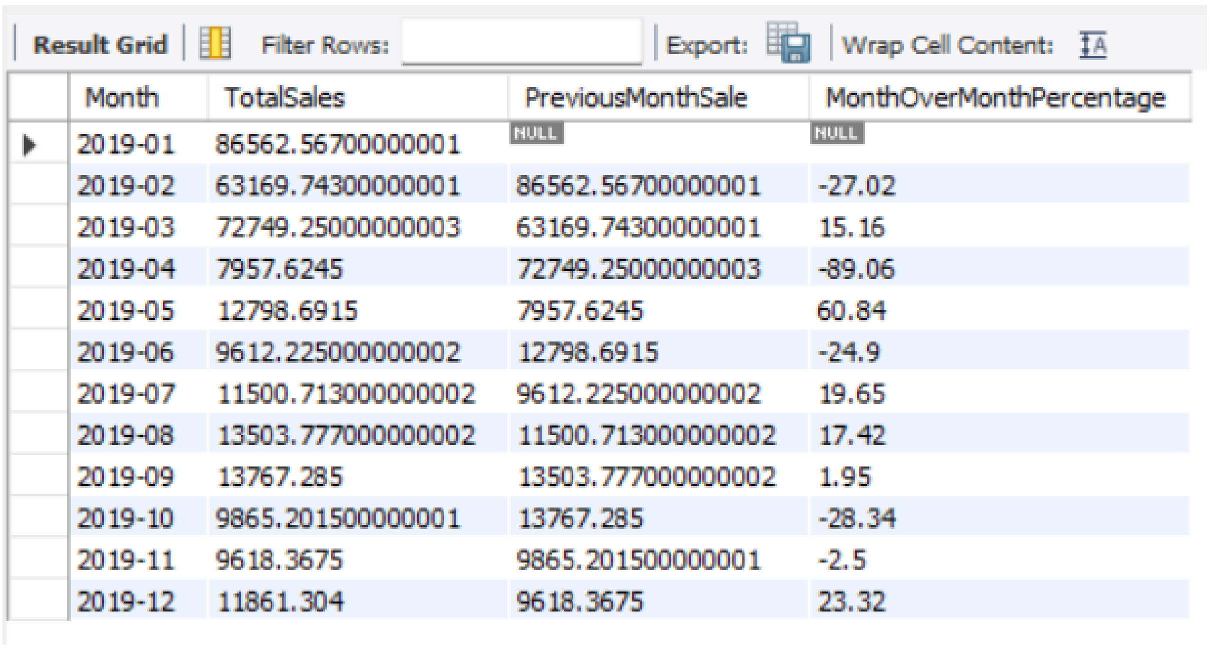
SELECT Month, TotalSales, LAG(TotalSales) OVER(ORDER BY Month) PreviousMonthSale

FROM MonthlySales)

SELECT Month, TotalSales,PreviousMonthSale, ROUND(((TotalSales-PreviousMonthSale)/PreviousMonthSale)*100,2) AS MonthOverMonthPercentage

FROM PreviousMonthSales;

Output:



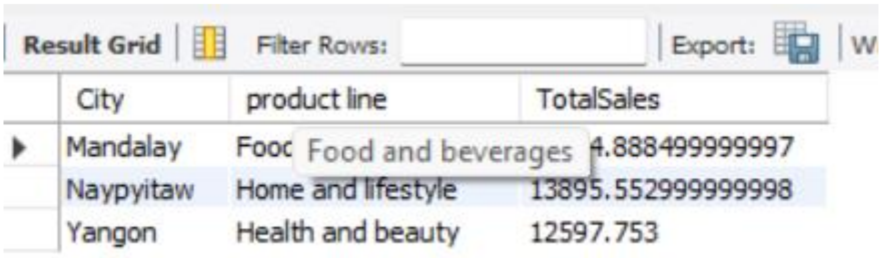
	Month	TotalSales	PreviousMonthSale	MonthOverMonthPercentage
▶	2019-01	86562.56700000001	NULL	NULL
	2019-02	63169.74300000001	86562.56700000001	-27.02
	2019-03	72749.25000000003	63169.74300000001	15.16
	2019-04	7957.6245	72749.25000000003	-89.06
	2019-05	12798.6915	7957.6245	60.84
	2019-06	9612.225000000002	12798.6915	-24.9
	2019-07	11500.713000000002	9612.225000000002	19.65
	2019-08	13503.777000000002	11500.713000000002	17.42
	2019-09	13767.285	13503.777000000002	1.95
	2019-10	9865.201500000001	13767.285	-28.34
	2019-11	9618.3675	9865.201500000001	-2.5
	2019-12	11861.304	9618.3675	23.32

15. **Find the top-selling product line in each city.**

SQL Query:

WITH CitySales AS(
SELECT City, `product line`, sum(sales) TotalSales
FROM supermarketanalysis
GROUP BY City, `product line`
ORDER BY City),
CitySalesRank AS(
SELECT City, `product line`, TotalSales, RANK() OVER(PARTITION BY City ORDER BY TotalSales) CityRank
FROM CitySales
)
SELECT City, `product line`, TotalSales FROM CitySalesRank
WHERE CityRank=1;

Output:



	City	product line	TotalSales
▶	Mandalay	Food and beverages	4.888499999997
	Naypyitaw	Home and lifestyle	13895.552999999998
	Yangon	Health and beauty	12597.753

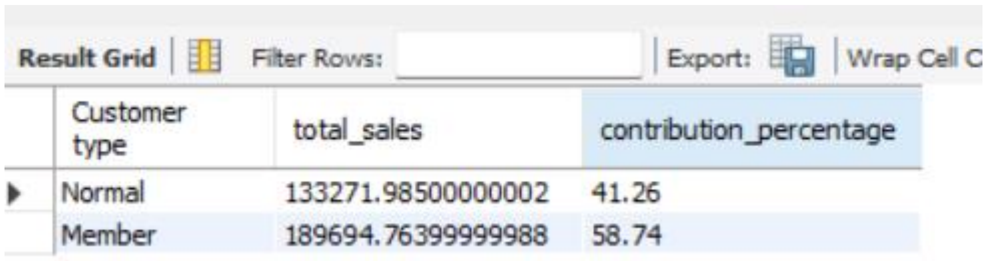
16. **Analyze the contribution of each customer type to the total sales, both in absolute values and percentages.**

SQL Query:

WITH OverallSales as(

```
SELECT SUM(sales) TotalSales
FROM supermarketanalysis
),
CustomerContribution as(
SELECT `Customer type`, SUM(`Sales`) AS total_sales, ROUND((SUM(`Sales`)/(SELECT
TotalSales FROM OverallSales))*100,2) AS contribution_percentage
FROM supermarketanalysis
GROUP BY `Customer type`)
SELECT * FROM CustomerContribution;
```

Output:



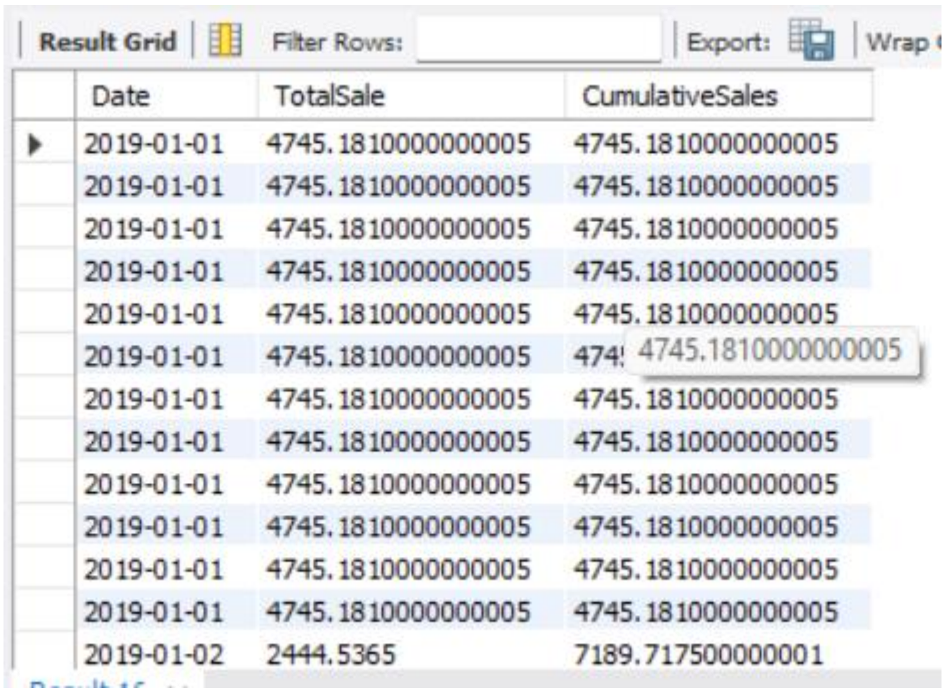
Customer type	total_sales	contribution_percentage
Normal	133271.98500000002	41.26
Member	189694.76399999988	58.74

17.calculates the cumulative sales over time.

SQL Query:

```
WITH DailySales as(
SELECT Date, SUM(Sales) AS TotalSale
FROM supermarketanalysis
GROUP BY Date
ORDER BY Date),
CumulativeSales as(
SELECT  Date, SUM(Sales) OVER( ORDER BY Date) CumulativeSales
FROM supermarketanalysis)
SELECT ds.Date, ds.TotalSale, cs.CumulativeSales FROM CumulativeSales cs
INNER JOIN DailySales ds ON cs.Date=ds.Date;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap

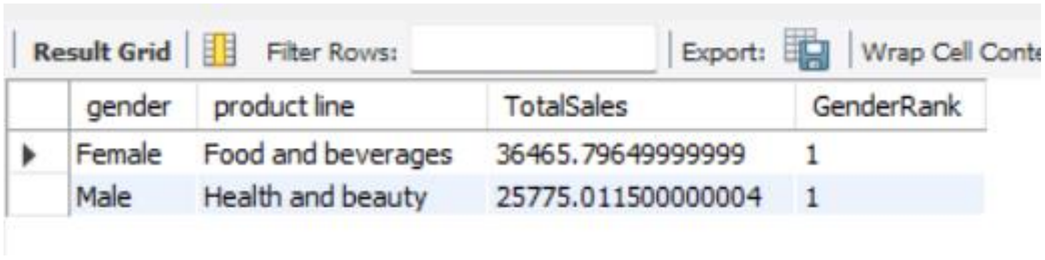
Date	TotalSale	CumulativeSales
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-01	4745.1810000000005	4745.1810000000005
2019-01-02	2444.5365	7189.717500000001

18.Product line with high sales regarding Gender.

SQL Query:

```
WITH GenderProduct AS
SELECT gender, `product line`, SUM(Sales) TotalSales
FROM supermarketanalysis
GROUP BY gender, `product line`),
GenderProductRank AS(
SELECT *, RANK() OVER(PARTITION BY gender order by TotalSales DESC) GenderRank
FROM GenderProduct)
SELECT * FROM GenderProductRank
WHERE GenderRank=1;
```

Output:



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

gender	product line	TotalSales	GenderRank
Female	Food and beverages	36465.79649999999	1
Male	Health and beauty	25775.011500000004	1