The ShopNest Portugal Experience

ShopNest is an online e-commerce platform which provides small business merchants from Portugal regions to showcase and sell their products to customers.The platform serves as a single point of contact between merchants and customers addition to that it facilitates shipping through logistics partners

Pre-work before loading data : Data cleaning

The following steps are performed:

- ➔ Checked column header names and used “Use first row as headers” for required tables
- ➔ Changed the data types
- ➔ Splitted time and date into different columns
- ➔ Removed an unnecessary row from sellers table where seller_city was given in numeric
- ➔ Removed duplicates and errors
- ➔ Handled whitespaces (performed trimming)
- ➔ Standardized text case to lowercase
- ➔ Checked column quality, distribution and profiling
- ➔ Identified outliers
- ➔ Handled null and blank values
- ➔ These replacings will ideally be done according to the business
    - ◆ Numerical data: can be replaced by mean, median(if skewed data, outliers present)
    - ◆ Categorical data: replaced by mode
    - ◆ Date : In our dataset the missed customer_delivery_date is filled with estimated_delivery_date and the missed carrier_delivery_date is filled with order_purchased_date

Tasks

Task 1: Top categories by total price

Identify and visually top 10 product categories by total sales

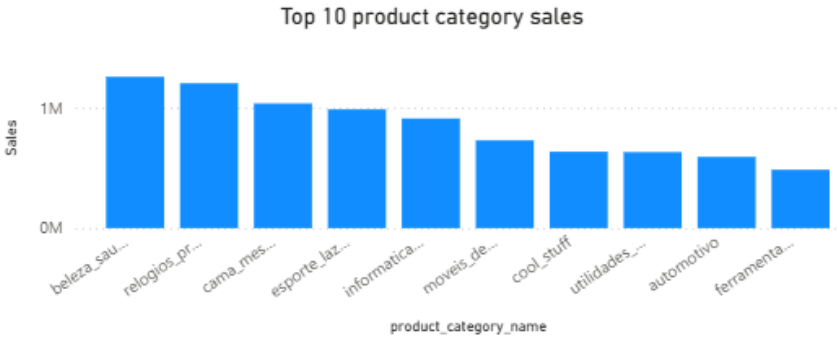
Action: we need to identify top 10 product category wise total sales

Tables connection:

- Product_category_name (1) - Products (many) based on product_category_name
- Products (1) - Order_items (many) based on product_id

Statistical and graphical analysis:

product_category_name	Sum of price
beleza_saude	12,58,681.34
relogios_presentes	12,05,005.68
cama_mesa_banho	10,36,988.68
esporte_lazer	9,88,048.97
informatica_acessorios	9,11,954.32
moveis_decoracao	7,29,762.49
cool_stuff	6,35,290.85
utilidades_domesticas	6,32,248.66
automotivo	5,92,720.11
ferramentas_jardim	4,85,256.46
Total	**84,75,957.56**



Explanation:

For statistical analysis:

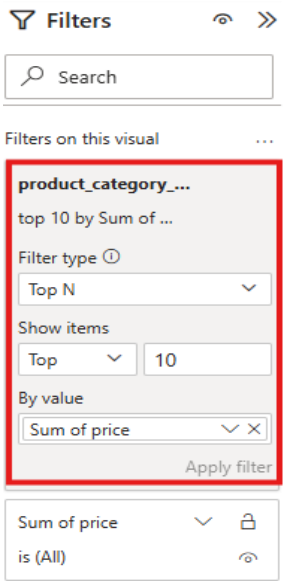
Matrix visual is chosen with product category name on rows and price on values

For graphical analysis:

Clustered column visual is chosen and used same columns for x,y axes respectively

To get the top 10 product category names, top N (top 10) filter on products category name from filter visuals is used

Please refer the below image



Insights: beleza_saude (health and beauty) is the product category which has highest number of sales and ferramentas_jardim (garden_tools) is the one which has lowest number of sales

Task 2: Delayed orders analysis

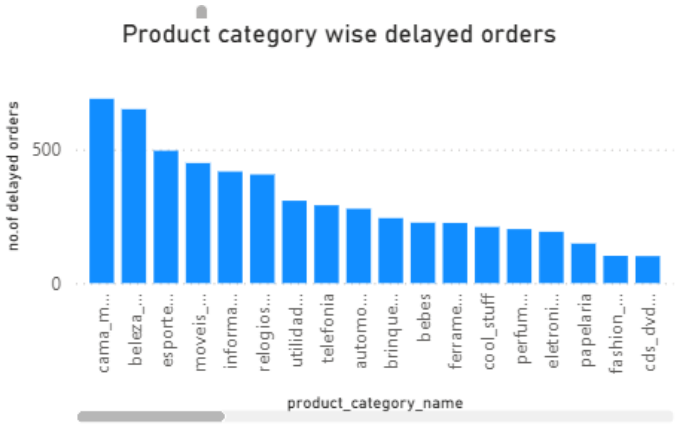
Determine the number of delayed orders in each category. An order is considered delayed if the actual delivery date is later than the estimated delivery date

Action: we need to identify category wise delayed orders

Tables connection: Orders (1) - order_items (many) based on order_id

Statistical and graphical analysis:

product_category_name	Sum of delayed
cama_mesa_banho	689
beleza_saude	650
esporte_lazer	495
moveis_decoracao	449
informatica_acessorios	417
relogios_presentes	406
utilidades_domesticas	308
telefonia	291
automotivo	278
brinquedos	243
bebes	226
ferramentas_jardim	225
Total	**6534**



Explanation:

To calculate if the order is delayed or on_time, we can create a conditional column called "delayed" in the orders table in data view

delayed = IF(Orders[Order_delivered_customer_date_corrected].[Date] > Orders[Order_estimated_delivery_date].[Date], 1,0)

1 → delayed

0 → on_time

For statistical analysis:

Matrix visual is chosen with product category name on rows and the new conditional column delayed on values

For graphical analysis:

Clustered bar visual is chosen with product category name on x-axis and the new conditional column delayed on y-axis

Insights: cama_mesa_banho (bed_bath_table) product category has more number of order delays to the customers and seguros_e_servicos (security_and_services) has 0 order delays

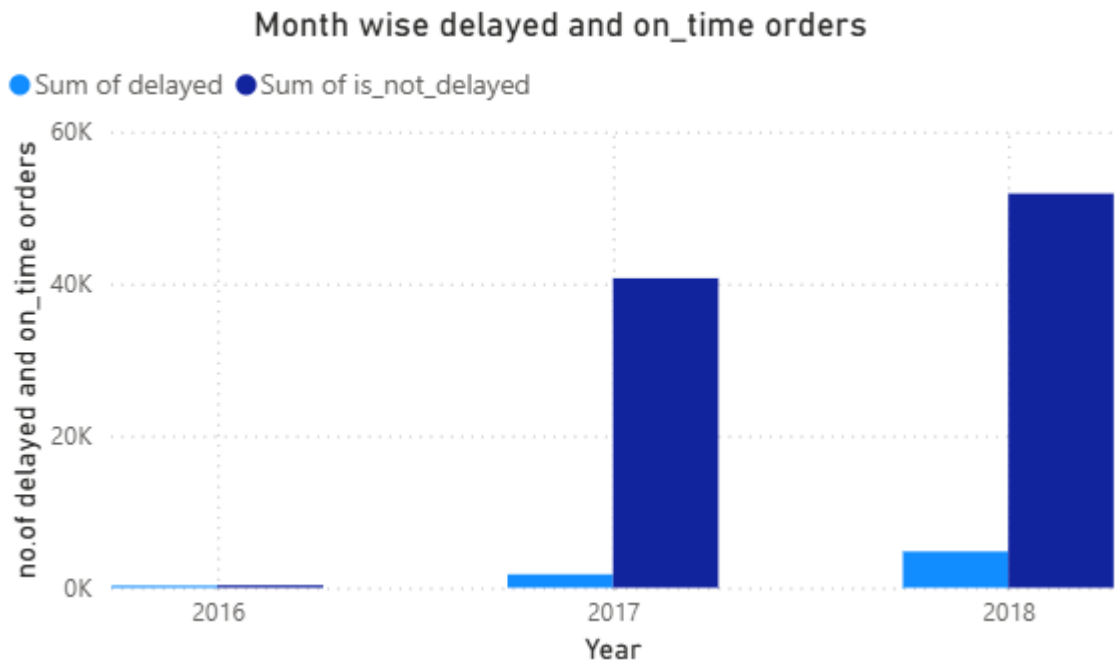
Task 3: Monthly comparison of delayed and on_time orders

Create a dynamic visual that compares the number of delayed orders to number of orders received earlier for each month

Action: we need to identify month wise number of delayed and on_time orders

Tables connection: Orders table is used

Graphical analysis:



Explanation:

A clustered column chart is used for visualizing month wise delayed and on_time orders

For delayed orders we have already created a conditional column called “delayed”

Like wise another conditional column named “is_not_delayed” is created in the orders table in data view

delayed = IF(Orders[Order_delivered_customer_date_corrected].[Date] > Orders[Order_estimated_delivery_date].[Date], 1,0)

1 → delayed

0 → on_time

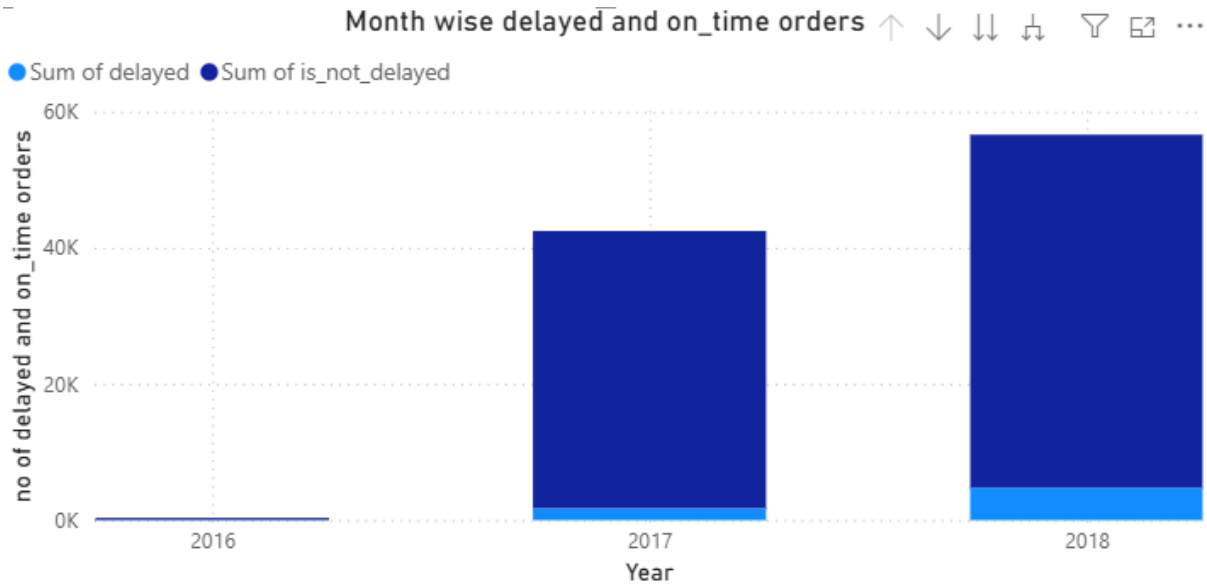
is_not_delayed = IF(Orders[Order_delivered_customer_date_corrected].[Date] <= Orders[Order_estimated_delivery_date].[Date], 1,0)

1 → on_time

0 → delayed

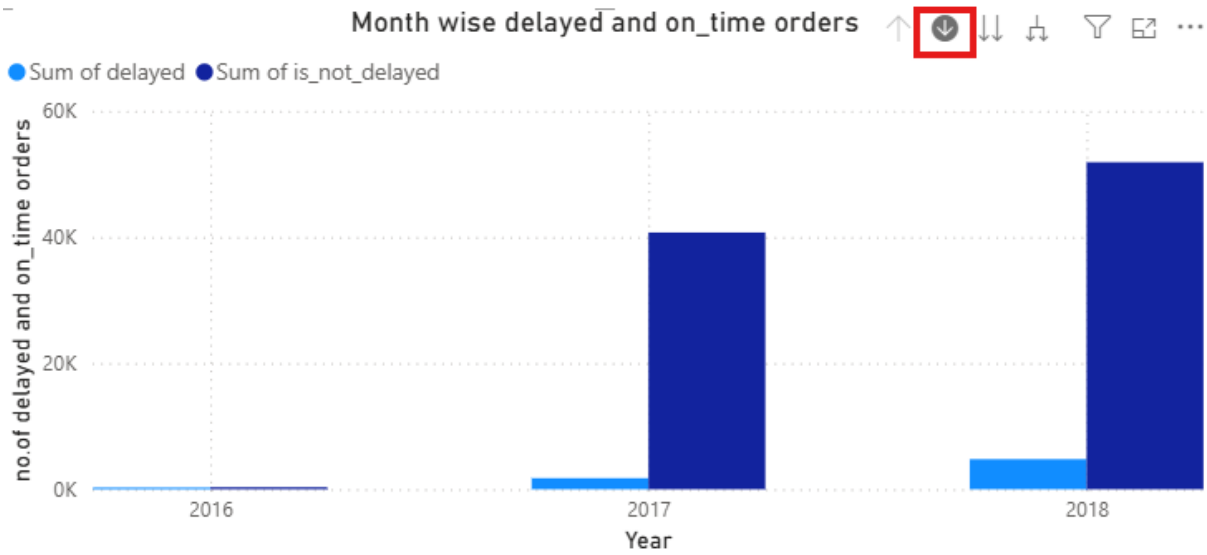
This time I tried to select a “Stacked column chart”, unfortunately the number of delayed orders are not visible unless drilling is performed. So to have a visual of that also without drilling, clustered column chart is chosen

Please have a look on the stacked column chart:

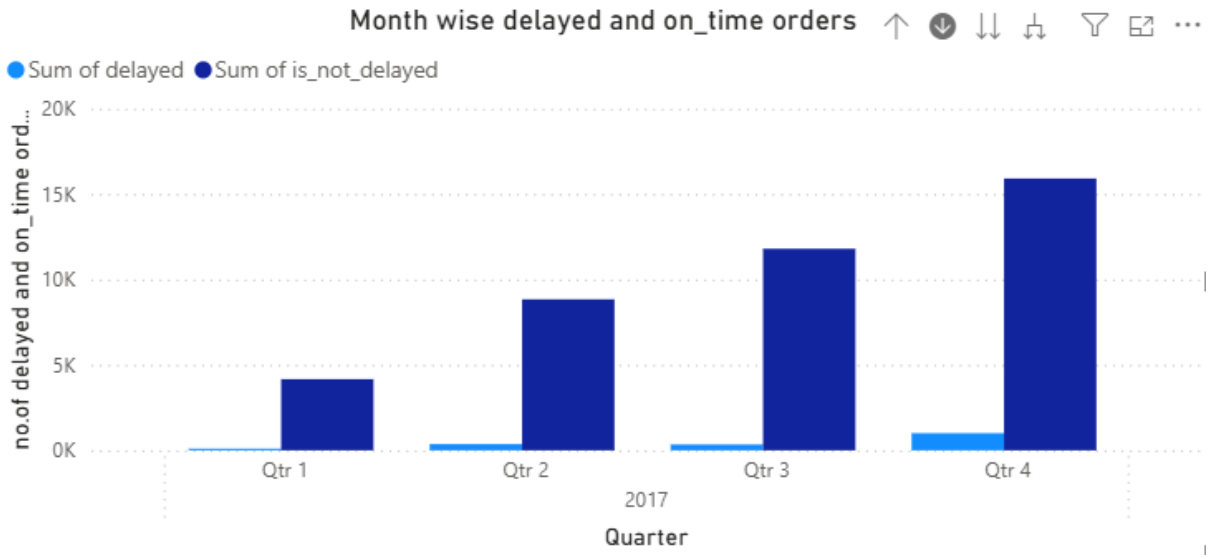


As we need to find month wise number of delayed and on_time orders, lets perform "DRILLING" in the clustered column chart

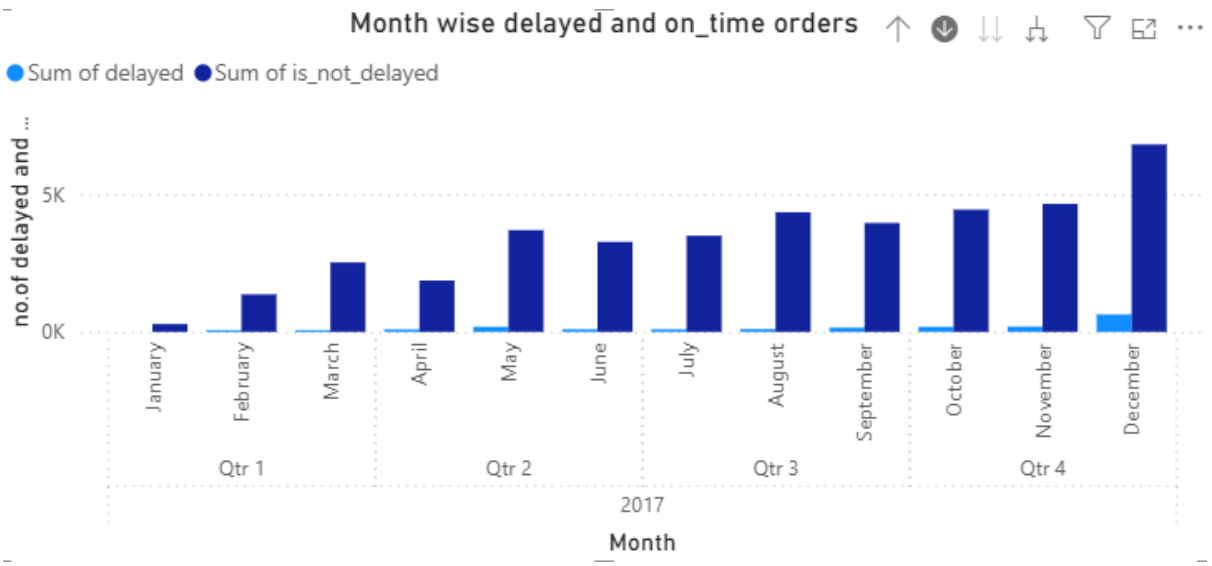
Drill down mode is ON:



Quarterly:



Monthly wise report:

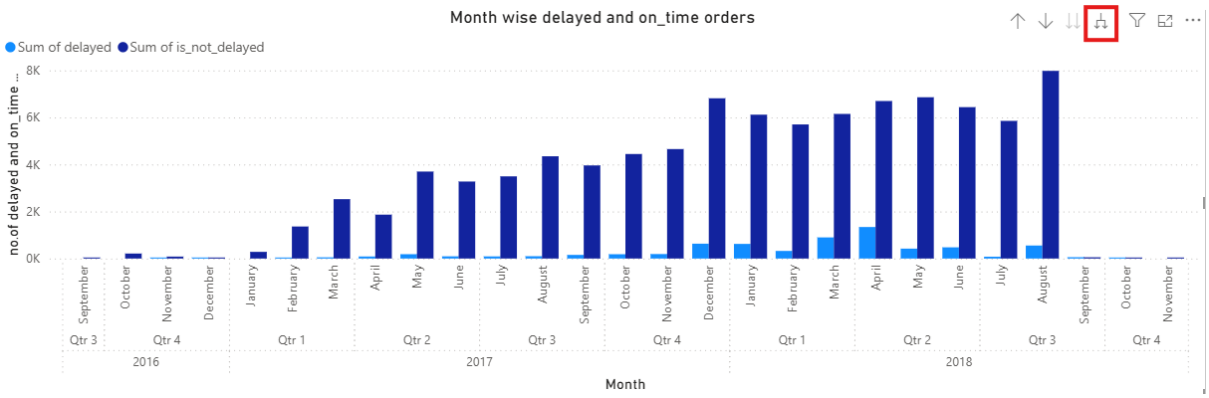


The below image is the drill up mode:

After one click on drill up we will be able to see quarter wise report

We can also visualize all years month wise orders at a time, we can use “hierarchy”

Please have a look on the below image:



Year	Quarter	Month	Sum of delayed	Sum of is_not_delayed
2018	Qtr 2	April	1340	6698
2018	Qtr 1	March	898	6149
2017	Qtr 4	December	631	6814
2018	Qtr 1	January	622	6117
2018	Qtr 3	August	553	7980
2018	Qtr 2	June	476	6437
2018	Qtr 2	May	421	6858
2018	Qtr 1	February	323	5702
2017	Qtr 4	November	190	4654
2017	Qtr 2	May	184	3698
2017	Qtr 4	October	183	4444
2017	Qtr 3	September	154	3959
2017	Qtr 3	August	98	4348
Total			**6535**	**92906**

Insights: August 2018 (Q2) - has highest number of delayed deliveries to the customers,
August 2018 (Q3) - has highest number of on_time deliveries to the customers,
September 2016 - November 2018 there are 0 number of delayed orders to customers,
In september 2018 there are only 2 orders which are not delivered on time

Task 4: Payment method analysis:

Analyze the most frequently used payment methods by customers using a visually appealing representation such as pie or other suitable visuals

Action: we need to find the most frequently used payment method by customers

Tables connection: Orders (1) - order_payments (many) based on order_id

Statistical and graphical analysis:

payment_type	count_of_payments
credit_card	76795
boleto	19784
voucher	5775
debit_card	1529
not_defined	3
Total	**103886**

For statistical analysis:

I have chosen Matrix visual with payment_type on rows and count_of_payments on values

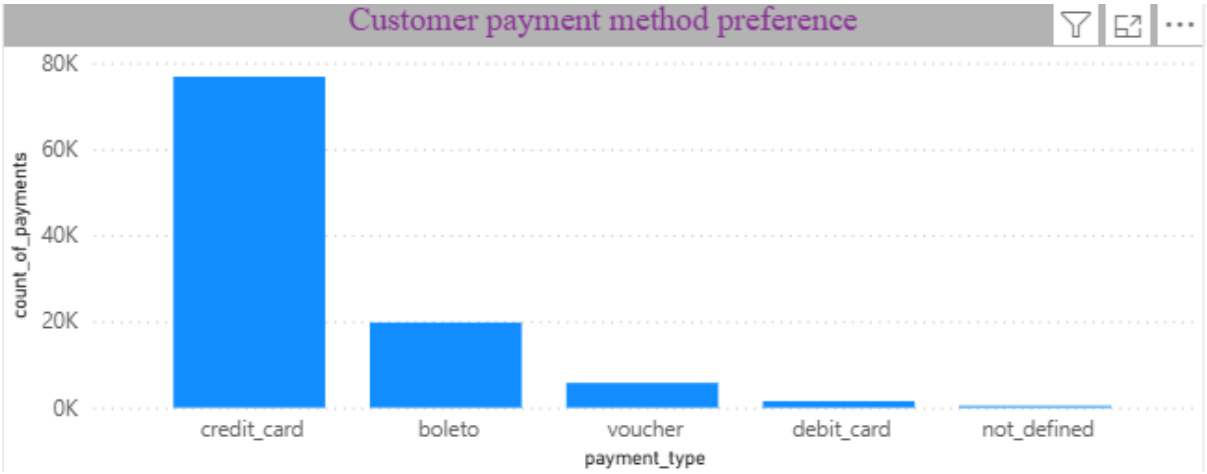
To find the count of payments a dax measure is used to calculate

```
count_of_payments = COUNTROWS('Order_payments')
```

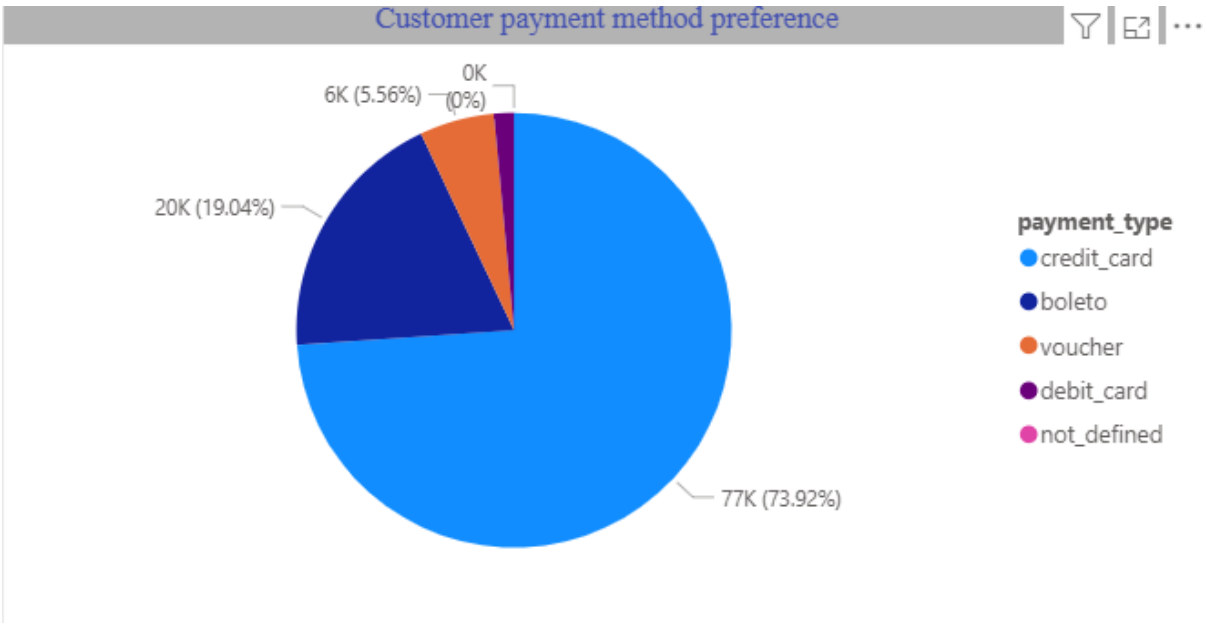
For graphical analysis:

I have chosen clustered column chart visual over pie chart as in the pie we are not able to see the small slice “not defined”

In the clustered column chart with payment_type on rows and count_of_payments on values



Please have a look on pie chart where a small slice “not defined” is not observed



Insights: credit card is the most frequently used payment methods by customers and debit card is less frequently used payment methods by customers

Task 5: Product rating analysis:

Determine the top 10 highest rated products and bottom 10 lowest rated products using bar or column chart

Action: we need to identify top 10 highest rated and bottom 10 lowest rated products

Tables connection: Orders (1) - order_reviews (many) based on order_id

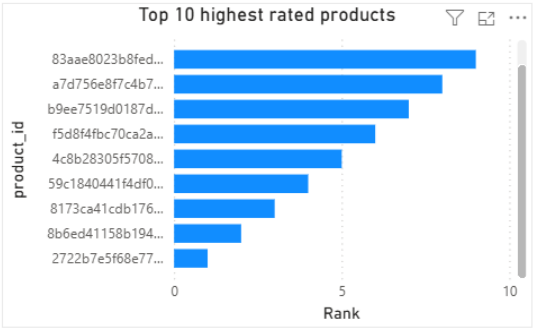
The cross filter is used as both in order to have data flow from both the sides

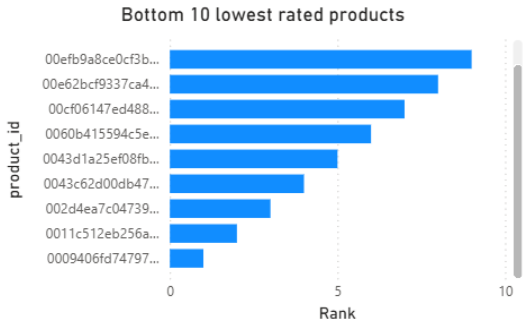
Graphical analysis:

For debugging purpose statistical visualization helps us:

product_id	ranking_key	Sum of tie_breaker	Sum of top_rank
2722b7e5f68e776d18fe901638034e54	5000013.00000000000000-0000005180	5180	1
8b6ed41158b194711f83b8da92757544	5000011.00000000000000-0000017926	17926	2
8173ca41cdb176462f9ae79821a48404	5000011.00000000000000-0000016719	16719	3
59c1840441f4df065f52760bf51a8442	5000011.00000000000000-0000011648	11648	4
4c8b28305f570899b6ded964ddd234a9	5000011.00000000000000-0000009950	9950	5
f5d8f4fbc70ca2a0038b9a0010ed5cb0	5000010.00000000000000-0000031654	31654	6
b9ee7519d0187d2389af62ba6c612963	5000010.00000000000000-0000023885	23885	7
a7d756e8f7c4b7e5b679e248a57d91ec	5000010.00000000000000-0000021497	21497	8
83aae8023b8feda53259f63e0ec06390	5000010.00000000000000-0000017012	17012	9
57552a168008a60472e3e6bb351422e7	5000009.00000000000000-0000011339	11339	10
Total		**166810**	**55**

product_id	ranking_key	Sum of tie_breaker	Sum of bottom_rank
0009406fd7479715e4bef61dd91f2462	1000001.00000000000000-0000000003	3	1
0011c512eb256aa0dbbb544d8dffcf6e	1000001.00000000000000-0000000006	6	2
002d4ea7c04739c130bb74d7e7cd1694	1000001.00000000000000-0000000020	20	3
0043c62d00db47eff6a6bc4cf6bfaeda	1000001.00000000000000-0000000033	33	4
0043d1a25ef08fb6f41b8fa6f91742ab	1000001.00000000000000-0000000034	34	5
0060b415594c5e1200324ef1a18493c4	1000001.00000000000000-0000000041	41	6
00cf06147ed4880ec5fbba2adbb20e1d	1000001.00000000000000-0000000089	89	7
00e62bcf9337ca4c5d5b4c5c8188f8d2	1000001.00000000000000-0000000108	108	8
00efb9a8ce0cf3b2f37892ab003edc10	1000001.00000000000000-0000000110	110	9
0103863bf3441460142ec23c74388e4c	1000001.00000000000000-0000000120	120	10
Total		**564**	**55**





Explanation:

To find the highest and lowest rated products first I tried to check using their average review scores and observed that most of the products have the same average review scores.

Then added count of review scores ie., avg_rating = average_review_scores + count_of_review_scores

Found we had same avg_rating for different products

So, multiplied the avg_review_scores with a large number ie., avg_review_scores * 1000000

Now, composite_number_calc = (avg_review_scores * 1000000) + count_of_review_scores

Why to use 1000000 why not 10 or 20 or any number?

Lets assume prod1 has avg_review_score = 4 and count_of_reviews = 100,

prod1 has avg_review_score = 5 and count_of_reviews = 1,
If we multiply avg_review_score with 10, for prod1 we get 4*10 + 100 = 140, prod2 = 5*1+100 = 105
We cant give fair ranking in such cases, in order to not affect avg_review_scores because of change in count of reviews, we need to multiply the avg_review_scores with large number
prod1=4*1000000+100=4000100
prod2=5*1000000+1=5000001, So fair ranking can be done

composite_number_calc used is:

composite_number_calc = AVERAGEX(RELATEDTABLE(Order_reviews),Order_reviews[review_score]) * 1000000 + COUNTROWS(RELATEDTABLE(Order_reviews))

In order to give ranking we can use RANKX() function which takes: table, expression, value, order, ties

What i understood is as we will be ranking the product_id based on expression the expression should be always unique to get unique rankings

The composite_number_calc itself is not able to provide us the unique rankings, we need to add some other tie-breaker

Lets rank each product_id which will be unique and will be used as tie-breaker
tie_breaker = RANKX(ALL(Products[product_id]),Products[product_id], ,ASC, Dense)

We need to add composite_number_calc and tie_breaker which can give us unique ranks

Adding didn't give the expected results, so i have concatenated those two
ranking_key = FORMAT(Products[composite_number_calc],"0.00000000000000") & "-" & FORMAT(Products[tie_breaker],"0000000000")

Finally top_rank and bottom_rank:

top_rank = RANKX(ALL(Products),
Products[ranking_key],
,

```
DESC,
Dense
)

bottom_rank = RANKX(ALL(Products),
Products[ranking_key],
,
ASC,
Dense
)
```

To debug we can use statistical method, table visual is chosen and product_id, ranking_key, tie_breaker, top_rank are used to check if the product_ids are having unique ranks for top_rank products, used bottom_rank for bottom_rank products
And yes, we achieved what we expected

For graphical analysis:
A bar graph was chosen with ranks on x-axis and product_ids on y-axis
To identify top10 and bottom 10 products we can use top N filter on top_rank and bottom_rank in the filter visuals

Insights: Health_beauty, sports_liesure, fashion_bags_accessories, furniture_decor, baby are the top 10 highest rated products
Bed_bath_table, auto, pet_shop, construction_tools_construction, books_general_interest are the bottom 10 lowest rated products

Task 6: State wise sales analysis

Identify and visually represent states with high and low sales, providing a clear understanding of regional sales performance

Action: we need to identify the states with high and low sales including regional sales performance

Tables connection:

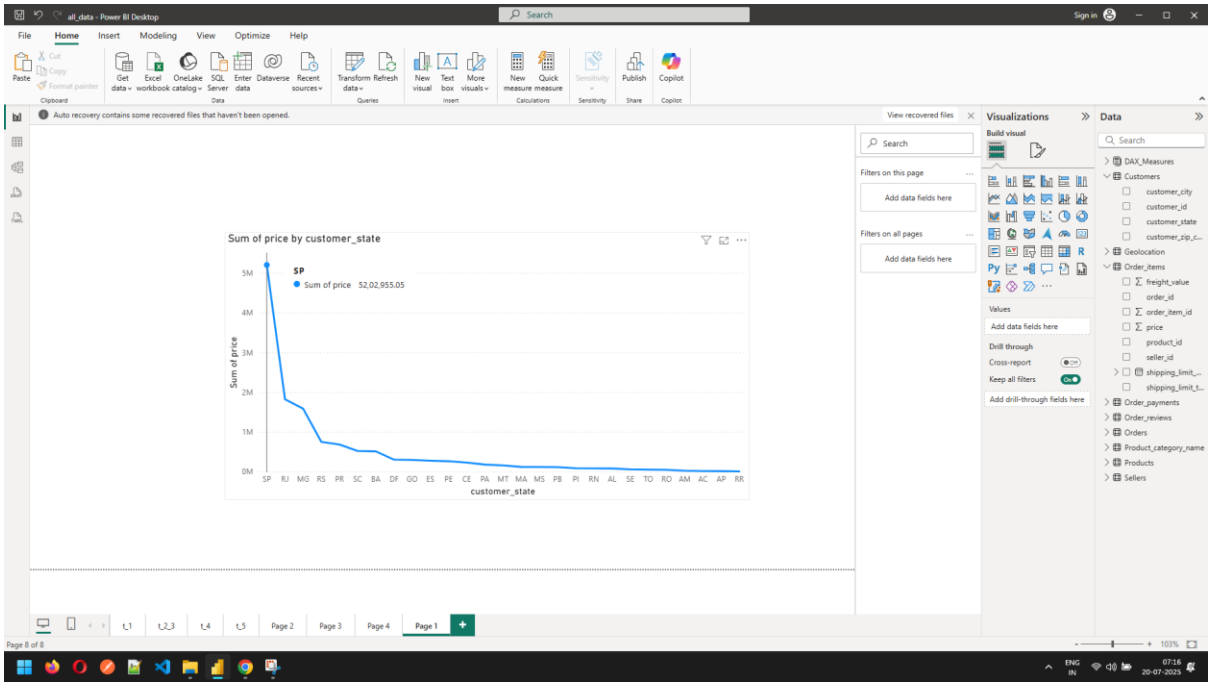
- Customers (1) - orders (many) based on customer_id
- Orders (1) - order_items (many) based on order_id

Statistical and visual analysis:

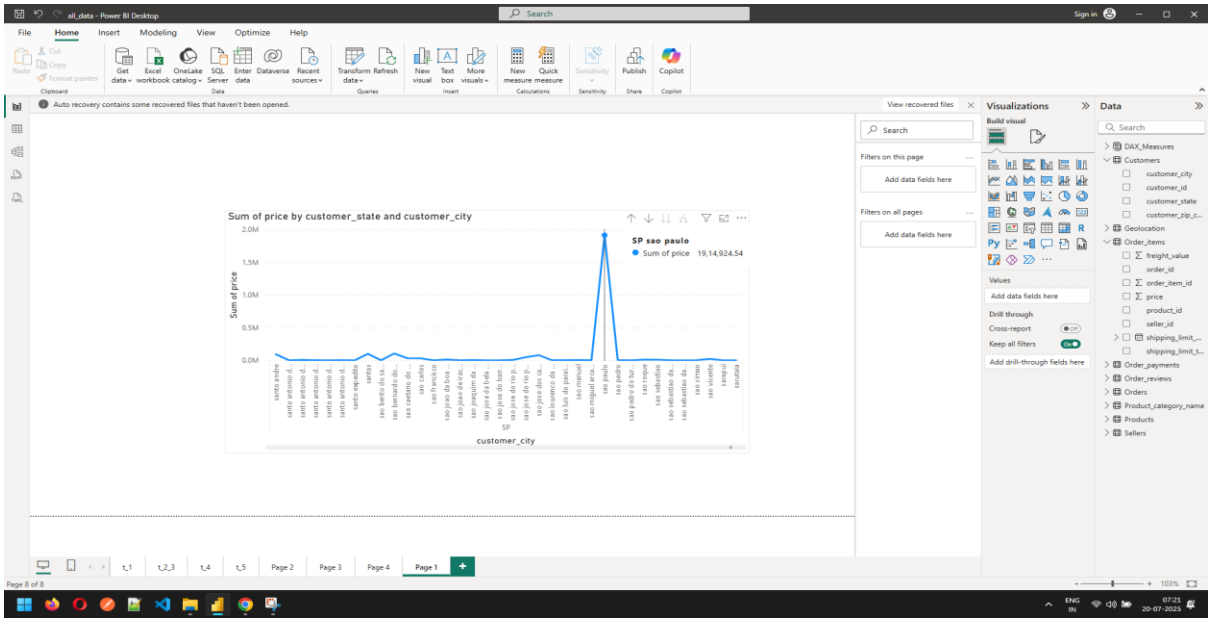
Highest sale state:

customer_state	Sum of price
⊟ **SP**	**52,02,955.05**
sao paulo	19,14,924.54
campinas	1,87,844.53
guarulhos	1,44,268.39
sao bernardo do campo	1,04,540.99
santos	98,777.09
santo andre	92,028.60
osasco	82,157.86
jundiai	81,310.30
sao jose dos campos	78,650.94
sorocaba	76,551.76
ribeirao preto	65,637.01
piracicaba	52,307.98
mogi das cruzes	52,007.78
barueri	51,328.35
Total	**52,02,955.05**

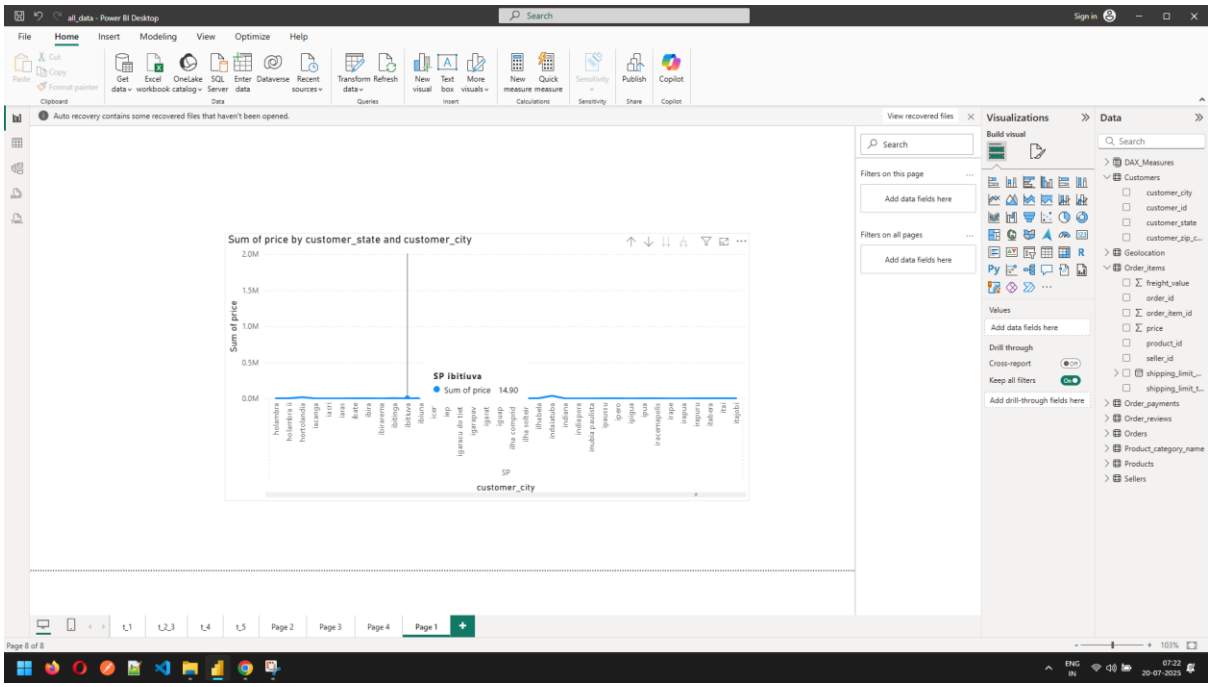
customer_state	Sum of price
⊟ **SP**	**52,02,955.05**
ibitiuva	14.90
aparecida d'oeste	19.90
nova luzitania	22.90
nova independencia	23.90
turmalina	28.00
boraceia	29.00
fernao	29.00
parisi	34.49
gabriel monteiro	34.90
florinia	41.50
agisse	43.00
queiroz	45.00
populina	47.60
elisiario	49.00
Total	**52,02,955.05**



Highest sales region from highest sales state:

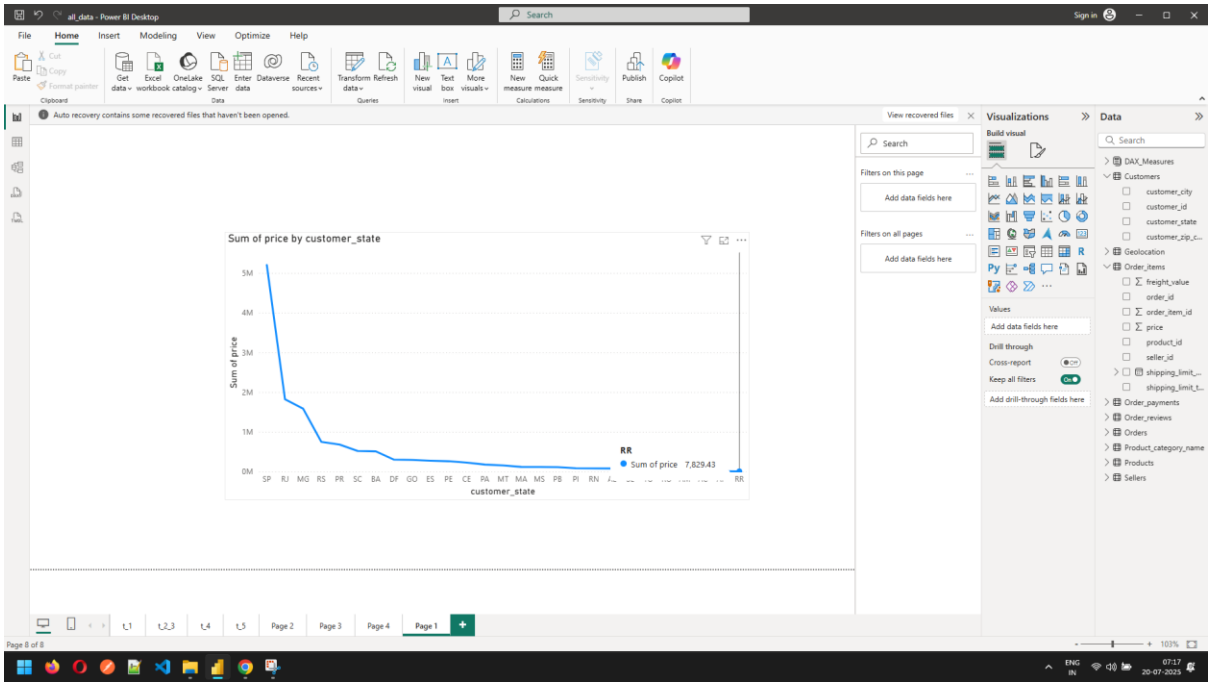


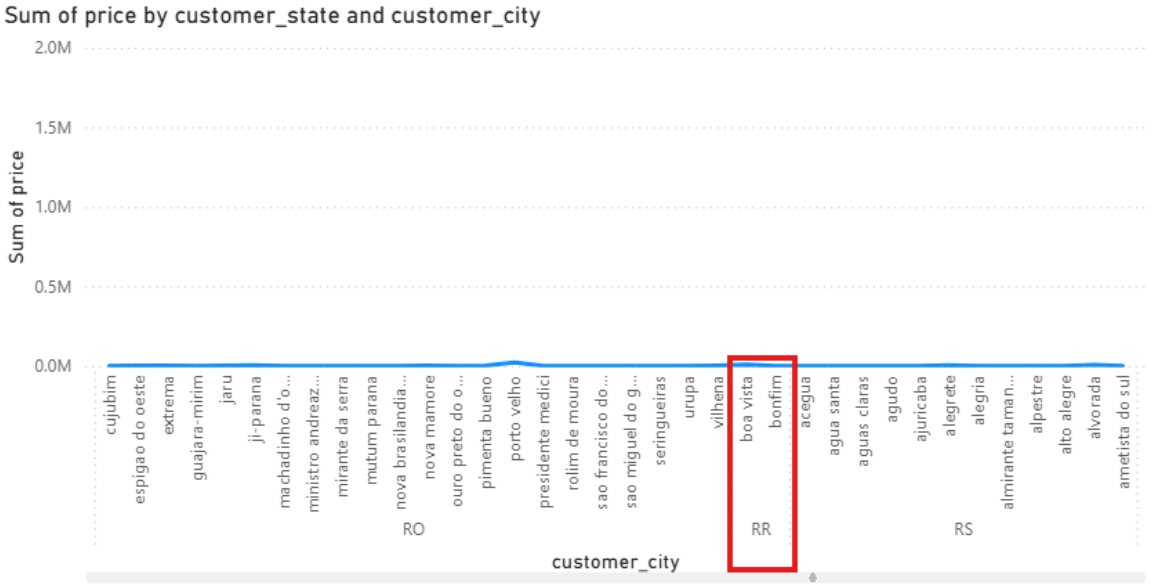
Lowest sales region from highest sales state:



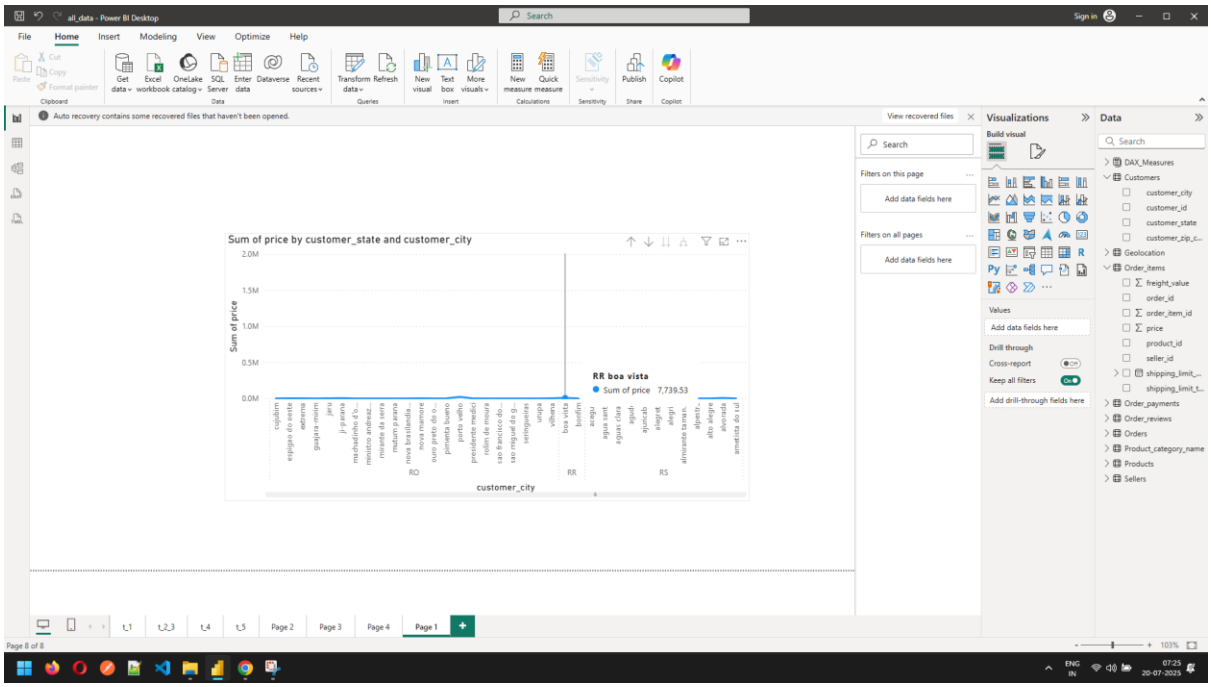
Lowest sales state:

customer_state	Sum of price
RR	**7,829.43**
boa vista	7,739.53
bonfim	89.90
Total	**7,829.43**

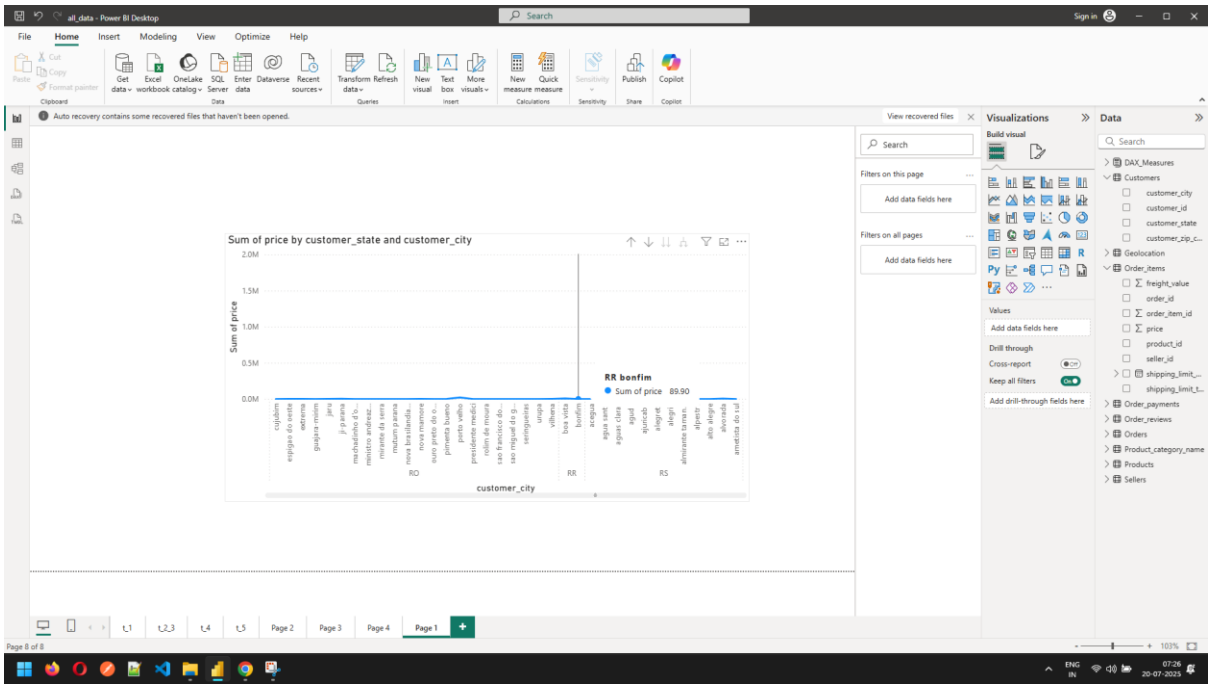

Sum of price by customer_state
Sum of price
customer_state
RR
Sum of price 7,829.43
Visualizations
Data


Sum of price by customer_state and customer_city
Sum of price
2.0M
1.5M
1.0M
0.5M
0.0M
customer_city
RO
RR
RS
cujubim
espigao do oeste
extrema
guajara-mirim
jaru
ji-parana
machadinho d'o...
ministro andreaz...
mirante da serra
mutum parana
nova brasilandia...
nova mamore
ouro preto do o...
pimenta bueno
porto velho
presidente medici
rolim de moura
sao francisco do...
sao miguel do g...
seringueiras
urupa
vilhena
boa vista
bonfim
acegua
agua santa
aguas claras
agudo
ajuricaba
alegrete
alegria
almirante taman...
alpestre
alto alegre
alvorada
ametista do sul

Highest sales region in lowest sales state:



Lowest sales region in lowest sales state:



Explanation:

For statistical analysis:

Matrix visual is chosen with customer state and city on x-axis and price on y-axis

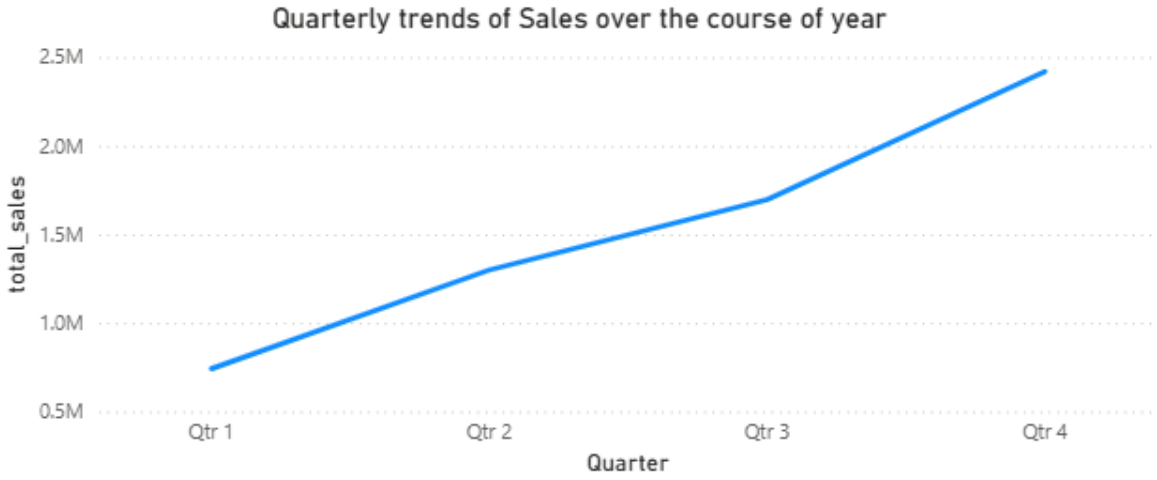
Used customer state filter from filter visuals to get highest and lowest sales states, we can use top N (top 1) and (bottom 1) filter and in by values we can drag “price”
For graphical analysis:
A line chart is chosen with customer state and city on x-axis and price on y-axis
The filters used in filter visuals for statistical analysis, is also performed for graphical analysis to get the highest and lowest sales states

Insights:
The state having more number of sales is “SP” with more number of sales in “sao paulo” and less number of sales in “ibitiuva”
The state having less number of sales is “RR” with more number of sales in “boa vista” and less number of sales in “bonfim”

Task 7: Seasonal sales patterns
Investigate and visualize any seasonal (quarterly) patterns or trends in sales data over the course of year
Action: we need to identify any quarterly trends of sales over the course of year
Tables connection: Orders (1) - order_items (many) based on order_id

Statistical and graphical analysis:

Year	Sum of price
⊟ 2017	61,55,806.98
⊟ Qtr 1	7,41,960.19
⊞ January	1,20,312.87
⊞ February	2,47,303.02
⊞ March	3,74,344.30
⊟ Qtr 2	12,99,036.97
⊞ April	3,59,927.23
⊞ May	5,06,071.14
⊞ June	4,33,038.60
⊟ Qtr 3	16,96,404.85
⊞ July	4,98,031.48
⊞ August	5,73,971.68
⊞ September	6,24,401.69
⊟ Qtr 4	24,18,404.97
⊞ October	6,64,219.43
⊞ November	10,10,271.37
⊞ December	7,43,914.17
Total	61,55,806.98



Explanation:

For statistical analysis:

Matrix visual is chosen with order_purchased_date on x-axis and price on y-axis

Using filter visuals, quarterly trends for 2017 is visualized

For graphical analysis:

Line chart is chosen with order_purchased_date on x-axis and price on y-axis

Using filter visuals, quarterly trends for 2017 is visualized

Insights: The sales are increased for every quarter in 2017

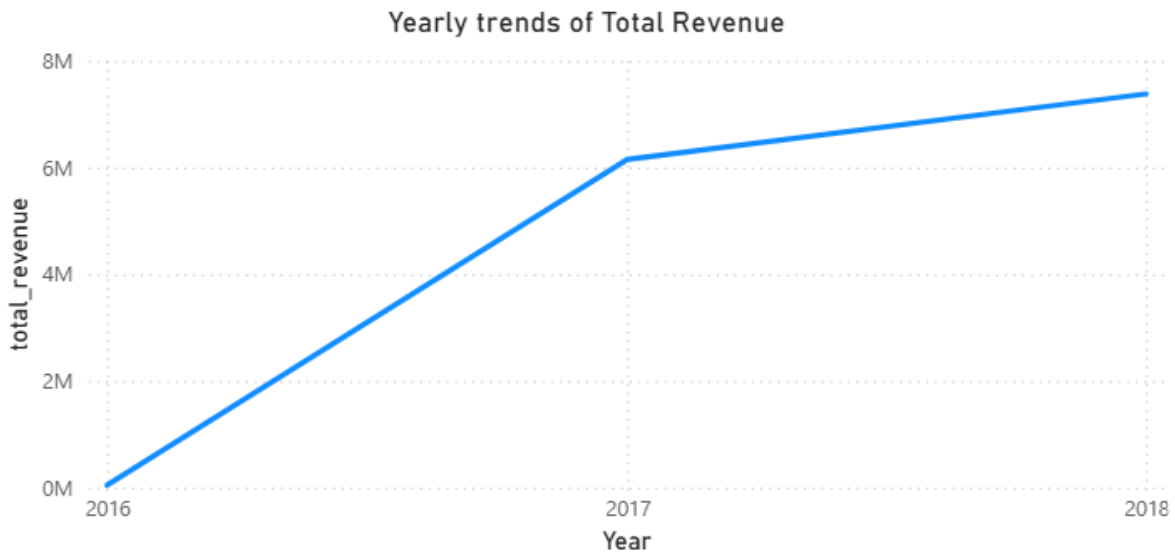
Task 8: Determine the total revenue generated by shopNest and analyze how it changes over time (yearly). Represent these visuals through suitable to highlight trends and patterns

Action: we need to identify the total revenue generated by ShopNest yearly

Tables connection: Orders (1) - order_items (many) based on order_id

Statistical and graphical analysis:

Year	Sum of price
2018	73,86,050.80
2017	61,55,806.98
2016	49,785.92
Total	**1,35,91,643.70**



Explanation:

For statistical analysis:

Matrix visual is chosen with order_purchased_date on x-axis and price on y-axis

For graphical analysis:

Line chart visual is chosen with order_purchased_date on x-axis and price on y-axis

Insights: Revenue of ShopNest peaked from 2016-2017 with increasing its revenue from 2017- 2018. This means Sales peaked in 2016-2017 and also increased from 2017-2018.