Northwind Sales Analysis

1- Customer Segmentation:

- RFM Analysis:

Query Explanation

We need to calculate days since last order, number of orders and total revenue for each customer, to get all these information I used join between two tables “Orders” and “Order Details”. I used MAX function to return the date of last order and used JULIANDAY function that returns an integer value that represents a number of days then subtract it from current date, to calculate total orders I used COUNT function with Distinct to count unique orders, to calculate total amount spent I used SUM function and Cast function to transform the result from float to integer. Group by customerid to calculate all result for each customer.

I created another view and select from previous view to create an conditional columns by splitting the customers into three categories using CASE expression.

I wrote another query to calculate customers number in each segment by using “Customer_Segments “view.

Queries:

```
create VIEW RFM_Analysis as
select o.customerid
,max(date(o.OrderDate)) LastOrder ,
, julianday(date('now'))- julianday(max(date(o.OrderDate))) Days_Since_lastOrders
, count(DISTINCT o.OrderID)    TotalOrders
, cast((sum(unitprice*quantity*(1-discount))) as int) TotalAmountSpent
from orders o left join 'Order Details' od
on o.OrderID=od.orderid
group by 1
order by 1
```

Customer segmentation

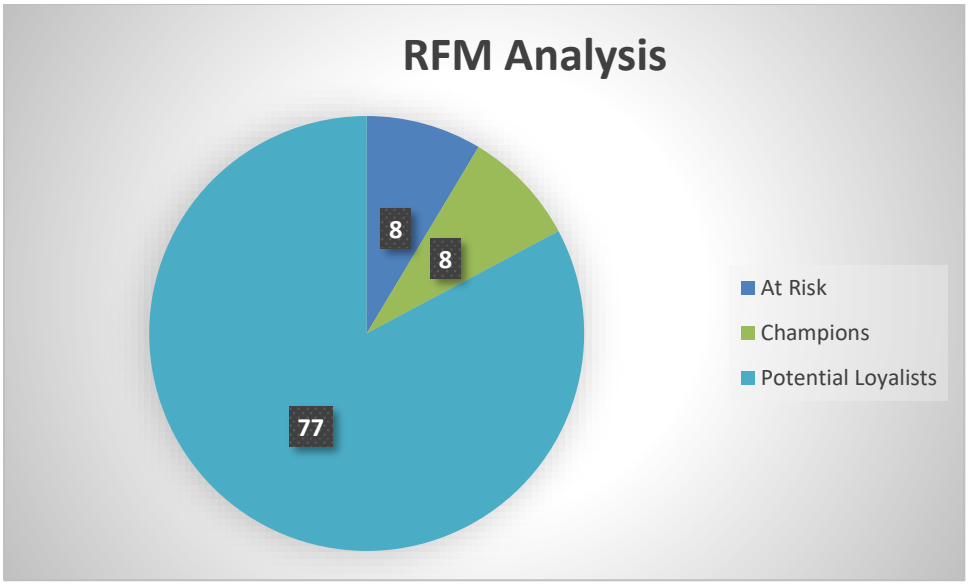
```
create view Customer_Segments as
select customerid
 , case
   when Days_Since_lastOrders < 450 and TotalOrders> 190
   and TotalAmountSpent > 5000000 then 'Champions'
   when TotalOrders BETWEEN 160 and 190  or TotalAmountSpent
   BETWEEN  4500000 and 5000000   then 'Potential Loyalists'
  Else  'At Risk'
 End  Customer_Segments
From  RFM_Analysis
order by 2
```

CustomerID	Customer_Segments
AROUT	At Risk
FURIB	At Risk
LETSS	At Risk
OCEAN	At Risk
RANCH	At Risk
RICAR	At Risk
TRAIH	At Risk
VINET	At Risk
ANATR	Champions
BSBEV	Champions
FOLIG	Champions
GOURL	Champions
HUNGC	Champions
LILAS	Champions
PRINI	Champions

CustomerID	Customer_Segments
PRINI	Champions
TORTU	Champions
ALFKI	Potential Loyalists
ANTON	Potential Loyalists
BERGS	Potential Loyalists
BLAUS	Potential Loyalists
BLONP	Potential Loyalists
BOLID	Potential Loyalists
BONAP	Potential Loyalists
BOTTM	Potential Loyalists
CACTU	Potential Loyalists
CENTC	Potential Loyalists
CHOPS	Potential Loyalists
COMMI	Potential Loyalists
CONSH	Potential Loyalists

Show customers number in each segment

```
select Customer_Segments
,count( Customer_Segments) CustomersNumber
from Customer_categories
group by Customer_Segments
```



- Insights

After calculating recency, Frequency and Monetary Value per customer

and splitting the customer into 3 segments :

- **Champions** are customers that request last order in less than 450 days and total orders more than 190 and Revenue more than 5000000

- **Potential** loyalists are customers that request orders between 160 and 190 order or spent money between 4500000 and 5000000

- **At Risk** are all other customers

Champions are 8

Potential loyalists are 77

At Risk are 8

- Order Value:

Query Explanation

I used same approach in writing queries like in the RFM analysis

Query:

High-Value, Medium-Value, Low-Value customers based on their

avarage order revenue value

```
create view avarage_order_revenue as
select o.customerid
,count(distinct o.OrderID) TotalOrders
,cast((sum(unitprice*quantity*(1-discount))) as int) TotalAmountSpent
,cast((sum(unitprice*quantity*(1-discount))) as int) /count(distinct o.OrderID)
avarage_order_revenue
from orders o inner join  'Order Details' od
on o.OrderID=od.orderid
group by 1
order by  avarage_order_revenue
```

customers segmentation

```
create view CustomerCategories as
select  customerid
, avarage_order_revenue
, case
when avarage_order_revenue < 26000 then ‘Low-Value’
when avarage_order_revenue > 29000 then ‘High-Value’
Else ' Medium-Value
End   CustomerCategory
```

from avarage_order_revenue

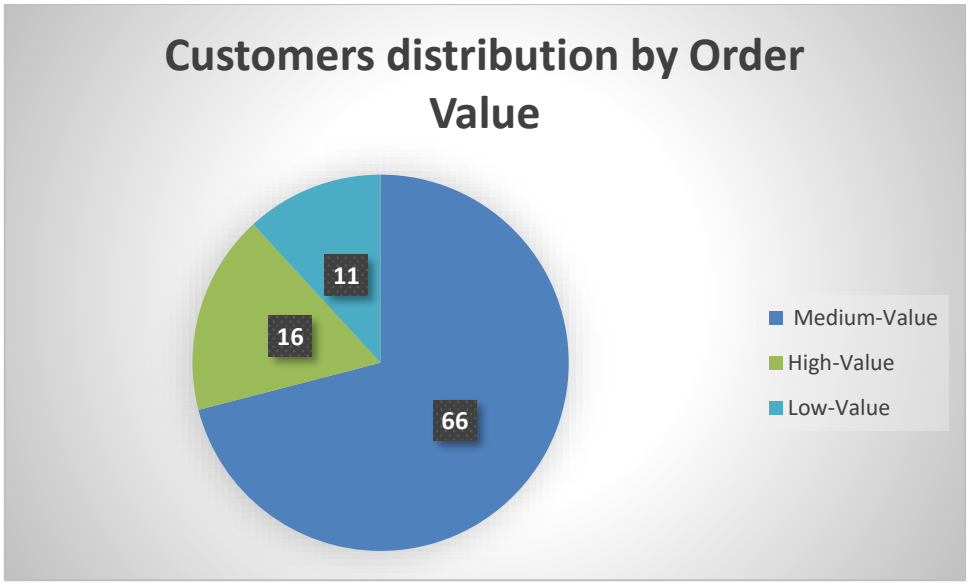
order by 3 desc

CustomerID	avarage_order_revenue	CustomerCategory
SAVEA	23789	Low-Value
ERNSH	24288	Low-Value
ALFKI	24328	Low-Value
QUEDE	25196	Low-Value
LEHMS	25205	Low-Value
LAMAI	25274	Low-Value
LINOD	25407	Low-Value
WHITC	25479	Low-Value
FOLKO	25521	Low-Value
LONEP	25896	Low-Value
NORTS	25933	Low-Value
FAMIA	29141	High-Value
BSBEV	29305	High-Value
THEBI	29456	High-Value
SANTG	29504	High-Value

CustomerID	avarage_order_revenue	CustomerCategory
THECR	29748	High-Value
PARIS	29850	High-Value
PERIC	30010	High-Value
MORGK	30051	High-Value
Val2	30316	High-Value
PICCO	30318	High-Value
FISSA	30448	High-Value
HUNGO	26001	Medium-Value
KOENE	26108	Medium-Value
OCEAN	26357	Medium-Value
REGGC	26395	Medium-Value
ISLAT	26408	Medium-Value
FURIB	26442	Medium-Value
ANTON	26452	Medium-Value
CENTC	26472	Medium-Value

Show number of customers per category

```
select CustomerCategory
count(CustomerCategory) CustomersNumber ,
from CustomerCategories
group by 1
```



- Insights

After calculating average order revenue value per customer by dividing total revenue over total orders

splitting the customer into 3 segments :

- **High-Value** are customers that have average order value more than 29000

- **Low-Value** are customers that have average order value less than 26000

- **Medium -Value** are all other customers

High-Value are 16

Medium-Value are 72

Low-Value are 11

Recommendations

-Providing more data or seek the feedback of “At Risk “ to understand what keeps them loyal , what can be improved and why they have not requested orders for along time.

- Suggest premium products or complementary items on ‘Low-Value’ to increase their average order value.

- Offer incentives for ‘Potential loyalist’, such as discounts on their favorite products or free shipping for repeat purchases, and analyze their purchase history to send personalized product recommendations and limited-time offers.

2- Product Analysis:

- The top 10 revenue generator products

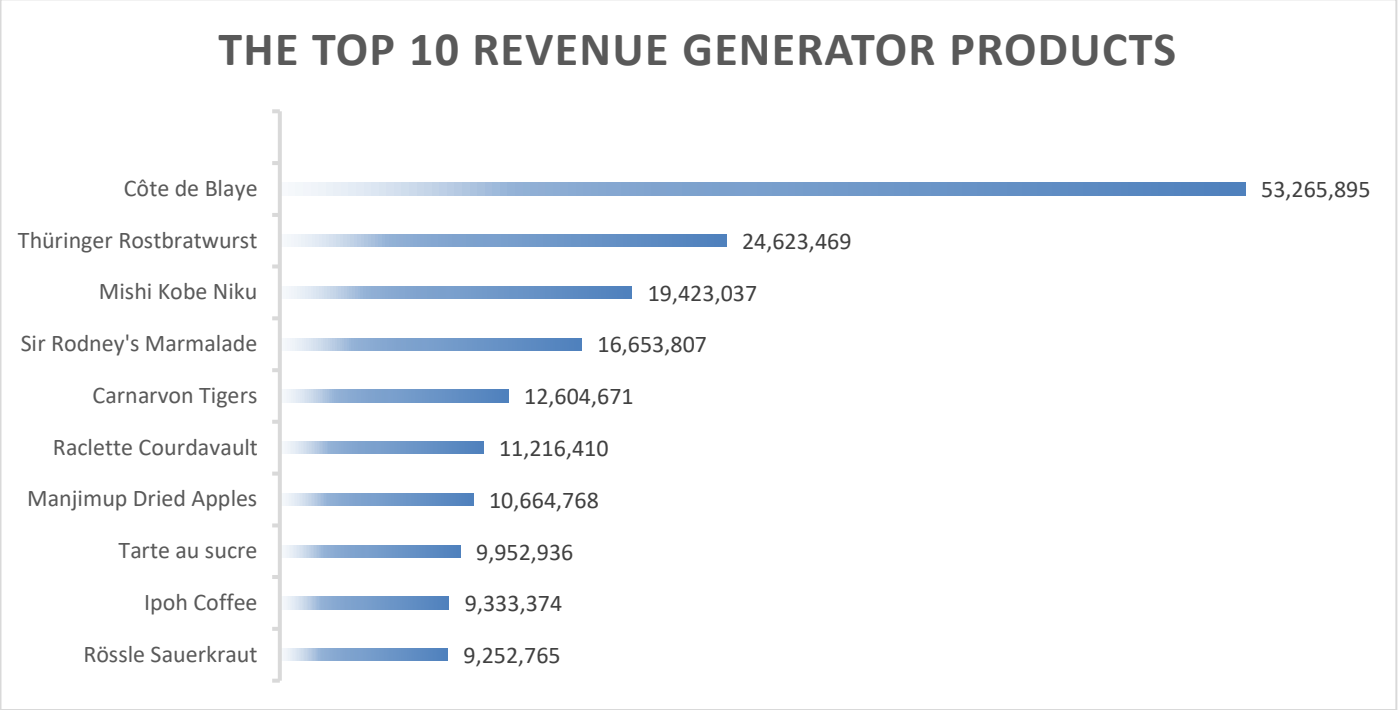
Query Explanation

I used inner join between two tables “Products” and “Order Details” to select sold products, select product name and calculate revenue using formula (unitprice*quantity(1-discount)), group by product name to calculate revenue for each product, and order the result by revenue from max to min and show the first 10 rows by using limit 10.

Query:

```
create view products_with_high_revenue as
 select p.productname
 ,cast(sum(od.unitprice*od.quantity*(1-od.discount)) as int) as TotalRevenue
 from Products p inner join 'Order Details' od
 on p.ProductID= od.productid
  group by 1
  order by TotalRevenue DESC
   limit 10
```

ProductName	TotalRevenue
Côte de Blaye	53265895
Thüringer Rostbratwurst	24623469
Mishi Kobe Niku	19423037
Sir Rodney's Marmalade	16653807
Carnarvon Tigers	12604671
Raclette Courdavault	11216410
Manjimup Dried Apples	10664768
Tarte au sucre	9952936
Ipoh Coffee	9333374
Rössle Sauerkraut	9252765



-Recommendations

-Invest in promoting these products through targeted ads, social media campaigns, and email newsletters to maintain or increase their visibility.

-Ensure these products are always in stock by closely monitoring inventory levels

-The top 10 most frequently ordered products

Query Explanation

I used inner join between two tables “Products” and “Order Details” to select sold products, select product name and calculate total orders by count ORDERID, group by product name to calculate total orders for each product, and order the result by total orders from max to min and show the first 10 rows by using limit 10.

Query:

```
create view products_with_high_sales as

select p.productname

count(distinct od.orderid) TotalOrders,

from Products p inner join 'Order Details' od

on p.ProductID= od.productid

group by 1

order by  TotalOrders DESC

  limit 10
```

ProductName	TotalOrders
Louisiana Hot Spiced Okra	8040
Teatime Chocolate Biscuits	8024
Outback Lager	8020
Sir Rodney's Marmalade	7999
Gumbär Gummibärchen	7999
Gudbrandsdalsost	7991
Raclette Courdavault	7982
Ravioli Angelo	7969
Konbu	7968
Gorgonzola Telino	7964

-Recommendations

-These products are popular but have lower prices, leading to higher order volumes but potentially lower revenue.

ProductName	UnitPrice
Louisiana Hot Spiced Okra	13.6
Teatime Chocolate Biscuits	7.3
Outback Lager	12
Sir Rodney's Marmalade	64.8
Gumbär Gummibärchen	24.9
Gudbrandsdalsost	28.8
Raclette Courdavault	44
Ravioli Angelo	15.6
Konbu	4.8
Gorgonzola Telino	10

-Offer discounts for bulk purchases to encourage even higher sales volumes

-Create loyalty programs for frequent buyers of these products to drive repeat purchase

- Slow Movers

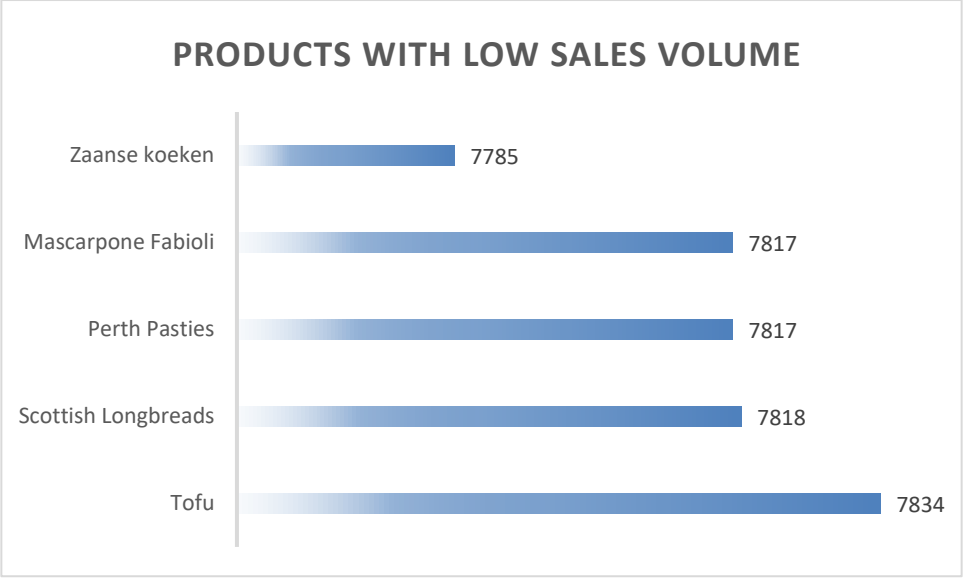
Query Explanation

I used same approach in previous query except ordering , order the result by total orders ascending from min to max and show first 5 rows by using limit 5.

Query:

```
create view products_with_low_sales_volume  as
select p.productname
,count(od.orderid) TotalOrders
from Products p inner join 'Order Details' od
on p.ProductID= od.productid
group by 1
order by TotalOrders ASC
limit 5
```

ProductName	TotalOrders
Zaanse koeken	7785
Mascarpone Fabioli	7817
Perth Pasties	7817
Scottish Longbreads	7818
Tofu	7834



-Recommendations

-Gather customer feedback to understand why these items are not selling well

-Offer discounts, flash sales, or include these products in bundles with top performers to clear inventory.

-Investigate reasons for low performance, such as pricing, quality, or lack of demand. Consider removing underperforming products or revamping them to better meet customer needs.

-Evaluate how these products are displayed on platform/store. Improve their visibility through better categorization and promotional banners.

3-Order Analysis:

-Seasonality

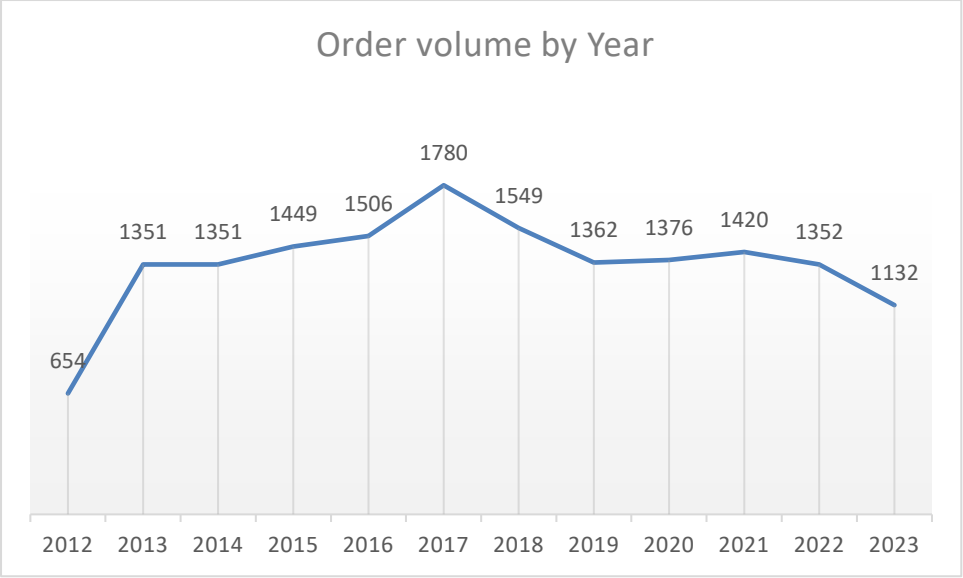
Query Explanation

I used STRFTIME function to extract the year from order date then calculate number of orders and group by years to summarize number of orders for each year.

Query:

```
 select strftime('%Y',orderdate) Years
,count(DISTINCT orderid) NumberOrders
from orders
group by 1
```

Years	NumberOrders
2012	654
2013	1351
2014	1351
2015	1449
2016	1506
2017	1780
2018	1549
2019	1362
2020	1376
2021	1420
2022	1352
2023	1132



Insights & Recommendations

2016, 2017, and 2018 were peak years

-Examine campaigns, promotions, or strategies used during these years to identify what resonated with customers

-The most popular order days

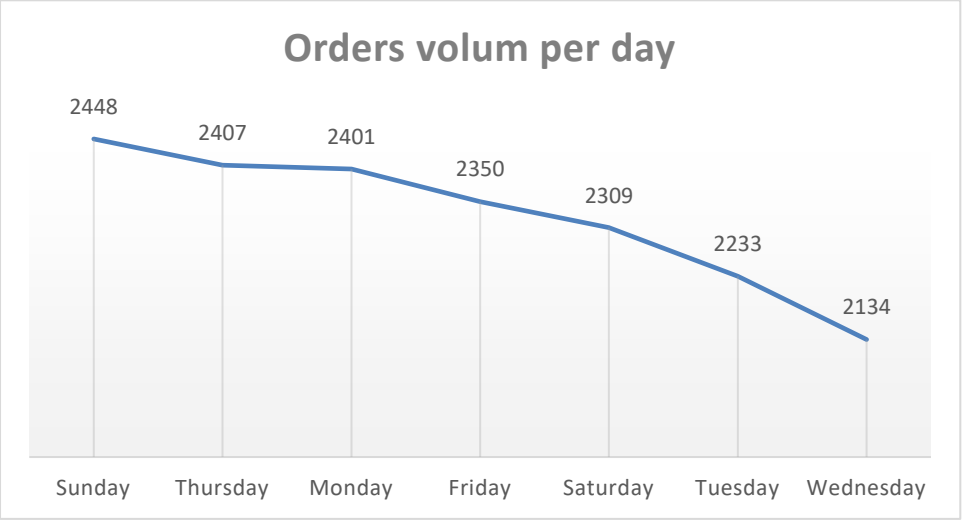
Query Explanation

I used STRFTIME(“%W”) function to extract day of the week from order date, and used case expression to transform days from integer to string, then calculate number of orders column and group by day of the week to calculate number of orders for each day and ordered the result from max to min and select top 2 days by limit 2.

Query:

```
 CREATE view popular_order_days as
select
  case strftime('%w',orderdate)
 when '0' then 'Saturday
'when '1' then 'Sunday
'when '2' then 'Monday
'WHEN '3' THEN 'Tuesday
'WHEN '4' THEN 'Wednesday
'WHEN '5' THEN 'Thursday
'WHEN '6' THEN 'Friday
End   Day_of_the_week
,count(DISTINCT orderid) NumberOrders
from orders
group by  Day_of_the_week
order by NumberOrders DESC
Limit 2
```

Day_of_the_week	NumberOrders
Sunday	2448
Thursday	2407



Insights & Recommendations

Days **Sunday** and **Thursday** are the most days have orders and **Wednesday** is the lowest day in the week

-Schedule flash sales, discounts, and promotional campaigns Sunday and Thursday to maximize revenue.

- Use social media and email marketing to remind customers of special offers on these days.

-The distribution of order quantities

Query Explanation

select order id from order details table and calculate sum of units sold for each order id by using SUM(quantity)

Query:

```
select orderid
 ,sum(quantity) Units_Sold
 from 'Order Details'
  group by 1
 order by Units_Sold DESC
```

The result is 16283 rows

This table contain of Top_30_order

in terms of units sold

OrderId	Units_Sold
13460	2308
17596	2283
13372	2258
13466	2233
25679	2205
18304	2203
21461	2195
22571	2195
17988	2191
23146	2190
14724	2188
14634	2187
23186	2183
19294	2176
19689	2176
19413	2171
11487	2170
11890	2168
21692	2167
25500	2166
13450	2163
16186	2163
17455	2162
23766	2159

-Recommendations

-For low-quantity orders, consider offering free shipping thresholds or volume discounts to encourage larger purchases

- For bulk orders, create special pricing tiers or loyalty programs to retain high-value customers

3-Employee Performance

- Total Revenue Generated

Query Explanation

I used inner join between two tables “Employees” and “order details” to Select employees who made revenue, I used CONCAT function to merge first name and last name in one string, group by full name to summarize total revenue for each employee and show result of revenue as integer number using function CAST, order the result by total revenue from max to min .

Query:

```
create view Total_Revenue_Generated as
select concat(e.FirstName,' ', e.LastName) FullName
,cast( sum(od.unitprice*od.quantity*(1-od.discount)) as int) TotalRevenue
from Employees e inner join orders o
on e.EmployeeID=o.EmployeeID inner join 'Order Details' od
on o.OrderID=od.orderid
group by FullName
order by TotalRevenue desc
```

FullName	TotalRevenue
Margaret Peacock	51488395
Steven Buchanan	51386459
Janet Leverling	50445573
Nancy Davolio	49659423
Robert King	49651899
Laura Callahan	49281136
Michael Suyama	49139966
Anne Dodsworth	49019678
Andrew Fuller	48314100

- Total Sales Volume

Query Explanation

I used same previous approach in writing the query except count of total orders instead sum of revenue

Query:

```
create view Total_Sales_Volume as
select concat(e.FirstName,' ', e.LastName) FullName
count(DISTINCT o.OrderID) TotalOrders ,
from Employees e inner join orders o
on e.EmployeeID=o.EmployeeID
group by FullName
order by TotalOrders desc
```

FullName	TotalOrders
Margaret Peacock	1908
Nancy Davolio	1846
Janet Leverling	1846
Steven Buchanan	1804
Laura Callahan	1798
Robert King	1789
Andrew Fuller	1771
Anne Dodsworth	1766
Michael Suyama	1754

- Average order value

Query Explanation

I used same previous approach in writing the query except calculating average order value by division sum of revenue over count of orders.

Query:

```
create view Average_order_value as
select concat(e.FirstName,' ', e.LastName) FullName
cast(sum(od.unitprice*od.quantity*(1-od.discount)) / count(DISTINCT
o.OrderID) as int) Average_order_value
from Employees e inner join orders o
on e.EmployeeID=o.EmployeeID inner join 'Order Details' od
on o.OrderID=od.orderid
group by FullName
```

FullName	Average_order_value
Andrew Fuller	27280
Anne Dodsworth	27757
Janet Leverling	27326
Laura Callahan	27408
Margaret Peacock	26985
Michael Suyama	28015
Nancy Davolio	26901
Robert King	27753
Steven Buchanan	28484

- Employee Performance Analysis in one Query

Query Explanation

Also we can analyze employees performance in one query and I prefer this solution.

Because the information in different two tables ORDERS and EMPLOYEES so I used inner join to select employees who have orders and revenue. Group by full name to summarize the result of Revenue, total orders and average order value for each employee.

Query:

```
create view Employee_Performance as
select concat(e.FirstName,' ', e.LastName) FullName
, cast(sum(od.unitprice*od.quantity*(1-od.discount)) as int) TotalRevenue
, count(DISTINCT o.OrderID) TotalOrders,
cast(sum(od.unitprice*od.quantity*(1-od.discount)) / count(DISTINCT o.OrderID) as
int) Average_order_value
from Employees e inner join orders o
on e.EmployeeID=o.EmployeeID inner join 'Order Details' od
on o.OrderID=od.orderid
group by FullName
```

FullName	TotalRevenue	TotalOrders	Average_order_value
Andrew Fuller	48314100	1771	27280
Anne Dodsworth	49019678	1766	27757
Janet Leverling	50445573	1846	27326
Laura Callahan	49281136	1798	27408
Margaret Peacock	51488395	1908	26985
Michael Suyama	49139966	1754	28015
Nancy Davolio	49659423	1846	26901
Robert King	49651899	1789	27753
Steven Buchanan	51386459	1804	28484

- Insights

- “ Margaret Peacock” and “Steven Buchanan” are the top employees in revenue generated, and “Andrew Fuller” is the lowest employee

- “Margaret Peacock” is top employee in terms of number of orders, and “Michael Suyama” is the lowest employee

- Recommendations

- Recognize top-performing employees and analyze their methods. Share best practices across the team to uplift others

- Provide targeted training programs for underperforming employees to improve their skills in sales, customer engagement, or operational tasks

- Keep employees motivated by fostering a positive work environment, providing career growth opportunities, and maintaining open communication