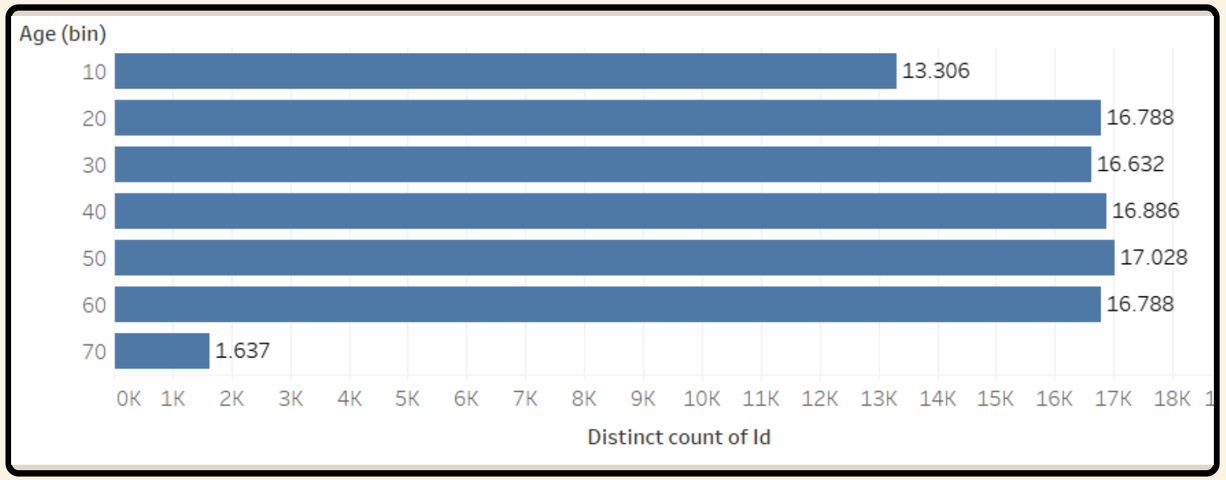
KELOMPOK 106

Jordi Ambat | Ika Sevi Saputri | Mutia Handiny
[jordiambat@gmail.com | mutiahandinyy@gmail.com | ikasevi.2018@student.uny.ac.id]

Check Point 1



WHO

Siapa target marketnya?(jenis kelamin/umur)

- Target marketing dari produk adalah merata dari jenis kelamin maupun usia sehngga kita dapat memaksimalkan keseluruhan penjualan tidak terkecuali.

WHAT

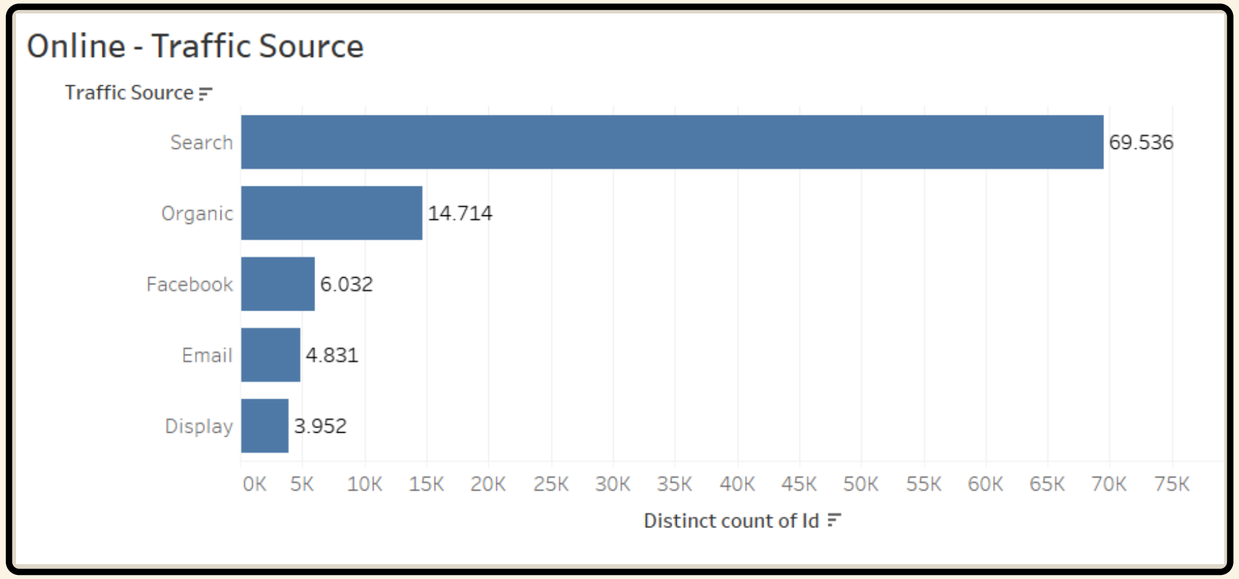
Apa yang harus dilakukan untuk meningkatkan tingkat kesetian user?

- Untuk meningkatkan tingkat kesetiaan user kita harus juga bisa menjawab apa yang menjadi keunggulan dan dari perusahaan kita, jika terdapat hal yang masih kurang kita dapat memperbaikinya

WHEN

Kapan promosi target market tepat dilakukan?

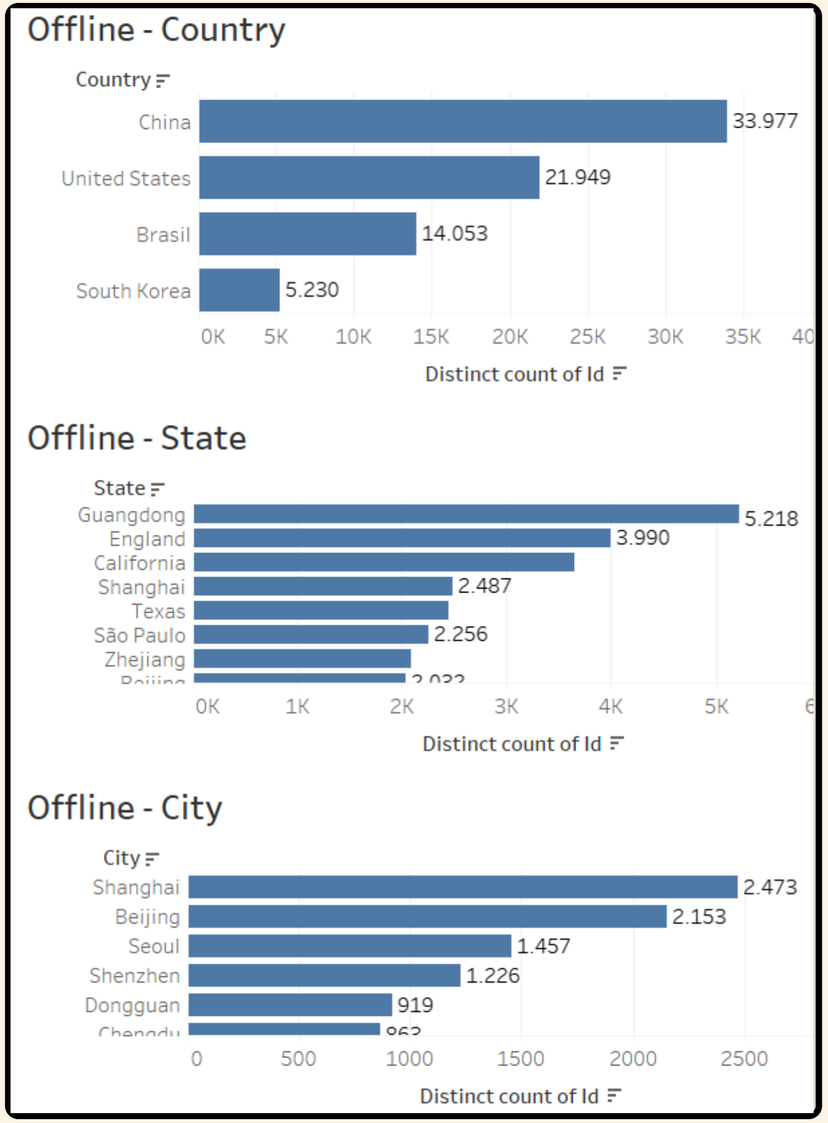
- Kita harus melihat lagi dari masing masing platform yang akan kita tingkatkan, untuk tiap platform memiliki waktu terbaik yang berbeda, dan untuk memaksimalkan promosi, promosi dapat dilakukan dengan memanfaatkan hari hari besar



WHERE

Dimana produk dapat di promosikan secara maksimal (Platform Digital) ?

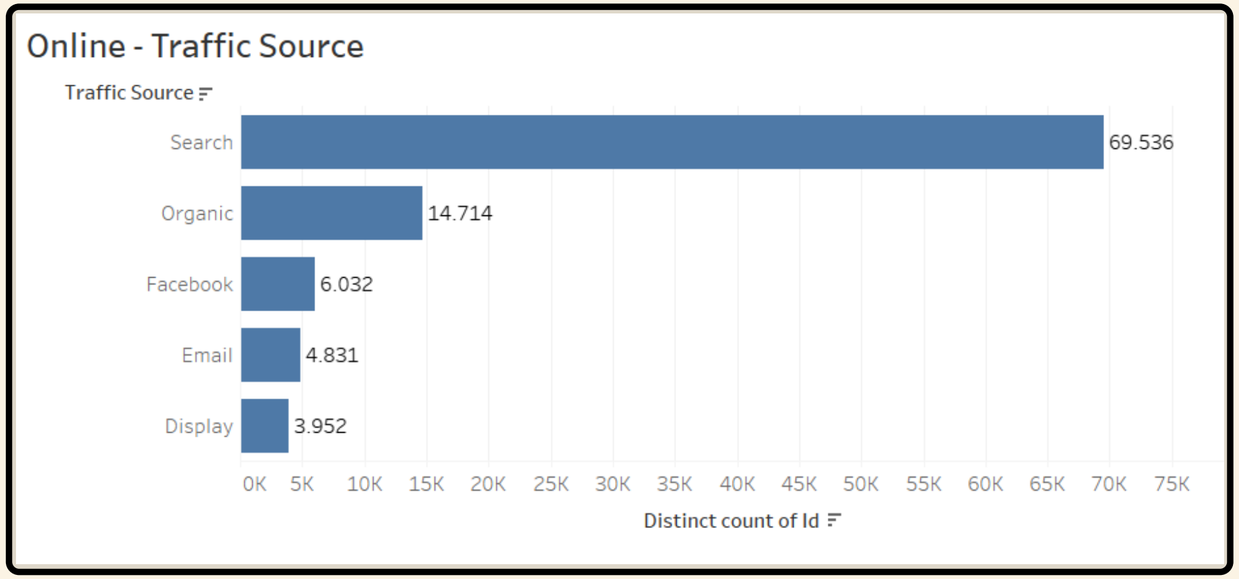
- Search Engine



WHERE

Daerah mana yang memiliki insight terbanyak terhadap produk?

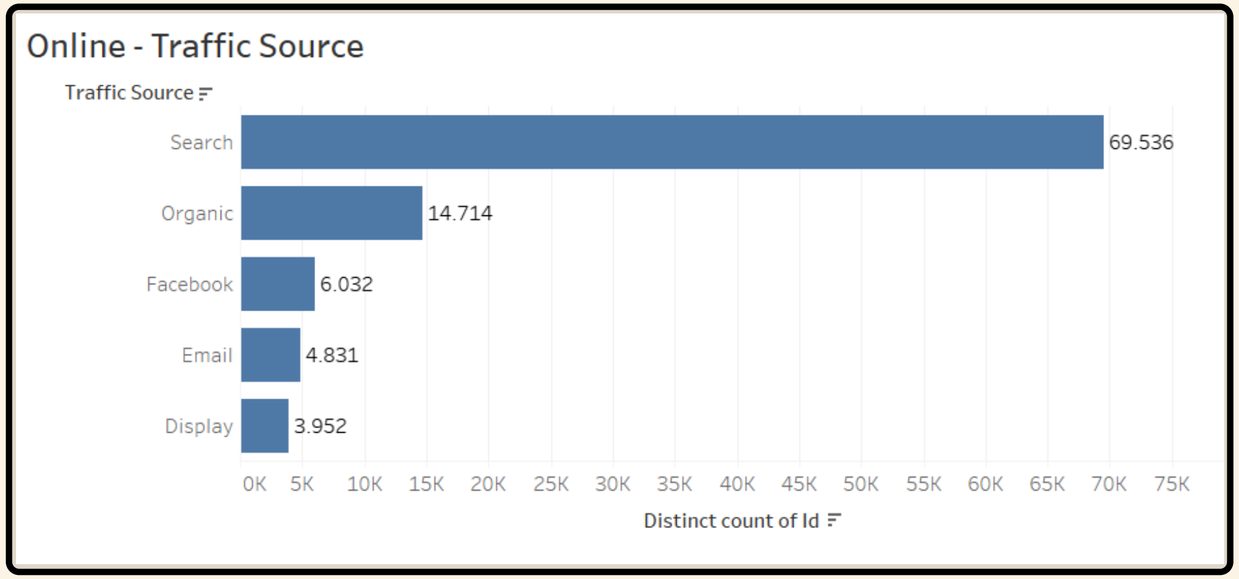
– ....



WHY

Kenapa harus mengetahui traffic yang paling tinggi

- Traffic adalah jumlah kunjungan dan berbagai aktivitas lengkap seseorang ketika mengunjungi suatu situs, dalam hal ini adalah situs promosi suatu produk. Dengan mengetahui traffic mana yang paling tinggi, maka dapat berpotensi terhadap perkembangan penjualan, yaitu dengan cara memaksimalkan penjualan/pomosi pada situs yang memiliki traffic yang paling tinggi.



HOW

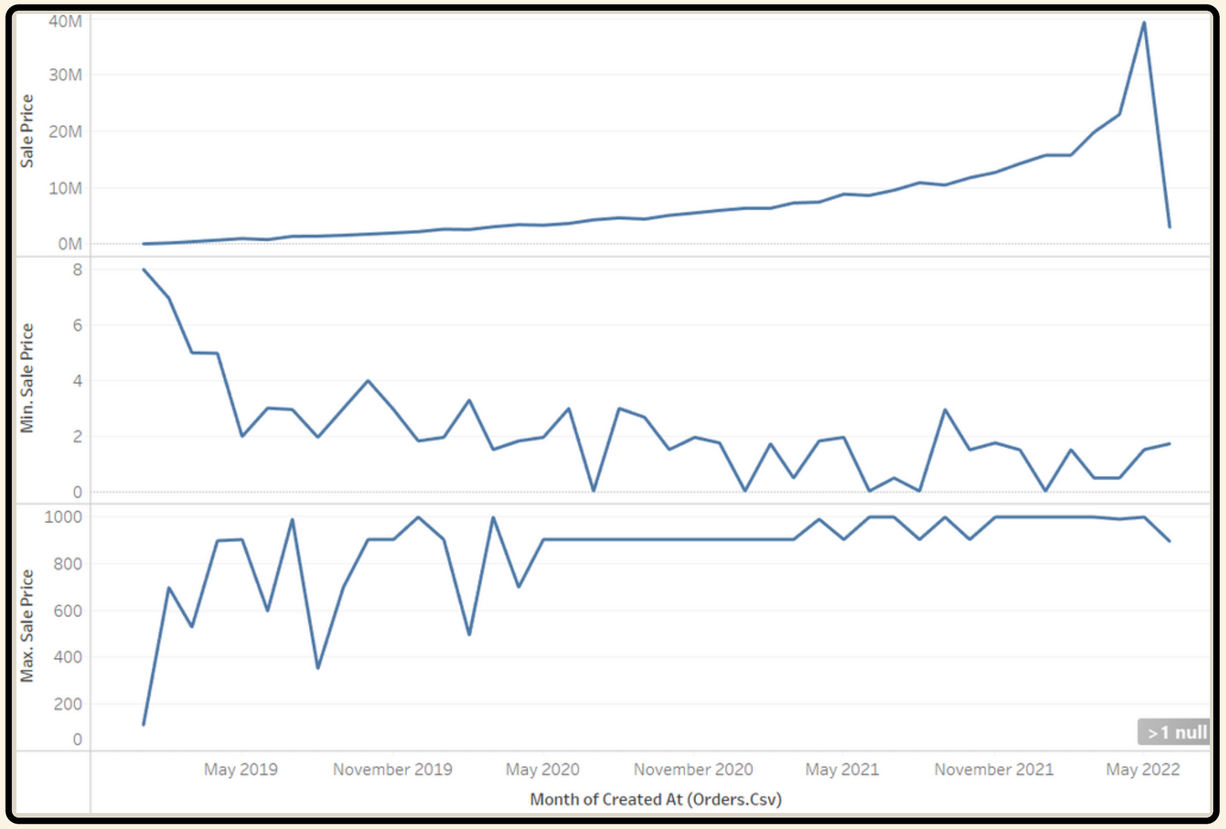
Bagaimana cara yang tepat untuk meningkatkan penjualan produk?

- Sesuai dengan *Who, Why* dan *Where* kita dapat memprioritaskan customer di daerah yang memiliki insight yang besar jika kita ingn mendapat profit, dan untuk mendapatkan traffic lebih dapat dilakukan *Search Engine Optimization*

WHO

Siapa yang dibutuhkan untuk libatkan agar pertubuhan sales dapat stabil dan membuat kerugian?

- Sales team, yang bisa berkomunikasi baik dengan customer, sehingga dalam meningkatkan suatu variabel demi mencapai suatu tujuan tanpa mengorbankan variabel lainnya



WHAT

Apa yang menjadi permalasahan utama?

- Tingkat penjualan produk rendah. yang dapat digambar dengan harga penjualan yang berulang berubah dengan tidak stabil

WHEN

Kapan masalah tersebut terjadi

- Pada tahun 2019-2022

WHERE

Dimana hal itu terjadi ?

- E-Commerce

WHY

Kenapa perusahaan membutuhkan sales team yang optimal?

1. Suatu tim yang dapat fokus menangani sales sangat dibutuhkan dikarena faktor yang menuntut agar penjualan tidak jatuh. Seperti rata-rata customer yang harus dikonfirmasi 8 kali untuk order agar mendapatkan respon. Juga sebagian besar penjualan memerlukan lima panggilan tindak lanjut sebelum kesepakatan ditutup. HIngga juga 35 hingga 50% dari semua penjualan pergi ke vendor yang merespons terlebih dahulu. Dimana dengan adanya team sales proses bisnis akan lebih efsien
2. Mendapatkan feedback dari customer lebih cepat & Perusahaan dapat melihat dari prespektif customer
3. Membangun hubungan yang lebih baik dengan customer

HOW

Bagaimana cara mnyelesaikan masalah terkait sales?

- Melakukan EDA, untuk melihat apakah ada kesalah dalam penarikan data atau ada kejanggalan dalam hasil data
- Memprediksi data dengan menggunakan timeseries forecasting agar kita bisa mengetahui low season sehingga dapat diantisipasi
- Forecasting juga dilakukan untuk mengkaji kebijakan presuhaan yang sudah dilakukan dan pengaruhnya terhadap hasil dimasa depan, seperti meningkatkan harga penjualan agar cost dapat tertutupi
- Team sales dapat mencoba mengatasi hal tersebut dengan memberikan waktu diskon sebagai uji coba

WHO

Siapa target marketnya?(jenis kelamin/umur)

- Target marketing dari produk adalah merata dari jenis kelamin maupun usia sehngga kita dapat memaksimalkan keseluruhan penjualan tidak terkecuali.

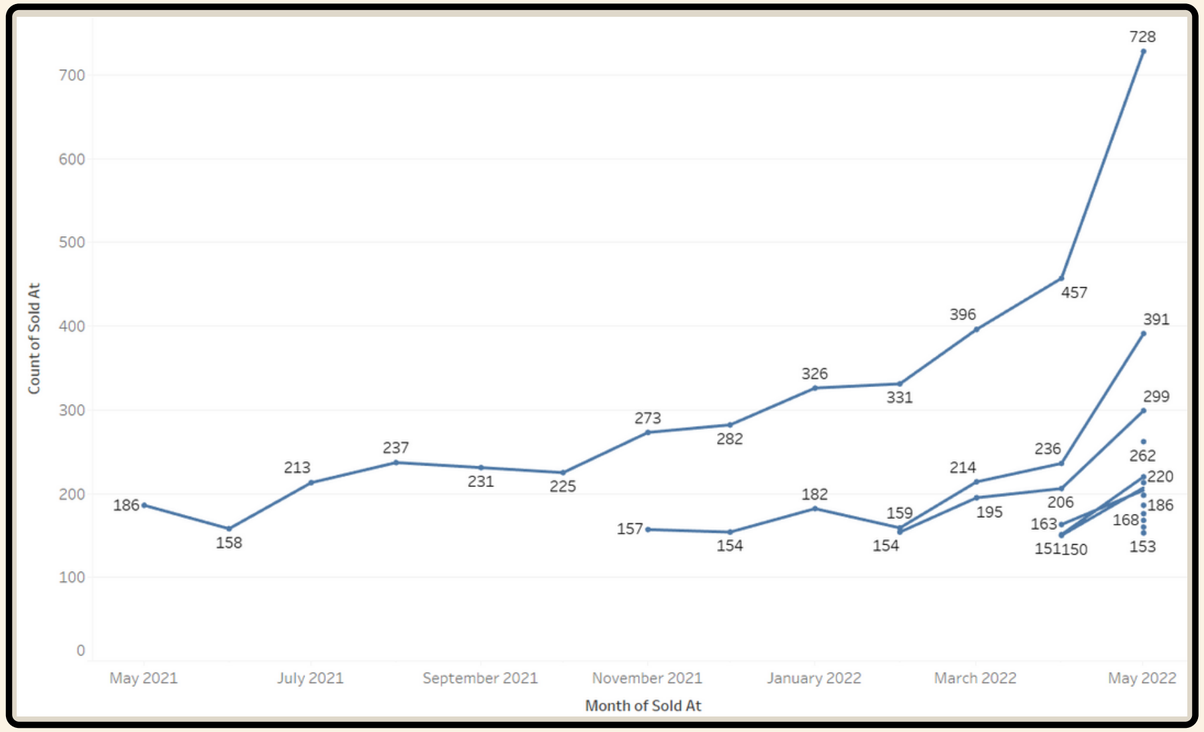
WHAT

Apa yang menjadi permasalahan utama?

- Produk yang harga penjualan tidak dapat menutupi cost, dimana ini sama dengan modal > pendapatan

Apa yang dapat jelaskan dengan melihat perbandingan harga retail dan cost dari produk

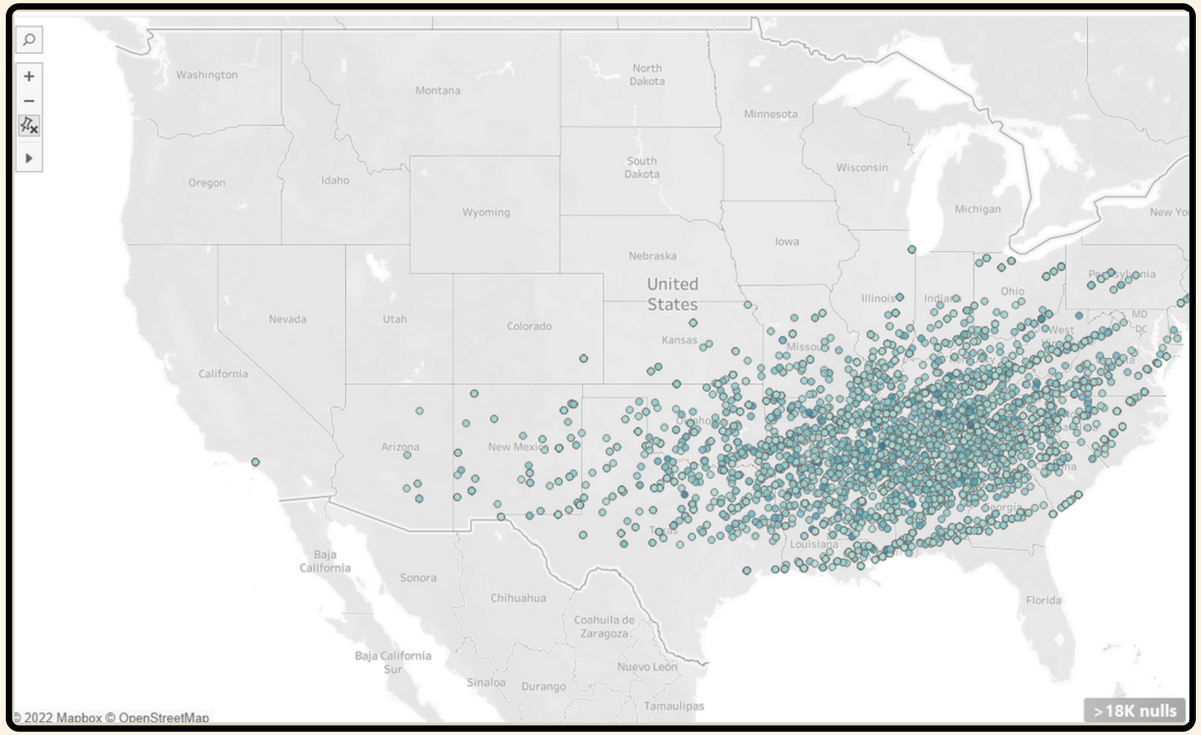
- Harga retail dari tiap produk lebih bisa menutupi cost dibandingkan pada variabel price



WHEN

Kapan penjualan produk mengalami peningkatan dan penurunan?

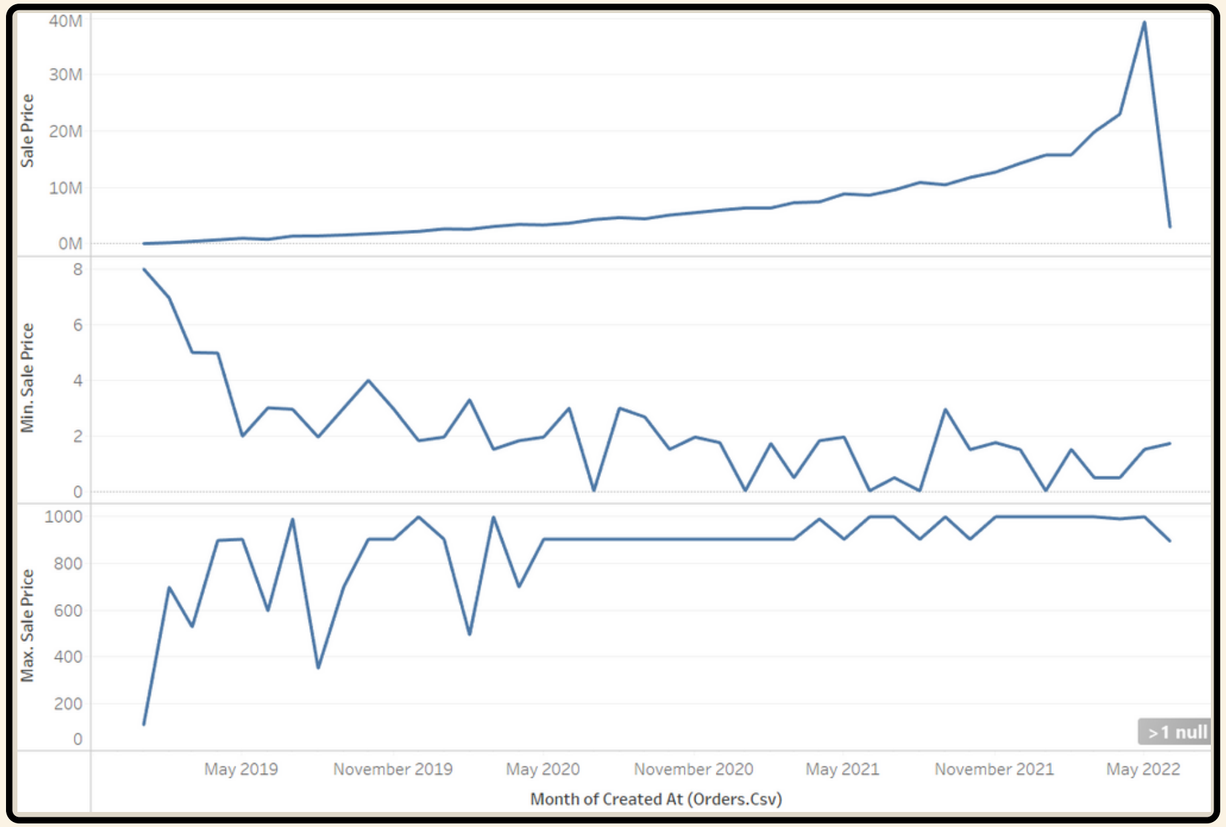
- Penjualan mengalami peningkatan seiring waktu dengan pola musiman dan untuk waktu akhir ini ada produk mengalami mengalami peningkatan yang sangat signifikan dan yang paling signifikan adalah dari brand "Allegra K"



WHERE

Dimana produk di distribusi

- Produk dari brand brand besar masih didominasi daerah amerika bagian selatan dan timur



WHY

Kenapa harus dilakukan analisis tiap produk terhadap harga jual dan biaya produksinya

- DIkarenakan hal mendasar dari bisnis adalah mendapatkan keuntungan sebanyak banyaknya maka parameter yang harus dipertahankan ialah
modal < pendapatan

HOW

Bagaimana cara yang tepat untuk mengatasi masalah dan mengkategorikan produk yang ada?

- Setiap Produk memiliki segmentasi yang berbeda dan untuk mengtasi masalah secara mendetail bisa dilakukan pengelompokan terlebih dahulu dengan melukan clustering dan selanjutnya bisa dilakukan analisis

Check Point 2

SALES

```
[4]  df = pd.read_csv('/content/order_items.csv')
```

```
[5]  df.shape
```

```
(180508, 11)
```

```
[6]  df.head(5)
```

	id	order_id	user_id	product_id	inventory_item_id	status	created_at	shipped_at	delive
0	59347	41038	33153	13606	160256	Shipped	2020-01-05 08:14:27+00:00	2020-01-06 20:56:00+00:00	
1	101072	69979	56200	13606	272958	Shipped	2022-05-17 07:06:19+00:00	2022-05-19 10:42:00+00:00	
2	63687	44006	35515	13606	171991	Complete	2021-01-21 04:56:58+00:00	2021-01-19 07:16:00+00:00	2021 03:38:00
3	16338	11275	9230	13606	44149	Cancelled	2022-05-31 08:55:48.353609+00:00	NaN	
4	85756	59418	47846	13606	231665	Cancelled	2022-05-22 11:36:45+00:00	NaN	

```
[7]  df.id.nunique()
```

```
180508
```

```
[13] df.order_id.nunique()
```

```
124512
```

```
[14] df.user_id.nunique()
```

```
79986
```

```
df.product_id.nunique()
```

```
29050
```

```
[15] df.inventory_item_id.nunique()
```

```
180508
```

```
[8]  df.info()
```

```
[15] df.inventory_item_id.nunique()
```

```
180508
```

```
[8] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180508 entries, 0 to 180507
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   id                 180508 non-null  int64  
 1   order_id           180508 non-null  int64  
 2   user_id            180508 non-null  int64  
 3   product_id         180508 non-null  int64  
 4   inventory_item_id  180508 non-null  int64  
 5   status             180508 non-null  object 
 6   created_at         180508 non-null  object 
 7   shipped_at         117502 non-null  object 
 8   delivered_at       63035 non-null   object 
 9   returned_at        17752 non-null   object 
 10  sale_price         180508 non-null  float64
dtypes: float64(1), int64(5), object(5)
memory usage: 15.1+ MB
```

Disk

[10]
```
df['created_at'] = pd.to_datetime(df['created_at'], format = '%Y%m%d %H:%M:')
df['shipped_at'] = pd.to_datetime(df['shipped_at'], format = '%Y%m%d %H:%M:')
df['delivered_at'] = pd.to_datetime(df['delivered_at'], format = '%Y%m%d %H:%M:')
df['returned_at'] = pd.to_datetime(df['returned_at'], format = '%Y%m%d %H:%M:')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180508 entries, 0 to 180507
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 180508 non-null  int64
 1   order_id           180508 non-null  int64
 2   user_id            180508 non-null  int64
 3   product_id         180508 non-null  int64
 4   inventory_item_id  180508 non-null  int64
 5   status             180508 non-null  object
 6   created_at         180508 non-null  datetime64[ns, UTC]
 7   shipped_at         117502 non-null  datetime64[ns, UTC]
 8   delivered_at       63035 non-null   datetime64[ns, UTC]
 9   returned_at        17752 non-null   datetime64[ns, UTC]
 10  sale_price         180508 non-null  float64
dtypes: datetime64[ns, UTC](4), float64(1), int64(5), object(1)
memory usage: 15.1+ MB
```

```
[12] df.isna().sum()
```

```
id                         0
order_id                   0
user_id                    0
product_id                 0
inventory_item_id          0
status                     0
created_at                 0
shipped_at             63006
delivered_at          117473
returned_at           162756
sale_price                 0
dtype: int64
```

MARKETING

```
[ ] #import libraries
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
[ ] #Load the data
    users = pd.read_csv('/content/users.csv')
```

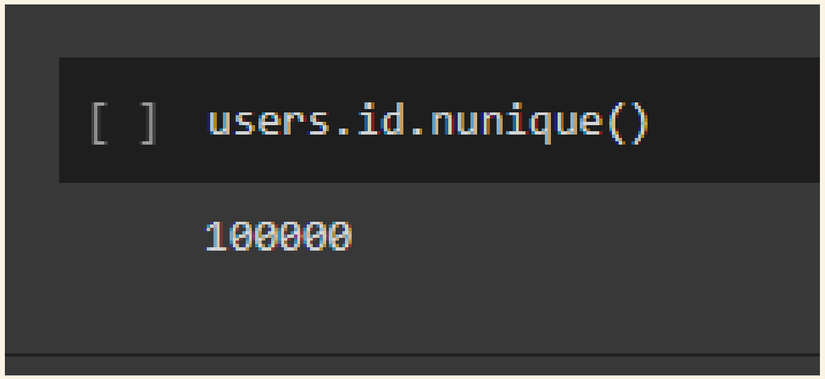
```
[ ] users.shape
```

```
(100000, 15)
```

```
users.head(5)
```

	id	first_name	last_name	email	age	gender	state	street_address	postal_code	city	country	latitude	longitude	traffic_source	created_at
0	19279	Heidi	Jackson	heidijackson@example.org	50	F	Mie	894 Nicholas Curve Suite 865	513-0836	Suzuka City	Japan	34.851814	136.508713	Search	2020-11-07 12:40:00+00:00
1	5678	Michael	Brooks	michaelbrooks@example.org	58	M	Acre	0549 Deanna Land	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	Email	2019-07-28 04:20:00+00:00
2	29694	Scott	Anderson	scottanderson@example.org	60	M	Acre	8979 Stephens Oval Apt. 816	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	Facebook	2019-08-16 17:06:00+00:00
3	29967	Mike	Beck	mikebeck@example.org	12	M	Acre	76404 Michael Way Apt. 377	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	Search	2019-05-09 12:08:00+00:00
4	47096	Holly	Kennedy	hollykennedy@example.org	68	F	Acre	23658 Santana Mission	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	Search	2019-05-24 15:53:00+00:00

MARKETING



MARKETING

```
[ ]  users.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 15 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   id              100000 non-null  int64  
 1   first_name      100000 non-null  object 
 2   last_name       100000 non-null  object 
 3   email           100000 non-null  object 
 4   age             100000 non-null  int64  
 5   gender          100000 non-null  object 
 6   state           100000 non-null  object 
 7   street_address  100000 non-null  object 
 8   postal_code     100000 non-null  object 
 9   city            99065 non-null   object 
 10  country         100000 non-null  object 
 11  latitude        100000 non-null  float64
 12  longitude       100000 non-null  float64
 13  traffic_source  100000 non-null  object 
 14  created_at      100000 non-null  object 
dtypes: float64(2), int64(2), object(11)
memory usage: 11.4+ MB
```

MARKETING

```
[ ]  users['created_at'] = pd.to_datetime(users['created_at'], format = '%Y%m%d %H:%M:')
```

```
users.info()
```

```
 7   street_address  100000 non-null  object
 8   postal_code     100000 non-null  object
 9   city            99065 non-null   object
 10  country         100000 non-null  object
 11  latitude        100000 non-null  float64
 12  longitude       100000 non-null  float64
 13  traffic_source  100000 non-null  object
 14  created_at      100000 non-null  datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](1), float64(2), int64(2), object(10)
memory usage: 11.4+ MB
```

MARKETING

```
[ ]  #Menghitung nilai yang kosong di
     users.isna().sum()
```

```
id                   0
first_name           0
last_name            0
email                0
age                  0
gender               0
state                0
street_address       0
postal_code          0
city               935
country              0
latitude             0
longitude            0
traffic_source       0
created_at           0
dtype: int64
```

```
[ ]  users_clean = users.dropna()
```

```
     users_clean.shape
```

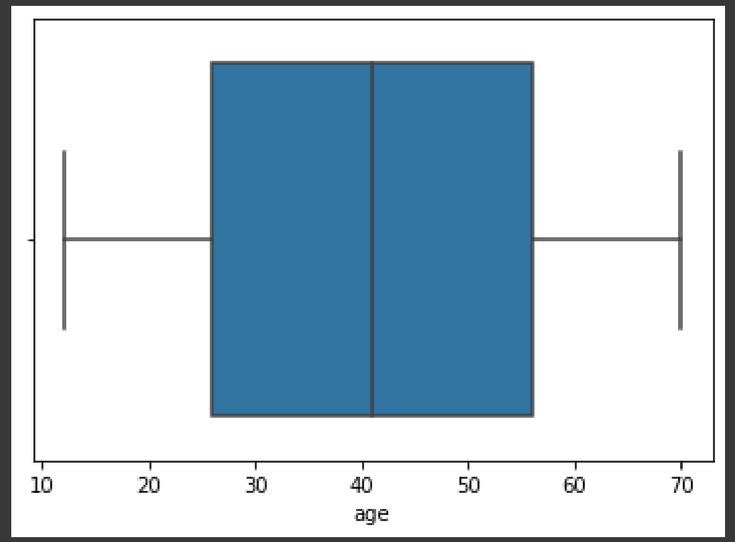
```
(99065, 15)
```

MARKETING

```
users_clean['age'].describe()
```

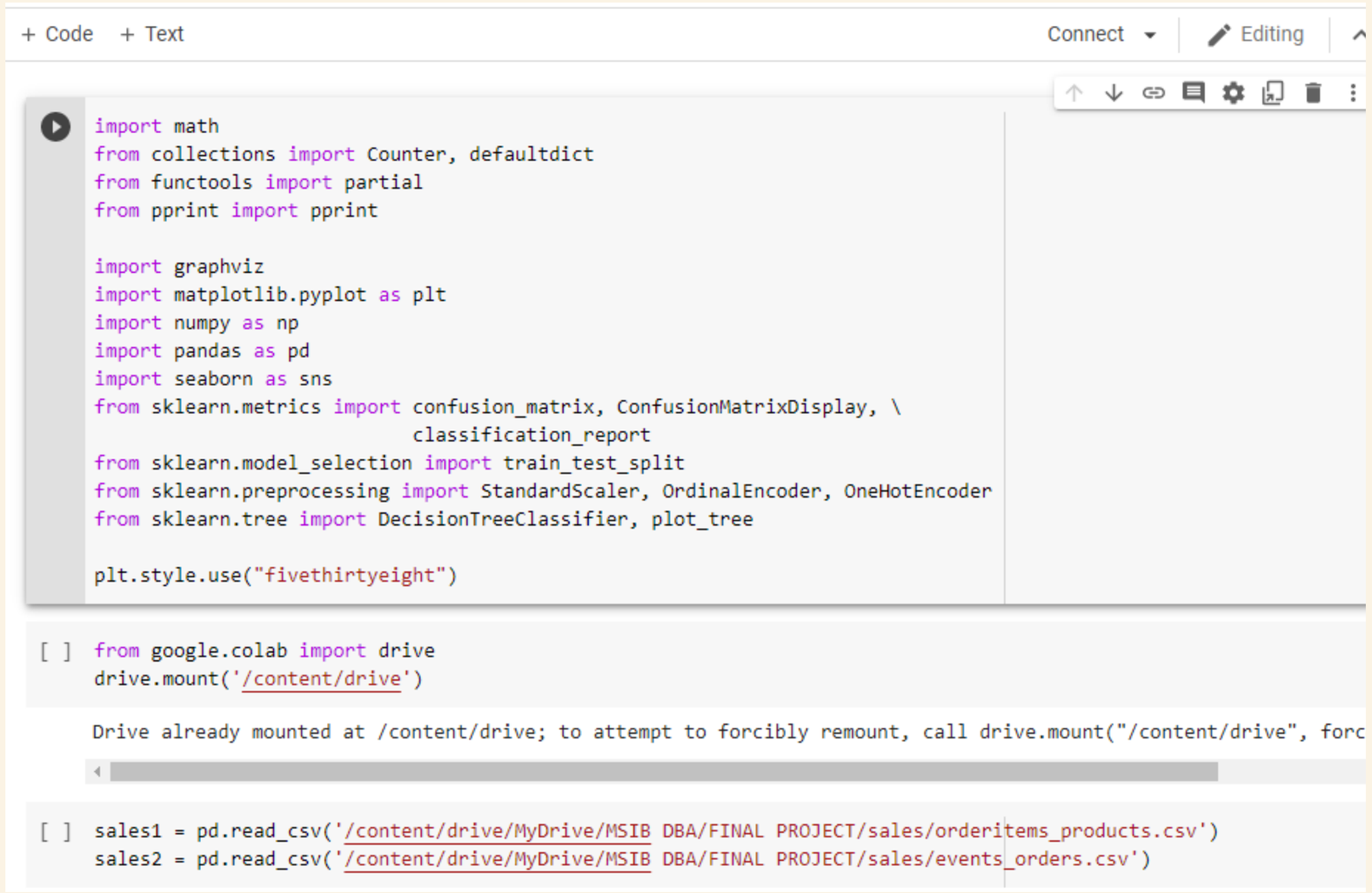
```
count    99065.000000
mean        41.054399
std         17.004206
min         12.000000
25%         26.000000
50%         41.000000
75%         56.000000
max         70.000000
Name: age, dtype: float64
```

```
sns.boxplot(x='age',data=users_clean);
```



Check Point 3

SALES



+ Code + Text Connect Editing

```
import math
from collections import Counter, defaultdict
from functools import partial
from pprint import pprint

import graphviz
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, \
                            classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OrdinalEncoder, OneHotEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree

plt.style.use("fivethirtyeight")
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
```

```
[ ] sales1 = pd.read_csv('/content/drive/MyDrive/MSIB DBA/FINAL PROJECT/sales/orderitems_products.csv')
    sales2 = pd.read_csv('/content/drive/MyDrive/MSIB DBA/FINAL PROJECT/sales/events_orders.csv')
```

SALES

```
[ ]  sales1.describe().T
```

	count	mean	std	min	25%	50%	75%	m
product_id	180508.0	15290.370105	8405.434843	1.0000	8050.00	16029.000000	22547.000000	29120.000(
inventory_item_id	180508.0	243722.590096	140711.306992	2.0000	121863.25	243832.500000	365533.000000	487394.000(
sale_price	180508.0	59.799867	65.844662	0.0200	24.90	39.990002	69.949997	999.000(
cost	180578.0	28.753807	30.626827	0.0083	11.40	19.955011	34.715039	557.151(
retail_price	180578.0	59.797859	65.841446	0.0200	24.90	39.990002	69.949997	999.000(

```
[ ]  sales2.describe().T
```

	count	mean	std	min	25%	50%	75%	max
user_id	2627403.0	5.034535e+04	28859.623713	1.0	25521.0	50506.0	75459.0	100000.0
event_id	3752901.0	1.009958e+06	660057.280667	1.0	463122.0	924444.0	1480509.0	2418734.0
sequence_number	3752901.0	3.770787e+00	2.851881	1.0	2.0	3.0	5.0	13.0
order_id	2627403.0	6.262531e+04	36016.272699	1.0	31571.0	62825.0	94014.0	124512.0
num_of_item	2627403.0	1.869644e+00	1.089224	1.0	1.0	1.0	2.0	4.0

```
[ ]  sales1.head(5)
```

```
[ ] cat = pd.Series(['city','state','browser','traffic_source','url','event_type','status'], dtype="category")
    sales2[cat].describe()
```

	city	state	browser	traffic_source	url	event_type	status
count	3714767	3752901	3752901	3752901	3752901	3752901	2627403
unique	8781	231	5	7	35530	6	5
top	Shanghai	Guangdong	Chrome	Search	/cart	product	Shipped
freq	91078	199811	1871800	1840062	944877	1194699	790368

```
[ ] sales1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180578 entries, 0 to 180577
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   product_id         180508 non-null  float64
 1   inventory_item_id  180508 non-null  float64
 2   status             180508 non-null  object 
 3   created_at         180508 non-null  object 
 4   shipped_at         117502 non-null  object 
 5   delivered_at       63035 non-null   object 
 6   returned_at        17752 non-null   object 
 7   sale_price         180508 non-null  float64
 8   cost               180578 non-null  float64
 9   category           180578 non-null  object 
10   name               180566 non-null  object
```

```
[ ]  na1 = sales1.isnull().sum()
     na2 = sales2.isnull().sum()
```

```
[ ]  na1
```

```
product_id              70
inventory_item_id       70
status                  70
created_at              70
shipped_at           63076
delivered_at        117543
returned_at         162826
sale_price              70
cost                     0
category                 0
name                    12
brand                  144
retail_price             0
department               0
sku                      0
dtype: int64
```

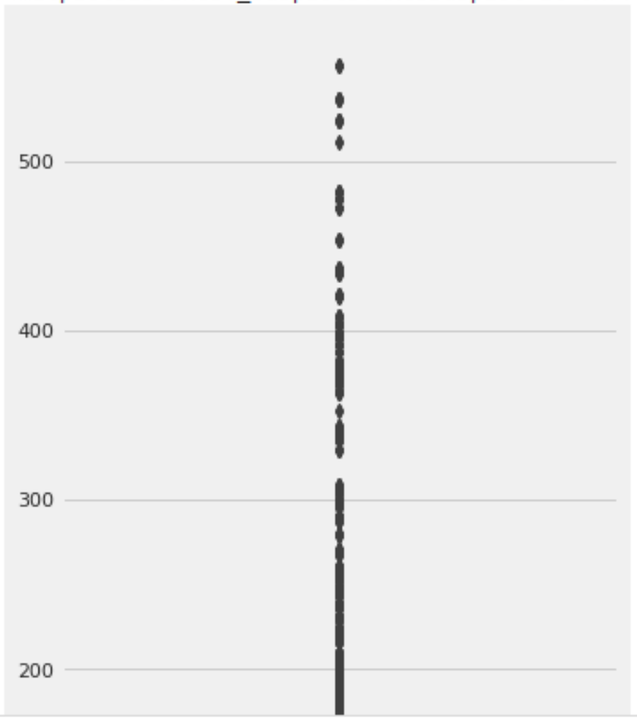
```
[ ]  na2
```

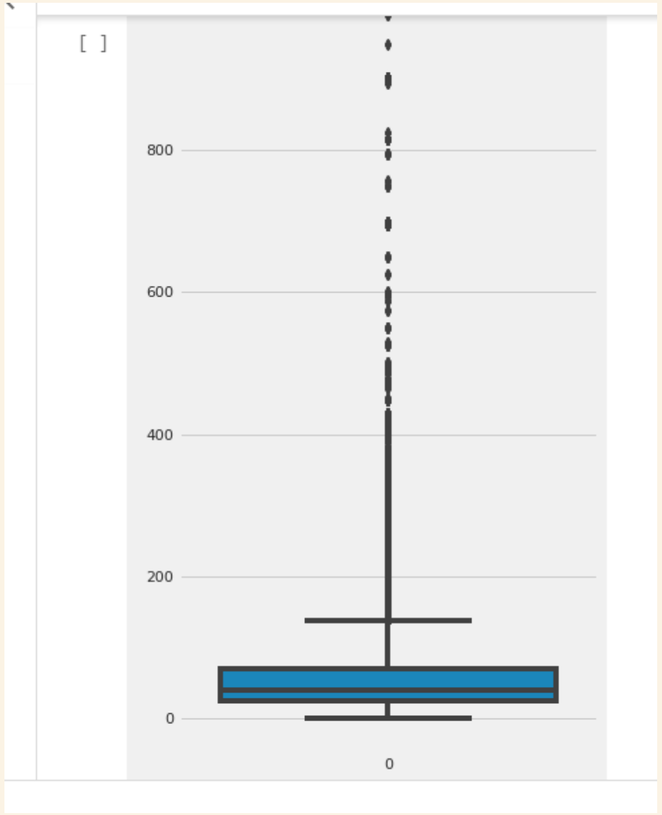
```
user_id            1125498
event_id                 0
sequence_number          0
session_id               0
ip_address               0
```

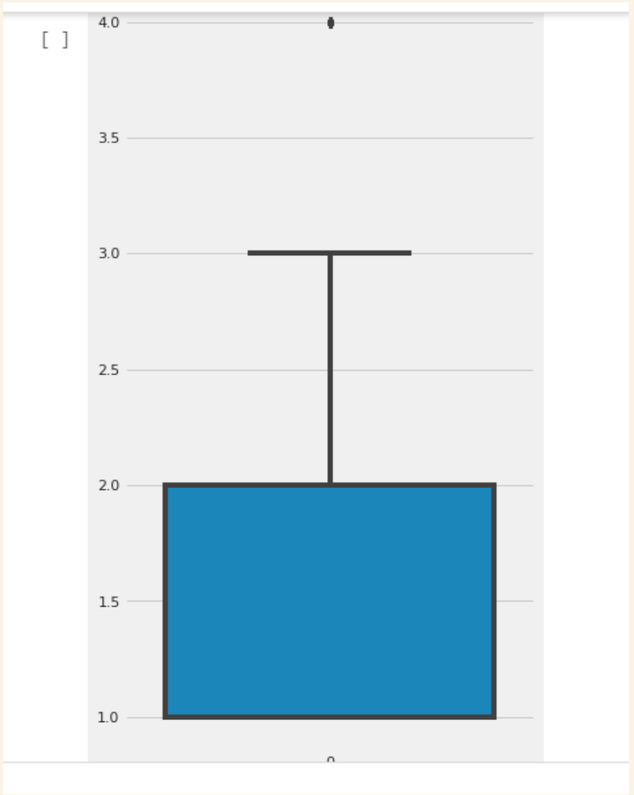
```
[ ]  na1_clean = sales1.dropna()
     na2_clean = sales2.dropna()
```

```
[ ]  fig, ax = plt.subplots(figsize=(5,10))
     sns.boxplot(data = sales1['cost'])
```

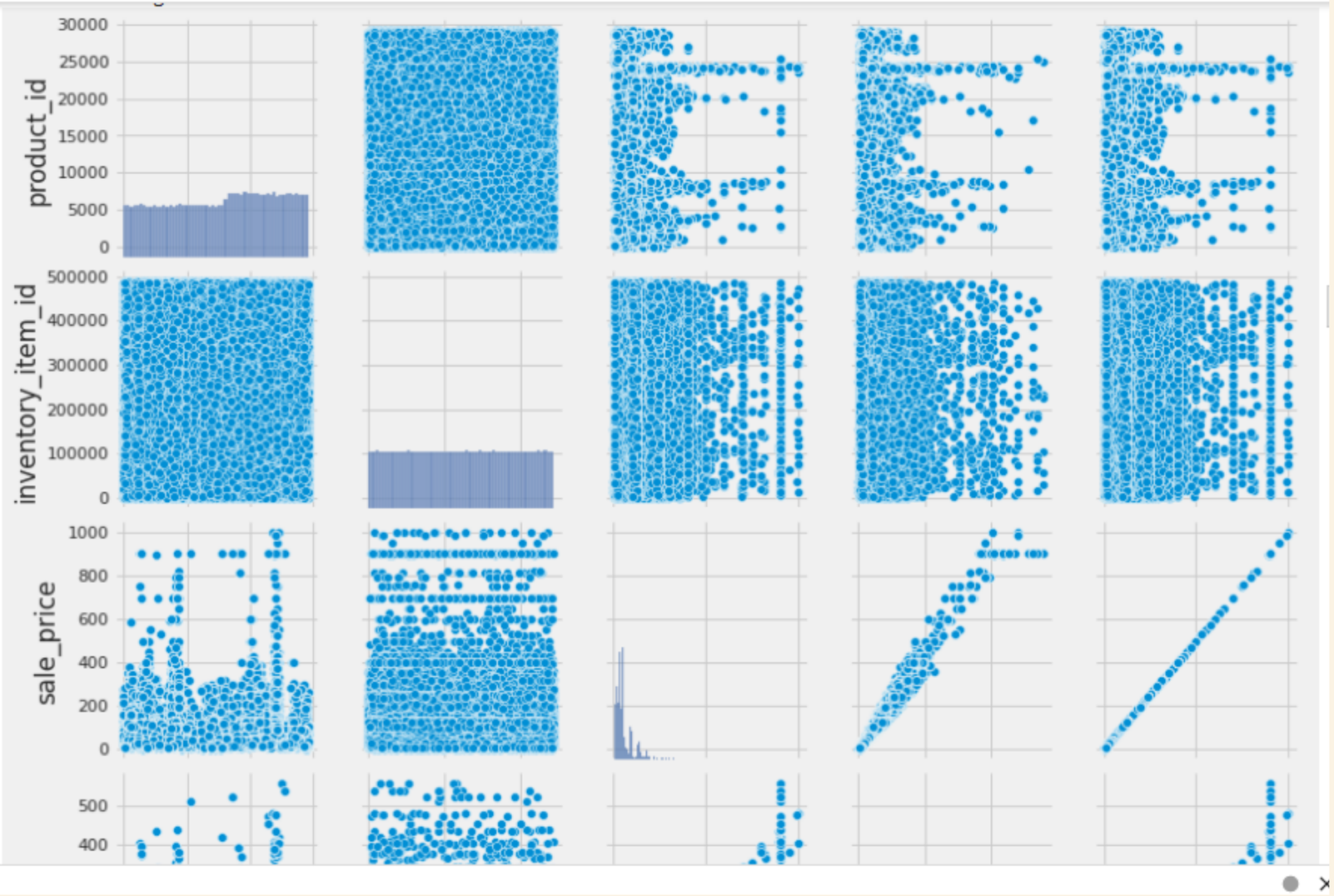
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4656f4fd0>
```




[ ]
800
600
400
200
0
0

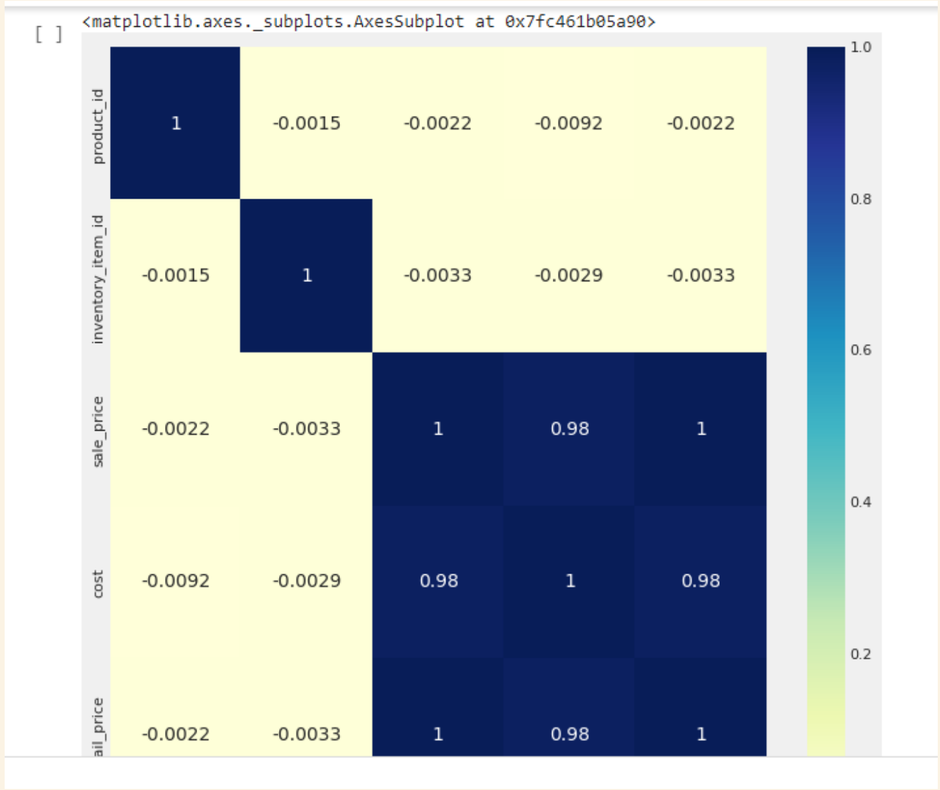

[ ]
4.0
3.5
3.0
2.5
2.0
1.5
1.0

[ ]

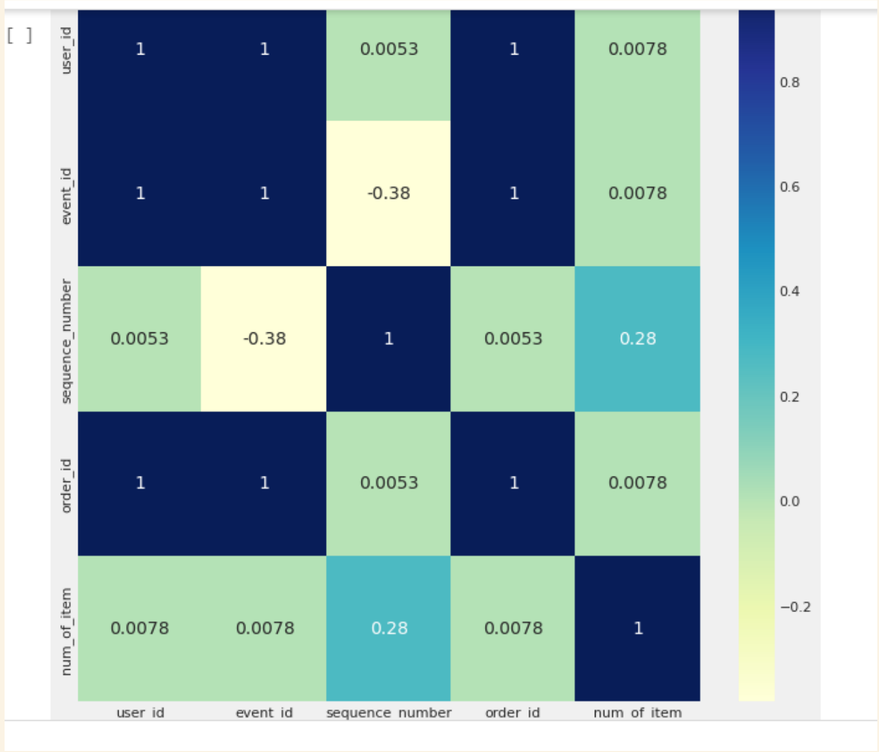

product_id
30000
25000
20000
15000
10000
5000
0
inventory_item_id
500000
400000
300000
200000
100000
0
sale_price
1000
800
600
400
200
0
500
400

[ ]

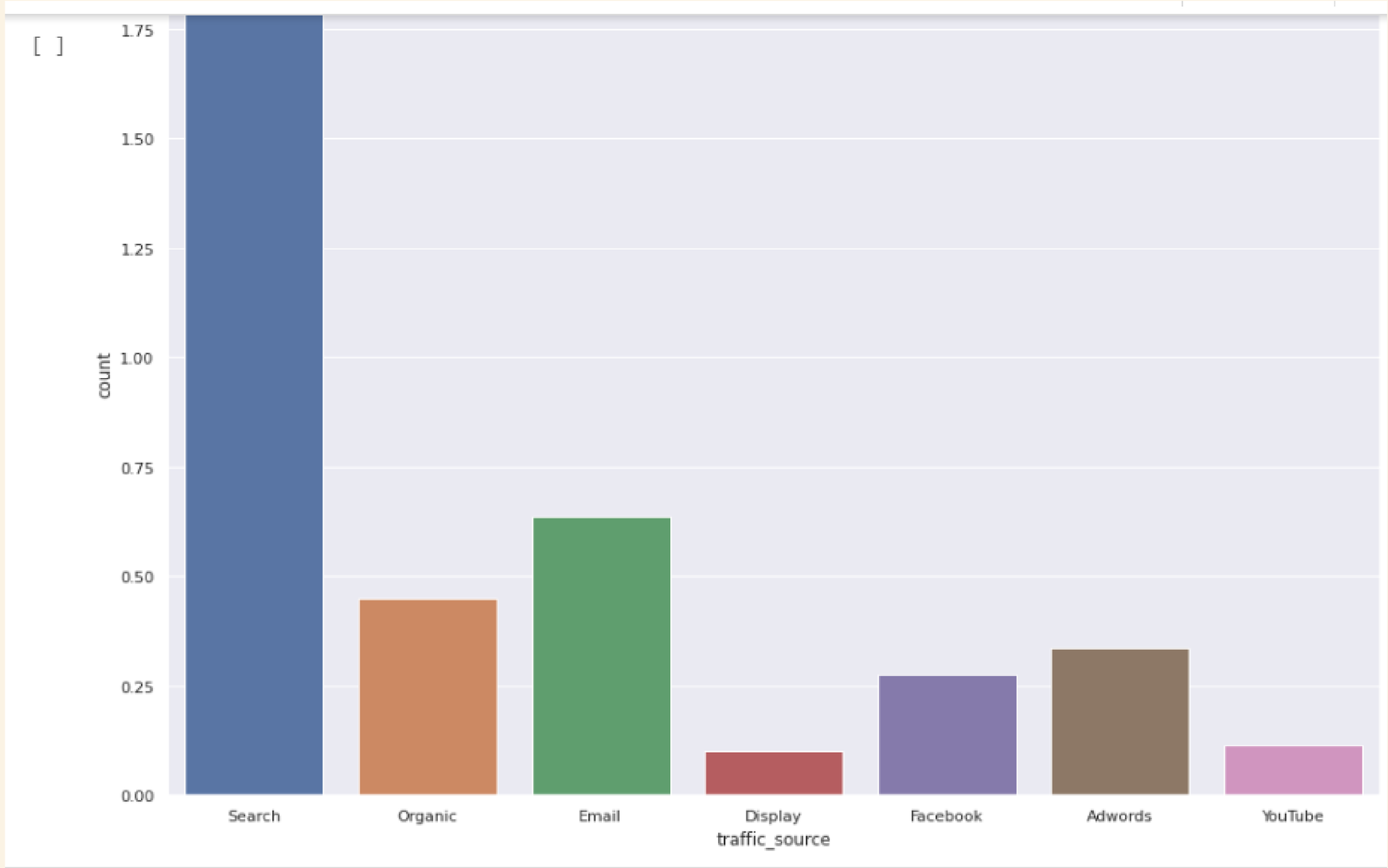
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc461b05a90>
```



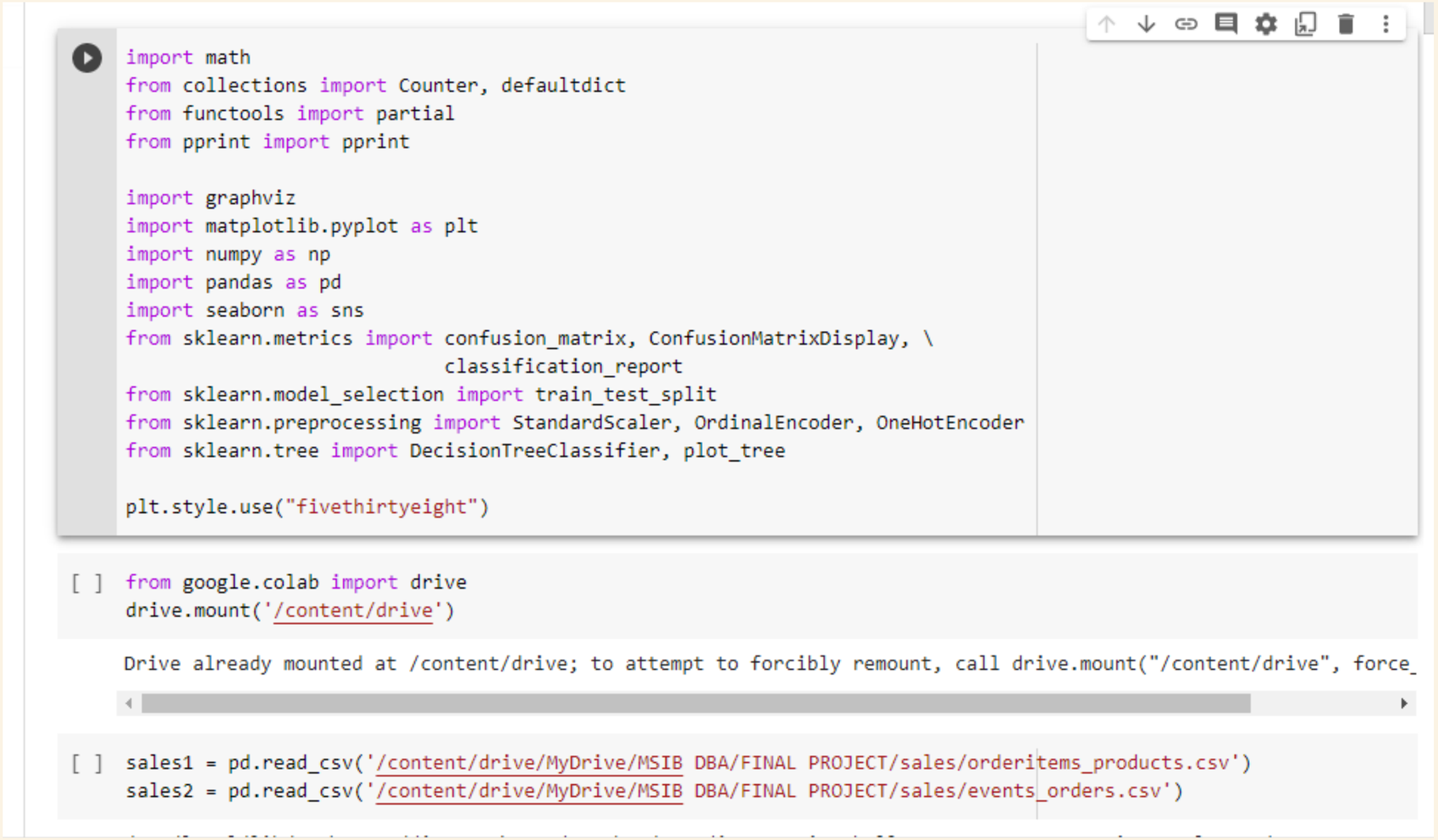
[ ]



[ ]


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
count
Search
Organic
Email
Display
Facebook
Adwords
YouTube
traffic_source

MARKETING



```
import math
from collections import Counter, defaultdict
from functools import partial
from pprint import pprint

import graphviz
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, \
                            classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OrdinalEncoder, OneHotEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree

plt.style.use("fivethirtyeight")
```

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_
```

```
[ ]  sales1 = pd.read_csv('/content/drive/MyDrive/MSIB DBA/FINAL PROJECT/sales/orderitems_products.csv')
     sales2 = pd.read_csv('/content/drive/MyDrive/MSIB DBA/FINAL PROJECT/sales/events_orders.csv')
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

DATASET USERS (Check Point 3)

```
[ ] #Load the data
    df = pd.read_csv('/content/drive/MyDrive/data/users.csv')
```

```
[ ] #Melihat kolom dan baris pada data
    df.shape
```

```
(100000, 15)
```

```
[ ] #Menampilkan Data 10 Teratas

    df.head(10)
```

	id	first_name	last_name	email	age	gender	state	street_address	postal_code	city	country	latitude	longitude	tra
0	19279	Heidi	Jackson	heidijackson@example.org	50	F	Mie	894 Nicholas Curve Suite 865	513-0836	Suzuka City	Japan	34.851814	136.508713	
1	5678	Michael	Brooks	michaelbrooks@example.org	58	M	Acre	0549 Deanna Land	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	
2	29694	Scott	Anderson	scottanderson@example.org	60	M	Acre	8979 Stephens Oval Apt. 816	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	
3	29967	Mike	Beck	mikebeck@example.org	12	M	Acre	76404 Michael Way Apt. 377	69917-400	Rio Branco	Brasil	-9.945568	-67.835610	

```
[ ] #Analis Statistik
    df.describe()
```

	id	age	latitude	longitude
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	50000.500000	41.057690	28.210711	25.257163
std	28867.657797	17.000799	22.176740	89.663548
min	1.000000	12.000000	-43.253132	-158.164931
25%	25000.750000	26.000000	26.057472	-50.000131
50%	50000.500000	41.000000	35.242914	4.812536
75%	75000.250000	56.000000	40.724087	116.385652
max	100000.000000	70.000000	64.865194	153.543292

Jika dilihat dari variabel Age, Rata-rata users berumur 41 tahun, dengan minimal umur pengguna(user) 12 tahun, dan umur maksimal 70 tahun

```
[ ] #Melihat data unik
    df.id.nunique()
```

```
100000
```

Tidak ada data terduplikasi

```
[ ]  #Melihat info yang terdapat pada dataset users
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 15 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   id              100000 non-null  int64  
 1   first_name      100000 non-null  object 
 2   last_name       100000 non-null  object 
 3   email           100000 non-null  object 
 4   age             100000 non-null  int64  
 5   gender          100000 non-null  object 
 6   state           100000 non-null  object 
 7   street_address  100000 non-null  object 
 8   postal_code     100000 non-null  object 
 9   city            99065 non-null   object 
 10  country         100000 non-null  object 
 11  latitude        100000 non-null  float64
 12  longitude       100000 non-null  float64
 13  traffic_source  100000 non-null  object 
 14  created_at      100000 non-null  object 
dtypes: float64(2), int64(2), object(11)
memory usage: 11.4+ MB
```

```
[ ]  #Menghitung nilai yang kosong di setiap kolom(missing value)
     df.isna().sum()
```

```
id                0
first_name        0
last_name         0
email             0
age               0
```

Terdapat missing value pada variabel "city"

```
[ ]  #Cleaning Data
     df_clean = df.dropna()
```

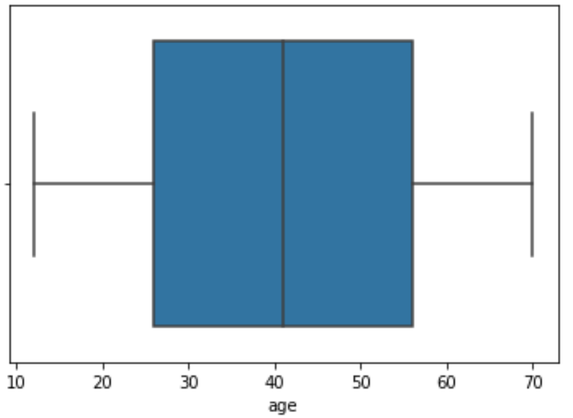
```
[ ]  df_clean.shape
```

```
(99065, 15)
```

```
[ ]  #Statistik Data
     df_clean['age'].describe()
```

```
count    99065.000000
mean        41.054399
std         17.004206
min         12.000000
25%         26.000000
50%         41.000000
75%         56.000000
max         70.000000
Name: age, dtype: float64
```

```
[ ]  #Outlier
     sns.boxplot(x='age',data=df_clean);
```



Tidak terdapat outlier pada variabel "Age"

```
[ ]  #Menampilkan Statistik Deskriptif
     df['traffic_source'].describe()
```

```
count     100000
unique         5
top       Search
freq       70181
Name: traffic_source, dtype: object
```

```
[ ]  #Mengetahui jumlah tiap kategori pada data
     df['traffic_source'].value_counts()
```

```
freq       70181
Name: traffic_source, dtype: object
```

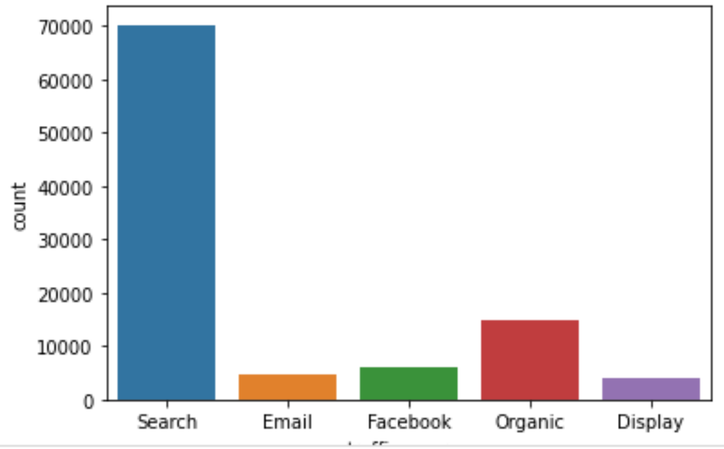
```
[ ]  #Mengetahui jumlah tiap kategori pada data
     df['traffic_source'].value_counts()
```

```
Search      70181
Organic     14866
Facebook     6098
Email        4863
Display      3992
Name: traffic_source, dtype: int64
```

Top three traffic (Search, Organic dan Facebook)

```
[ ]  #Visualisasi jumlah tiap kategori pada data pada variabel traff
     sns.countplot(df['traffic_source'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfe190cc50>
```



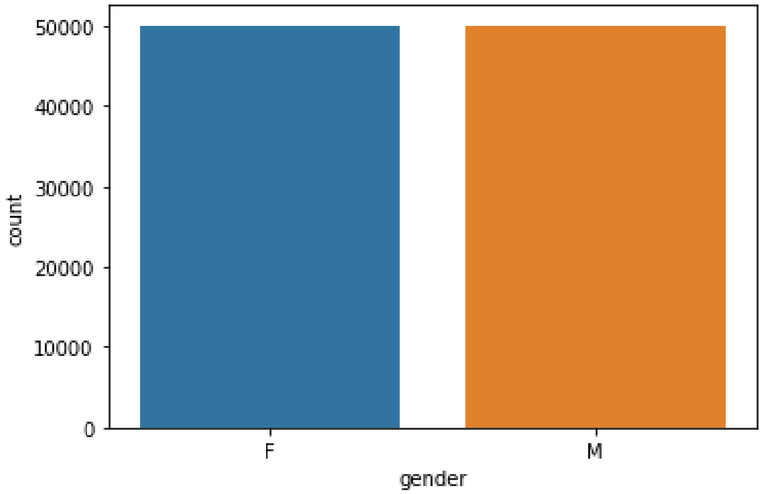
Top 1 traffic ada pada search

```
[ ]  #Melihat Jumlah customer berdasarkan gender
     df['gender'].value_counts()
```

```
F    50053
M    49947
Name: gender, dtype: int64
```

```
[ ]  #Visualisasi jumlah cust berdasarkan gender
     sns.countplot(df['gender'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfe1918690>
```



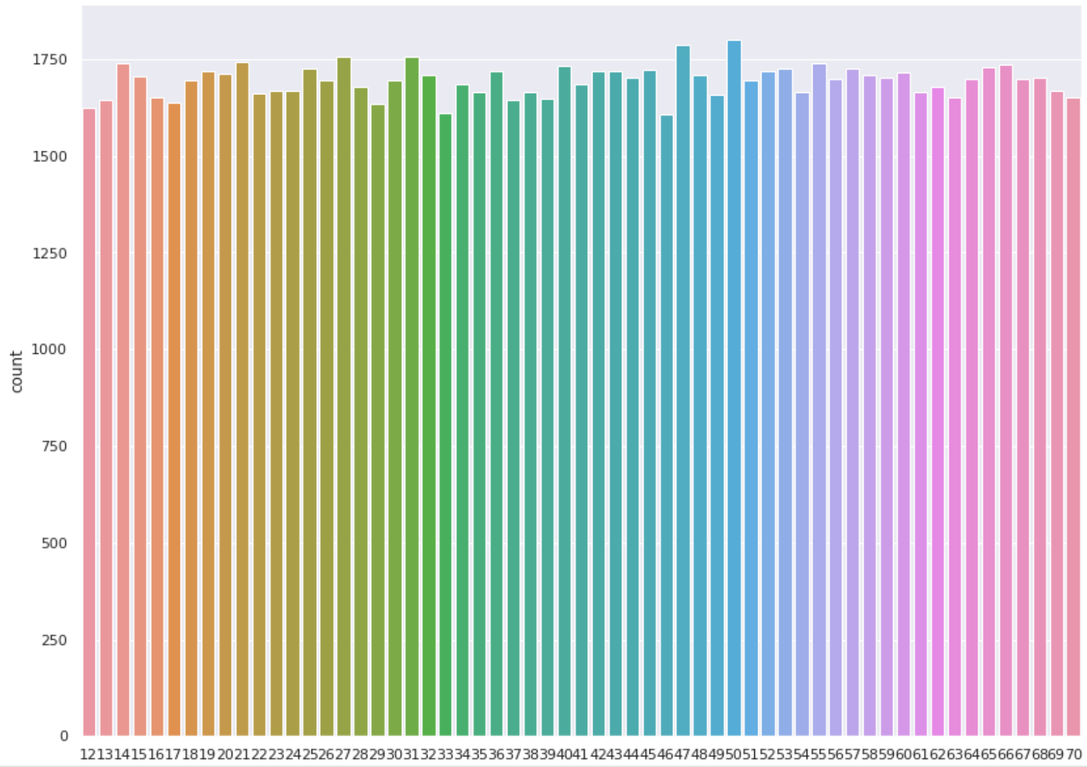
Jika diperhatikan berdasrkan visualisasi countplot, jumlah users F dan M mengalami kemiripan (hampir seimbang), bahkan cenderung sama

```
[ ]  #Melihat umur customer/users
     df['age'].value_counts()
```

```
50    1802
47    1786
27    1757
31    1756
21    1742
55    1741
14    1740
66    1735
40    1734
65    1731
25    1727
57    1727
53    1725
45    1723
19    1721
43    1721
42    1721
52    1721
36    1720
60    1715
20    1711
48    1710
58    1709
32    1709
15    1706
59    1704
68    1703
44    1701
64    1700
```

```
sns.countplot(df['age'])
```

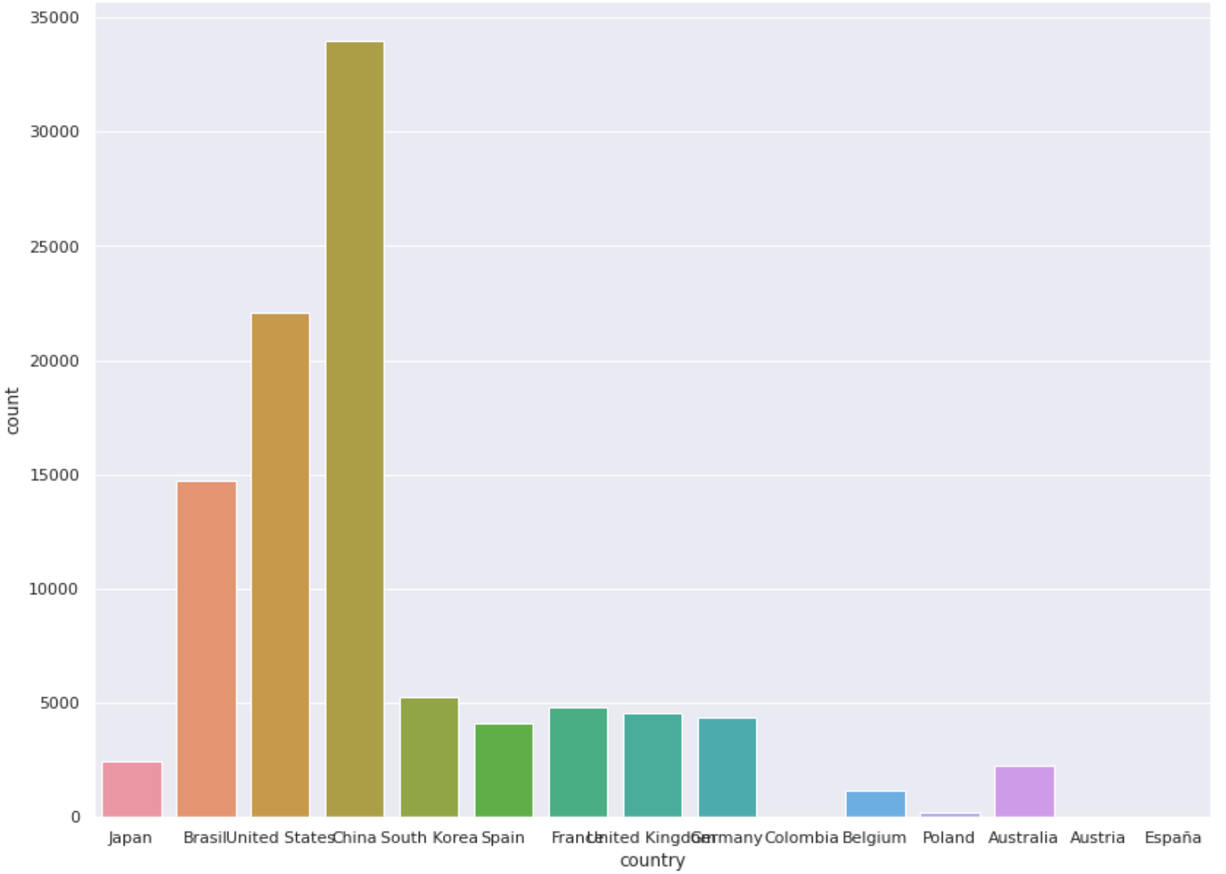
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfe13cb890>
```



```
[ ]  #Mengetahui jumlah tiap kategori pada data
     df['country'].value_counts()
```

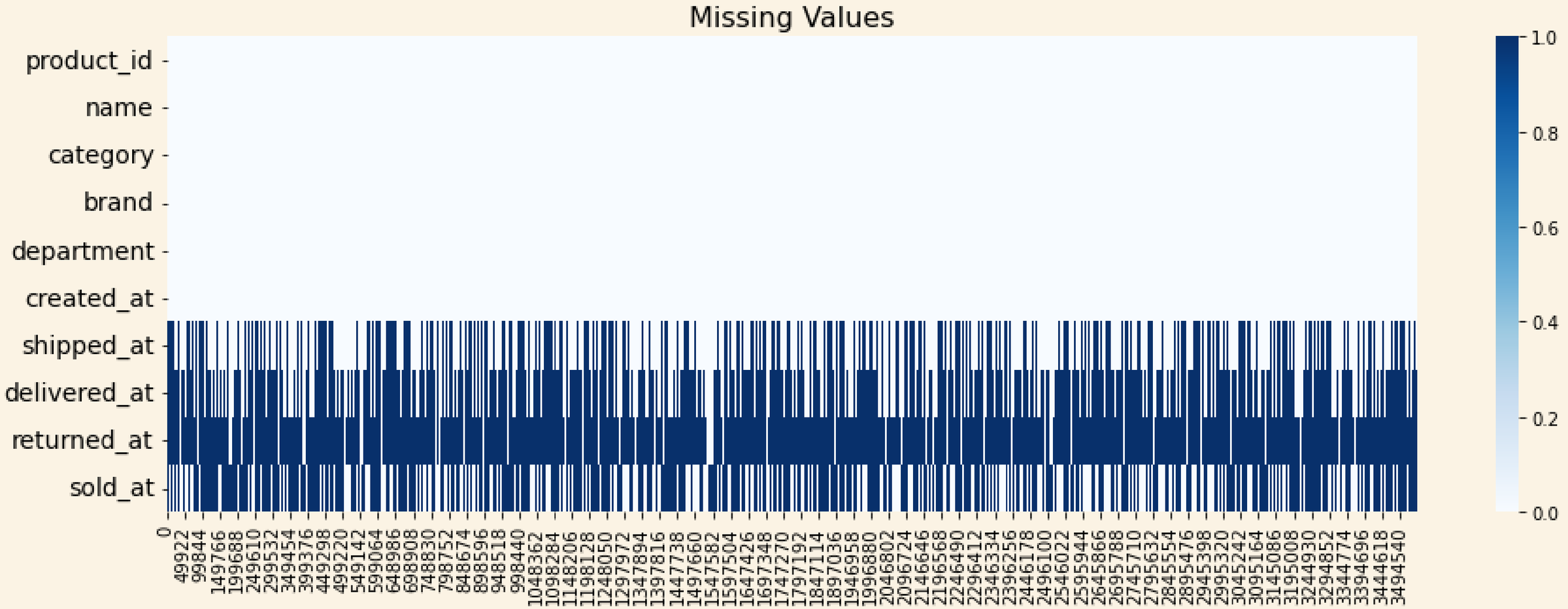
```
China             34001
United States     22107
Brasil            14717
South Korea        5279
France             4789
United Kingdom     4538
Germany            4340
Spain              4129
Japan              2425
Australia          2233
Belgium            1200
Poland              213
Colombia             16
Austria               8
España                5
Name: country, dtype: int64
```

```
[ ]  #Visualisasi
     sns.set(rc={'figure.figsize':(13.7,10.27)})
     sns.countplot(df['country'])
```



Check Point 4

PRODUCT


Missing Values
product_id
name
category
brand
department
created_at
shipped_at
delivered_at
returned_at
sold_at
1.0
0.8
0.6
0.4
0.2
0.0

PRODUCT

created_at, Length: 180172
shipped_at, Length: 77625,
delivered_at, Length: 42494,
sold_at, Length: 180172,
returned_at, Length: 12065,

PRODUCT

```
encoder = OneHotEncoder()
encoder.fit(df.select_dtypes(include="object"))
X_onehot = pd.DataFrame(encoder.transform(df.select_dtypes(include="object")).toarray(),columns=encoder.get_feature_names_out())

X_preprocessed = pd.concat([df.select_dtypes(exclude="object"), X_onehot],axis=1)

scaler = StandardScaler()
scaler.fit(X_preprocessed)
X_scaled = pd.DataFrame(scaler.transform(X_preprocessed),columns=X_preprocessed.columns)

X_scaled.describe()
```

	product_id	retail_price	cost	sale_price	clusters_of_5	clusters_of_10	clusters_of_15
count	3.544381e+06	3.544381e+06	3.544381e+06	3.544381e+06	3.544381e+06	3.544381e+06	3.544381e+06
mean	9.301817e-18	-5.645241e-18	-1.283009e-19	-5.645241e-18	-7.576169e-17	-8.682765e-17	-9.019555e-17
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-1.906984e+00	-9.068294e-01	-9.292008e-01	-9.068294e-01	-1.380424e+00	-1.539444e+00	-1.617255e+00
25%	-8.481075e-01	-5.430103e-01	-5.705320e-01	-5.430103e-01	-6.744635e-01	-8.518029e-01	-9.278856e-01
50%	1.420139e-01	-3.004643e-01	-2.770756e-01	-3.004643e-01	3.149691e-02	-1.641616e-01	-8.726534e-03
75%	8.525631e-01	1.391504e-01	1.794120e-01	1.391504e-01	7.374573e-01	8.673002e-01	9.104325e-01
max	1.580960e+00	1.423714e+01	1.723249e+01	1.423714e+01	1.443418e+00	1.554941e+00	1.599802e+00

PRODUCT

```
K = 5
kmeans_5 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_5.fit(df)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
Done fitting kMeans in 41.438s
```

```
K = 2
kmeans_2 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_2.fit(df)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
Done fitting kMeans in 13.618s
```

```
K = 10
kmeans_10 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_10.fit(df)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
Done fitting kMeans in 54.775s
```

```
K = 15
kmeans_15 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_15.fit(df)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
K = 5
kmeans_5 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_5.fit(X_scaled)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
Done fitting kMeans in 28.890s
```

```
K = 10
kmeans_10 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_10.fit(X_scaled)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

```
Done fitting kMeans in 64.925s
```

```
K = 15
kmeans_15 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_15.fit(X_scaled)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

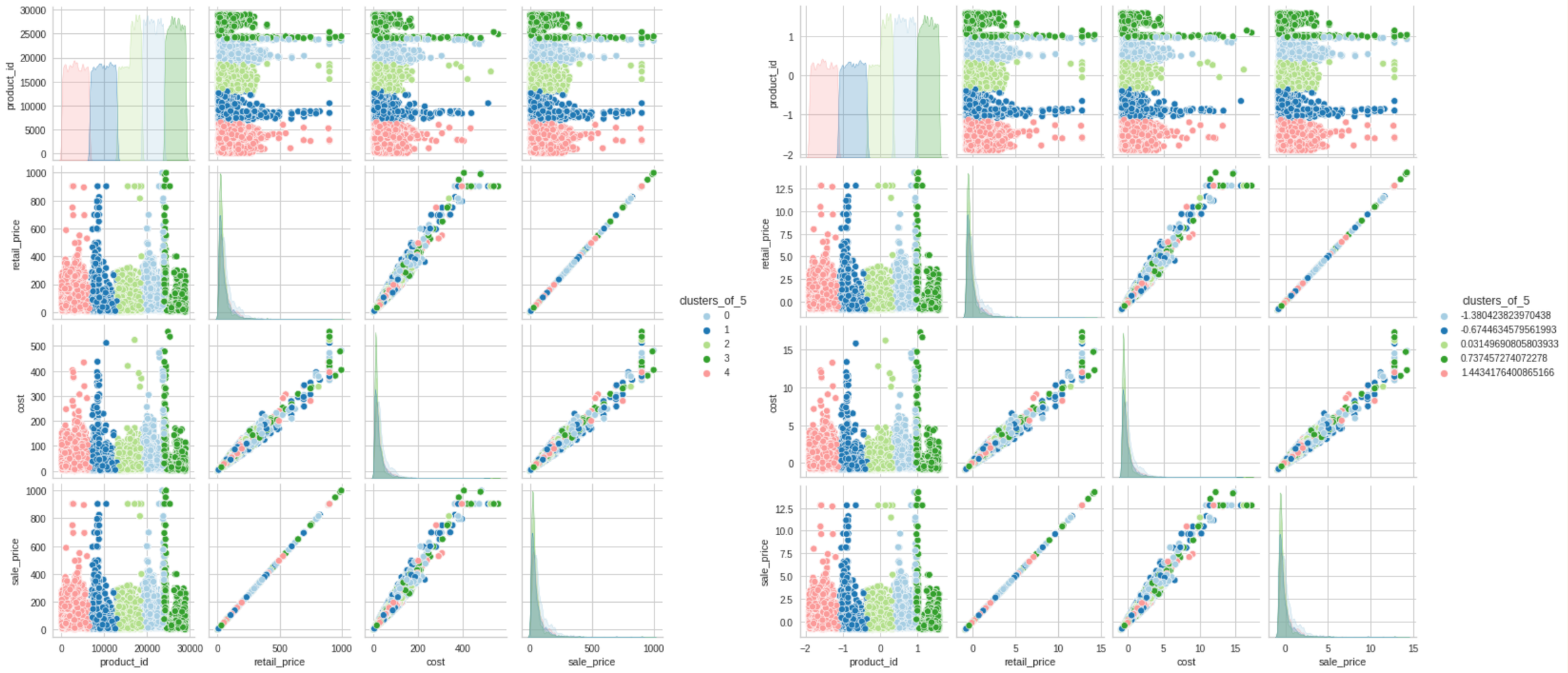
```
Done fitting kMeans in 93.204s
```

```
K = 5
kmeans_5 = KMeans(n_clusters=K, random_state=11)

start = time()
kmeans_5.fit(df)
print(f"Done fitting kMeans in {time()-start:.3f}s")
```

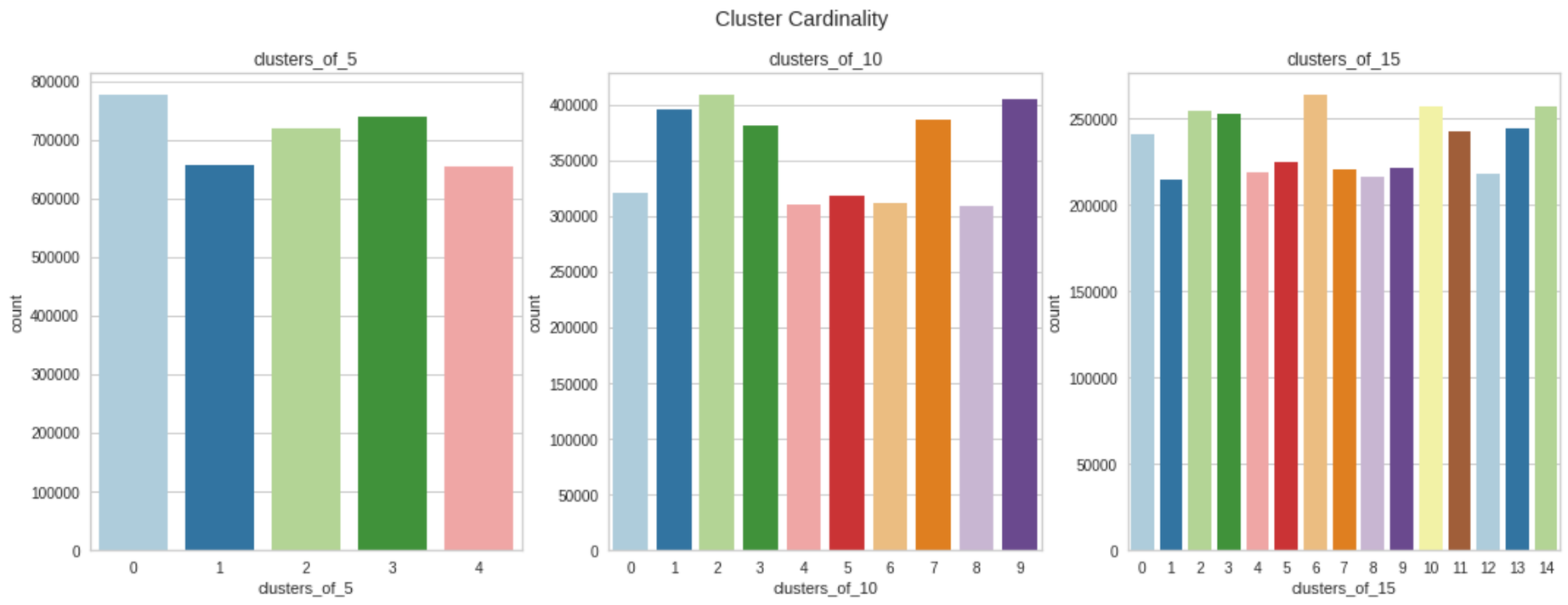
```
df = df.assign(
    clusters_of_5 = kmeans_5.predict(df),
    clusters_of_10 = kmeans_10.predict(df),
    clusters_of_15 = kmeans_15.predict(df),
)

with pd.option_context("display.max_columns", None):
    display(df.head())
```

	product_id	retail_price	cost	sale_price	clusters_of_5	clusters_of_10	clusters_of_15
0	16898	25	13	25	2	2	0
1	16898	25	13	25	2	2	0
2	16898	25	13	25	2	2	0
3	16898	25	13	25	2	2	0
4	16898	25	13	25	2	2	0

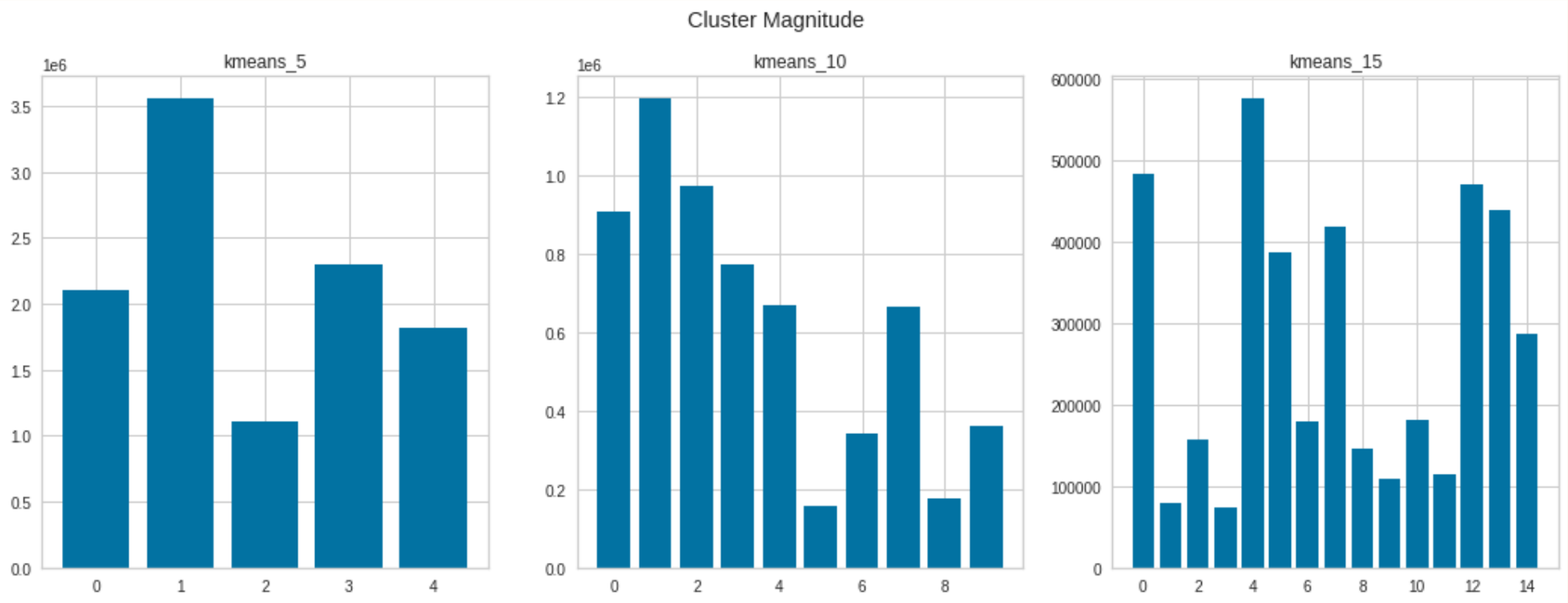
PRODUCT


product_id
retail_price
cost
sale_price
clusters_of_5
0
1
2
3
4
clusters_of_5
-1.380423823970438
-0.6744634579561993
0.03149690805803933
0.737457274072278
1.4434176400865166

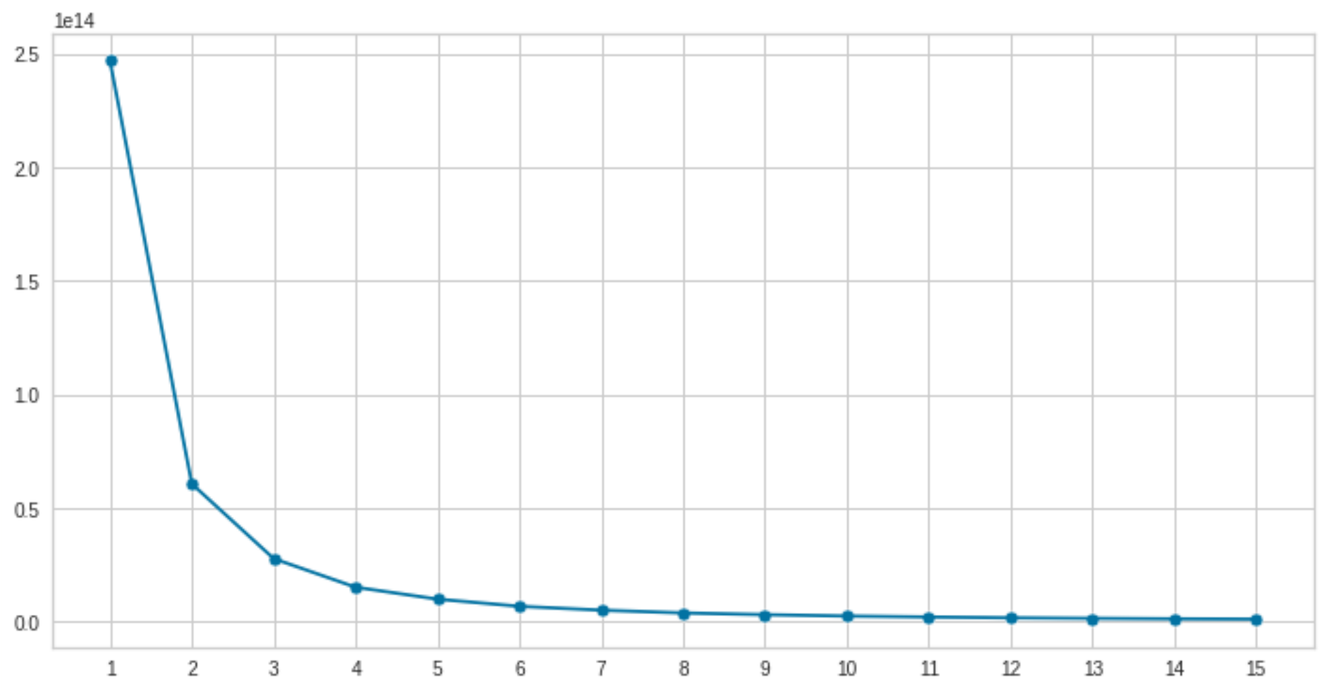
PRODUCT


Cluster Cardinality
clusters_of_5
clusters_of_10
clusters_of_15
count
clusters_of_5
clusters_of_10
clusters_of_15

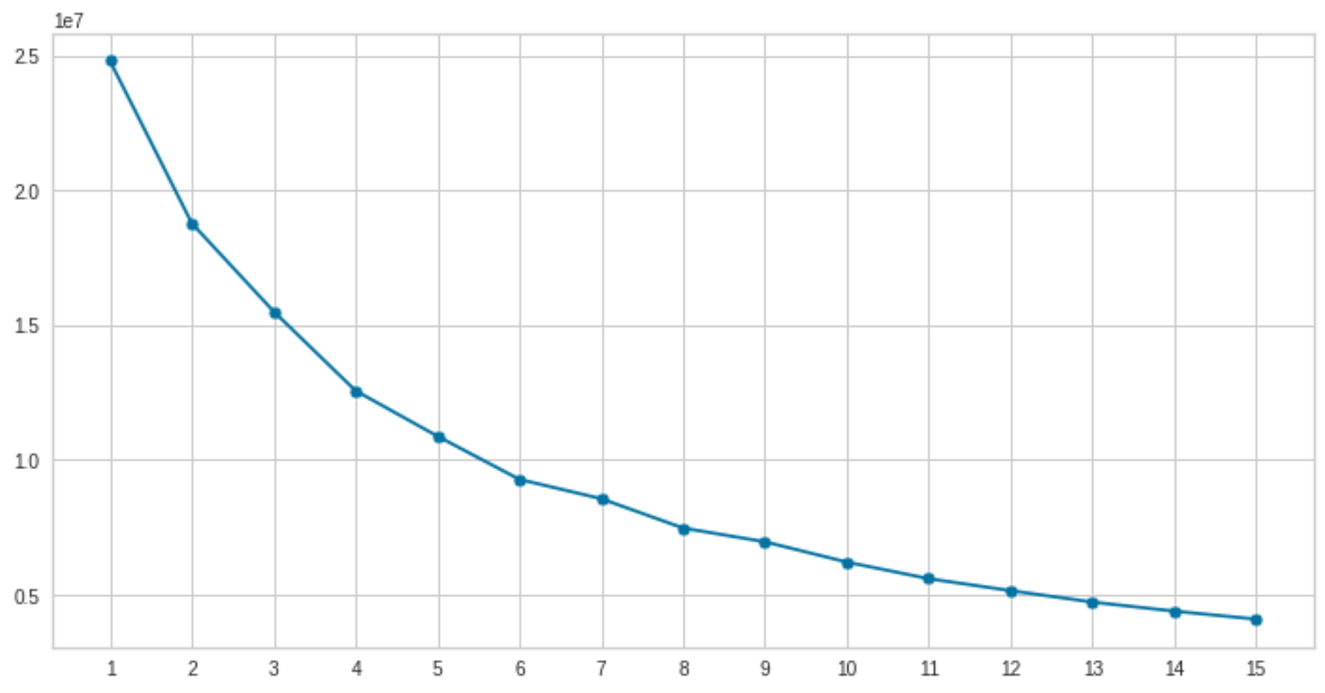
PRODUCT


Cluster Magnitude
kmeans_5
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
kmeans_10
1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
kmeans_15
600000
500000
400000
300000
200000
100000
0
0
2
4
6
8
10
12
14

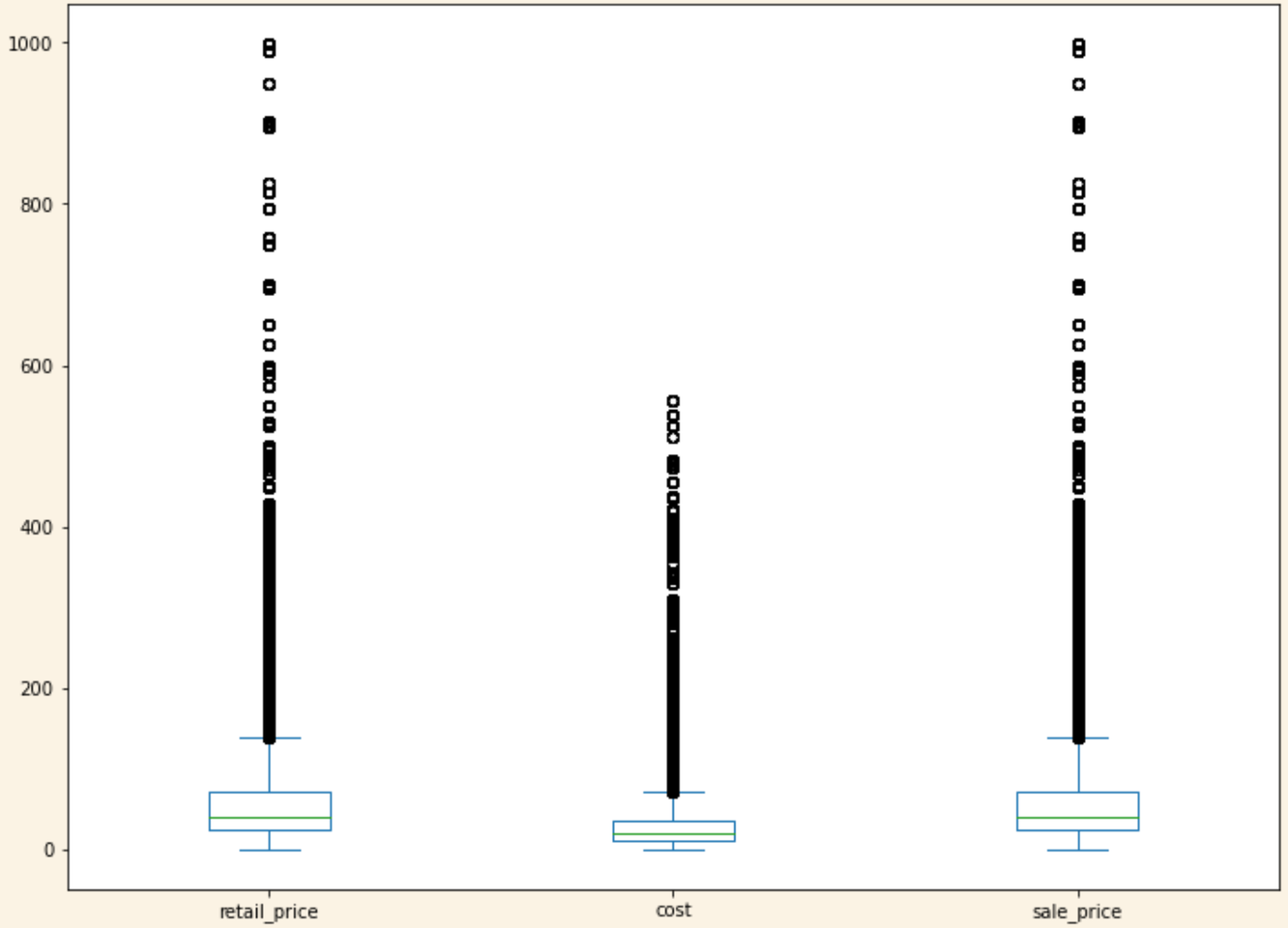
PRODUCT



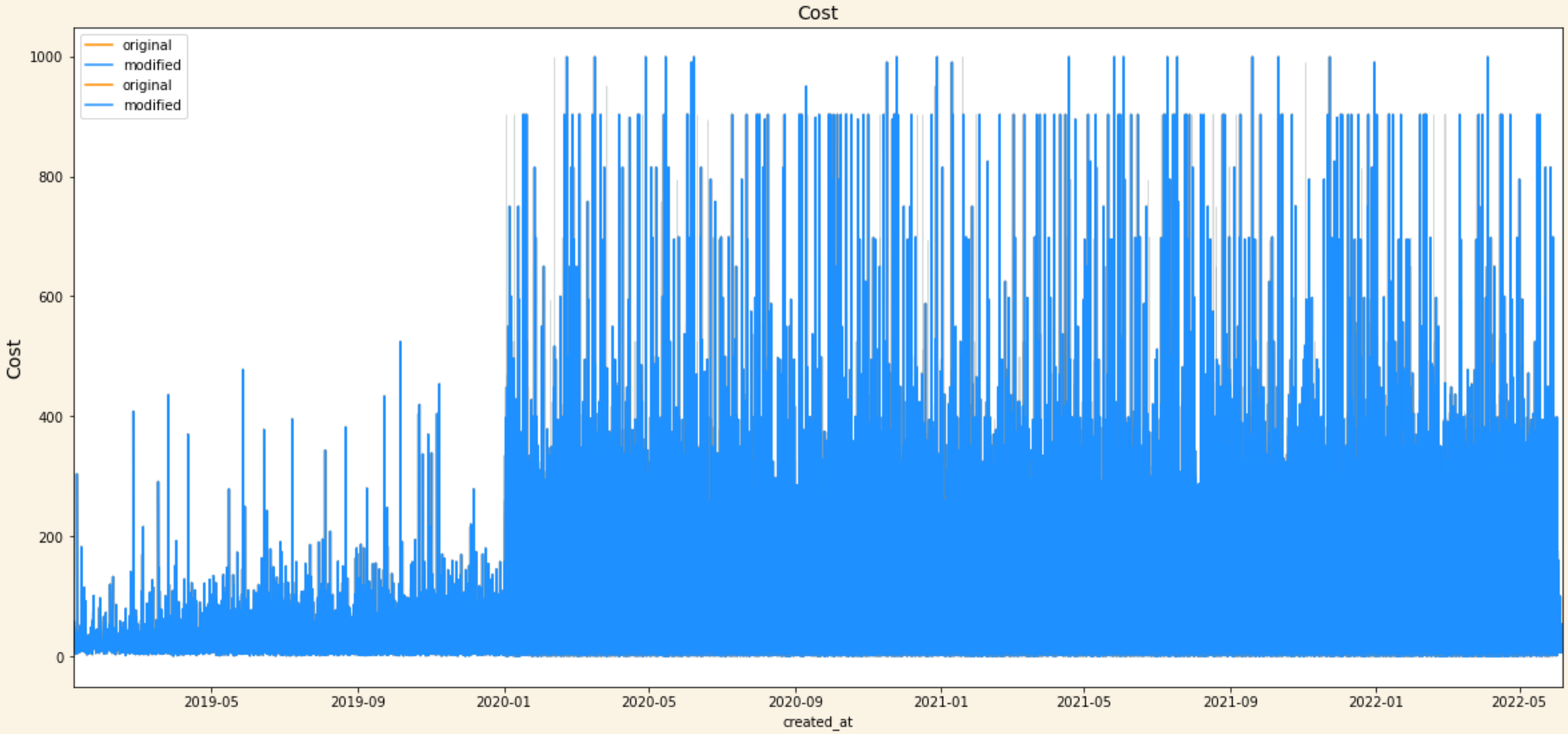
PRODUCT



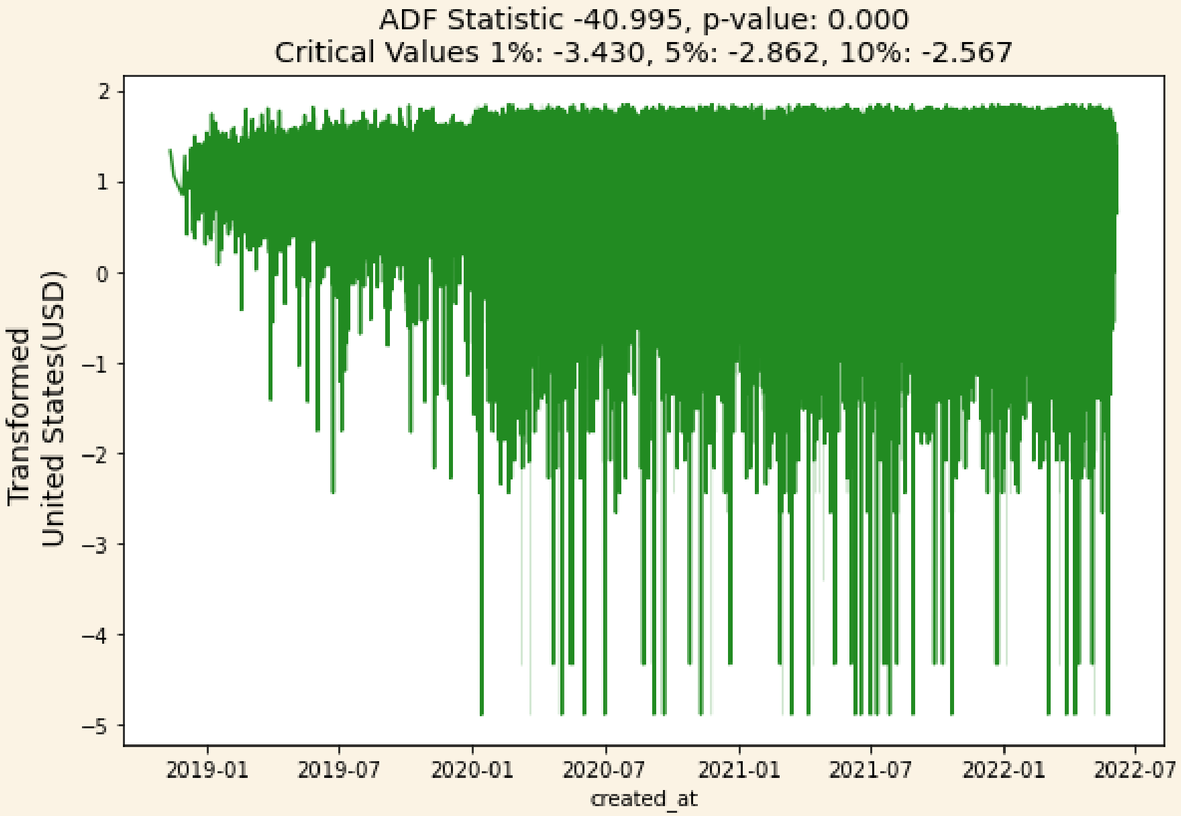
PRODUCT

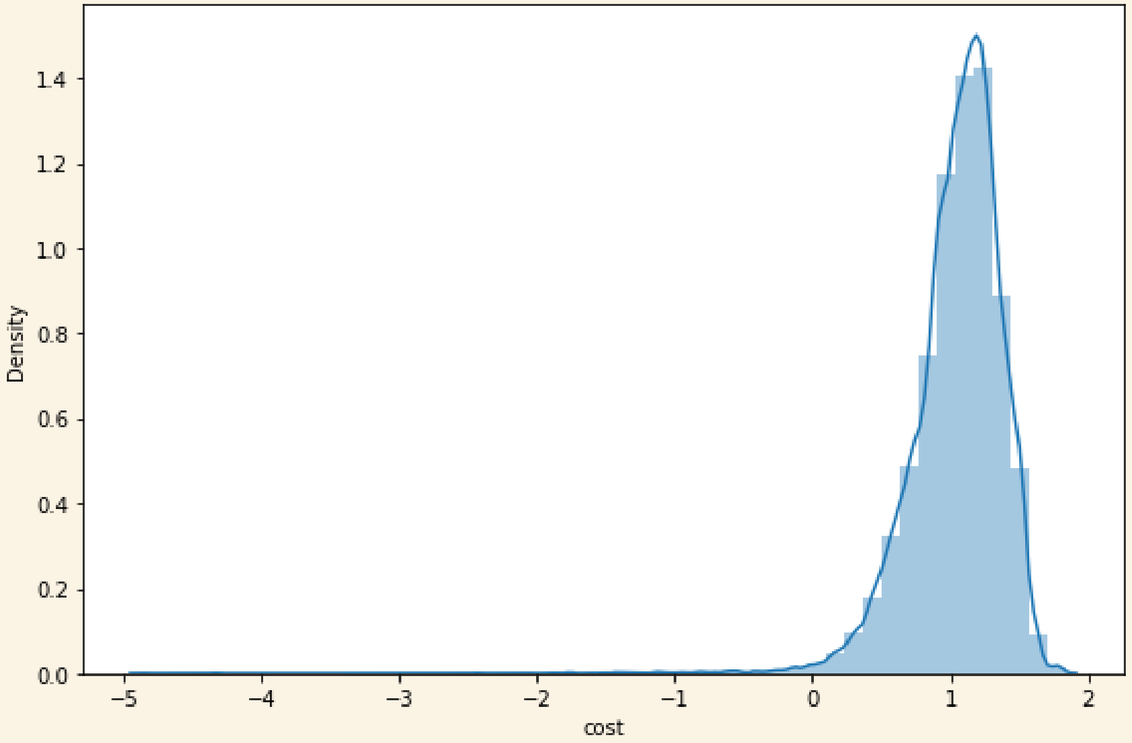


PRODUCT

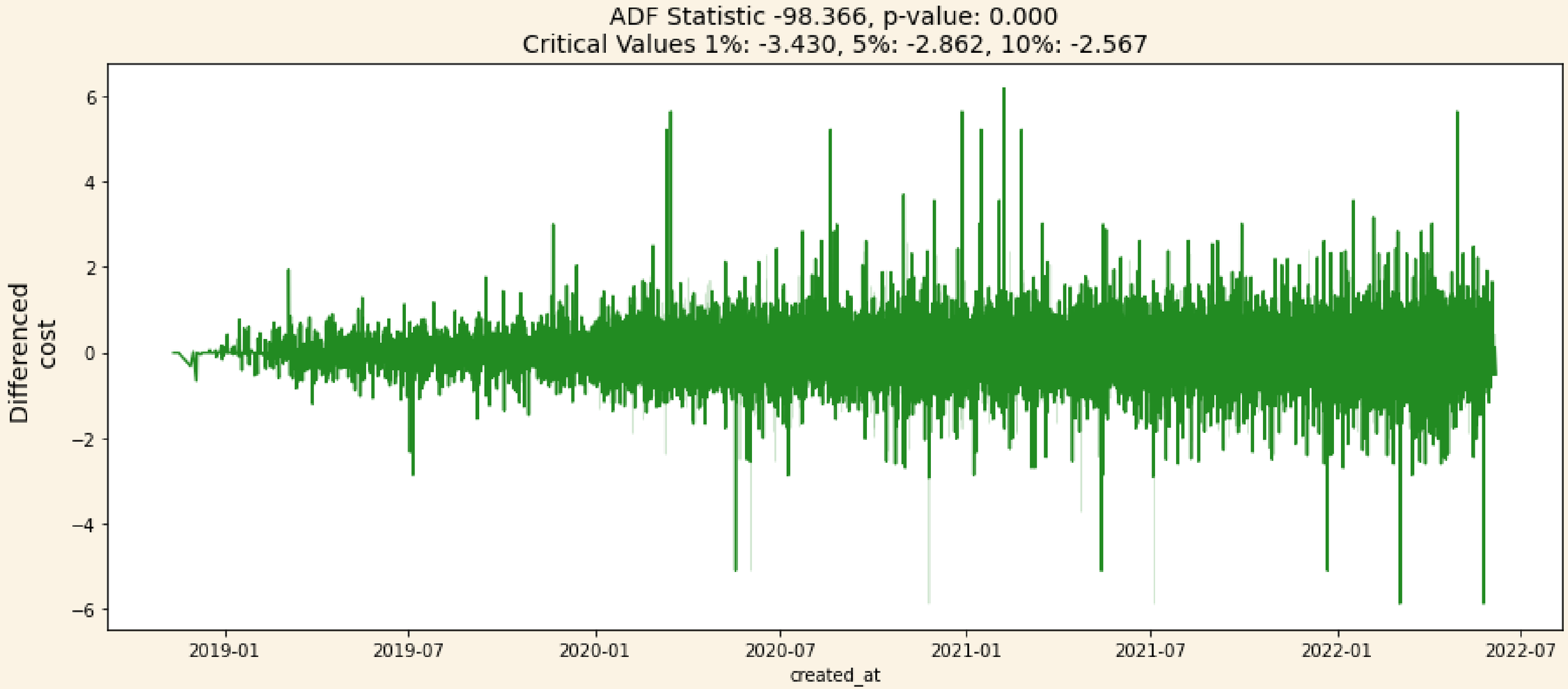

Cost
original
modified
original
modified
1000
800
600
400
200
0
Cost
2019-05
2019-09
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
created_at

PRODUCT


ADF Statistic -40.995, p-value: 0.000
Critical Values 1%: -3.430, 5%: -2.862, 10%: -2.567
Transformed United States(USD)
created_at
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07


Density
cost

PRODUCT


ADF Statistic -98.366, p-value: 0.000
Critical Values 1%: -3.430, 5%: -2.862, 10%: -2.567
Differenced cost
6
4
2
0
−2
−4
−6
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
created_at

PRODUCT