

Kafin Ann Sulaimillah, A.Md.Tr.

Portfolio



Hi, let's get to know each other!

Hi, I'm Kafin Ann Sulaimillah or you can call me Kafin. I am a graduate of D3 Telecommunication Engineering from Bandung State Polytechnic. Currently, I am pursuing expertise in the field of Data Analysis and am interested in building a career in this field.

Through this portfolio, I want to share my enthusiasm for Data Analysis. Previously I had several certificates of expertise in the field of Data Analysis and worked on several projects.

This portfolio is proof of the projects I have carried out before and my abilities in Data Analysis. Hopefully with this portfolio, we can discuss possible opportunities in the future.

Please contact me at :

Email : kafinazkiya@gmail.com

Linkedin : kafinazkiyaaa

Background

E-commerce, or electronic commerce, is a method of shopping and doing business online via the internet. Users can purchase products or services, conduct transactions, and exchange information electronically. Reviews and ratings from customers who have used a product or service are very important in e-commerce. Reviews provide information about quality and previous user satisfaction, helping potential buyers in decision making. Ratings also provide a quick picture of overall quality, influencing the reputation and long-term success of an e-commerce business. Negative ratings and comments in e-commerce provide important feedback about product or service shortcomings. This helps sellers to improve quality and build trust by responding well to negative feedback.

SMART Questions

S	M	A	R	T
What products and which cities are the targets of negative reviews?	How can we measure the level of dissatisfaction from reviews with low ratings?	Is it possible to identify common patterns in the language or phrases used in negative reviews?	What are the common themes of negative reviews that can serve as a guide for product or service improvement?	How many negative reviews have you received over the past year, and how have these complaints trended over the past year?










1. Prepare all required libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import string
from collections import Counter
from wordcloud import WordCloud
from sklearn.decomposition import PCA
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.preprocessing import Normalizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from googletrans import Translator
from pandas.tseries.offsets import MonthEnd
from datetime import datetime, timedelta
```

Data Wrangling

2. Gathering Data

The first step that needs to be taken at this stage is to prepare the dataset required for analysis, namely the dataset "E-Commerce Public Data Set " which is sourced from <https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce>, as in the image below:

Name	Date modified	Type	Size
 customers_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	8,823 KB
 geolocation_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	59,838 KB
 order_items_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	15,077 KB
 order_payments_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	5,642 KB
 order_reviews_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	14,113 KB
 orders_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	17,242 KB
 product_category_name_translation	7/23/2023 1:22 PM	Microsoft Excel Com...	3 KB
 products_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	2,324 KB
 sellers_dataset	7/23/2023 1:22 PM	Microsoft Excel Com...	171 KB

After collecting all the data from the data set source, we need to read the dataset using Python to find out what information we can get from each file that we can use for analysis and decision making, because the files we have are in CSV format, the Pandas library provides function to read files, like the one we used on one of the files:

```
# Membaca File CSV
customers_df = pd.read_csv(r".\E-Commerce Public Dataset\customers_dataset.csv")
# Menampilkan 5 teratas
customers_df.head()
```

✓ 4.6s

Then with this function we can find out the information for each file:

a. customers_df

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruze	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP

b. geoloc_df

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	1037	-23.545621	-46.639292	sao paulo	SP
1	1046	-23.546081	-46.644820	sao paulo	SP
2	1046	-23.546129	-46.642951	sao paulo	SP
3	1041	-23.544392	-46.639499	sao paulo	SP
4	1035	-23.541578	-46.641607	sao paulo	SP

c. orderitem_df

order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
2cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.90	13.90
a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03 11:05:13	239.90	13.90
da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18 14:48:30	199.00	13.90
38114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018-08-15 10:10:18	12.99	13.90
e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017-02-13 13:57:51	199.90	13.90

d. orderpayment_df

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

e. orderreviews_df

	review_id	order_id	review_score	review_comment_title
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN
1	80e641a11e56f04c1ad469d5645fdfe	a548910a1c6147796b98fdf73dbeba33	5	NaN
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	NaN
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	NaN

review_comment_message	review_creation_date	review_answer_timestamp
NaN	2018-01-18 00:00:00	2018-01-18 21:46:59
NaN	2018-03-10 00:00:00	2018-03-11 03:05:13
NaN	2018-02-17 00:00:00	2018-02-18 14:36:24
Recebi bem antes do prazo estipulado.	2017-04-21 00:00:00	2017-04-21 22:02:06
Parabéns lojas lannister adorei comprar pela l...	2018-03-01 00:00:00	2018-03-02 10:26:53

f. orders_df

	order_id	customer_id	order_status	order_purchase_timestamp
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39

order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
2017-10-02 11:07:15	2017-10-04 19:55:00	2017-10-10 21:25:13	2017-10-18 00:00:00
2018-07-26 03:24:27	2018-07-26 14:31:00	2018-08-07 15:27:45	2018-08-13 00:00:00
2018-08-08 08:55:23	2018-08-08 13:50:00	2018-08-17 18:06:29	2018-09-04 00:00:00
2017-11-18 19:45:59	2017-11-22 13:39:59	2017-12-02 00:28:42	2017-12-15 00:00:00
2018-02-13 22:20:29	2018-02-14 19:46:34	2018-02-16 18:17:02	2018-02-26 00:00:00

g. productcategory_df

	product_category_name	product_category_name_english
0	beleza_saude	health_beauty
1	informatica_acessorios	computers_accessories
2	automotivo	auto
3	cama_mesa_banho	bed_bath_table
4	moveis_decoracao	furniture_decor

h. products_df

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1.0
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1.0
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1.0
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1.0
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4.0

product_weight_g	product_length_cm	product_height_cm	product_width_cm
225.0	16.0	10.0	14.0
1000.0	30.0	18.0	20.0
154.0	18.0	9.0	15.0
371.0	26.0	4.0	26.0
625.0	20.0	17.0	13.0

i. sellers_df

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP

3. Assessing Data

Before entering the data analysis stage, at this assessing stage we must identify the problems contained in the dataset so that we can ensure quality data. In the process of assessing data we examine the dataframes one by one:

a. customers_df

- The initial stage in assessing data is assessing the dataframe information, the Pandas library has a function, namely .info(), to check the dataset information:

```
customers_df.info()
✓ 0.2s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   customer_id                          99441 non-null  object
1   customer_unique_id                   99441 non-null  object
2   customer_zip_code_prefix             99441 non-null  int64
3   customer_city                        99441 non-null  object
4   customer_state                       99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

We can judge the information produced from the dataframe as not having *missing value* In the dataframe, you can see that in the Non-Null column there is no difference in values between the columns so there is no need to check the *missing value* further, then using this information we can assess the suitability of the data type, we can see that each column has a data type that corresponds to the contents of the column.

- Inspect**Data Duplication**

```
print("Jumlah duplikasi: ", customers_df.duplicated().sum())
✓ 0.4s
Jumlah duplikasi: 0
```

The Pandas library has a function for calculating duplicate data, namely `.duplicated().sum()`, you can see the results of the duplication calculation, namely there are no duplicates in this dataframe.

- Inspect *Inaccurate Value*
Inaccurate value is a problem that arises when the value in data does not match the observation results. This problem generally arises due to human error or system error. To check *Inaccurate Value* pandas has a function to display dataframe descriptions, namely `.describe()` :

```
customers_df.describe()
✓ 0.0s
```

customer_zip_code_prefix	
count	99441.000000
mean	35137.474583
std	29797.938996
min	1003.000000
25%	11347.000000
50%	24416.000000
75%	58900.000000
max	99990.000000

It can be seen in the description above, there is no unreasonable data in the description, so the value can be said to be accurate.

- b. geoloc_df
- Checking dataset information:

```
geoloc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   geolocation_zip_code_prefix  1000163 non-null int64
1   geolocation_lat              1000163 non-null float64
2   geolocation_lng              1000163 non-null float64
3   geolocation_city             1000163 non-null object
4   geolocation_state            1000163 non-null object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
```

We can judge the information produced from the dataframe as not having *missing value* In the dataframe, you can see that in the Non-Null column there is no difference in values between the columns so there is no need to check the values *missing value* further, then using this information we can assess the suitability of the data type, we can see that each column has a data type that corresponds to the contents of the column.

- Inspect **Data Duplication**

```
print("Jumlah duplikasi: ",geoloc_df.duplicated().sum())
Jumlah duplikasi: 261831
```

You can see the results of the duplication calculation, namely 261831 duplications in this dataframe.

- Inspect *Inaccurate Value*

```
geoloc_df.describe()
```

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng
count	1.000163e+06	1.000163e+06	1.000163e+06
mean	3.657417e+04	-2.117615e+01	-4.639054e+01
std	3.054934e+04	5.715866e+00	4.269748e+00
min	1.001000e+03	-3.660537e+01	-1.014668e+02
25%	1.107500e+04	-2.360355e+01	-4.857317e+01
50%	2.653000e+04	-2.291938e+01	-4.663788e+01
75%	6.350400e+04	-1.997962e+01	-4.376771e+01
max	9.999000e+04	4.506593e+01	1.211054e+02

It can be seen in the description above, there is no unreasonable data in the description, so the value can be said to be accurate.

- c. orderreview_df

- Checking dataset information:

```
orderreviews_df.info()

Python

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   review_id             99224 non-null  object
1   order_id              99224 non-null  object
2   review_score          99224 non-null  int64
3   review_comment_title  11568 non-null  object
4   review_comment_message 40977 non-null  object
5   review_creation_date  99224 non-null  object
6   review_answer_timestamp 99224 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
```

We can assess the information generated from the dataframe. **missing value** in the dataframe, you can see that the review_comment_title and review_comment_message columns have Non-Null values which are different from the other columns, you need to check the values. **missing value** further, then on this information we can judge **data type compatibility**, it can be seen that each column has an appropriate data type and some do not match its contents, such as the review_creation_date and review_answer_timestamp columns should have a data type *datetime* not *object*.

- Checking Amount **Missing Value**

To calculate the amount **missing value** Furthermore, the Pandas library has a function namely .isna().sum() to calculate **missing value** which is found in a dataset, such as:

```
orderreviews_df.isna().sum()

review_id          0
order_id           0
review_score       0
review_comment_title 87656
review_comment_message 58247
review_creation_date 0
review_answer_timestamp 0
dtype: int64
```

It can be seen in the table description above that the review_comment_title column has a missing value of 87656 and review_comment_message has a missing value of 58247.

- Inspect **Data Duplication** and **Inaccurate Value**

```
print("Jumlah duplikasi: ", orderreviews_df.duplicated().sum())
orderreviews_df.describe()

Jumlah duplikasi: 0

review_score
count  99224.000000
mean    4.086421
std     1.347579
min     1.000000
25%     4.000000
50%     5.000000
75%     5.000000
max     5.000000
```

You can see the results of the duplication calculation, namely that there is no duplication in this dataframe and can be seen in the description above, there is no data that makes no sense from the description, so the value can be said to be accurate.

Then assessing the data is carried out further down to the last dataframe, and the following are the results of assessing the data for the entire dataset:

No	Dataframe	Missing Value	Datatype Conformity	Data Duplication	Inaccurate Value
1	customers_df	-	-	-	-
2	geoloc_df	-	-	261831	-
3	orderitem_df	-	√	-	-
4	orderpayment_df	-	-	-	-

5	orderreview_df	√	√	-	-
6	orders_df	√	√	-	-
7	productcategory_df	-	-	-	-
8	products_df	√	-	-	-
9	sellers_df	-	-	-	-

4. Cleaning Data

After finding errors in the dataset in the previous stage, in stage *cleaning* or this cleaning we clean or correct the errors found:

- a. geoloc_df
In this dataframe we find duplicate data, to delete duplicate data the Pandas Library has the .drop_duplicates function, like:

```
geoloc_df.drop_duplicates(inplace=True)

print("Jumlah duplikasi: ", geoloc_df.duplicated().sum())

Jumlah duplikasi: 0
```

It can be seen in the picture above after doing *itdrop duplicates* there is no duplication in the geoloc_df dataframe.

- b. orderitem_df
In this dataframe we find a data type mismatch in this dataframe, the Pandas library has a function to change the data type in this case we have to change the data type *todatetime* so the function used is .to_datetime like:

```
datetime_columns = ["shipping_limit_date"]

for column in datetime_columns:
    orderitem_df[column] = pd.to_datetime(orderitem_df[column])

orderitem_df.info()
✓ 1.5s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   order_id              112650 non-null object
1   order_item_id         112650 non-null int64
2   product_id           112650 non-null object
3   seller_id             112650 non-null object
4   shipping_limit_date   112650 non-null datetime64[ns]
5   price                 112650 non-null float64
6   freight_value         112650 non-null float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
```

It can be seen from the image above, after changing the data type, the data type in the shipping_limit_date column changes to datetime.

- c. orderreview_df
 - Missing Value
To deal with missing values there are several ways to deal with them, in the case of this dataframe the data has a type of qualitative data, namely comments (reviews) which are grouped quantitatively or by rating, the method that will be used in this dataframe is *imputation* or fill in the data with a certain value such as the highest value, but to avoid sampling bias or when the sample does not represent the population as a whole, for example when the highest value (mode) of the data is a positive comment while the data that must be filled in is not only in the high rating range but also the low rating Also, there will be sampling bias in negative ratings but there are positive comments in them. To maintain the credibility of the data we use multiple imputation techniques, namely by:

- Divide the dataframe into 5 parts according to each rating:

```
import pandas as pd

# Membagi dataset berdasarkan review score
review_score_1 = orderreviews_df[orderreviews_df['review_score'] == 1]
review_score_2 = orderreviews_df[orderreviews_df['review_score'] == 2]
review_score_3 = orderreviews_df[orderreviews_df['review_score'] == 3]
review_score_4 = orderreviews_df[orderreviews_df['review_score'] == 4]
review_score_5 = orderreviews_df[orderreviews_df['review_score'] == 5]
```

- Carry out the data cleaning process for each dataframe, starting from the review_score_1 dataframe:
 1. Calculating missing values in the review_score_1 dataframe:

```
review_score_1.isna().sum()

[ ] Python

... review_id      0
order_id      0
review_score   0
review_comment_title  9551
review_comment_message  2679
review_creation_date  0
review_answer_timestamp  0
dtype: int64
```

You can see that in the review_comment_title column there are 9551 missing values, so the missing values will be filled with data values that frequently appear in that column.

2. To see the contents of the data in a column and the amount, using the Pandas library we can use the .value_counts function:

```
review_score_1.review_comment_title.value_counts()

Python

review_comment_title
Não recomendo      44
Ruim                37
não recomendo      34
Não recebi o produto  30
Produto errado     30
..
Não recebi ainda    1
Comprei dois filtros...  1
Irritante           1
nao funciona telefones  1
Empres não confiável  1
Name: count, Length: 1217, dtype: int64
```

It can be seen in the image above, the data that often appears is Não recommendation with 44 data, this value will be used as the value to fill in the empty column.

3. To fill empty columns with previously obtained data, we have the .fillna function:

```
review_score_1['review_comment_title'].fillna(value='Não recomendo', inplace=True)

✓ 0.0s
```

After the column has been successfully filled in, check again whether the review_comment_title column still has missing values:

```
review_score_1.isna().sum()

✓ 0.0s

review_id      0
order_id      0
review_score   0
review_comment_title  0
review_comment_message  2679
review_creation_date  0
review_answer_timestamp  0
dtype: int64
```

The missing value in the review_comment_title column has been successfully removed.

4. It can be seen in the last image that the review_comment_message column has 2679 missing values, columns that have no content, like the previous process, missing values will be filled with data values that frequently appear in that column. To see the contents of the data in the column and the amount, use the Pandas library we can use the .value_counts function:

```
review_score_1.review_comment_message.value_counts()

✓ 0.0s
```

And generate data:

review_comment_message	..
I did not recieve the product 30	One of the products arrived with the seal broken, leaked, and the packaging was very dirty. \r\n\r\nI don't recommend it... 1
I didn't receive the product 12	Goodnight. I need a position from Baratheon on My request. Or a refund on my card since the product was not delivered. 1
I didn't receive 11	I didn't receive everything I purchased. I have already sent a complaint to Stark, but I have not yet received any response 1
I haven't received 10 yet	
I still haven't received the product 10	I canceled this order on 02/27, and until today I have not received a response.\r\nI did not receive the product and

my card was not refunded.\r\nI am very dissatisfied. 1
My product arrived and I already have to return it, because

it is defective, it does not hold the charge 1

It can be seen in the image above, the data that often appears is Não recebi product with 30 data, this value will be used as the value to fill in the empty column.

5. Fill in the empty values with previously obtained data:

```
review_score_1['review_comment_message'].fillna(value='Não recebi o produto', inplace=True)
✓ 0.0s
```

After the column has been successfully filled in, check again whether the review_comment_message column still has missing values:

```
review_score_1.isna().sum()
✓ 0.0s

review_id      0
order_id      0
review_score   0
review_comment_title  0
review_comment_message  0
review_creation_date  0
review_answer_timestamp  0
dtype: int64
```

And it can be seen in the image above that the review_score_1 dataframe has no missing values. The above processes are carried out the same for the dataframes that have not been cleaned, namely review_score_2 to 5.

- Datatype
In this dataframe we find a data type mismatch in this dataframe, the Pandas library has a function to change the data type in this case we have to change the data type to *datetime* so the function used is *.to_datetime* like:

```
datetime_columns = ["review_creation_date", "review_answer_timestamp"]

for column in datetime_columns:
    review_score_1[column] = pd.to_datetime(review_score_1[column])

for column in datetime_columns:
    review_score_2[column] = pd.to_datetime(review_score_2[column])

for column in datetime_columns:
    review_score_3[column] = pd.to_datetime(review_score_3[column])

for column in datetime_columns:
    review_score_4[column] = pd.to_datetime(review_score_4[column])

for column in datetime_columns:
    review_score_5[column] = pd.to_datetime(review_score_5[column])
```

Then after success, check the data types of these 5 dataframes again:

```
review_score_1.info()
review_score_2.info()
review_score_3.info()
review_score_4.info()
review_score_5.info()
✓ 0.3s

<class 'pandas.core.frame.DataFrame'>
Index: 11424 entries, 5 to 99223
Data columns (total 7 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   review_id                   11424 non-null  object
1   order_id                   11424 non-null  object
2   review_score                11424 non-null  int64
3   review_comment_title       11424 non-null  object
4   review_comment_message     11424 non-null  object
5   review_creation_date       11424 non-null  datetime64[ns]
6   review_answer_timestamp    11424 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(1), object(4)
```

And the review_creation_date and review_answer_timestamp columns have changed to the datetime data type.

d. orders_df

- Missing Value
In this dataframe, missing values were found in the column:

```
orders_df.isna().sum()
✓ 0.1s Python
```

order_id	0
customer_id	0
order_status	0
order_purchase_timestamp	0
order_approved_at	160
order_delivered_carrier_date	1783
order_delivered_customer_date	2965
order_estimated_delivery_date	0

dtype: int64

The order_approved_at, order_delivered_carrier_date, order_delivered_customer_date columns have the datetime data type, which is included in the continuous data category, so that if imputation is carried out directly there is no worry of sampling bias.

- 1. Look at the data that often appears in the order_approve_at column:

```
orders_df.order_approved_at.value_counts()
[136] ✓ 0.2s
```

```
order_approved_at
2018-02-27 04:31:10    9
2017-11-07 07:30:38    7
2018-02-27 04:31:01    7
2018-02-06 05:31:52    7
2017-11-07 07:30:29    7
..
2018-08-22 11:50:14    1
2017-09-22 11:27:36    1
2018-03-07 16:40:32    1
2017-08-08 10:50:15    1
2018-03-09 11:20:28    1
Name: count, Length: 90733, dtype: int64
```

It can be seen that the data that appears frequently, namely 2018-02-27 04:31:10 9 times, will be used as a reference value to fill in the empty columns.

- 2. Fill in the empty columns with the values obtained previously:

```
orders_df['order_approved_at'].fillna(value="2018-02-27 04:31:10", inplace=True)
✓ 0.0s
```

+ Code

+ Markdown

Do the same with the order_delivered_carrier_date and order_delivered_customer_date columns, then check the number of missing values in the dataframe:

```
orders_df.isna().sum()
✓ 0.0s
```

order_id	0
customer_id	0
order_status	0
order_purchase_timestamp	0
order_approved_at	0
order_delivered_carrier_date	0
order_delivered_customer_date	0
order_estimated_delivery_date	0

dtype: int64

Cleaning missing values in this dataframe was successful.

- Datatype
In this dataframe we found a data type mismatch in this dataframe, the Pandas library has a function to change the data type in this case we have to change the data type in the order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_time columns todatetime so the function used is .to_datetime like:

```
datetime_columns = ["order_purchase_timestamp", "order_approved_at", "order_delivered_carrier_date",
"order_delivered_customer_date", "order_estimated_delivery_date"]
for column in datetime_columns:
    orders_df[column]= pd.to_datetime(orders_df[column])
```

then check the data type:

```
orders_df.info()
[141] ✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   order_id                             99441 non-null  object
1   customer_id                          99441 non-null  object
2   order_status                         99441 non-null  object
3   order_purchase_timestamp             99441 non-null  datetime64[ns]
4   order_approved_at                   99441 non-null  datetime64[ns]
5   order_delivered_carrier_date         99441 non-null  datetime64[ns]
6   order_delivered_customer_date        99441 non-null  datetime64[ns]
7   order_estimated_delivery_date        99441 non-null  datetime64[ns]
dtypes: datetime64[ns](5), object(3)
```

Dan kolom order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_time sudah berubah ke tipe data datetime.

e. products_df

In this dataframe, missing value errors were also found. Apart from using the imputation technique, missing values can be overcome using the drop technique or deleting all columns that have no data or missing values. In this dataframe, the drop technique was used because the columns containing missing values do not play an important role in answering the question. business :

```
products_df.isna().sum()
✓ 0.0s

product_id          0
product_category_name    610
product_name_lenght    610
product_description_lenght  610
product_photos_qty      610
product_weight_g        2
product_length_cm       2
product_height_cm       2
product_width_cm        2
dtype: int64
```

So the .dropna technique was carried out, and the missing values were successfully resolved:

```
products_df.dropna(inplace=True)
✓ 0.0s

products_df.isna().sum()
✓ 0.0s

product_id          0
product_category_name    0
product_name_lenght    0
product_description_lenght  0
product_photos_qty      0
product_weight_g        0
product_length_cm       0
product_height_cm       0
product_width_cm        0
dtype: int64
```

Data Exploration (*Exploratory Data Analysis*)

Once the data is ready to use, it's time to explore the data to search the best answers to the SMART Questions that we compiled previously, in this data analysis we focus on negative reviews:

- 1. Pursing the orderreview_df dataframe only negative ratings
Because during data cleaning, the orderreview_df dataframe has been separated according to its rating, this time we only need to reunite the dataframes with low ratings, namely 1 and 2, combining dataframes that have the same column structure can be done directly with the .concat function which is owned by the pandas library:


```
import pandas as pd

# Gabungkan berdasarkan baris
orderreviews_lowrate_df = pd.concat([review_score_1, review_score_2], axis=0)

# Reset indeks jika diperlukan
orderreviews_lowrate_df.reset_index(drop=True, inplace=True)
orderreviews_lowrate_df.head()
```

✓ 2.8s Python

	review_id	order_id	review_score	review_comment_title	review_comment_mess
0	15197aa66ff4d0650b5434f1b46cda19	b18dcdcf73be66366873cd26c5724d1dc	1	Não recomendo	Não recebi o proc
1	373cbeecea8286a2b66c97b1b157ec46	583174fbe37d3d5f0d6661be3aad1786	1	Não chegou meu produto	Péss
2	2c5e27fc178bde7ac173c9c62c31b070	0ce9a24111d850192a933fcaab6fbad3	1	Não recomendo	Não gostei ! Comprei q por le
3	58044bca115705a48fe0e00a21390c54	68e55ca79d04a79f20d4bfc0146f4b66	1	Não recomendo	Sempre compro Internet e a entr

2. Perform data preprocessing

To find more details about negative reviews, of course we have to know the contents of the review sentences sent by customers. To process a large number of sentences per sentence, we need to do data preprocessing so that the data turns into a cleaner and more structured format, making it easier. For further analysis, the data preprocessing stages carried out this time are:

```
# Lowercasing
orderreviews_lowrate_df['review_comment_message'] = orderreviews_lowrate_df['review_comment_message'].str.lower()

# Text cleaning
def clean_text(text):
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text
orderreviews_lowrate_df['review_comment_message'] = orderreviews_lowrate_df['review_comment_message'].apply(clean_text)

# Your token
orderreviews_lowrate_df['tokens'] = orderreviews_lowrate_df['review_comment_message'].apply(word_tokenize)

# Stopword Removal
# Uses Portuguese stopwords
stop_words_portuguese = set(stopwords.words('portuguese'))
# Removed Portuguese stopwords
orderreviews_lowrate_df['tokens'] = orderreviews_lowrate_df['tokens'].apply(lambda tokens: [word for word in tokens if word not in stop_words_portuguese])

# Stemming
ps = PorterVoices()
orderreviews_lowrate_df['stemmed_tokens'] = orderreviews_lowrate_df['tokens'].apply(lambda tokens: [ps.stem(word.lower()) for word in tokens])

# Show results
orderreviews_lowrate_df.head()
```

- a. Lowercasing
To avoid two identical words being considered different due to differences in upper and lower case letters, all letters in review_comment_message are changed to lower case using the .str function provided by the pandas library.
- b. Text cleaning
To clean the review text from irrelevant or unnecessary punctuation characters, the text is cleaned by removing punctuation marks, the text becomes cleaner and easier to process and analyze using the clean_text and str.maketrans functions provided by the pandas library.
- c. Your token
To separate the text into words so that we can analyze each word separately, this process uses the word_tokenize function provided by the NLTK library.
- d. Stopword Removal
to eliminate words that are common and often appear in the text, but have little contribution to meaning in the analysis so that the focus of the analysis is more focused on words that have important meaning and a greater contribution to the interpretation and analysis of the text. Stopword removal is used using the stopwords module from the NLTK library.
- e. Stemming
Stemming is done to change the words in the token into the basic form of the word, such as running into running, this is done to help in analysis by grouping words that have the same basic word even though they have different forms.

The following are the results of Data Preprocessing:

	review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp	tokens	stemmed_tokens
0	15197aa66ff4d0650b5434f1b46cda19	b18dcdcf73be66366873cd26c5724d1dc	1	I do not recommend	I didn't receive the product	2018-04-13	2018-04-16 00:39:37	[received, product]	[received, product]
1	373cbeecea8286a2b66c97b1b157ec46	583174fbe37d3d5f0d6661be3aad1786	1	My product didn't arrive	terrible	2018-08-15	2018-08-15 04:10:37	[terrible]	[terrible]

2	2c5e27fc178bde7ac173c9c62c31b070	Oce9a2411d850192a933fcaab6fbad3	1	I do not recommend	I didn't like it, I bought it in a poke	2017-12-13	2017-12-16 07:14:07	[I liked, I bought, cat, hare]	[I liked, I bought, cat, hare]
3	58044bca115705a48fe0e00a21390c54	68e55ca79d04a79f20d4bfc0146f4b66	1	I do not recommend	I always buy online and delivery takes place...	2018-04-08	2018-04-09 12:22:39	[always, buy, internet, delivery, occurs, an...	[several, buy, internet, delivery, ocorr, ant....
4	9fd59cd04b42f600df9f25e54082a8d1	3c314f50bc654f3c4e317b055681dff9	1	I do not recommend	my order never arrived	2017-04-21	2017-04-23 05:37:03	[nothing, arrive, request]	[nothing, arrive, request]

3. N-gram Analysis

To answer section questions**Achievable** we can find out general patterns in phrases/words that contain negative review_comment_message by using N-gram analysis, we can calculate the appearance of a sequence of words or characters consisting of n consecutive elements (N-gram), in this data analysis the elements we calculate that is N=3, The following code is used to analyze trigrams:

```
# Function to get trigrams from a text
def extract_trigrams(text):
    words = text.split()
    return [f"{words[i]} {words[i+1]} {words[i+2]}" for i in range(len(words) - 2)]
# Combines all text in the review_comment_message column
all_comments = ' '.join(orderreviews_lowrate_df['review_comment_message'].dropna())
# Trigram extraction from long text
all_trigrams = extract_trigrams(all_comments)
# Count the number of occurrences of each trigram
trigram_counts = Counter(all_trigrams)
# Counts the total trigrams in the dataset
total_trigrams = len(all_trigrams)
# Calculates the percentage of each trigram
trigram_percentages = {trigram: (count / total_trigrams) * 100 for trigram, count in trigram_counts.items()}
# Sorts trigrams by highest frequency
sorted_trigrams = dict(sorted(trigram_counts.items(), key=lambda item: item[1], reverse=True))
# Displays top 10 trigrams
top_10_trigrams = dict(list(sorted_trigrams.items())[:10])
# Initialize the translator
translator = Translator()
# Function to translate trigrams into Indonesian
def translate_to_indonesian(trigram):
    try:
        # Translate trigram to Indonesian
        translated = translator.translate(trigram, src='pt', start='id')
        return translated.text
    except Exception as e:
        print(f"Translation failed for {trigram}: {str(e)}")
        return trigram
# Displays results with trigrams translated into Indonesian
print("Top 10 Trigrams (Translation to Indonesian):")
for trigram, count in top_10_trigrams.items():
    translated_trigram = translate_to_indonesian(trigram)
    print(f"{translated_trigram}: {count} times, {trigram_percentages[trigram]:.2f}%")
```

To run the code, several functions and libraries are used, the first function is extract_trigrams, which helps extract trigrams or groups of three words from each review. To manage data efficiently, the Pandas library is used, which facilitates processing and grouping data in the form of DataFrames. Finally, the translate_to_indonesian function, which is provided by the Translator library from Googletrans to translate trigrams from Portuguese to Indonesian. then generate data:

```
Top 10 Trigram (Terjemahan ke Bahasa Indonesia):
Saya menerima produk: 4609 kali, 2.27%
Saya tidak menerima: 4446 kali, 2.19%
Produk tidak: 1807 kali, 0.89%
Produk tidak menerima: 1346 kali, 0.66%
tidak dikirim: 407 kali, 0.20%
Saya belum menerimanya: 404 kali, 0.20%
produk o: 368 kali, 0.18%
produk yang dibeli: 267 kali, 0.13%
sampai sekarang: 248 kali, 0.12%
Produk Produk: 245 kali, 0.12%
```

Of the 10 highest data produced, the percentage of data still cannot represent the entire data, namely that the data has negative reviews, so further analysis must be carried out, namely using clustering.

4. Clustering Analysis

Clustering analysis is used to group similar or similar sentences based on certain features/characteristics in the review_comment_message column which have high similarities in one cluster and significant differences with other clusters, so that natural sentence patterns can be identified. In clustering analysis the TfidfVectorizer function from the library is used. sklearn.feature_extraction.text to extract features from review text using the TF-IDF method then KMeans from the sklearn.cluster library to form groups based on these features (characteristics) and finally use the fit_transform function to process the data, and value_counts to evaluate the results clustering. Here's the complete code:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X = tfidf_vectorizer.fit_transform(orderreviews_lowrate_df['review_comment_message'])

# Lakukan K-means clustering dengan inisialisasi yang stabil menggunakan random_state
num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)

# Tambahkan label kluster ke DataFrame
orderreviews_lowrate_df['cluster_label'] = kmeans.labels_

# Evaluasi hasil clustering
cluster_counts = orderreviews_lowrate_df['cluster_label'].value_counts()
total_data = len(orderreviews_lowrate_df)
cluster_percentages = (cluster_counts / total_data) * 100

# Menampilkan jumlah data dan persentase masing-masing cluster
for label, count, percentage in zip(cluster_counts.index, cluster_counts, cluster_percentages):
    print(f"Cluster Label {label}: {count} data ({percentage:.2f}%")
```

And from the code produces clustering:

```
Cluster Label 2: 9103 data (62.46%)
Cluster Label 3: 3825 data (26.24%)
Cluster Label 0: 1022 data (7.01%)
Cluster Label 1: 625 data (4.29%)
```

And the resulting cluster 2 has the highest amount of data, so it can be concluded that there are 9103 or 62.46% of the total data that have characteristics and types of reviews with the same words.

Let's look at the contents of each cluster, using the code:

```
unique_labels = orderreviews_lowrate_df['cluster_label'].value_counts()

# Tampilkan isi dari setiap cluster untuk 5 cluster teratas
num_clusters_to_display = 5

for idx, label in enumerate(unique_labels.index):
    if idx >= num_clusters_to_display:
        break

    cluster_data = orderreviews_lowrate_df[orderreviews_lowrate_df['cluster_label'] == label]['review_comment_message']
    cluster_texts = cluster_data.tolist()[:5]

    print("Cluster Label:", label)
    for idx, original_text in enumerate(cluster_texts):
        print(f"Text {idx + 1}: {original_text}")
```

With result :

Cluster Label: 2

Text 1: terrible

Text 2: I didn't like it, I bought it in a poke

Text 3: I always buy online and delivery occurs before the agreed deadline, which I believe is the maximum deadline at Stark, the maximum deadline has already passed and I still haven't received the product

Text 4: my order never arrived

Text 5: I only received 1 midea split style control missing remote control for consul air conditioning

Cluster Label: 3

Text 1: I didn't receive the product

Text 2: I didn't receive the product

Text 3: I didn't receive the product

Text 4: I didn't receive the product

Text 5: I didn't receive the product

Cluster Label: 0

Text 1: this was the request bucket with 128 pieces building blocks 2 units R 2500 each was not delivered sold and delivered targaryen

eva mat number letters 36 pieces children 1 unit r 3590 this was delivered

Text 2: I bought the product on February 25th and today, March 29th, it was not delivered to my home. I don't know if the post office in Brazil is terrible or if it was the store itself that took so long to post

Text 3: here it is described as delivered but until now I have not received it

Text 4: product was delivered with one of the handles having problems, 1 fixing pin is missing

Text 5: of the two products purchased, only one was delivered

Cluster Label: 1

Text 1: I haven't received it yet

Text 2: I've been waiting for more than twenty days and I still haven't received my product and I can't get any information

Text 3: the product has not yet arrived

Text 4: I haven't received the product yet

Text 5: I haven't received it yet

Because the results still use Portuguese, we need to translate them into Indonesian using the translator function provided by the GoogleTrans library, here are the translation results:

Cluster Label: 2

Text 1 (Translated): very bad

Text 2 (Translated): I don't like that I bought a cat for a rabbit

Text 3 (Translated): Always buy over the internet and delivery occurs before the deadline which I believe is the maximum deadline within the maximum deadline have sold out and have not received the product

Text 4 (Translated): There is no request for my order

Text 5 (Translated): I only received 1 midea split style control does not have a remote control for the consul air conditioner

```
=====
Cluster Label: 3
Text 1 (Translated): I did not receive the product
Text 2 (Translated): I did not receive the product
Text 3 (Translated): I did not receive the product
Text 4 (Translated): I did not receive the product
Text 5 (Translated): I did not receive the product
=====
Cluster Label: 0
Text 1 (Translated): This is a bucket order with 128 pieces assembly 2 un r 2500 each not delivered and delivered targaryen carpet eva n° letters 36 pieces children 1 un r 3590 this was delivered
Text 2 (Translated): I bought the product on February 25 and today Marco 29 was not delivered to my residence, I don't know if this is the Brazilian post office and it's bad or the shop itself takes the post
Text 3 (Translated): here describes only conveyed as far as I have not received
Text 4 (Translated): The product was delivered with one of the handles with the problem of missing 1 pin
Text 5 (Translated): of the two products purchased, only one was sent
=====
Cluster Label: 1
Text 1 (Translated): I haven't received it yet
Text 2 (Translated): I have been waiting for more than twenty days and I have not received my product and I cannot get any information
Text 3 (Translated): The product has not arrived
Text 4 (Translated): I have not received the product
Text 5 (Translated): I haven't received it yet
=====
```

5. Time Series Analysis

To answer questions**Time-Bond** We need to analyze the number of review_comment_message columns from time to time over the last 1 year to be able to determine review trends. To calculate the number of reviews in the last year, of course we use several functions, namely the .max() Python function which is used to get the latest date in the dataframe, then use the timedelta() function from the datetime module to calculate the date of the last 1 year and use the resample function provided by the pandas library is used to group months, here is the complete code:

```
# Ambil tanggal terakhir dalam dataset
last_date_in_dataset = orderreviews_lowrate_df['review_creation_date'].max()

# Hitung tanggal 1 tahun sebelum tanggal terakhir
one_year_ago = last_date_in_dataset - timedelta(days=365)

# Filter data hanya untuk satu tahun terakhir
data_last_year = orderreviews_lowrate_df[orderreviews_lowrate_df['review_creation_date'] >= one_year_ago]

# Hitung jumlah ulasan per bulan
review_counts_per_month = data_last_year.resample('M', on='review_creation_date').size()
print(review_counts_per_month)
```

Then generate data:

```
review_creation_date
2017-08-31      20
2017-09-30     521
2017-10-31     540
2017-11-30     640
2017-12-31    1478
2018-01-31     926
2018-02-28     981
2018-03-31    1832
2018-04-30    1392
2018-05-31     903
2018-06-30     823
2018-07-31     583
2018-08-31    1092
Freq: M, dtype: int64
```

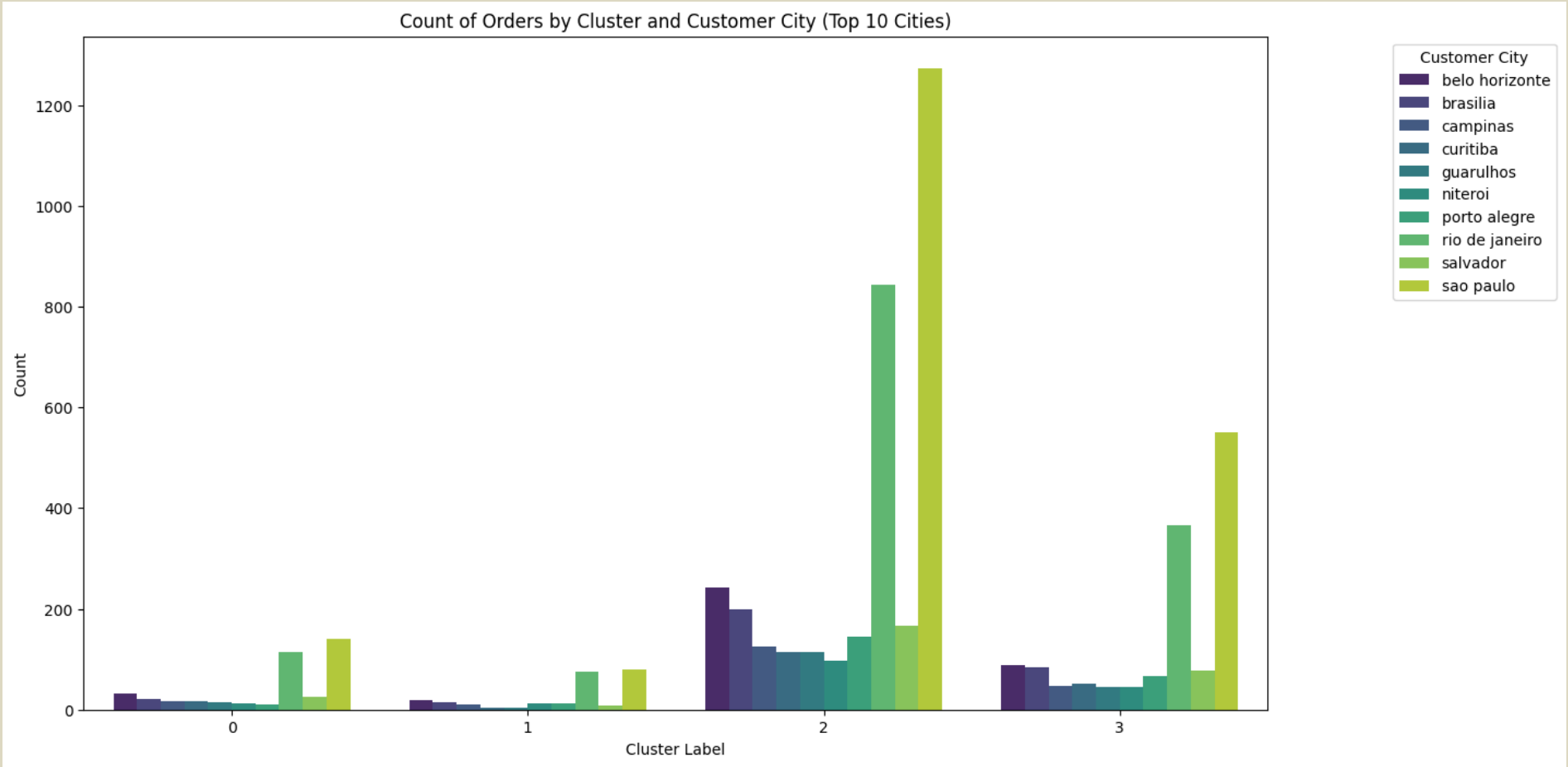
Data Visualization and Conclusions

- 1. **Specifics:**What products and which cities are the targets of negative reviews?

To find the cities with the most customers to which deliveries are sent, we need to create a graph by comparing the cluster label column with the customer city column in the dataframe using the matplotlib library, as follows:

```
# Ambil 10 kota terbanyak
top_10_cities = city_df['customer_city'].value_counts().nlargest(10).index
# Filter DataFrame hanya untuk 10 kota teratas
top_10_cities_df = city_df[city_df['customer_city'].isin(top_10_cities)]
# Group by 'cluster_label' and 'customer_city', then count the occurrences
grouped_data = top_10_cities_df.groupby(['cluster_label', 'customer_city']).size().reset_index(name='count')
# Create the plot
plt.figure(figsize=(14, 8))
sns.barplot(data=grouped_data, x='cluster_label', y='count', hue='customer_city', palette='viridis')
# Add title and labels
plt.title('Count of Orders by Cluster and Customer City (Top 10 Cities)')
plt.xlabel('Cluster Label')
plt.ylabel('Count')
# Show legend
plt.legend(title='Customer City', loc='upper right', bbox_to_anchor=(1.25, 1))
plt.show()
```

And generate a graph:



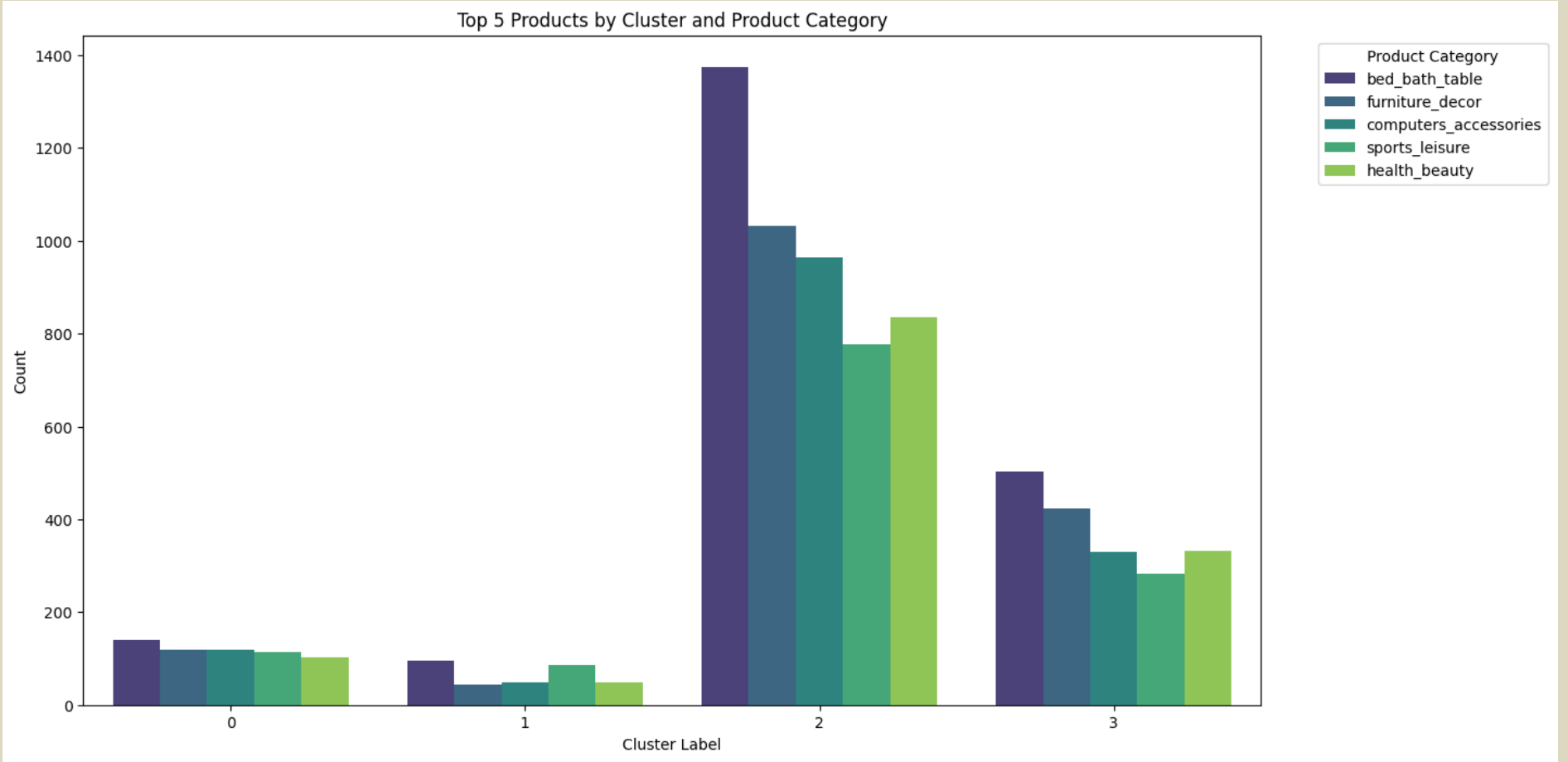
From the resulting graph, the highest customer city is the city of Sao Paulo, and the second highest city is the City of Rio de Janeiro in the four clusters.

To find the most products purchased by customers, we need to create a graph by comparing the cluster label column with the product_category column in the dataframe using the matplotlib library, as follows:

```
# Function to get top n products in each cluster
def get_top_products(cluster_data, n=5):
    top_products = cluster_data.sort_values(by='count', ascending=False).head(n)
    return top_products

# Group by 'cluster_label' and 'product_category_name_english', then count the occurrences
grouped_data = all_df.groupby(['cluster_label', 'product_category_name_english']).size().reset_index(name='count')
# Get the top 5 products in each cluster
top_products_by_cluster = grouped_data.groupby('cluster_label', group_keys=False).apply(get_top_products, n=5)
# Create the plot
plt.figure(figsize=(14, 8))
sns.barplot(data=top_products_by_cluster, x='cluster_label', and='count', hue='product_category_name_english', palette='viridis')
# Add title and labels
plt.title('Top 5 Products by Cluster and Product Category')
plt.xlabel('Cluster Label')
plt.ylabel('Count')
# Show legend
plt.legend(title='Product Category', place='upper right', bbox_to_anchor=(1.25, 1))
plt.show()
```

The code produces a graph:



From the resulting graph, the highest product category purchased by customers is the bed_bath_table category.

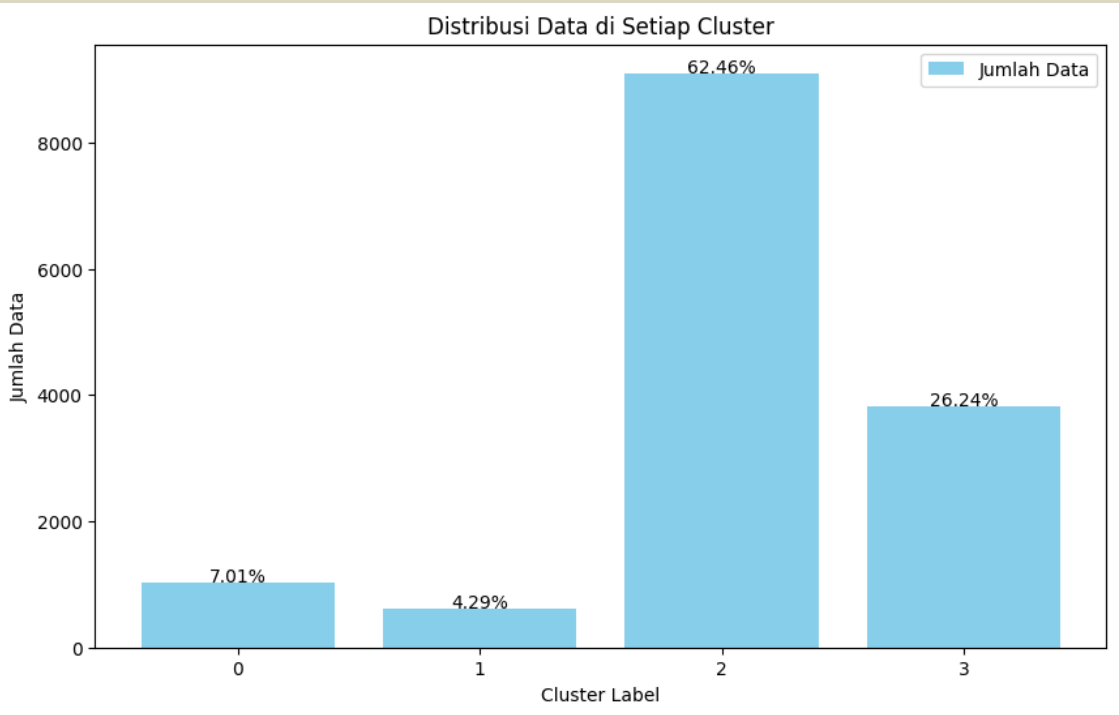
2. **Measurable** : How can we measure the level of dissatisfaction from reviews with low ratings?

Measuring dissatisfaction from reviews is by looking at word patterns and grouping similar word patterns into one group or called clustering. The clustering used in this dataset is kmeans clustering, then visualized using matplotlib to see the data distribution for each cluster, with the following code:

```
# Hitung jumlah data di setiap cluster
cluster_counts = orderreviews_lowrate_df['cluster_label'].value_counts()
# Hitung total data
total_data = len(orderreviews_lowrate_df)
# Hitung persentase jumlah data di setiap cluster
cluster_percentages = (cluster_counts / total_data) * 100
# Plot grafik batang dengan presentase
plt.figure(figsize=(10, 6))
bars = plt.bar(cluster_counts.index, cluster_counts.values, color='skyblue', label='Jumlah Data')
plt.xlabel('Cluster Label')
plt.ylabel('Jumlah Data')
plt.title('Distribusi Data di Setiap Cluster')
plt.xticks(cluster_counts.index)
# Tambahkan label persentase di atas setiap bar
for bar, percentage in zip(bars, cluster_percentages):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 10, f'{percentage:.2f}%', ha='center')
plt.legend()
plt.show()
```

And generate a graph:

From the graph above, cluster 2 is produced which has the highest of data, so it can be concluded there are 9103 or 62.46% of the data that have characteristics and reviews with the same words, content of the reviews for each clustering:



amount that total types of with the

Cluster Label: 2
Text 1 (Translated): very bad
Text 2 (Translated): I don't like that I bought a cat
Text 3 (Translated): Always buy over the internet occurs before the deadline which I believe is the deadline within the maximum deadline have sold out and have not received the product
Text 4 (Translated): There is no request for my order
Text 5 (Translated): I only receive 1 midea split style control does not have a remote control for the consul air conditioner
=====
Cluster Label: 3
Text 1 (Translated): I did not receive the product
Text 2 (Translated): I did not receive the product
Text 3 (Translated): I did not receive the product
Text 4 (Translated): I did not receive the product
Text 5 (Translated): I did not receive the product
=====
Cluster Label: 0
Text 1 (Translated): This is a bucket order with 128 pieces assembly 2 un r 2500 each not delivered and delivered targaryen carpet eva n° letters 36 pieces children 1 un r 3590 this was delivered
Text 2 (Translated): I bought the product on February 25 and today Marco 29 was not delivered to my residence, I don't know if this is the Brazilian post office and it's bad or the shop itself takes the post
Text 3 (Translated): here describes only conveyed as far as I have not received
Text 4 (Translated): The product was delivered with one of the handles with the problem of missing 1 pin
Text 5 (Translated): of the two products purchased, only one was sent
=====
Cluster Label: 1
Text 1 (Translated): I haven't received it yet
Text 2 (Translated): I have been waiting for more than twenty days and I have not received my product and I cannot get any information
Text 3 (Translated): The product has not arrived
Text 4 (Translated): I have not received the product
Text 5 (Translated): I haven't received it yet
=====

for a rabbit and delivery maximum

From the reviews that have been clustered according to the similarity of words, and we can conclude that: Clusters 1, 2, and 3 have the topic or core of the review that the customer did not receive the product, and cluster 0 is a customer with other negative reviews such as incomplete products and so on, so we can conclude that**92.99% of reviews are negative**has a topic, namely products that are not accepted by customers.

3. **Achievable:** Is it possible to identify common patterns and similarities in the language or phrases used in negative reviews?

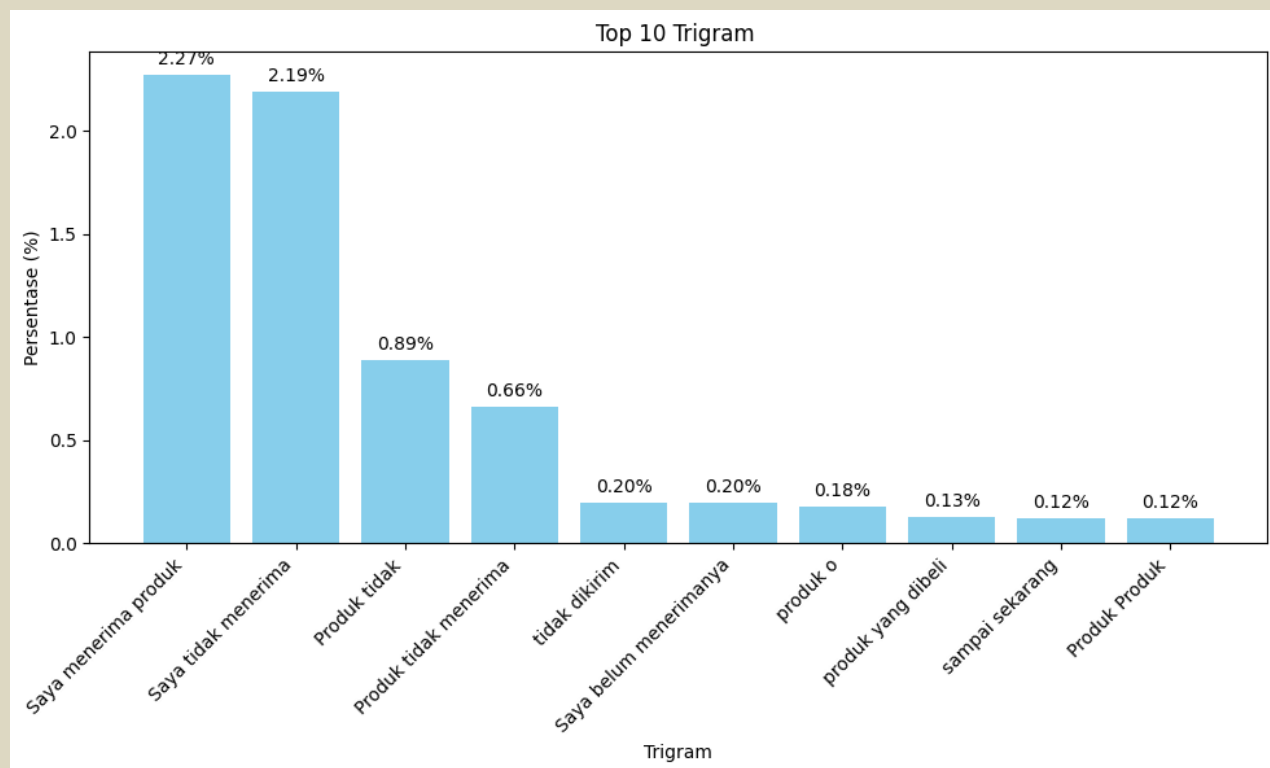
Identifying general patterns and similarities in language or phrases is very possible with N-Gram Analysis, and has been visualized using matplotlib with the code:

```

import matplotlib.pyplot as plt
# Data
trigrams = [
    "I received the product",
    "I do not accept",
    "Product no",
    "Product not received",
    "not sent",
    "I haven't received it yet",
    "product o",
    "purchased product",
    "until now",
    "Product Products"
]
counts = [4609, 4446, 1807, 1346, 407, 404, 368, 267, 248, 245]
percentages = [2.27, 2.19, 0.89, 0.66, 0.20, 0.20, 0.18, 0.13, 0.12, 0.12]
# Plot
fig, ax = plt.subplots(figsize=(10, 6))
ax.bar(trigrams, percentages, color='skyblue')
ax.set_ylabel('Percentage (%)')
ax.set_xlabel('Trigram')
ax.set_title('Top 10 Trigram ')
# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')
# Display the percentages on top of the bars
for i in range(len(trigrams)):
    ax.text(i, percentages[i] + 0.05, f'{percentages[i]:.2f}%', ha='center')
plt.tight_layout()
plt.show()

```

and generate a graph:



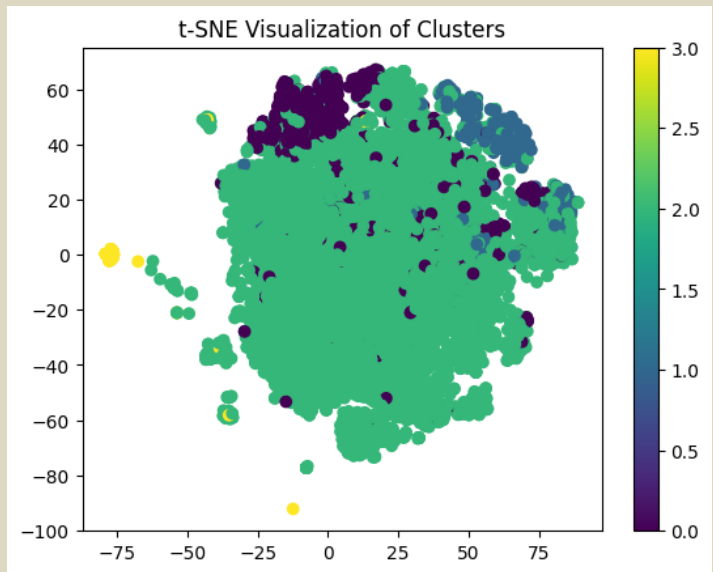
From the Trigram that has been created a sequence of words or characters consisting of 3 consecutive elements and 10 patterns/phrases were obtained from reviews that were often mentioned, but did not represent the entire data so clustering was carried out and 4 clusters were obtained for similar combinations of patterns/phrases and visualized using t-SNE graphs with the matplotlib library, as follows:

```

# Reduksi dimensi menggunakan TruncatedSVD (untuk mengubah matriks sparse menjadi dense)
# dan kemudian menggunakan t-SNE pada matriks dense hasil SVD
tsvd = TruncatedSVD(n_components=50) # Ubah n_components sesuai kebutuhan
normalizer = Normalizer(copy=False)
tsne = TSNE(n_components=2, random_state=42)
# Pipeline untuk menggabungkan proses reduksi dimensi
pipeline = Pipeline([
    ('svd', tsvd),
    ('norm', normalizer),
    ('tsne', tsne)
])
# Perform t-SNE on the TF-IDF data
tsne_result = pipeline.fit_transform(X.toarray())
# Get the cluster labels
cluster_labels = orderreviews_lowrate_df['cluster_label']
# Plot hasil clustering menggunakan t-SNE
plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=cluster_labels, cmap='viridis')
plt.title('t-SNE Visualization of Clusters')
plt.colorbar()
plt.show()

```

And generate a graph:



It can be seen from the density and difference in color on the graph, the dots are very close and densely spaced, this shows that the sentence/phrase/word patterns from the review have high similarity, and can be seen in green or showing that cluster two dominates as stated in previous question.

4. **Relevant** : What are the common themes of negative reviews that can serve as a guide for product or service improvement?

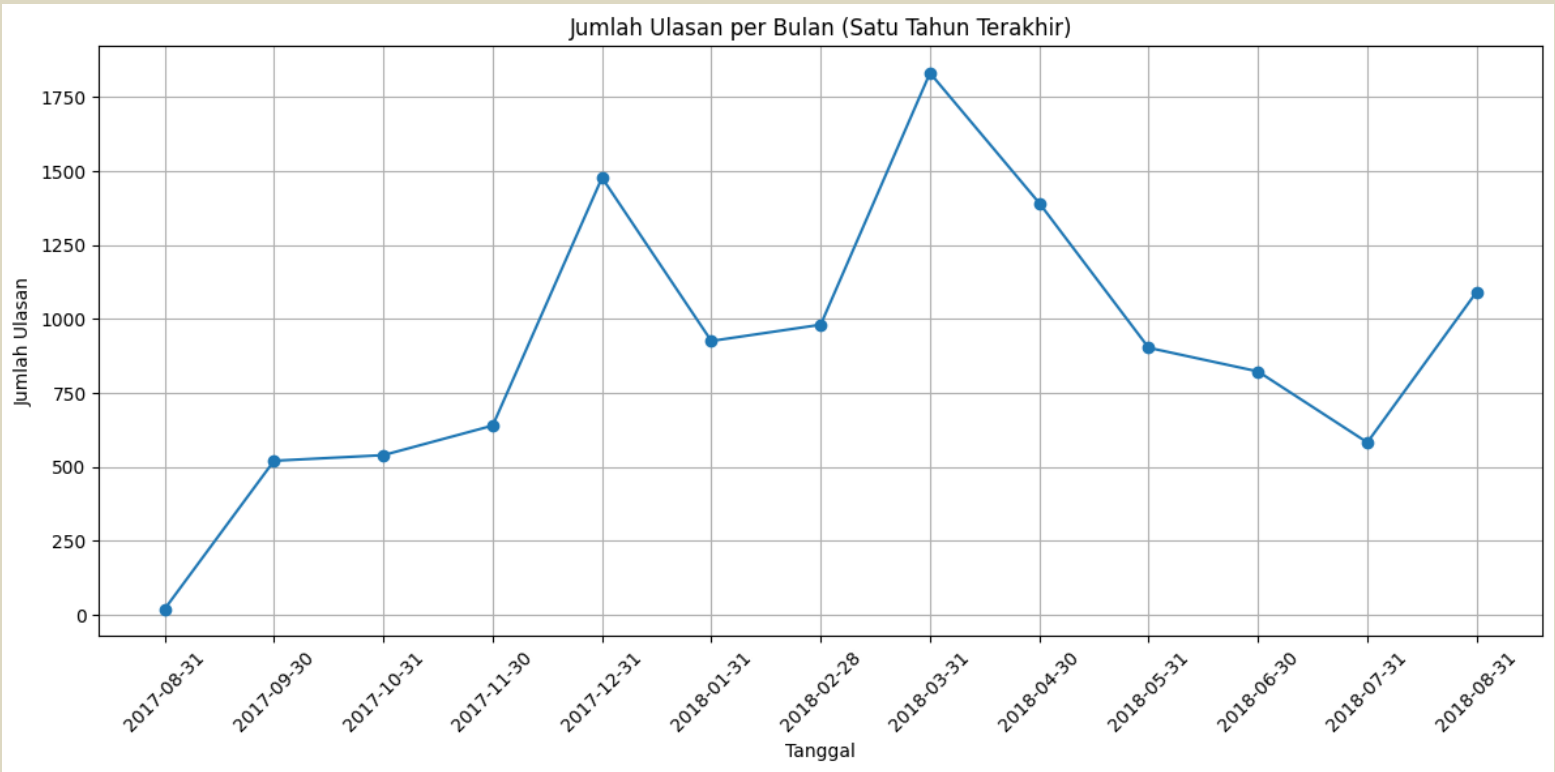
From question number 2, the general theme or topic of the negative reviews is concluded **the product is not accepted by the customer**, This can be used as a guide for improving services so that reviews can be good in the following year.

5. **Time-Bound** : How many negative reviews have you received over the past year, and how have these complaints trended over the past year?

The results of Time Series Analysis are visualized using matplotlib, through the code:

```
# Ambil tanggal terakhir dalam dataset
last_date_in_dataset = orderreviews_lowrate_df['review_creation_date'].max()
# Hitung tanggal 1 tahun sebelum tanggal terakhir
one_year_ago = last_date_in_dataset - timedelta(days=365)
# Filter data hanya untuk satu tahun terakhir
data_last_year = orderreviews_lowrate_df[orderreviews_lowrate_df['review_creation_date'] >= one_year_ago]
# Hitung jumlah ulasan per bulan
review_counts_per_month = data_last_year.resample('M', on='review_creation_date').size()
# Buat urutan tanggal dengan frekuensi M (akhir setiap bulan)
date_sequence = pd.date_range(start=one_year_ago, periods=len(review_counts_per_month), freq=MonthEnd())
# Ambil tanggal dalam format tahun-bulan-hari
formatted_dates = [date.strftime('%Y-%m-%d') for date in date_sequence]
# Plot time series
plt.figure(figsize=(12, 6))
plt.plot(formatted_dates, review_counts_per_month.values, marker='o')
plt.xlabel('Tanggal')
plt.ylabel('Jumlah Ulasan')
plt.title('Jumlah Ulasan per Bulan (Satu Tahun Terakhir)')
plt.grid(True)
plt.xticks(rotation=45) # Rotasi label tanggal sebesar 45 derajat
plt.tight_layout()
plt.show()
```

And generate a graph:



From the data we obtained over the past year, it can be seen in the table, during the period from August 2017 to August 2018, we can see quite significant variations in the number of reviews submitted. There was a notable spike in December 2017, with the number of reviews reaching its highest peak of 1478 reviews. However, after that, there was a decline in January 2018, although there was a slight increase in February 2018, then in March 2018, the number of reviews again increased markedly, even surpassing the previous highest number after December 2017, namely 1832 reviews. After March 2018, there was a continuous decline until June 2018. However, July 2018 saw an increase in the number of reviews again. Overall, despite irregular monthly

data variations, it can be seen that December 2017 and March 2018 were the months where the number of reviews reached their highest peak during the period.

Overall conclusion:

92.99% of negative reviews have a topic, namely products that are not accepted by customers with the majority of the recipient city being Sao Paulo and the majority of the product categories purchased are bed_bath_table and in the last 1 year the reviews have not had a regular number every month and the highest peak was reached in December 2017 and March 2018.

Thank You! You can see the entire code at the link: