

Implementasi Koding Data Mining

Market basket Analysis



Disusun Oleh

| | |
|--------------------------------|---------------------|
| Giraldo Stevanus | 220441100064 |
| Abib Maulana Aan Nafudu | 220441100118 |
| Wisnu Ary Swadana | 220441100121 |
| Fairuz Abdullah | 220441100070 |

Program Studi Sistem Informasi
Fakultas Teknik
Universitas Trunojoyo Madura
Tahun Ajaran 2024 /2025

HASIL DAN PEMBAHASAN

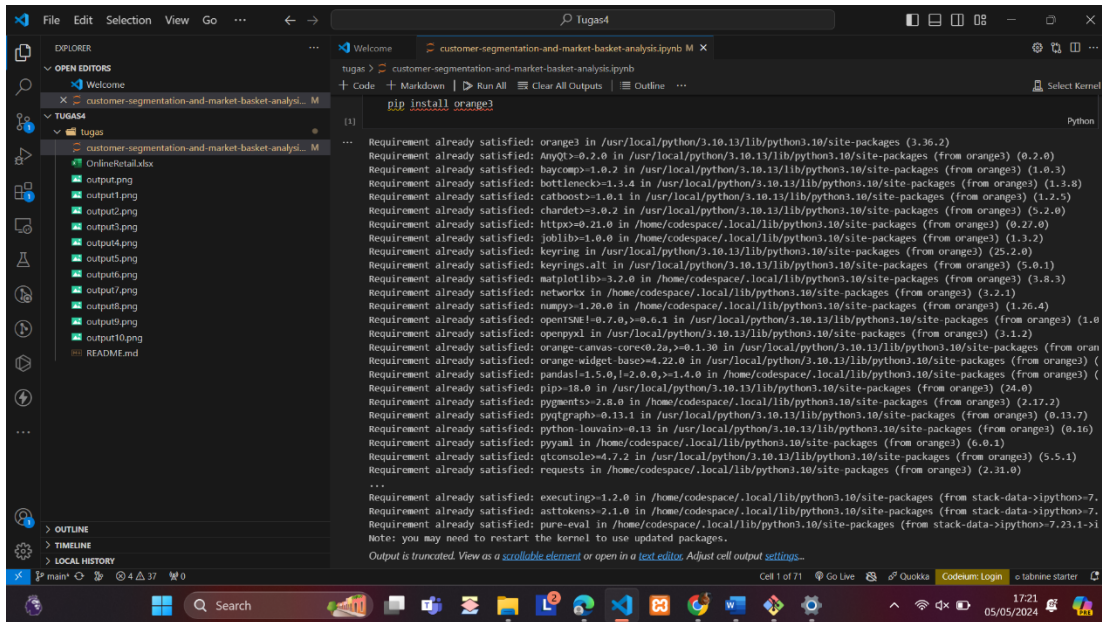
4.1 Implementasi

Analisis segmentasi pelanggan dan analisis keranjang belanja pasar merupakan dua teknik penting dalam bidang data mining yang digunakan untuk mendapatkan pemahaman yang lebih dalam tentang perilaku pelanggan dan pola pembelian mereka. Tujuan dari analisis segmentasi pelanggan adalah memahami dan mengelompokkan pelanggan berdasarkan karakteristik, preferensi, dan perilaku pembelian mereka. Ini dilakukan dengan menggunakan teknik pembelajaran mesin, seperti clustering atau segmentasi pelanggan, untuk mengidentifikasi kelompok pelanggan yang memiliki pola pembelian serupa. Manfaat dari analisis segmentasi pelanggan termasuk kemampuan perusahaan untuk menyesuaikan strategi pemasaran, harga, dan layanan kepada setiap segmen pelanggan secara lebih efektif, serta membantu dalam pengambilan keputusan terkait pengembangan produk dan strategi promosi.

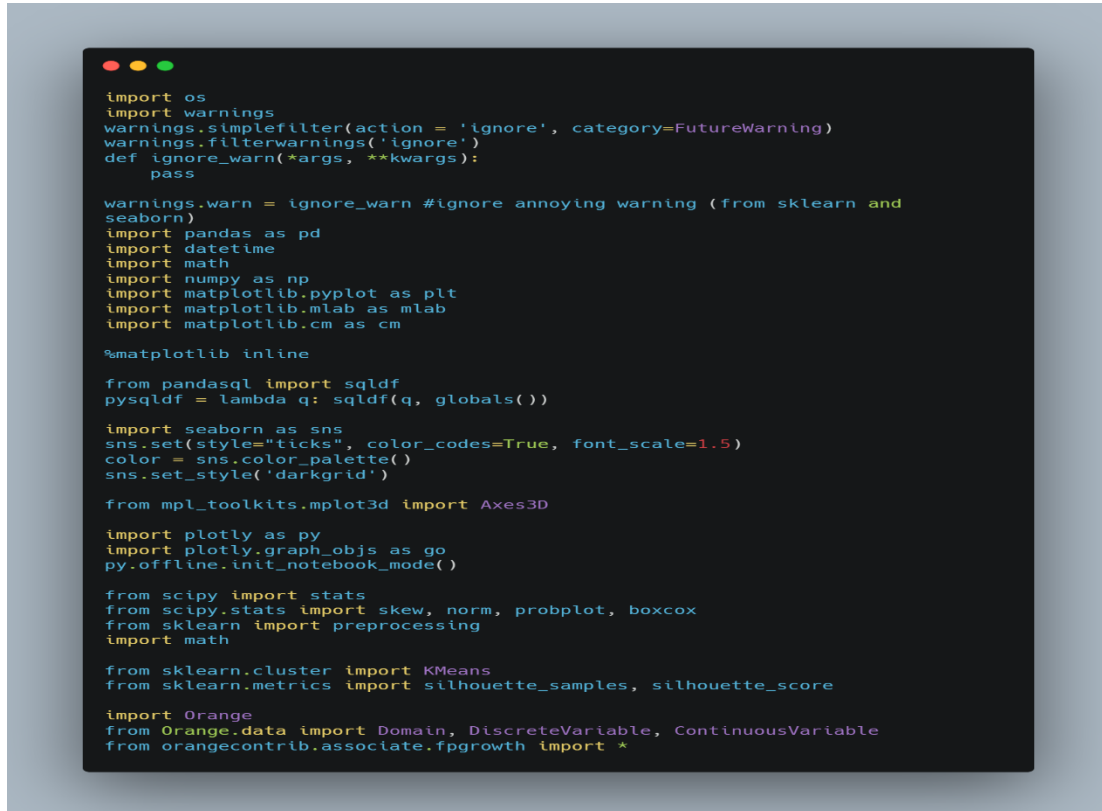
Sementara itu, tujuan dari analisis keranjang belanja pasar adalah mengidentifikasi pola dan asosiasi antara item yang dibeli secara bersamaan oleh pelanggan. Metodenya melibatkan penggunaan algoritma asosiasi seperti Apriori atau FP-Growth untuk menemukan aturan asosiasi yang mengungkap pola pembelian bersama yang signifikan. Manfaat dari analisis keranjang belanja pasar termasuk kemampuan perusahaan untuk membuat strategi penjualan silang dan penjualan tambahan yang lebih efektif. Dengan memahami pola pembelian, perusahaan dapat menyesuaikan penempatan produk, promosi, dan bundel produk yang lebih baik untuk meningkatkan penjualan dan kepuasan pelanggan. Kedua teknik ini merupakan bagian penting dari analisis data untuk bisnis e-commerce dan ritel, membantu perusahaan untuk meningkatkan pemahaman mereka tentang pelanggan dan meningkatkan kinerja bisnis mereka melalui pengambilan keputusan yang lebih terinformasi.

1. Ketergantungan Beban dan Pengaturan Konfigurasi

Perintah `!conda install -y orange3` adalah perintah yang digunakan di lingkungan seperti Jupyter Notebook atau Google Colab untuk menginstal paket Python bernama



The screenshot shows a Jupyter Notebook window titled 'Tugas4'. The code cell contains the command `pip install orange3`. The output shows a list of requirements that are already satisfied, including `orange3` (3.36.2) and various dependencies like `AnyQt`, `baycomp`, `bottleneck`, `catboost`, `chardet`, `httpx`, `joblib`, `keyring`, `keyrings.alt`, `matplotlib`, `networkx`, `numpy`, `openpyxl`, `orange-canvas-core`, `orange-widget-base`, `pandas`, `pip`, `pygments`, `pyqtgraph`, `python-louvain`, `pyyaml`, `qtconsole`, `requests`, `stack-data`, `asttokens`, `pure-eval`, and `requests`. The output is truncated at the bottom.



```
import os
import warnings
warnings.filterwarnings(action = 'ignore', category=FutureWarning)
warnings.filterwarnings('ignore')
def ignore_warn(*args, **kwargs):
    pass

warnings.warn = ignore_warn #ignore annoying warning (from sklearn and seaborn)
import pandas as pd
import datetime
import math
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib.cm as cm

%matplotlib inline

from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())

import seaborn as sns
sns.set(style="ticks", color_codes=True, font_scale=1.5)
color = sns.color_palette()
sns.set_style('darkgrid')

from mpl_toolkits.mplot3d import Axes3D

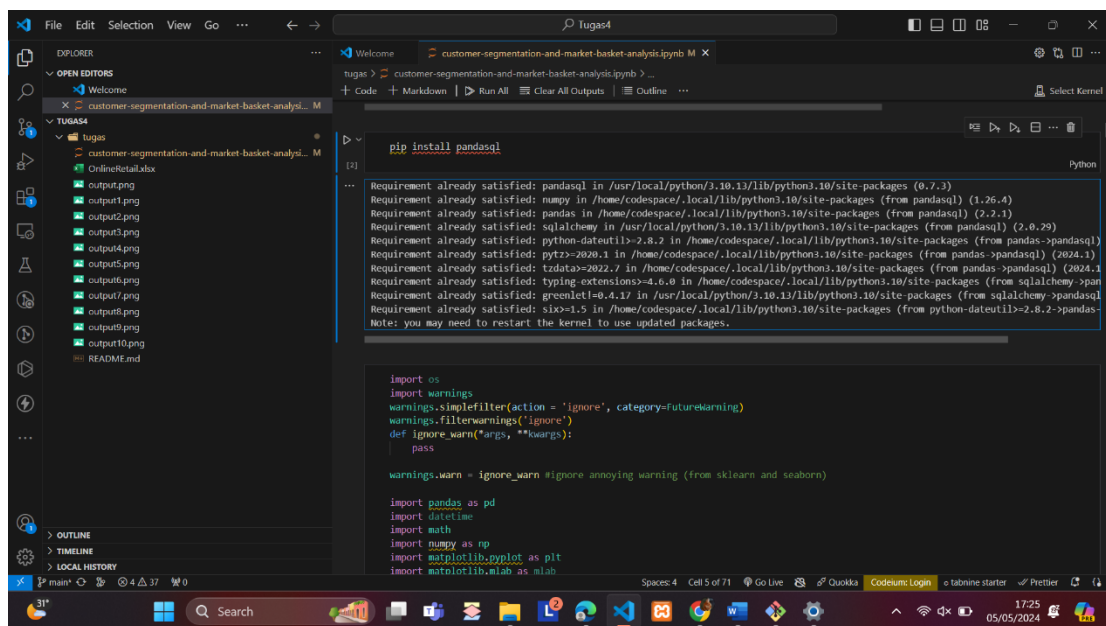
import plotly as py
import plotly.graph_objs as go
py.offline.init_notebook_mode()

from scipy import stats
from scipy.stats import skew, norm, probplot, boxcox
from sklearn import preprocessing
import math

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

import Orange
from Orange.data import Domain, DiscreteVariable, ContinuousVariable
from orangecontrib.associate.fpgrowth import *
```

Orange3 melalui Anaconda, sebuah manajer paket yang mengelola instalasi paket dan lingkungan Python. Pada dasarnya, perintah ini melakukan instalasi paket dengan menggunakan Anaconda. Opsi -y digunakan untuk memberikan persetujuan otomatis terhadap semua prompt yang mungkin muncul selama proses instalasi, sehingga instalasi berjalan tanpa interaksi pengguna yang diperlukan. Dengan demikian, perintah tersebut akan menginstal paket Python Orange3 secara otomatis dan mengkonfirmasi setiap prompt yang muncul selama proses instalasi.



```

pip install pandasql

Requirement already satisfied: pandasql in /usr/local/python3.10.13/lib/python3.10/site-packages (0.7.3)
Requirement already satisfied: numpy in /home/codespace/.local/lib/python3.10/site-packages (from pandasql) (1.26.4)
Requirement already satisfied: pandas in /home/codespace/.local/lib/python3.10/site-packages (from pandasql) (2.2.1)
Requirement already satisfied: sqlalchemy in /usr/local/python3.10.13/lib/python3.10/site-packages (from pandasql) (2.0.29)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/codespace/.local/lib/python3.10/site-packages (from pandas->pandasql)
Requirement already satisfied: pytz>=2020.1 in /home/codespace/.local/lib/python3.10/site-packages (from pandas->pandasql) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/codespace/.local/lib/python3.10/site-packages (from pandas->pandasql) (2024.1)
Requirement already satisfied: typing-extensions>=4.6.0 in /home/codespace/.local/lib/python3.10/site-packages (from sqlalchemy->pandasql)
Requirement already satisfied: greenlet<=0.4.17 in /usr/local/python3.10.13/lib/python3.10/site-packages (from sqlalchemy->pandasql)
Requirement already satisfied: six>=1.5 in /home/codespace/.local/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandasql)
Note: you may need to restart the kernel to use updated packages.

import os
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.filterwarnings('ignore')
def ignore_warn(*args, **kwargs):
    pass

warnings.warn = ignore_warn #ignore annoying warning (from sklearn and seaborn)

import pandas as pd
import datetime
import math
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab

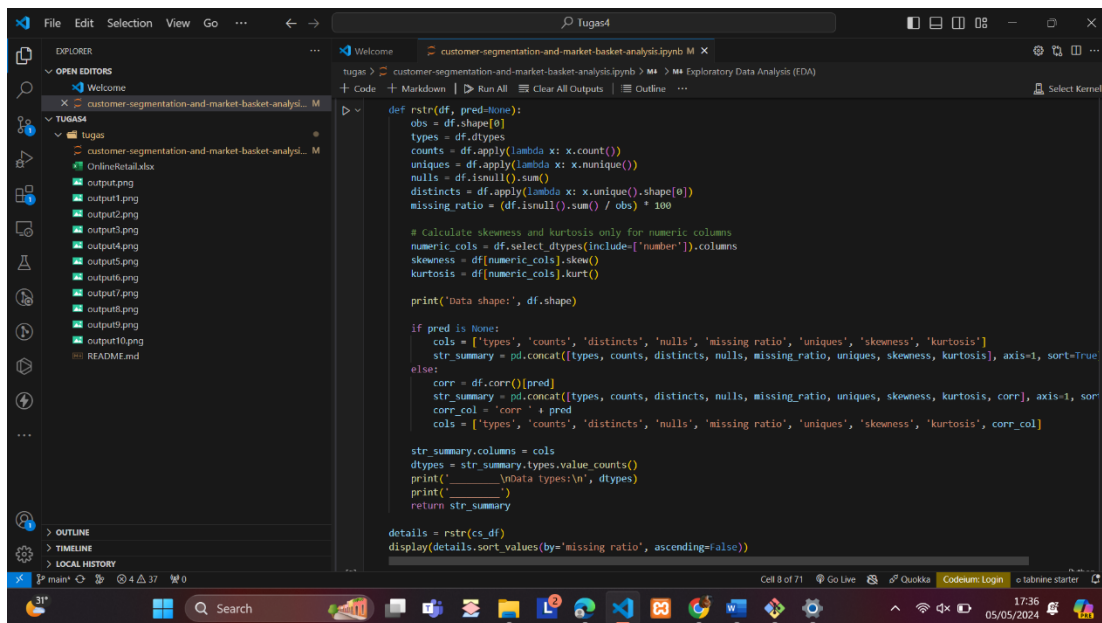
```

```
cs_df = pd.read_excel(io=r'/workspaces/tugas/OnlineRetail.xlsx')
```

Data alur dari syntax yang ada diatas merupakan awalan dari langkah untuk membuat konfigurasi data dalam mengolahnya.

2. Memberi *Title* Pada Sistem

Analisis Data Eksplorasi (Exploratory Data Analysis/EDA) adalah tahap awal dalam proses analisis data yang dilakukan untuk memahami karakteristik dan struktur dataset. Tujuannya adalah untuk mendapatkan wawasan awal tentang data, mengidentifikasi pola atau anomali yang menarik, serta merencanakan langkah-langkah analisis selanjutnya. Proses EDA melibatkan penggunaan berbagai teknik statistik dan visualisasi data, seperti ringkasan statistik, grafik univariat, dan grafik bivariat. Selain itu, EDA juga mencakup pembersihan data untuk menangani nilai yang hilang atau outlier. Melalui EDA, analis data dapat membuat asumsi awal, mengevaluasi kebutuhan preprocessing data, dan memperoleh pemahaman yang lebih baik tentang dataset yang akan digunakan dalam analisis lebih lanjut. Ini merupakan tahap kritis dalam siklus analisis data yang membantu dalam pembuatan keputusan yang lebih baik dan pemahaman yang lebih dalam tentang data.



```
def rstr(df, pred=None):
    obs = df.shape[0]
    types = df.dtypes
    counts = df.apply(lambda x: x.count())
    uniques = df.apply(lambda x: x.unique())
    nulls = df.isnull().sum()
    distincts = df.apply(lambda x: x.unique().shape[0])
    missing_ratio = (df.isnull().sum() / obs) * 100

    # Calculate skewness and kurtosis only for numeric columns
    numeric_cols = df.select_dtypes(include=['number']).columns
    skewness = df[numeric_cols].skew()
    kurtosis = df[numeric_cols].kurt()

    print("Data shape:", df.shape)

    if pred is None:
        cols = ['types', 'counts', 'distincts', 'nulls', 'missing_ratio', 'uniques', 'skewness', 'kurtosis']
        str_summary = pd.concat([types, counts, distincts, nulls, missing_ratio, uniques, skewness, kurtosis], axis=1, sort=True)
    else:
        corr = df.corr()[pred]
        str_summary = pd.concat([types, counts, distincts, nulls, missing_ratio, uniques, skewness, kurtosis, corr], axis=1, sort=True)
        corr_col = 'corr' + pred
        cols = ['types', 'counts', 'distincts', 'nulls', 'missing_ratio', 'uniques', 'skewness', 'kurtosis', corr_col]

    str_summary.columns = cols
    dtypes = str_summary.dtypes.value_counts()
    print("\nData types:\n", dtypes)
    print("\n")
    return str_summary

details = rstr(cs_df)
display(details.sort_values(by='missing_ratio', ascending=False))
```

Data shape: (541909, 8)

Data types:

types

object 4

float64 2

datetime64[ns] 1

int64 1

Name: count, dtype: int64

The screenshot shows a Jupyter Notebook window with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'TUGAS4' with a sub-project 'customer-segmentation-and-market-basket-analysis'. The code editor displays a table with the following columns: types, counts, distincts, nulls, missing ratio, uniques, skewness, and kurtosis. The table contains data for various columns: CustomerID (float64, 406829, 4373, 135080, 24.926694, 4372, 0.029835, -1.179982), Description (object, 540455, 4224, 1454, 0.268311, 4223, NaN, NaN), Country (object, 541909, 38, 0, 0.000000, 38, NaN, NaN), InvoiceDate (datetime64[ns], 541909, 23260, 0, 0.000000, 23260, NaN, NaN), InvoiceNo (object, 541909, 25900, 0, 0.000000, 25900, NaN, NaN), Quantity (int64, 541909, 722, 0, 0.000000, 722, -0.264076, 119769.160031), StockCode (object, 541909, 4070, 0, 0.000000, 4070, NaN, NaN), and UnitPrice (float64, 541909, 1630, 0, 0.000000, 1630, 186.506972, 59005.719097). Below the table, there is a section titled 'Let's see the description of each column:' followed by a list of descriptions for each column. At the bottom, there is a code cell with the command 'cs_df.describe()'.

| | types | counts | distincts | nulls | missing ratio | uniques | skewness | kurtosis |
|-------------|----------------|--------|-----------|--------|---------------|---------|------------|---------------|
| CustomerID | float64 | 406829 | 4373 | 135080 | 24.926694 | 4372 | 0.029835 | -1.179982 |
| Description | object | 540455 | 4224 | 1454 | 0.268311 | 4223 | NaN | NaN |
| Country | object | 541909 | 38 | 0 | 0.000000 | 38 | NaN | NaN |
| InvoiceDate | datetime64[ns] | 541909 | 23260 | 0 | 0.000000 | 23260 | NaN | NaN |
| InvoiceNo | object | 541909 | 25900 | 0 | 0.000000 | 25900 | NaN | NaN |
| Quantity | int64 | 541909 | 722 | 0 | 0.000000 | 722 | -0.264076 | 119769.160031 |
| StockCode | object | 541909 | 4070 | 0 | 0.000000 | 4070 | NaN | NaN |
| UnitPrice | float64 | 541909 | 1630 | 0 | 0.000000 | 1630 | 186.506972 | 59005.719097 |

Let's see the description of each column:

- **InvoiceNo:** A unique identifier for the invoice. An invoice number shared across rows means that those transactions were performed in a single invoice (multiple purchases).
- **StockCode:** Identifier for items contained in an invoice.
- **Description:** Textual description of each of the stock item.
- **Quantity:** The quantity of the item purchased.
- **InvoiceDate:** Date of purchase.
- **UnitPrice:** Value of each item.
- **CustomerID:** Identifier for customer making the purchase.
- **Country:** Country of customer.

```
cs_df.describe()
```

```
print('Check if we had negative quantity and prices at same register:',  
      'No' if cs_df[(cs_df.Quantity<0) & (cs_df.UnitPrice<0)].shape[0]  
      == 0 else 'Yes', '\n')  
print('Check how many register we have where quantity is negative',  
      'and prices is 0 or vice-versa:',  
      cs_df[(cs_df.Quantity<=0) & (cs_df.UnitPrice<=0)].shape[0])  
print('\nWhat is the customer ID of the registers above:',  
      cs_df.loc[(cs_df.Quantity<=0) & (cs_df.UnitPrice<=0),  
                ['CustomerID']].CustomerID.unique())
```

```

print('\n% Negative Quantity:
{:3.2%}'.format(cs_df[(cs_df.Quantity<0)].shape[0]/cs_df.shape[0]))
print('\nAll register with negative quantity has Invoice start with:',
      cs_df.loc[(cs_df.Quantity<0) & ~(cs_df.CustomerID.isnull()),
                'InvoiceNo'].apply(lambda x: x[0]).unique())
print('\nSee an example of negative quantity and others related
records:')
display(cs_df[(cs_df.CustomerID==12472) & (cs_df.StockCode==22244)])

```

| InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|-----------|-----------|-------------|----------|---------------------|-----------|------------|----------------|
| 285657 | 561916 | M | 1 | 2011-08-01 11:44:00 | 0.0 | 15581.0 | United Kingdom |
| 298054 | 562973 | 23157 | 240 | 2011-08-11 11:42:00 | 0.0 | 14911.0 | EIRE |
| 314745 | 564651 | 23270 | 96 | 2011-08-26 14:19:00 | 0.0 | 14646.0 | Netherlands |
| 314746 | 564651 | 23268 | 192 | 2011-08-26 14:19:00 | 0.0 | 14646.0 | Netherlands |
| 314747 | 564651 | 22955 | 144 | 2011-08-26 14:19:00 | 0.0 | 14646.0 | Netherlands |
| 314748 | 564651 | 21786 | 144 | 2011-08-26 14:19:00 | 0.0 | 14646.0 | Netherlands |
| 358655 | 568158 | PADS | 1 | 2011-09-25 12:22:00 | 0.0 | 16133.0 | United Kingdom |
| 361825 | 568384 | M | 1 | 2011-09-27 09:46:00 | 0.0 | 12748.0 | United Kingdom |
| 379913 | 569716 | 22778 | 2 | 2011-10-06 08:17:00 | 0.0 | 15804.0 | United Kingdom |
| 395529 | 571035 | M | 1 | 2011-10-13 12:50:00 | 0.0 | 12446.0 | RSA |
| 420404 | 572893 | 21208 | 5 | 2011-10-26 14:36:00 | 0.0 | 18059.0 | United Kingdom |
| 436428 | 574138 | 23234 | 216 | 2011-11-03 11:26:00 | 0.0 | 12415.0 | Australia |
| 436597 | 574175 | 22065 | 12 | 2011-11-03 11:47:00 | 0.0 | 14110.0 | United Kingdom |
| 436961 | 574252 | M | 1 | 2011-11-03 13:24:00 | 0.0 | 12437.0 | France |
| 439361 | 574469 | 22385 | 12 | 2011-11-04 11:55:00 | 0.0 | 12431.0 | Australia |
| 446125 | 574879 | 22625 | 2 | 2011-11-07 13:22:00 | 0.0 | 13014.0 | United Kingdom |
| 446793 | 574920 | 22899 | 1 | 2011-11-07 16:34:00 | 0.0 | 13985.0 | United Kingdom |
| 446794 | 574920 | 23480 | 1 | 2011-11-07 16:34:00 | 0.0 | 13985.0 | United Kingdom |
| 454463 | 575579 | 22437 | 20 | 2011-11-10 11:49:00 | 0.0 | 13081.0 | United Kingdom |
| 454464 | 575579 | 22089 | 24 | 2011-11-10 11:49:00 | 0.0 | 13081.0 | United Kingdom |
| 479079 | 577129 | 22464 | 4 | 2011-11-17 19:52:00 | 0.0 | 15602.0 | United Kingdom |
| 479546 | 577168 | M | 1 | 2011-11-18 10:42:00 | 0.0 | 12603.0 | Germany |
| 480649 | 577314 | 23407 | 2 | 2011-11-18 13:23:00 | 0.0 | 12444.0 | Norway |
| 485985 | 577696 | M | 1 | 2011-11-21 11:57:00 | 0.0 | 16406.0 | United Kingdom |
| 502122 | 578841 | 84826 | 12540 | 2011-11-25 15:57:00 | 0.0 | 13256.0 | United Kingdom |

Check if we had negative quantity and prices at same register: No

Check how many register we have where quantity is negative and prices is 0 or vice-versa: 1336

What is the customer ID of the registers above: [nan]

% Negative Quantity: 1.96%

All register with negative quantity has Invoice start with: ['c']

See an example of negative quantity and others related records:

| InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|-----------|-----------|----------------------------------|----------|---------------------|-----------|------------|---------|
| 1973 | CS36548 | 22244 3 HOOK HANGER MAGIC GARDEN | -4 | 2010-12-01 14:33:00 | 1.95 | 12472.0 | Germany |
| 9438 | 537201 | 22244 3 HOOK HANGER MAGIC GARDEN | 12 | 2010-12-05 14:19:00 | 1.95 | 12472.0 | Germany |
| 121980 | 546843 | 22244 3 HOOK HANGER MAGIC GARDEN | 12 | 2011-03-17 12:40:00 | 1.95 | 12472.0 | Germany |

```

print('Check register with UnitPrice negative:')
display(cs_df[(cs_df.UnitPrice<0)])
print('Sales records with Customer ID and zero in Unit Price:',cs_df[(cs_df.UnitPrice==0) & ~(cs_df.CustomerID.isnull())].shape)
cs_df[(cs_df.UnitPrice==0) & ~(cs_df.CustomerID.isnull())]

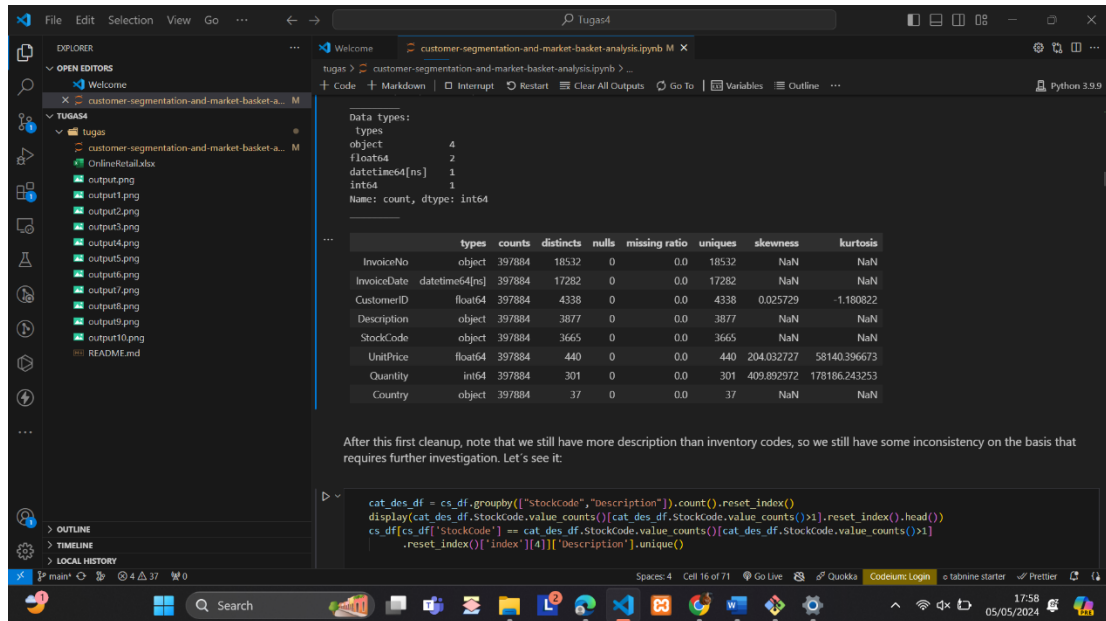
```

Check register with UnitPrice negative:

| InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|-----------|-----------|-------------------|----------|---------------------|-----------|------------|----------------|
| 299983 | A563186 | B Adjust bad debt | 1 | 2011-08-12 14:51:00 | -11062.06 | NaN | United Kingdom |
| 299984 | A563187 | B Adjust bad debt | 1 | 2011-08-12 14:52:00 | -11062.06 | NaN | United Kingdom |

Sales records with Customer ID and zero in Unit Price: 40

Seperti yang dapat Anda lihat, tidak ada catatan di mana kuantitas dan harga bernilai negatif, tetapi ada 1.336 catatan di mana salah satu dari keduanya bernilai negatif dan yang lainnya bernilai 0. Namun, perhatikan bahwa untuk semua catatan ini, kami tidak memiliki ID pelanggan. Jadi kami menyimpulkan bahwa kami dapat menghapus semua catatan dalam kuantitas atau harga dan negatif. Selain itu, dengan ringkasan di atas kita melihat bahwa ada 135.080 catatan tanpa identifikasi pelanggan yang juga dapat kita abaikan.



```
# Remove rows with missing CustomerID
cs_df = cs_df.dropna(subset=['CustomerID'])

# Remove negative or return transactions
cs_df = cs_df[(cs_df['Quantity'] >= 0) & (cs_df['UnitPrice'] > 0)]

details = rstr(cs_df)
display(details.sort_values(by='distincts', ascending=False))
```

| | StockCode | count |
|---|-----------|-------|
| 0 | 23196 | 4 |

| | StockCode | count |
|---|------------------|--------------|
| 1 | 23236 | 4 |
| 2 | 23203 | 3 |
| 3 | 17107D | 3 |
| 4 | 23535 | 3 |

Hal ini memberikan beberapa deskripsi untuk salah satu item tersebut dan kami menyaksikan cara sederhana di mana kualitas data dapat rusak dalam kumpulan data apa pun. Kesalahan pengejaan yang sederhana dapat berakhir dengan penurunan kualitas data dan analisis yang salah.

```
unique_desc = cs_df[["StockCode",
"Description"]].groupby(by=["StockCode"]).\
                apply(pd.DataFrame.mode).reset_index(drop=True)
q = '''
select df.InvoiceNo, df.StockCode, un.Description, df.Quantity,
df.InvoiceDate,
        df.UnitPrice, df.CustomerID, df.Country
from cs_df as df INNER JOIN
        unique_desc as un on df.StockCode = un.StockCode
'''

cs_df = pysqldf(q)
```

```
cs_df.InvoiceDate = pd.to_datetime(cs_df.InvoiceDate)
cs_df['amount'] = cs_df.Quantity*cs_df.UnitPrice
cs_df.CustomerID = cs_df.CustomerID.astype('Int64')

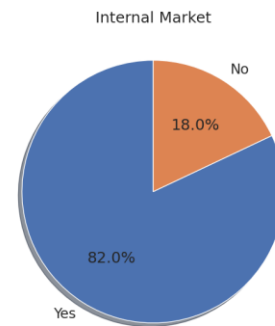
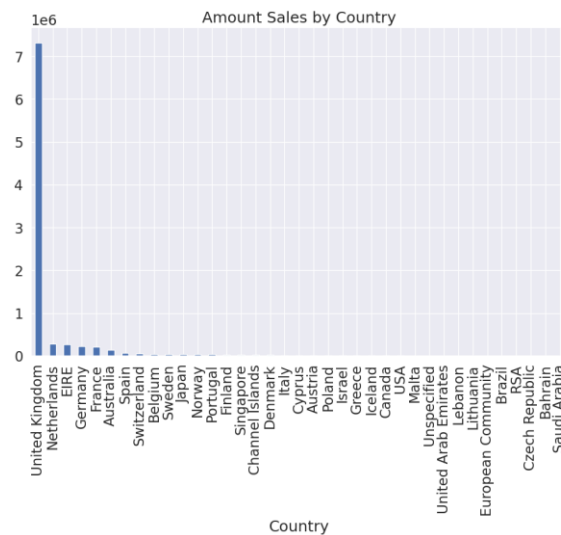
details = rstr(cs_df)
```

```
display(details.sort_values(by='distincts', ascending=False))
```

```
fig = plt.figure(figsize=(25, 7))
f1 = fig.add_subplot(121)
g = cs_df.groupby(["Country"]).amount.sum().sort_values(ascending =
False).plot(kind='bar', title='Amount Sales by Country')
cs_df['Internal'] = cs_df.Country.apply(lambda x: 'Yes' if x=='United
Kingdom' else 'No' )
f2 = fig.add_subplot(122)
market = cs_df.groupby(["Internal"]).amount.sum().sort_values(ascending
= False)
g = plt.pie(market, labels=market.index, autopct='%1.1f%%',
shadow=True, startangle=90)
plt.title('Internal Market')
plt.show()
```

The screenshot shows a Jupyter Notebook environment with the following components:

- EXPLORER:** Displays the file structure, including a folder named 'TUGAS4' containing 'customer-segmentation-and-market-basket-analysis.ipynb' and several 'output.png' files.
- CODE CELL:** Contains the Python code for data analysis and visualization, including grouping by country and internal status, and creating a bar chart and a pie chart.
- DATA SHAPES:** Shows the shape of the data as (397884, 9).
- DATA TYPES:** Lists the data types for each column: object, int64, float64, and datetime64[ns].
- SUMMARY TABLE:** A table with columns: types, counts, distincts, nulls, missing ratio, uniques, skewness, and kurtosis. It lists various columns from the dataset, including InvoiceNo, InvoiceDate, CustomerID, StockCode, Description, amount, UnitPrice, Quantity, and Country.



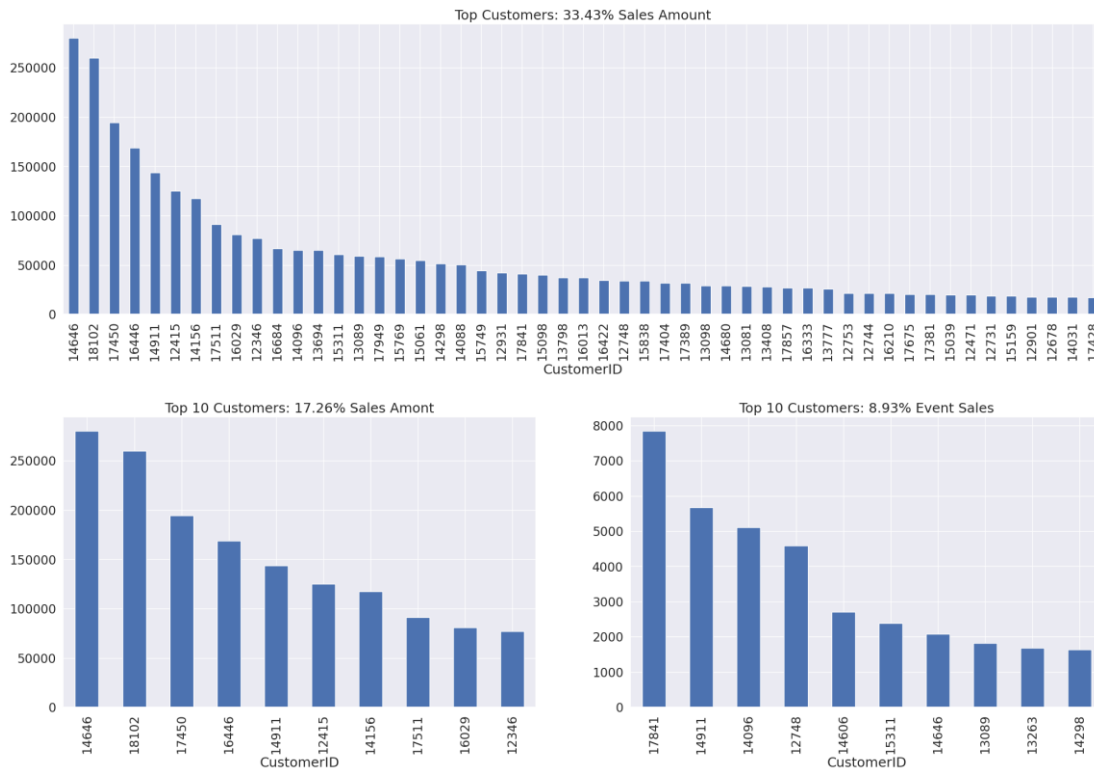
```
fig = plt.figure(figsize=(25, 7))
PercentSales = np.round((cs_df.groupby(["CustomerID"]).amount.sum().\
                           sort_values(ascending =
False)[:51].sum())/cs_df.groupby(["CustomerID"]).\
                           amount.sum().sort_values(ascending =
False).sum()) * 100, 2)
g = cs_df.groupby(["CustomerID"]).amount.sum().sort_values(ascending =
False)[:51].\
    plot(kind='bar', title='Top Customers: {:.3.2f}% Sales
Amount'.format(PercentSales))

fig = plt.figure(figsize=(25, 7))
f1 = fig.add_subplot(121)
PercentSales = np.round((cs_df.groupby(["CustomerID"]).amount.sum().\
                           sort_values(ascending =
False)[:10].sum())/cs_df.groupby(["CustomerID"]).\
                           amount.sum().sort_values(ascending =
False).sum()) * 100, 2)
g = cs_df.groupby(["CustomerID"]).amount.sum().sort_values(ascending =
False)[:10]\
```

```

        .plot(kind='bar', title='Top 10 Customers: {:.2f}% Sales
Amount'.format(PercentSales))
f1 = fig.add_subplot(122)
PercentSales
= np.round((cs_df.groupby(["CustomerID"]).amount.count().\
            sort_values(ascending =
False)[:10].sum())/cs_df.groupby(["CustomerID"]).\
            amount.count().sort_values(ascending =
False).sum()) * 100, 2)
g = cs_df.groupby(["CustomerID"]).amount.count().sort_values(ascending
= False)[:10].\
    plot(kind='bar', title='Top 10 Customers: {:.2f}% Event
Sales'.format(PercentSales))

```



```

AmountSum =
cs_df.groupby(["Description"]).amount.sum().sort_values(ascending =
False)

```

```

inv = cs_df[["Description",
"InvoiceNo"]].groupby(["Description"]).InvoiceNo.unique().\
    agg(np.size).sort_values(ascending = False)

fig = plt.figure(figsize=(25, 7))
f1 = fig.add_subplot(121)
Top10 = list(AmountSum[:10].index)
PercentSales = np.round((AmountSum[Top10].sum()/AmountSum.sum()) * 100,
2)
PercentEvents = np.round((inv[Top10].sum()/inv.sum()) * 100, 2)
g = AmountSum[Top10].\
    plot(kind='bar', title='Top 10 Products in Sales Amount: {:.2f}%\
of Amount and {:.2f}% of Events'.\
        format(PercentSales, PercentEvents))

f1 = fig.add_subplot(122)
Top10Ev = list(inv[:10].index)
PercentSales = np.round((AmountSum[Top10Ev].sum()/AmountSum.sum()) *
100, 2)
PercentEvents = np.round((inv[Top10Ev].sum()/inv.sum()) * 100, 2)
g = inv[Top10Ev].\
    plot(kind='bar', title='Events of top 10 most sold products:\
{:.2f}% of Amount and {:.2f}% of Events'.\
        format(PercentSales, PercentEvents))

fig = plt.figure(figsize=(25, 7))
Top15ev = list(inv[:15].index)
PercentSales = np.round((AmountSum[Top15ev].sum()/AmountSum.sum()) *
100, 2)
PercentEvents = np.round((inv[Top15ev].sum()/inv.sum()) * 100, 2)
g = AmountSum[Top15ev].sort_values(ascending = False).\
    plot(kind='bar',

```

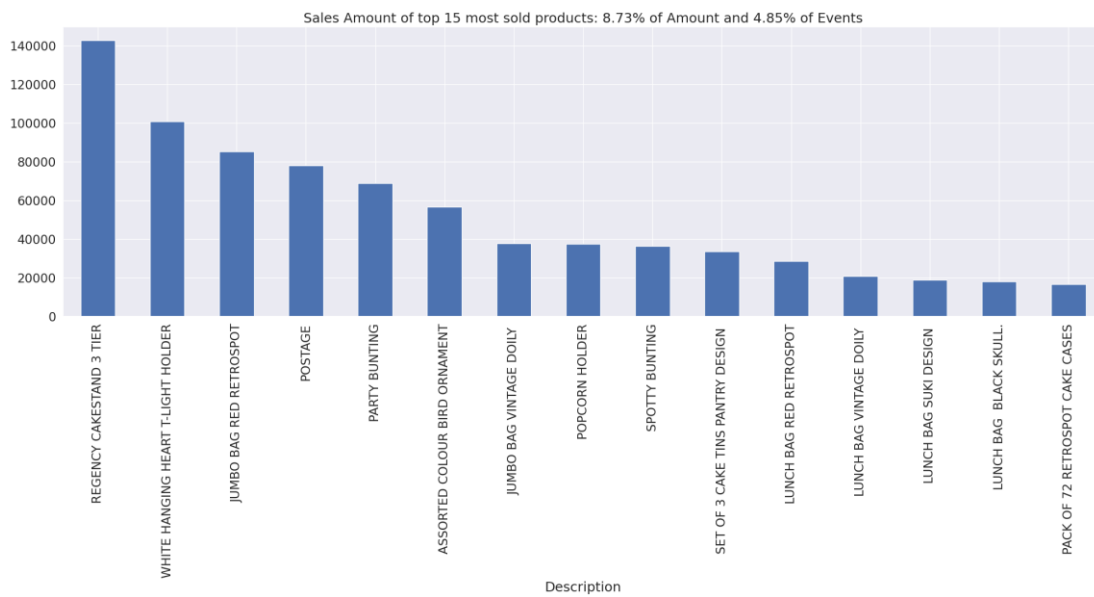
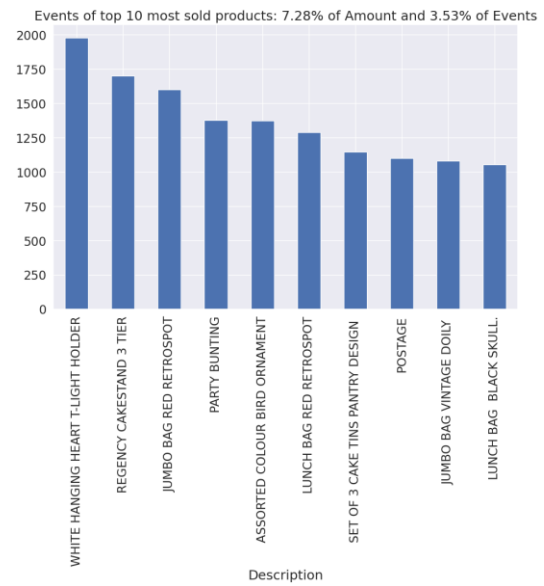
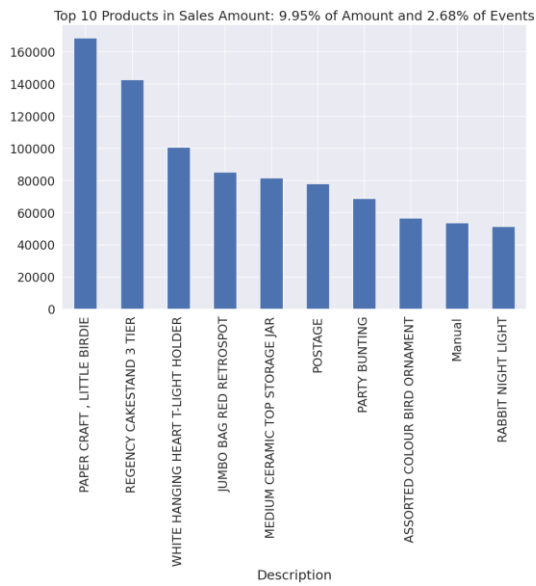
```

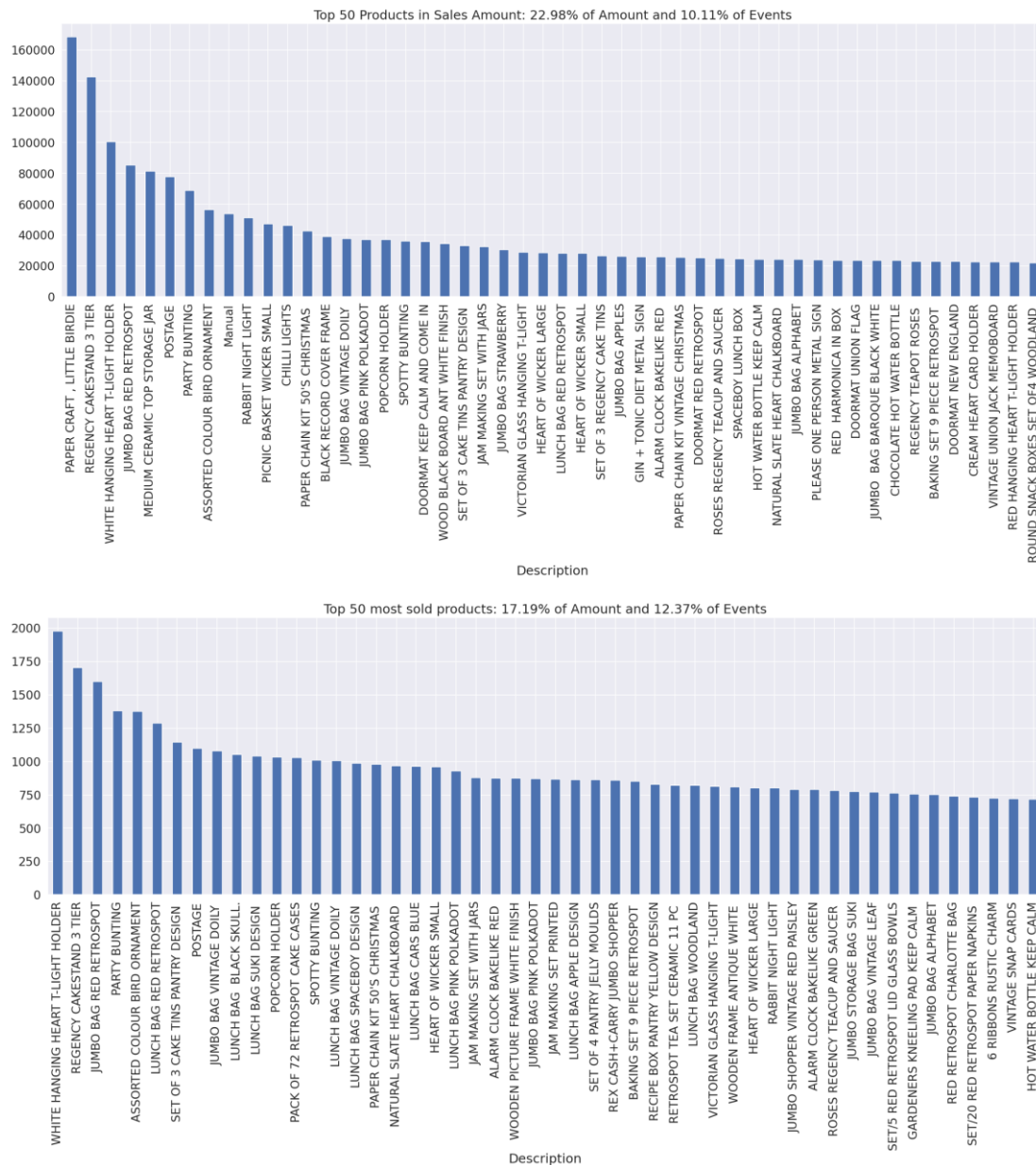
        title='Sales Amount of top 15 most sold products: {:.3.2f}% of
Amount and {:.3.2f}% of Events'.\
        format(PercentSales, PercentEvents))

fig = plt.figure(figsize=(25, 7))
Top50 = list(AmoutSum[:50].index)
PercentSales = np.round((AmoutSum[Top50].sum()/AmoutSum.sum()) * 100,
2)
PercentEvents = np.round((inv[Top50].sum()/inv.sum()) * 100, 2)
g = AmoutSum[Top50].\
    plot(kind='bar',
        title='Top 50 Products in Sales Amount: {:.3.2f}% of Amount and
{:.3.2f}% of Events'.\
        format(PercentSales, PercentEvents))

fig = plt.figure(figsize=(25, 7))
Top50Ev = list(inv[:50].index)
PercentSales = np.round((AmoutSum[Top50Ev].sum()/AmoutSum.sum()) *
100, 2)
PercentEvents = np.round((inv[Top50Ev].sum()/inv.sum()) * 100, 2)
g = inv[Top50Ev].\
    plot(kind='bar', title='Top 50 most sold products: {:.3.2f}% of
Amount and {:.3.2f}% of Events'.\
        format(PercentSales, PercentEvents))

```





3. Segmentasi Pelanggan

Segmentasi pelanggan dalam data mining adalah proses penting untuk mengelompokkan pelanggan ke dalam segmen berdasarkan kesamaan karakteristik atau perilaku. Langkah-langkahnya meliputi pemilihan variabel yang relevan, seperti data demografis atau riwayat pembelian, pemilihan metode segmentasi yang sesuai,

dan pengelompokan pelanggan ke dalam segmen menggunakan algoritma yang dipilih. Evaluasi segmen dilakukan untuk memastikan kualitas segmentasi yang optimal. Proses ini terus berlanjut karena karakteristik pelanggan dan lingkungan bisnis dapat berubah dari waktu ke waktu. Segmentasi pelanggan membantu perusahaan dalam menyediakan layanan yang lebih personalisasi, meningkatkan retensi pelanggan, dan meningkatkan efisiensi pemasaran dengan menargetkan pesan yang sesuai ke setiap segmen. Dengan memahami lebih baik basis pelanggan mereka, perusahaan dapat meningkatkan kinerja bisnis mereka secara keseluruhan.

The screenshot shows a Jupyter Notebook interface with a file explorer on the left, a code editor in the center, and a console output at the bottom. The code defines a 'Recency' variable and calculates the number of days since the last purchase for each customer. The output shows a summary statistics table for the 'recency' variable.

```

import datetime
import pandas as pd

# calculate reference date
reference_date = cs_df['InvoiceDate'].max() + pd.Timedelta(days=1)
print('Reference Date:', reference_date)

# calculate days since last purchase
cs_df['days_since_last_purchase'] = (reference_date - cs_df['InvoiceDate']) / pd.Timedelta(1, 'D')

# Calculate customer recency
customer_history_df = cs_df[['CustomerID', 'days_since_last_purchase']].groupby('CustomerID').min().reset_index()
customer_history_df.rename(columns={'days_since_last_purchase': 'recency'}, inplace=True)

# Display summary statistics
customer_history_df.describe().transpose()

```

| | count | mean | std | min | 25% | 50% | 75% | max |
|------------|--------|--------------|-------------|---------|-----------|-----------|------------|------------|
| CustomerID | 4338.0 | 15300.408022 | 1721.808492 | 12346.0 | 13813.25 | 15299.5 | 16778.75 | 18267.0 |
| recency | 4338.0 | 93.049317 | 100.013298 | 1.0 | 18.072396 | 51.089931 | 142.730556 | 374.122917 |

Kami akan memplot Recency Distribution dan QQ-plot untuk mengidentifikasi penyimpangan substantif dari normalitas, seperti outlier, skewness, dan kurtosis.

```

def QQ_plot(data, measure):
    fig = plt.figure(figsize=(20,7))

    #Get the fitted parameters used by the function
    (mu, sigma) = norm.fit(data)

```

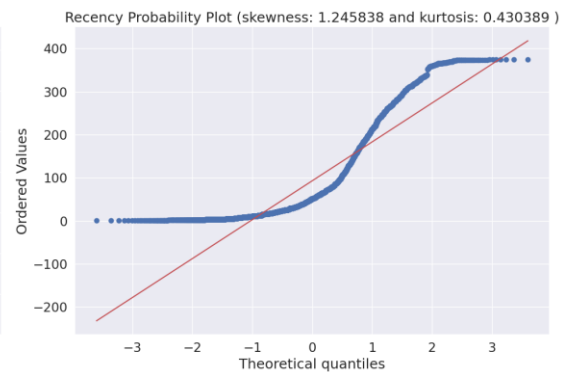
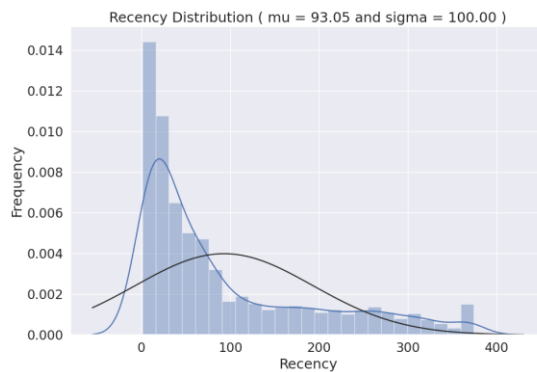
```

#Kernel Density plot
fig1 = fig.add_subplot(121)
sns.distplot(data, fit=norm)
fig1.set_title(measure + ' Distribution ( mu = {:.2f} and sigma = {:.2f} )'.format(mu, sigma), loc='center')
fig1.set_xlabel(measure)
fig1.set_ylabel('Frequency')

#QQ plot
fig2 = fig.add_subplot(122)
res = probplot(data, plot=fig2)
fig2.set_title(measure + ' Probability Plot (skewness: {:.6f} and kurtosis: {:.6f} )'.format(data.skew(), data.kurt()), loc='center')

plt.tight_layout()
plt.show()
QQ_plot(customer_history_df.recency, 'Recency')

```



```

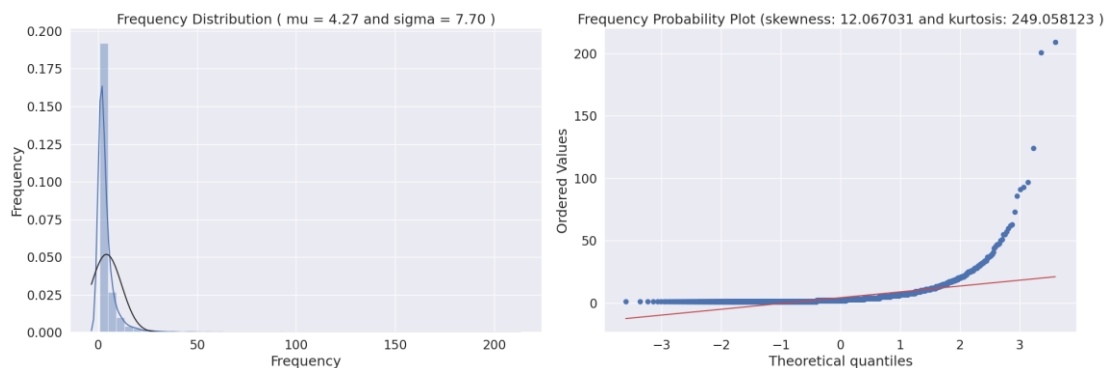
customer_freq = (cs_df[['CustomerID',
'InvoiceNo']].groupby(["CustomerID",
'InvoiceNo']).count().reset_index()).\
                groupby(["CustomerID"]).count().reset_index()
customer_freq.rename(columns={'InvoiceNo': 'frequency'}, inplace=True)
customer_history_df = customer_history_df.merge(customer_freq)

```

```
QQ_plot(customer_history_df.frequency, 'Frequency')
```

Dari grafik pertama, terlihat jelas bahwa distribusi frekuensi penjualan miring, menunjukkan puncak di sebelah kiri dan ekor yang panjang di sebelah kanan, menyimpang dari distribusi normal dan menunjukkan bias positif. Probability Plot lebih lanjut mengkonfirmasi pengamatan ini, karena data sales recency tidak sejajar dengan garis merah diagonal yang mewakili distribusi normal, yang mengindikasikan distribusi miring ke kanan. Dengan nilai skewness sebesar 1,25, ada kekurangan simetri yang jelas, yang menunjukkan bahwa frekuensi penjualan condong ke kanan, seperti yang terlihat pada plot Distribusi Penjualan.

Kecondongan ini menunjukkan bahwa ekor kanan distribusi lebih panjang dibandingkan ekor kiri. Selain itu, nilai kurtosis positif sebesar 0,43 menunjukkan bahwa frekuensi penjualan memiliki ekor yang besar, yang menunjukkan adanya pencilan dalam data.



Dari grafik pertama di atas, kita dapat melihat bahwa distribusi jumlah penjualan miring, memiliki puncak di sebelah kiri dan ekor yang panjang di sebelah kanan. Ini menyimpang dari distribusi normal dan bias positif.

Dari Probability Plot, kita dapat melihat bahwa jumlah penjualan juga tidak sejajar dengan diagonal, khususnya di sebelah kanan.

Dengan skewness positif sebesar 19,3, kami mengkonfirmasi kurangnya

kesimetrisan yang tinggi dan dengan 478 Kurtosis mengindikasikan bahwa itu adalah distribusi yang terlalu berekor tebal dan memiliki outlier, tentunya lebih dari 10 yang sangat ekstrim.

Mari kita lihat ringkasan statistik dari kumpulan data ini:

```
customer_history_df.describe()
```

4. Proses Data Nilai Uang

Setelah kita membuat dataset nilai pelanggan, kita akan melakukan beberapa preprocessing pada data. Untuk pengelompokan kita, kita akan menggunakan algoritma pengelompokan K-means. Salah satu persyaratan agar algoritma ini berfungsi dengan baik adalah pemusatan rata-rata nilai variabel. Pemusatan rata-rata nilai variabel berarti bahwa kita akan mengganti nilai aktual variabel dengan nilai standar, sehingga variabel tersebut memiliki rata-rata 0 dan varians 1. Hal ini memastikan bahwa semua variabel berada dalam rentang yang sama dan perbedaan rentang nilai tidak menyebabkan algoritma tidak bekerja dengan baik. Ini mirip dengan penskalaan fitur.

Masalah lain yang dapat Anda selidiki adalah rentang nilai yang sangat besar yang dapat diambil oleh setiap variabel. Ini

Masalah ini terutama terlihat pada variabel jumlah uang. Untuk mengatasi masalah ini, kita akan mentransformasikan semua variabel pada skala log. Transformasi ini, bersama dengan standarisasi, akan memastikan bahwa input ke algoritma kita adalah satu set homogen dari nilai yang diskalakan dan diubah.

Poin penting tentang langkah prapemrosesan data adalah bahwa terkadang kita membutuhkannya untuk menjadi reversibel. Dalam kasus kita, kita akan mendapatkan hasil pengelompokan dalam bentuk variabel log yang ditransformasi dan diskalakan. Tetapi untuk membuat kesimpulan dalam hal data asli, kita perlu mengubah balik

semua variabel sehingga kita mendapatkan kembali angka RFM yang sebenarnya. Hal ini dapat dilakukan dengan menggunakan kemampuan preprocessing Python.

```
customer_history_df['recency_log'] =  
customer_history_df['recency'].apply(math.log)  
customer_history_df['frequency_log'] =  
customer_history_df['frequency'].apply(math.log)  
customer_history_df['amount_log'] =  
customer_history_df['amount'].apply(math.log)  
feature_vector = ['amount_log', 'recency_log', 'frequency_log']  
X_subset = customer_history_df[feature_vector] #.as_matrix()  
scaler = preprocessing.StandardScaler().fit(X_subset)  
X_scaled = scaler.transform(X_subset)  
pd.DataFrame(X_scaled, columns=X_subset.columns).describe().T
```

```
fig = plt.figure(figsize=(20,14))  
f1 = fig.add_subplot(221); sns.regplot(x='recency', y='amount',  
data=customer_history_df)  
f1 = fig.add_subplot(222); sns.regplot(x='frequency', y='amount',  
data=customer_history_df)  
f1 = fig.add_subplot(223); sns.regplot(x='recency_log', y='amount_log',  
data=customer_history_df)  
f1 = fig.add_subplot(224); sns.regplot(x='frequency_log',  
y='amount_log', data=customer_history_df)  
  
fig = plt.figure(figsize=(15, 10))  
ax = fig.add_subplot(111, projection='3d')  
  
xs =customer_history_df.recency_log  
ys = customer_history_df.frequency_log  
zs = customer_history_df.amount_log  
ax.scatter(xs, ys, zs, s=5)
```

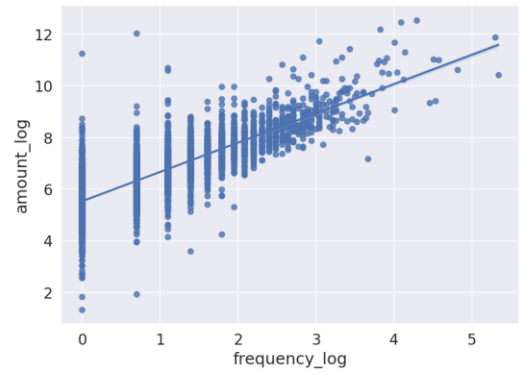
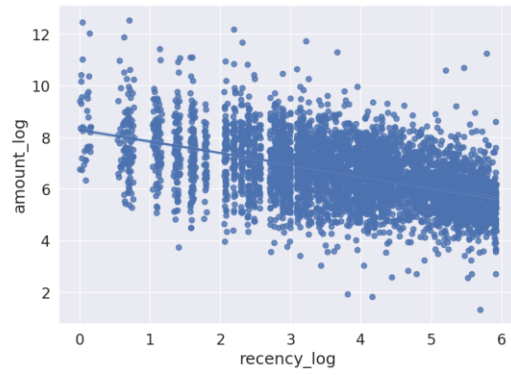
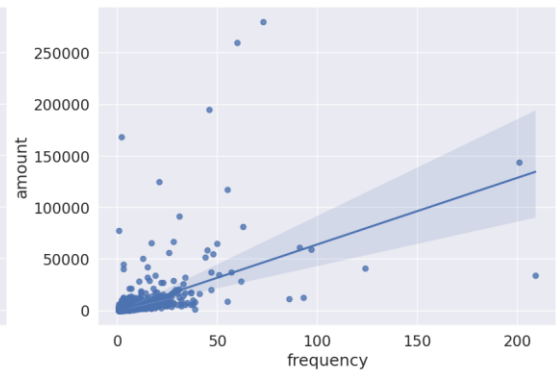
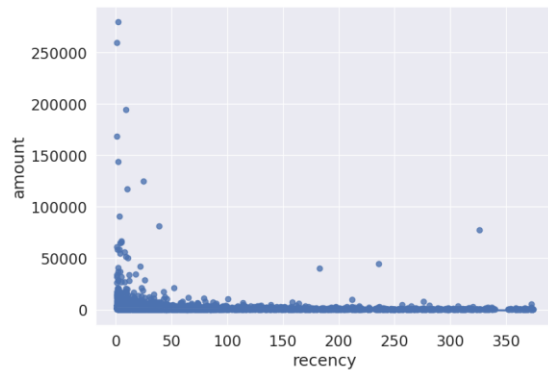
```

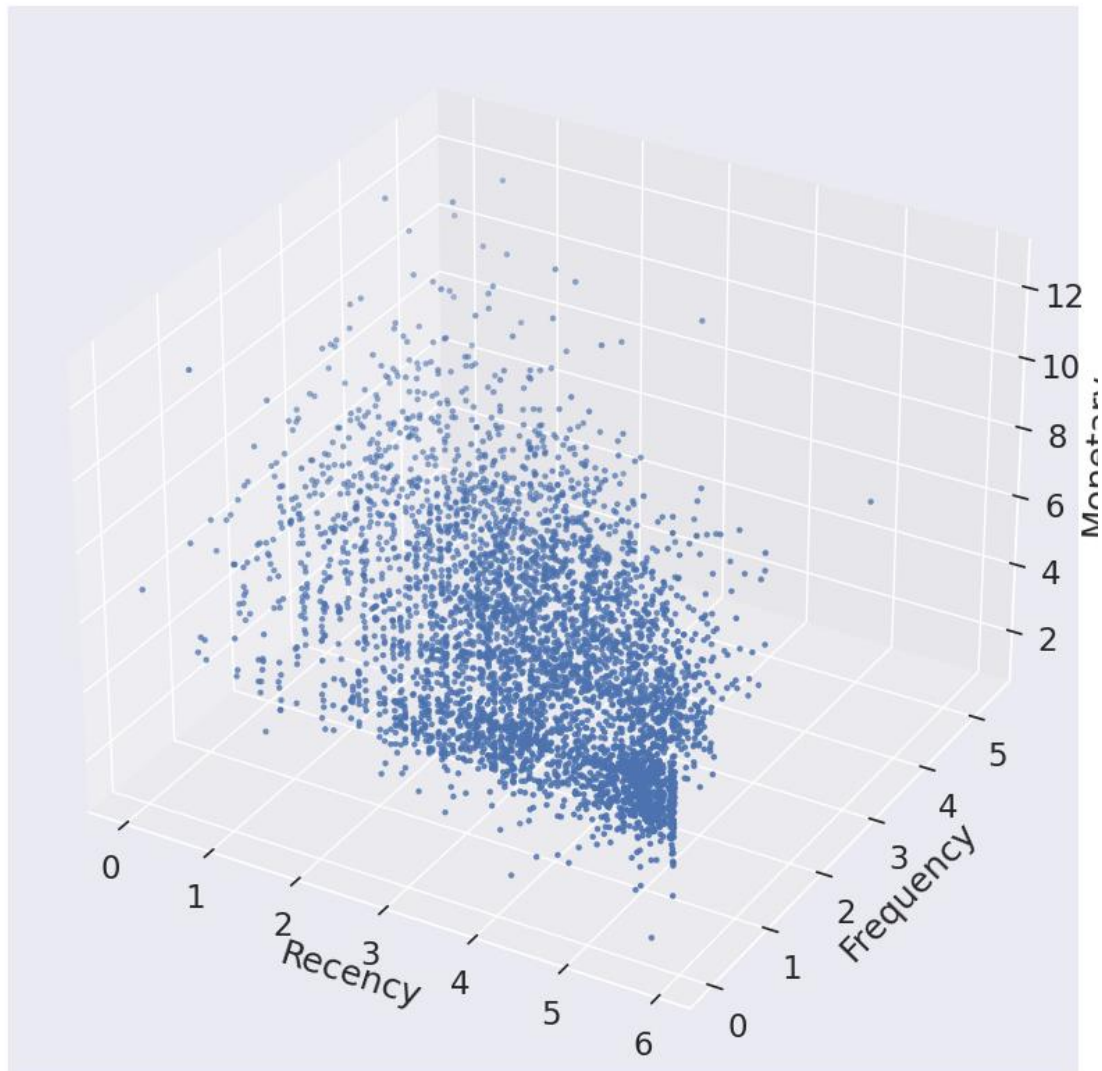
ax.set_xlabel('Recency')
ax.set_ylabel('Frequency')
ax.set_zlabel('Monetary')

plt.show()

```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---------------|--------|-----------------------|--------------|-------------------|-------------------|-------------------|--------------|--------------|
| amount_log | 4338.0 | - 7.01042 6e-16 | 1.0001 15 | - 4.1792 80 | - 0.6841 83 | - 0.0609 42 | 0.6542 44 | 4.7213 95 |
| recency_log | 4338.0 | 9.82770 0e-18 | 1.0001 15 | - 2.7475 18 | - 0.6551 96 | 0.0960 33 | 0.8387 07 | 1.5353 05 |
| frequency_log | 4338.0 | - 9.99149 5e-17 | 1.0001 15 | - 1.0486 10 | - 1.0486 10 | - 0.2790 44 | 0.7382 67 | 4.8827 14 |





5. Pengelompokan Segmen Data Dengan Metode K-means

Pengelompokan untuk segmen merupakan salah satu teknik dalam analisis data yang digunakan untuk mengelompokkan data ke dalam kelompok atau segmen berdasarkan kesamaan karakteristik atau pola tertentu. Salah satu metode yang umum digunakan untuk melakukan pengelompokan adalah metode K-means.

Metode K-means adalah algoritma pengelompokan yang bertujuan untuk membagi data menjadi K kelompok atau segmen berdasarkan kedekatan antara titik data. Algoritma ini bekerja dengan cara menginisialisasi pusat-pusat

kelompok secara acak, kemudian mengiterasikan proses di mana setiap titik data dikelompokkan ke dalam kelompok terdekat berdasarkan jarak Euclidean antara titik data dan pusat kelompok. Pusat kelompok kemudian diperbarui dengan menghitung rata-rata dari semua titik data yang termasuk dalam kelompok tersebut, dan proses ini diulangi hingga konvergensi atau hingga tidak ada perubahan yang signifikan dalam pusat kelompok.

Untuk memperbaiki kinerja K-means dan menghindari inisialisasi yang buruk, digunakanlah metode K-means++. Metode ini merupakan sebuah algoritma inisialisasi yang canggih yang bertujuan untuk menentukan lokasi awal pusat kelompok yang lebih baik. Dalam K-means++, pusat-pusat awal kelompok dipilih secara acak dari titik-titik data, namun dengan probabilitas yang berdasarkan pada jarak antara titik data dan pusat kelompok yang sudah dipilih sebelumnya.

Metode siku (elbow method) adalah teknik yang digunakan untuk menentukan jumlah optimal kelompok dalam algoritma K-means. Ide utamanya adalah untuk mengidentifikasi titik di mana penambahan jumlah kelompok tidak lagi memberikan penurunan yang signifikan dalam varians dalam kelompok. Grafik yang menunjukkan jumlah kelompok terhadap varians intra-kelompok sering kali memiliki bentuk seperti lengkungan, dan titik di mana lengkungan tersebut memiliki siku adalah jumlah kelompok yang optimal.

Secara teoritis, pengelompokan menggunakan algoritma K-means memungkinkan untuk membagi data menjadi kelompok-kelompok yang memiliki kesamaan dalam fitur atau atribut tertentu. K-means++ membantu meningkatkan kinerja algoritma dengan memilih inisialisasi pusat kelompok yang lebih baik, sedangkan metode siku membantu menentukan jumlah kelompok yang optimal untuk data tertentu. Dengan demikian, teknik ini memungkinkan analisis data untuk memahami struktur data dengan lebih baik dan mengidentifikasi pola atau

kelompok yang mungkin tersembunyi dalam data.

```
cl = 50
corte = 0.1

anterior = 1000000000000000
cost = []
K_best = cl

for k in range (1, cl+1):
    # Create a kmeans model on our data, using k
    # clusters. random_state helps ensure that the algorithm returns the
    # same results each time.
    model = KMeans(
        n_clusters=k,
        init='k-means++', #'random',
        n_init=10,
        max_iter=300,
        tol=1e-04,
        random_state=101)

    model = model.fit(X_scaled)

    # These are our fitted labels for clusters -- the first cluster has
    # label 0, and the second has label 1.
    labels = model.labels_

    # Sum of distances of samples to their closest cluster center
    interia = model.inertia_

    if (K_best == cl) and (((anterior - interia)/anterior) < corte):
        K_best = k - 1
        cost.append(interia)
        anterior = interia
```

```

plt.figure(figsize=(8, 6))
plt.scatter(range (1, cl+1), cost, c='red')
plt.show()

# Create a kmeans model with the best K.
print('The best K suggest: ',K_best)
model = KMeans(n_clusters=K_best, init='k-means++',
n_init=10,max_iter=300, tol=1e-04, random_state=101)

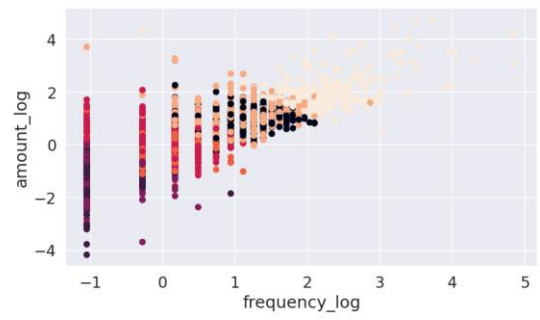
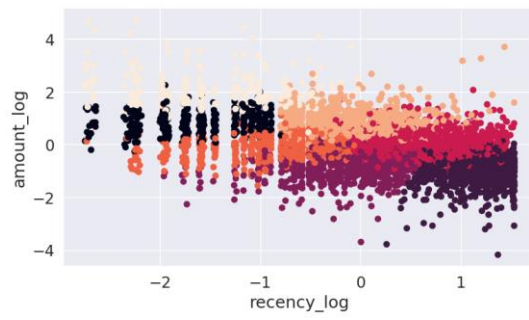
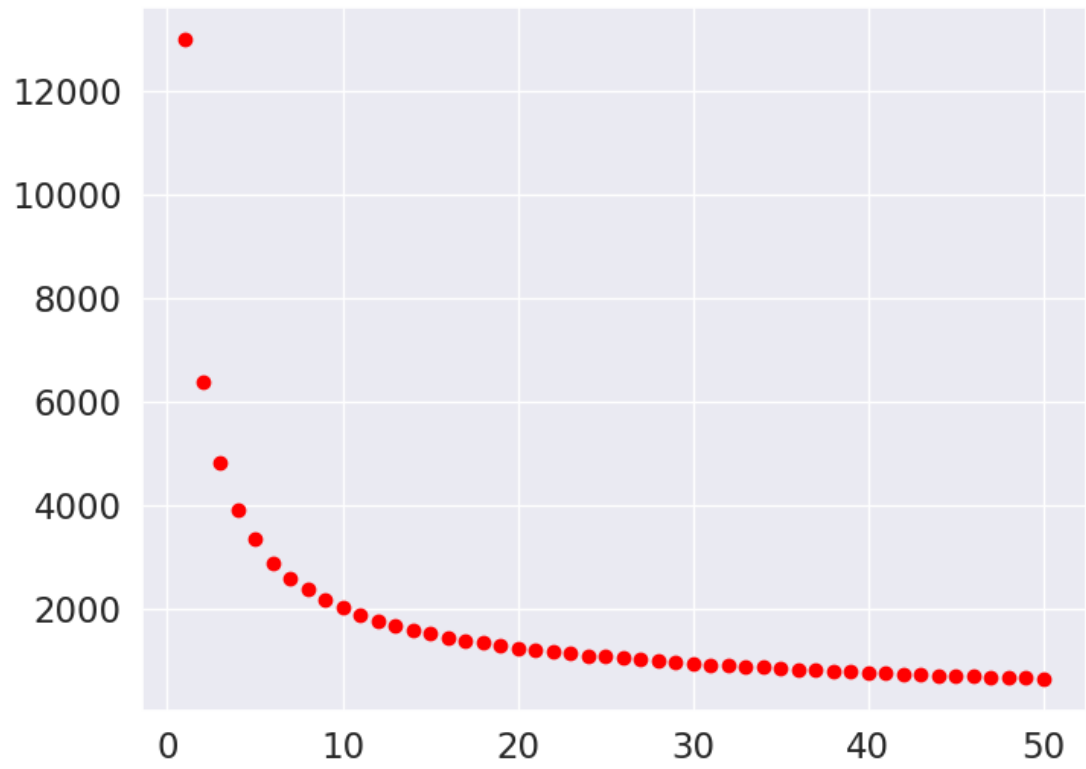
# Note I'm scaling the data to normalize it! Important for good
results.
model = model.fit(X_scaled)

# These are our fitted labels for clusters -- the first cluster has
label 0, and the second has label 1.
labels = model.labels_

# And we'll visualize it:
#plt.scatter(X_scaled[:,0], X_scaled[:,1],
c=model.labels_.astype(float))
fig = plt.figure(figsize=(20,5))
ax = fig.add_subplot(121)
plt.scatter(x = X_scaled[:,1], y = X_scaled[:,0],
c=model.labels_.astype(float))
ax.set_xlabel(feature_vector[1])
ax.set_ylabel(feature_vector[0])
ax = fig.add_subplot(122)
plt.scatter(x = X_scaled[:,2], y = X_scaled[:,0],
c=model.labels_.astype(float))
ax.set_xlabel(feature_vector[2])
ax.set_ylabel(feature_vector[0])

plt.show()

```



6. Analisis Siluet Pada Pengelompokan K-means

Analisis Siluet adalah metode evaluasi yang digunakan untuk mengevaluasi kualitas pengelompokan atau klastering dalam algoritma pengelompokan seperti K-means. Metode ini memberikan pengukuran tentang seberapa baik setiap titik data cocok dengan kelompoknya sendiri dibandingkan dengan kelompok lain. Evaluasi ini berguna untuk menentukan seberapa baik pengelompokan tersebut telah berhasil memisahkan titik-titik data ke dalam kelompok yang sesuai.

Proses analisis siluet dimulai dengan menghitung nilai siluet untuk setiap titik data dalam dataset. Nilai siluet diperoleh dengan mengukur rasio antara jarak rata-rata antara titik tersebut dengan titik-titik dalam kelompok yang sama (intra-cluster distance) dan jarak rata-rata antara titik tersebut dengan titik-titik dalam kelompok lain (inter-cluster distance). Semakin tinggi nilai siluet, semakin baik titik data tersebut dikelompokkan dengan kelompoknya sendiri dibandingkan dengan kelompok lain.

Setelah mendapatkan nilai siluet untuk setiap titik data, nilai siluet rata-rata dari seluruh dataset dihitung. Nilai siluet rata-rata ini berkisar dari -1 hingga 1. Nilai siluet yang mendekati 1 menunjukkan bahwa pengelompokan tersebut berhasil, sedangkan nilai siluet yang mendekati -1 menunjukkan bahwa titik data seharusnya dikelompokkan ke dalam kelompok lain.

Dengan menganalisis nilai siluet rata-rata dari seluruh dataset, kita dapat mengevaluasi kualitas pengelompokan secara keseluruhan. Nilai siluet yang tinggi menunjukkan bahwa pengelompokan telah berhasil memisahkan titik-titik data dengan baik, sementara nilai siluet yang rendah menandakan bahwa ada ketidakcocokan antara titik-titik data dalam kelompok yang sama.

Dalam konteks pengelompokan menggunakan algoritma K-means, analisis siluet dapat digunakan untuk mengevaluasi seberapa baik model K-means telah berhasil memisahkan data ke dalam kelompok-kelompok yang sesuai dengan struktur sebenarnya dari data tersebut. Dengan memahami nilai siluet, kita dapat menentukan apakah jumlah kelompok yang dipilih adalah optimal dan apakah data dapat dibagi dengan baik ke dalam kelompok-kelompok yang bermakna.

```
cluster_centers = dict()

for n_clusters in range(3, K_best+1, 2):
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3)
    fig.set_size_inches(25, 7)
    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(X_scaled) + (n_clusters + 1) * 10])

    clusterer = KMeans(n_clusters=n_clusters, init='k-means++',
n_init=10, max_iter=300, tol=1e-04, random_state=101)
    cluster_labels = clusterer.fit_predict(X_scaled)

    silhouette_avg = silhouette_score(X = X_scaled, labels =
cluster_labels)
    cluster_centers.update({n_clusters
: {'cluster_center': clusterer.cluster_centers_,
'silhouette_score': silhouette_avg,
'labels': cluster_labels}
})

    sample_silhouette_values = silhouette_samples(X = X_scaled, labels
= cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
```

```

        ith_cluster_silhouette_values =
sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.Spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                        0, ith_cluster_silhouette_values,
                        facecolor=color, edgecolor=color, alpha=0.7)

        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10 # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8,
0.9, 1])
    colors = cm.Spectral(cluster_labels.astype(float) / n_clusters)

    centers = clusterer.cluster_centers_
    y = 0
    x = 1
    ax2.scatter(X_scaled[:, x], X_scaled[:, y], marker='.', s=30, lw=0,
alpha=0.7, c=colors, edgecolor='k')
    ax2.scatter(centers[:, x], centers[:, y], marker='o', c="white",
alpha=1, s=200, edgecolor='k')
    for i, c in enumerate(centers):

```

```

        ax2.scatter(c[x], c[y], marker='$_d$' % i, alpha=1, s=50,
edgecolor='k')
        ax2.set_title("{} Clustered data".format(n_clusters))
        ax2.set_xlabel(feature_vector[x])
        ax2.set_ylabel(feature_vector[y])

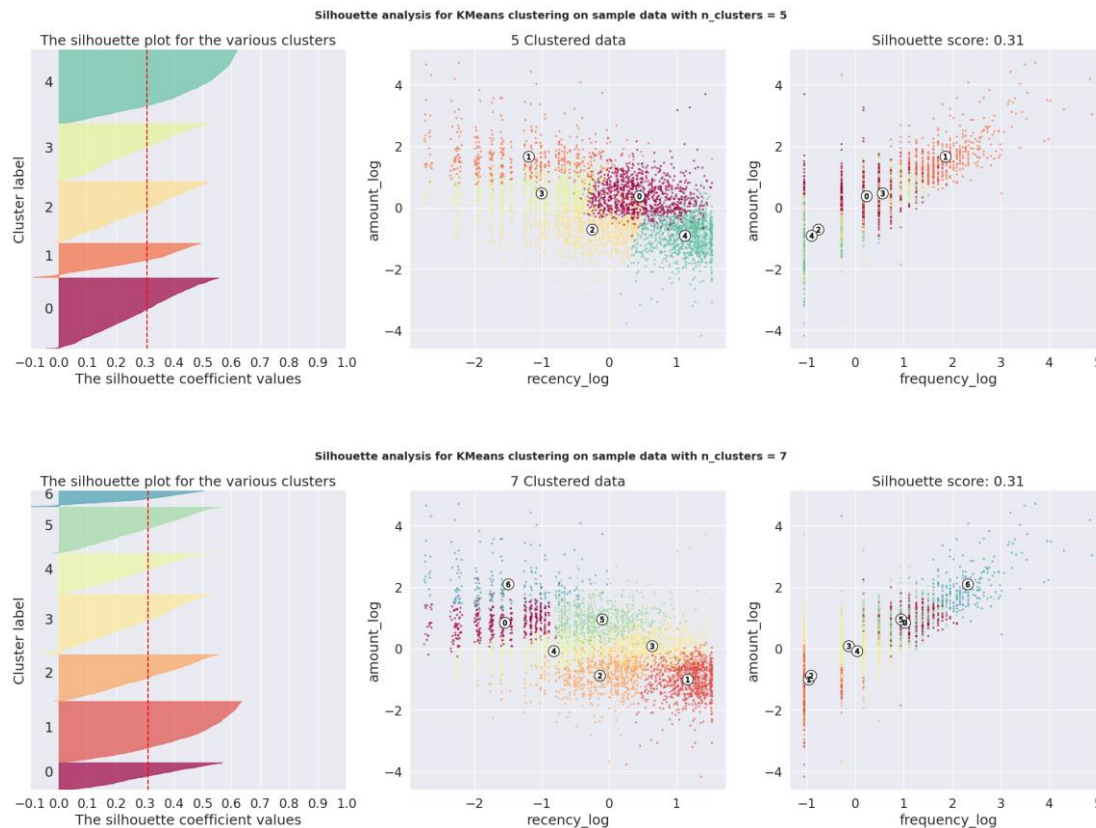
        x = 2
        ax3.scatter(X_scaled[:, x], X_scaled[:, y], marker='.', s=30, lw=0,
alpha=0.7, c=colors, edgecolor='k')
        ax3.scatter(centers[:, x], centers[:, y], marker='o', c="white",
alpha=1, s=200, edgecolor='k')
        for i, c in enumerate(centers):
            ax3.scatter(c[x], c[y], marker='$_d$' % i, alpha=1, s=50,
edgecolor='k')
        ax3.set_title("Silhouette score:
{:1.2f}".format(cluster_centers[n_clusters]['silhouette_score']))
        ax3.set_xlabel(feature_vector[x])
        ax3.set_ylabel(feature_vector[y])

        plt.suptitle(("Silhouette analysis for KMeans clustering on sample
data with n_clusters = %d" % n_clusters),
                    fontsize=14, fontweight='bold')

        plt.show()

```





7. Pusat Kluster

Cluster center (pusat kluster) adalah titik representatif yang mewakili kelompok atau kluster tertentu dalam algoritma pengelompokan seperti K-means. Dalam konteks algoritma K-means, setiap kelompok atau kluster direpresentasikan oleh satu titik pusat, yang merupakan nilai rata-rata dari semua titik data dalam kelompok tersebut.

Proses pembentukan pusat kluster dimulai dengan inisialisasi pusat-pusat kluster secara acak pada awalnya. Setelah inisialisasi, algoritma K-means menghitung jarak antara setiap titik data dengan semua pusat kluster yang ada, dan kemudian menetapkan setiap titik data ke kelompok dengan pusat kluster terdekat. Setelah semua titik data dikelompokkan, pusat kluster diperbarui dengan menghitung rata-rata dari semua titik data yang termasuk dalam kelompok tersebut.

Proses ini diulangi secara iteratif, dengan titik-titik data direlokasi ke kelompok-

kelompok baru dan pusat klaster diperbarui setiap iterasi. Iterasi dilakukan hingga tidak ada perubahan yang signifikan dalam lokasi pusat klaster atau hingga kriteria konvergensi terpenuhi.

Pusat klaster memiliki arti penting dalam algoritma K-means karena mereka menentukan bentuk dan posisi dari setiap kelompok. Pusat klaster yang optimal akan mewakili dengan baik pola data dalam kelompoknya dan memastikan bahwa jarak antara titik data dalam kelompok tersebut relatif kecil.

Dalam prakteknya, hasil dari algoritma K-means sering kali dievaluasi berdasarkan posisi dan karakteristik pusat klaster untuk menentukan apakah pengelompokan yang dihasilkan adalah representasi yang baik dari data yang ada.

```
features = ['amount', 'recency', 'frequency']
for i in range(3, K_best+1, 2):
    print("for {} clusters the silhouette score is {:.2f}".format(i,
cluster_centers[i]['silhouette_score']))
    print("Centers of each cluster:")
    cent_transformed =
scaler.inverse_transform(cluster_centers[i]['cluster_center'])
    print(pd.DataFrame(np.exp(cent_transformed), columns=features))
    print('-'*50)
```

for 3 clusters the silhouette score is 0.33

Centers of each cluster:

| | amount | recency | frequency |
|---|-------------|------------|-----------|
| 0 | 955.108413 | 35.220128 | 3.042955 |
| 1 | 3859.014223 | 8.763853 | 9.651473 |
| 2 | 259.270172 | 119.903139 | 1.183191 |

for 5 clusters the silhouette score is 0.31

Centers of each cluster:

| | amount | recency | frequency |
|---|-------------|------------|-----------|
| 0 | 1161.573046 | 83.524872 | 3.155363 |
| 1 | 5952.818406 | 8.576574 | 13.683600 |
| 2 | 296.081689 | 31.515878 | 1.295759 |
| 3 | 1314.406149 | 11.107092 | 4.290844 |
| 4 | 231.257842 | 212.118311 | 1.144017 |

for 7 clusters the silhouette score is 0.31

Centers of each cluster:

| | amount | recency | frequency |
|---|--------------|------------|-----------|
| 0 | 2134.498750 | 5.204186 | 6.460064 |
| 1 | 205.931613 | 226.183423 | 1.085344 |
| 2 | 240.188728 | 37.193024 | 1.130437 |
| 3 | 815.752704 | 108.048616 | 2.280582 |
| 4 | 666.191186 | 14.347347 | 2.667995 |
| 5 | 2408.833819 | 38.804916 | 5.996435 |
| 6 | 10200.920787 | 5.633035 | 20.695211 |

8. . Wawasan Klaster

Cluster insight merujuk pada pemahaman atau wawasan yang diperoleh dari hasil pengelompokan atau klastering data. Setelah data dikelompokkan ke dalam kelompok-kelompok atau klaster, analisis cluster insight bertujuan untuk menggali informasi yang berguna dari setiap klaster yang terbentuk. Ini melibatkan eksplorasi karakteristik atau pola unik yang dimiliki oleh setiap klaster, serta mengidentifikasi perbedaan atau kesamaan antar klaster.

Analisis cluster insight dapat mengungkapkan berbagai hal, termasuk:

1. ****Tren atau pola unik****: Identifikasi tren atau pola yang spesifik untuk setiap klaster, seperti preferensi pelanggan, perilaku pembelian, atau fitur produk yang

populer.

2. ****Perbedaan antar kluster****: Memahami perbedaan signifikan antara kluster dalam hal fitur atau atribut tertentu, seperti demografi, lokasi geografis, atau preferensi produk.
3. ****Karakteristik pusat kluster****: Memahami posisi dan karakteristik pusat kluster untuk setiap kluster, termasuk jarak antara titik data dalam kluster dan kepadatan kluster.
4. ****Keterkaitan antar kluster****: Mengidentifikasi keterkaitan atau hubungan antar kluster, seperti kluster yang saling bersaing atau saling melengkapi.
5. ****Segmentasi pasar****: Membuat segmentasi pasar yang lebih mendalam dengan memahami preferensi dan perilaku pelanggan dalam setiap kluster.

Analisis cluster insight penting dalam mengambil keputusan bisnis yang lebih baik dan merumuskan strategi yang lebih efektif. Dengan memahami karakteristik dan pola dalam setiap kluster, perusahaan dapat mengidentifikasi peluang pasar baru, meningkatkan pelayanan pelanggan, dan mengoptimalkan strategi pemasaran dan penjualan.

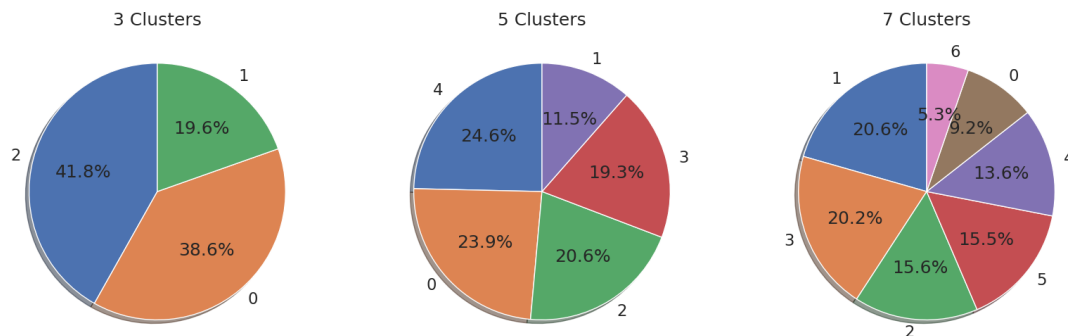
9. Menelusuri Cluster

```
10. customer_history_df['clusters_3'] = cluster_centers[3]['labels']
11. customer_history_df['clusters_5'] = cluster_centers[5]['labels']
12. customer_history_df['clusters_7'] = cluster_centers[7]['labels']
13. display(customer_history_df.head())
14.
15. fig = plt.figure(figsize=(20,7))
16. f1 = fig.add_subplot(131)
```

```

17. market = customer_history_df.clusters_3.value_counts()
18. g = plt.pie(market, labels=market.index, autopct='%1.1f%%',
    shadow=True, startangle=90)
19. plt.title('3 Clusters')
20. f1 = fig.add_subplot(132)
21. market = customer_history_df.clusters_5.value_counts()
22. g = plt.pie(market, labels=market.index, autopct='%1.1f%%',
    shadow=True, startangle=90)
23. plt.title('5 Clusters')
24. f1 = fig.add_subplot(133)
25. market = customer_history_df.clusters_7.value_counts()
26. g = plt.pie(market, labels=market.index, autopct='%1.1f%%',
    shadow=True, startangle=90)
27. plt.title('7 Clusters')
28. plt.show()

```



```

x_data = ['Cluster 0', 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4',
'Cluster 5', 'Cluster 6']
colors = ['rgba(93, 164, 214, 0.5)', 'rgba(255, 144, 14, 0.5)',
'rgba(44, 160, 101, 0.5)', 'rgba(255, 65, 54, 0.5)',
'rgba(22, 80, 57, 0.5)', 'rgba(127, 65, 14, 0.5)', 'rgba(207,
114, 255, 0.5)', 'rgba(127, 96, 0, 0.5)']
cutoff_quantile = 95

for n_clusters in range(3, K_best+1, 2):

```

```

    cl = 'clusters_' + str(n_clusters)
    for fild in range(0, 3):
        field_to_plot = features[fild]
        y_data = list()
        ymax = 0
        for i in np.arange(0, n_clusters):
            y0 =
customer_history_df[customer_history_df[cl]==i][field_to_plot].values
            y0 = y0[y0<np.percentile(y0, cutoff_quantile)]
            if ymax < max(y0): ymax = max(y0)
            y_data.insert(i, y0)

        traces = []

        for xd, yd, cls in zip(x_data[:n_clusters], y_data,
colors[:n_clusters]):
            traces.append(go.Box(y=yd, name=xd, boxpoints=False,
jitter=0.5, whiskerwidth=0.2, fillcolor=cls,
marker=dict( size=1, ),
line=dict(width=1),
            ))

        layout = go.Layout(
            title='Difference in {} with {} Clusters and {:.12f}
Score'.\
            format(field_to_plot, n_clusters,
cluster_centers[n_clusters]['silhouette_score']),
            yaxis=dict( autorange=True, showgrid=True, zeroline=True,
dtick = int(ymax/10),
            gridcolor='black', gridwidth=0.1,
zerolinecolor='rgb(255, 255, 255)', zerolinewidth=2, ),
            margin=dict(l=40, r=30, b=50, t=50, ),
            paper_bgcolor='white',
            plot_bgcolor='white',

```

```

        showlegend=False
    )

    fig = go.Figure(data=traces, layout=layout)
    py.offline.iplot(fig)

```

10. Membuat Perhitungan Data Transaksi

```

items = list(cs_df.Description.unique())
grouped = cs_df.groupby('InvoiceNo')
transaction_level = grouped.aggregate(lambda x:
tuple(x)).reset_index()[['InvoiceNo', 'Description']]
transaction_dict = {item:0 for item in items}
output_dict = dict()
temp = dict()
for rec in transaction_level.to_dict('records'):
    invoice_num = rec['InvoiceNo']
    items_list = rec['Description']
    transaction_dict = {item:0 for item in items}
    transaction_dict.update({item:1 for item in items if item in
items_list})
    temp.update({invoice_num:transaction_dict})

new = [v for k,v in temp.items()]
transaction_df = pd.DataFrame(new)

```

```

def prune_dataset(input_df, length_trans = 2, total_sales_perc = 0.5,
                  start_item = None, end_item = None, TopCols = None):
    if 'total_items' in input_df.columns:
        del(input_df['total_items'])
    item_count = input_df.sum().sort_values(ascending =
False).reset_index()
    total_items = sum(input_df.sum().sort_values(ascending = False))
    item_count.rename(columns={item_count.columns[0]: 'item_name',

```

```

item_count.columns[1]='item_count'},
inplace=True)
    if TopCols:
        input_df['total_items'] = input_df[TopCols].sum(axis = 1)
        input_df = input_df[input_df.total_items >= length_trans]
        del(input_df['total_items'])
        return input_df[TopCols],
item_count[item_count.item_name.isin(TopCols)]
    elif end_item > start_item:
        selected_items =
list(item_count[start_item:end_item].item_name)
        input_df['total_items'] = input_df[selected_items].sum(axis =
1)
        input_df = input_df[input_df.total_items >= length_trans]
        del(input_df['total_items'])
        return input_df[selected_items],item_count[start_item:end_item]
    else:
        item_count['item_perc'] = item_count['item_count']/total_items
        item_count['total_perc'] = item_count.item_perc.cumsum()
        selected_items = list(item_count[item_count.total_perc <
total_sales_perc].item_name)
        input_df['total_items'] = input_df[selected_items].sum(axis =
1)
        input_df = input_df[input_df.total_items >= length_trans]
        del(input_df['total_items'])
        return input_df[selected_items],
item_count[item_count.total_perc < total_sales_perc]

```

11. Penambahan Aturan Asosiasi Dengan Pertumbuhan FP

```

confidence = 0.6
rules_df = pd.DataFrame()

```



```

if len(itemsets) < 1000000:
    rules = [(P, Q, supp, conf)
    for P, Q, supp, conf in association_rules(itemsets, confidence)
        if len(Q) == 1 ]

    names = {item: '{}={}'.format(var.name, val)
        for item, var, val in OneHot.decode(mapping, data_tran,
mapping)}

    eligible_ante = [v for k,v in names.items() if v.endswith("1")]

    N = input_assoc_rules.shape[0]

    rule_stats = list(rules_stats(rules, itemsets, N))

    rule_list_df = []
    for ex_rule_frm_rule_stat in rule_stats:
        ante = ex_rule_frm_rule_stat[0]
        cons = ex_rule_frm_rule_stat[1]
        named_cons = names[next(iter(cons))]
        if named_cons in eligible_ante:
            rule_lhs = [names[i][:-2] for i in ante if names[i] in
eligible_ante]
            ante_rule = ', '.join(rule_lhs)
            if ante_rule and len(rule_lhs)>1 :
                rule_dict = {'support' : ex_rule_frm_rule_stat[2],
                    'confidence' : ex_rule_frm_rule_stat[3],
                    'coverage' : ex_rule_frm_rule_stat[4],
                    'strength' : ex_rule_frm_rule_stat[5],
                    'lift' : ex_rule_frm_rule_stat[6],
                    'leverage' : ex_rule_frm_rule_stat[7],
                    'antecedent': ante_rule,
                    'consequent':named_cons[:-2] }
                rule_list_df.append(rule_dict)

```

```

rules_df = pd.DataFrame(rule_list_df)
print("Raw rules data frame of {} rules
generated".format(rules_df.shape[0]))
if not rules_df.empty:
    pruned_rules_df =
rules_df.groupby(['antecedent', 'consequent']).max().reset_index()
else:
    print("Unable to generate any rule")

```

Penambahan aturan asosiasi dengan menggunakan algoritma Pertumbuhan FP (FP-Growth) adalah salah satu teknik dalam data mining yang digunakan untuk menemukan aturan asosiasi atau pola-pola tersembunyi dalam data transaksional. Konsep dasar dari FP-Growth adalah mengidentifikasi itemset yang sering muncul bersama dalam transaksi, dan dari itemset tersebut, aturan asosiasi yang kuat dapat diekstraksi.

Berikut adalah langkah-langkah utama dalam algoritma FP-Growth:

1. **Konstruksi Pohon Pertumbuhan Frequent Pattern (FP-Tree)**:**

- FP-Growth menggunakan struktur data FP-Tree untuk merepresentasikan pola itemset yang sering muncul bersama dalam data transaksional.
- Setiap node dalam FP-Tree mewakili item atau itemset yang muncul dalam satu transaksi, dan cabang-cabang dari setiap node mewakili item-item yang sering muncul bersama dalam transaksi yang sama.

2. **Konstruksi Header Table**:**

- Header table digunakan untuk menyimpan referensi ke semua node yang memiliki item yang sama dalam FP-Tree.
- Header table membantu dalam menemukan jalur-jalur yang sesuai dengan pola-pola itemset yang dicari.

3. ****Ekstraksi Itemset Frequent****:

- Dengan menggunakan FP-Tree dan header table, itemset frequent (itemset yang muncul di atas tingkat minimum support) dapat diekstraksi secara efisien.
- Itemset-itemset frequent ini akan digunakan untuk membentuk aturan asosiasi yang kuat.

4. ****Pembentukan Aturan Asosiasi****:

- Aturan asosiasi yang kuat dapat dibentuk dari itemset-itemset frequent yang telah diekstraksi.
- Aturan asosiasi ini memiliki dua bagian: "antecedent" (bagian kiri) dan "consequent" (bagian kanan), yang menunjukkan hubungan antara item-item dalam transaksi.

Algoritma FP-Growth efisien dalam menemukan pola-pola tersembunyi dalam data transaksional, terutama saat bekerja dengan dataset yang besar. Dengan memahami pola-pola itemset frequent, perusahaan dapat membuat strategi pemasaran yang lebih efektif, melakukan penempatan produk yang lebih baik, dan mengoptimalkan proses bisnis mereka secara keseluruhan.