```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime

#Exploring the user dataset
data= pd.read_csv('user_dataset.csv')
data.info
data.describe
```

```
<bound method NDFrame.describe of
user_name  customer_zip_code  \
0      861eff4711a542e4b93843c6dd7febb0              14409
1      290c77bc529b7ac935b93aa66c333dc3               9790
2      060e732b5b29e8181a18229c7b0b2b5e               1151
3      259dac757896d24d7702b9acbbff3f3c               8775
4      345ecd01c38d18a9036ed96c73b8d066              13056
...                                 ...                ...
99436  1a29b476fee25c95fbafc67c5ac95cf8               3937
99437  d52a67c98be1cf6a5c84435bd38d095d               6764
99438  e9f50caf99f032f0bf3c55141f019d99              60115
99439  73c2643a0a458b49f58cea58833b192e              92120
99440  84732c5050c01db9b23e19ba39899398               6703

                 customer_city    customer_state
0         KABUPATEN PEKALONGAN       JAWA TENGAH
1                   KOTA BEKASI        JAWA BARAT
2                KOTA TANGERANG            BANTEN
3      KABUPATEN BANDUNG BARAT        JAWA BARAT
4           KOTA JAKARTA TIMUR       DKI JAKARTA
...                        ...               ...
99436           KOTA TANGERANG            BANTEN
99437           KOTA PONTIANAK  KALIMANTAN BARAT
99438       KABUPATEN SIDOARJO        JAWA TIMUR
99439        KABUPATEN CIANJUR        JAWA BARAT
99440            KOTA SUKABUMI        JAWA BARAT

[99441 rows x 4 columns]>
```

```
#checking for null values
data.isnull().sum()
```

```
user_name            0
customer_zip_code    0
customer_city        0
customer_state       0
dtype: int64
```

```
# Remove any rows with missing or invalid 'UserID' or 'UserZIPCode'
data.dropna( inplace=True)
```

```
# Save the cleaned dataset to a new file
cleaned_file_path = "cleaned_user_dataset.csv"
data.to_csv(cleaned_file_path, index=False)

#Exploring the feedback dataset
df= pd.read_csv('feedback_dataset.csv')
df.info
df.describe
df.head
```

```
<bound method NDFrame.head of                                 feedback_id
order_id  \
0      7bc2406110b926393aa56f80a40eba40
73fc7af87114b39712e6da79b0a377eb
1      80e641a11e56f04c1ad469d5645fdfde
a548910a1c6147796b98fdf73dbeba33
2      228ce5500dc1d8e020d8d1322874b6f0
f9e4b658b201a9f2ecdecbb34bed034b
3      e64fb393e7b32834bb789ff8bb30750e
658677c97b385a9be170737859d3511b
4      f7c4243c7fe1938f181bec41a392bdeb
8e6bfb81e283fa7e4f11123a3fb894f1
...                                  ...
...
99995  f3897127253a9592a73be9bdfdf4ed7a
22ec9f0669f784db00fa86d035cf8602
99996  b3de70c89b1510c4cd3d0649fd302472
55d4004744368f5571d1f590031933e4
99997  1adeb9d84d72fe4e337617733eb85149
7725825d039fc1f0ceb7635e3f7d9206
99998  be360f18f5df1e0541061c87021e6d93
f8bd3f2000c28c5342fedeb5e50f2e75
99999  efe49f1d6f951dd88b51e6ccd4cc548f
90531360ecb1eec2a1fbb265a0db0508

       feedback_score feedback_form_sent_date feedback_answer_date
0                   4     2018-01-18 00:00:00  2018-01-18 21:46:59
1                   5     2018-03-10 00:00:00  2018-03-11 03:05:13
2                   5     2018-02-17 00:00:00  2018-02-18 14:36:24
3                   5     2017-04-21 00:00:00  2017-04-21 22:02:06
4                   5     2018-03-01 00:00:00  2018-03-02 10:26:53
...               ...                     ...                  ...
99995               5     2017-12-09 00:00:00  2017-12-11 20:06:42
99996               5     2018-03-22 00:00:00  2018-03-23 09:10:43
99997               4     2018-07-01 00:00:00  2018-07-02 12:59:13
99998               1     2017-12-15 00:00:00  2017-12-16 01:29:43
99999               1     2017-07-03 00:00:00  2017-07-03 21:01:49

[100000 rows x 5 columns]>
```

```
# Convert 'feedback_score' to integer and ensure IDs are strings
df['feedback_score'] = pd.to_numeric(df['feedback_score'],
errors='coerce')  # Coerce invalid entries to NaN
df['feedback_id'] = df['feedback_id'].astype(str)
df['order_id'] = df['order_id'].astype(str)

# Convert dates to datetime
df['feedback_form_sent_date'] =
pd.to_datetime(df['feedback_form_sent_date'], errors='coerce')
df['feedback_answer_date'] =
pd.to_datetime(df['feedback_answer_date'], errors='coerce')

# Remove rows with missing or invalid data
df = df.dropna(subset=['feedback_id', 'order_id', 'feedback_score',
'feedback_form_sent_date', 'feedback_answer_date'])

# Ensure feedback_score is within a valid range (e.g., 1-5)
df = df[df['feedback_score'].between(1, 5)]

# Save the cleaned dataset
output_file = "cleaned_feedback_data.csv"
df.to_csv(output_file, index=False)

#Exploring the payment dataset
df= pd.read_csv('payment_dataset.csv')
df.info
df.describe
df.head

<bound method NDFrame.head of                                 order_id
payment_sequential payment_type  \
0       70b7e94ea46d3e8b5bc12a50186edaf0                  1
credit_card
1       859f516f2fc3f95772e63c5757ab0d5b                  1
credit_card
2       ff36cbc44b8f228e0449c92ef089c843                  1
credit_card
3       2b7dbe9be72b8f9733844c31055c0825                  1
credit_card
4       6ae2e8b8fac02522481d2a2f4ca4412c                  1
credit_card
...                                  ...                ...
...
103881  0406037ad97740d563a178ecc7a2075c                  1
blipay
103882  32609bbb3dd69b3c066a6860554a77bf                  1
credit_card
103883  28bbae6599b09d39ca406b747b6632b1                  1
blipay
103884  744bade1fcf9ff3f31d860ace076d422                  2
credit_card
```

```
103885  1a57108394169c0b47d8f876acc9ba2d                    2
credit_card

        payment_installments  payment_value
0                         24       274840.0
1                         24       609560.0
2                         24       756490.0
3                         24       345390.0
4                         24       433430.0
...                      ...            ...
103881                     1       363310.0
103882                     1        47770.0
103883                     1       191580.0
103884                     0        58690.0
103885                     0       129940.0

[103886 rows x 5 columns]>
```

```python
# Type Conversion
df['order_id'] = df['order_id'].astype(str)
df['payment_sequential'] = df['payment_sequential'].astype(int)
df['payment_installments'] = df['payment_installments'].astype(int)
df['payment_value'] = df['payment_value'].astype(float)

# Text Capitalization
df['payment_type'] = df['payment_type'].str.capitalize()

# Value Replacement
df['payment_type'] = df['payment_type'].str.replace('_', ' ')

# Remove rows with missing or invalid data
df = df.dropna(subset=[ 'order_id', 'payment_sequential',
'payment_installments', 'payment_value'])

# Save the cleaned dataset
output_file = "cleaned_payment_data.csv"
df.to_csv(output_file, index=False)

#Exploring the products dataset
df= pd.read_csv('products_dataset.csv')
df.info
df.describe
df.head
```

```
<bound method NDFrame.head of                                 product_id
product_category  \
0      1e9e8ef04dbcff4541ed26657ea517e5                  perfumery
1      3aa071139cb16b67ca9e5dea641aaa2f                        art
2      96bd76ec8810374ed1b65e291975717f             sports_leisure
3      cef67bcfe19066a932b7673e239eb23d                       baby
4      9dc1a7de274444849c219cff195d0b71                  housewares
```

```
...                                     ...                          ...
32946  a0b7d5a992ccda646f2d34e418fff5a0            furniture_decor
32947  bf4538d88321d0fd4412a93c974510e6  construction_tools_lights
32948  9a7c6041fa9592d9d9ef6cfe62a71f8c              bed_bath_table
32949  83808703fc0706a22e264b9d75f04a2e      computers_accessories
32950  106392145fca363410d287a815be6de4              bed_bath_table

       product_name_lenght  product_description_lenght
product_photos_qty  \
0                     40.0                       287.0
1.0
1                     44.0                       276.0
1.0
2                     46.0                       250.0
1.0
3                     27.0                       261.0
1.0
4                     37.0                       402.0
4.0
...                    ...                         ...
...
32946                 45.0                        67.0
2.0
32947                 41.0                       971.0
1.0
32948                 50.0                       799.0
1.0
32949                 60.0                       156.0
2.0
32950                 58.0                       309.0
1.0

       product_weight_g  product_length_cm  product_height_cm  \
0                 225.0               16.0               10.0
1                1000.0               30.0               18.0
2                 154.0               18.0                9.0
3                 371.0               26.0                4.0
4                 625.0               20.0               17.0
...                 ...                ...                ...
32946           12300.0               40.0               40.0
32947            1700.0               16.0               19.0
32948            1400.0               27.0                7.0
32949             700.0               31.0               13.0
32950            2083.0               12.0                2.0

       product_width_cm
0                  14.0
1                  20.0
2                  15.0
3                  26.0
```

```
4                   13.0
...                  ...
32946               40.0
32947               16.0
32948               27.0
32949               20.0
32950                7.0

[32951 rows x 9 columns]>
```

```
# Remove any rows with missing or invalid rows
df.dropna( inplace=True)

# Type Conversion
df['product_id'] = df['product_id'].astype(str)
df['product_category'] = df['product_category'].astype(str)
df['product_name_lenght'] = df['product_name_lenght'].astype(int)

# Save the cleaned dataset
output_file = "cleaned_products_data.csv"
df.to_csv(output_file, index=False)

#Exploring the seller dataset
df= pd.read_csv('seller_dataset.csv')
df.info
df.describe
df.head
```

```
<bound method NDFrame.head of                                 seller_id
seller_zip_code  \
0     3442f8959a84dea7ee197c632cb2df15            13023
1     d1b65fc7debc3361ea86b5f14c68d2e2            13844
2     ce3ad9de960102d0677a81f5d0bb7b2d            20031
3     c0f3eea2e14555b6faeea3dd58c1b1c3             4195
4     51a04a8a6bdcb23deccc82b0b80742cf            12914
...                                ...              ...
3090  98dddbc4601dd4443ca174359b237166            87111
3091  f8201cab383e484733266d1906e2fdfa            88137
3092  74871d19219c7d518d0090283e03c137             4650
3093  e603cf3fec55f8697c9059638d6c8eb5            96080
3094  9e25199f6ef7e7c347120ff175652c3b            12051

                      seller_city        seller_state
0               KOTA JAKARTA TIMUR         DKI JAKARTA
1              KOTA PADANG PANJANG      SUMATERA BARAT
2               KOTA JAKARTA BARAT         DKI JAKARTA
3                   KOTA TANGERANG              BANTEN
4               KABUPATEN LAMONGAN           JAWA TIMUR
...                            ...                ...
3090  KABUPATEN HULU SUNGAI TENGAH  KALIMANTAN SELATAN
3091  KABUPATEN KOTAWARINGIN TIMUR   KALIMANTAN TENGAH
```

```
3092                 KOTA TANGERANG                BANTEN
3093             KABUPATEN GROBOGAN           JAWA TENGAH
3094                   KOTA KEDIRI            JAWA TIMUR

[3095 rows x 4 columns]>
```

```
# Type Conversion
df['seller_id'] = df['seller_id'].astype(str)
df['seller_zip_code'] = df['seller_zip_code'].astype(str)

# Remove any rows with missing or invalid rows
df.dropna( inplace=True)

# Text Capitalization
df['seller_city'] = df['seller_city'].str.upper()
df['seller_state'] = df['seller_state'].str.upper()

# Save the cleaned dataset
output_file = "cleaned_seller_data.csv"
df.to_csv(output_file, index=False)

#Exploring the order item dataset
df= pd.read_csv('order_item_dataset.csv')
df.info
df.describe
df.head
```

```
<bound method NDFrame.head of                                     order_id
order_item_id  \
0       00010242fe8c5a6d1ba2dd792cb16214              1
1       00018f77f2f0320c557190d7a144bdd3              1
2       000229ec398224ef6ca0657da4fc703e              1
3       00024acbcdf0a6daa1e931b038114c75              1
4       00042b26cf59d7ce69dfabb4e55b4fd9              1
...                                          ...             ...
112645  fffc94f6ce00a00581880bf54a75a037              1
112646  fffcd46ef2263f404302a634eb57f7eb              1
112647  fffce4705a9662cd70adb13d4a31832d              1
112648  fffe18544ffabc95dfada21779c9644f              1
112649  fffe41c64501cc87c801fd61db3f6244              1

                                 product_id
seller_id  \
0       4244733e06e7ecb4970a6e2683c13e61
48436dade18ac8b2bce089ec2a041202
1       e5f2d52b802189ee658865ca93d83a8f
dd7ddc04e1b6c2c614352b383efe2d36
2       c777355d18b72b67abbeef9df44fd0fd
5b51032eddd242adc84c38acab88f23d
3       7634da152a4610f1595efa32f14722fc
9d7a1d34a5052409006425275ba1c2b4
```

```
4       ac6c3623068f30de03045865e4e10089
df560393f3a51e74553ab94004ba5c87
...                                  ...
...
112645  4aa6014eceb682077f9dc4bffebc05b0
b8bc237ba3788b23da09c0f1f3a3288c
112646  32e07fd915822b0765e448c4dd74c828
f3c38ab652836d21de61fb8314b69182
112647  72a30483855e2eafc67aee5dc2560482
c3cfdc648177fdbbbb35635a37472c53
112648  9c422a519119dcad7575db5af1ba540e
2b3e4a2a3ea8e01938cabda2a3e5cc79
112649  350688d9dc1e75ff97be326363655e01
f7ccf836d21b2fb1de37564105216cc1

       pickup_limit_date     price  shipping_cost
0          9/19/2017 9:45   58900.0        13290.0
1          5/3/2017 11:05  239900.0        19930.0
2         1/18/2018 14:48  199000.0        17870.0
3         8/15/2018 10:10   12990.0        12790.0
4         2/13/2017 13:57  199900.0        18140.0
...                   ...       ...            ...
112645     5/2/2018 4:11  299990.0        43410.0
112646    7/20/2018 4:31  350000.0        36530.0
112647  10/30/2017 17:14   99900.0        16950.0
112648    8/21/2017 0:04   55990.0         8720.0
112649   6/12/2018 17:10   43000.0        12790.0

[112650 rows x 7 columns]>

# Remove any rows with missing or invalid rows
df.dropna( inplace=True)

# Convert 'pickup_limit_date' to datetime
df['pickup_limit_date'] = pd.to_datetime(df['pickup_limit_date'],
format='%m/%d/%Y %H:%M')

# Convert 'price' and 'shipping_cost' to numeric types
df['price'] = pd.to_numeric(df['price'], errors='coerce')
df['shipping_cost'] = pd.to_numeric(df['shipping_cost'],
errors='coerce')

# Save the cleaned dataset
output_file = "cleaned_order_item_data.csv"
df.to_csv(output_file, index=False)

#Exploring the order dataset
df= pd.read_csv('order_dataset.csv')
df.info
df.describe
df.head
```

```
<bound method NDFrame.head of                                  order_id
user_name  \
0      e481f51cbdc54678b7cc49136f2d6af7
7c396fd4830fd04220f754e42b4e5bff
1      53cdb2fc8bc7dce0b6741e2150273451
af07308b275d755c9edb36a90c618231
2      47770eb9100c2d0c44946d9cf07ec65d
3a653a41f6f9fc3d2a113cf8398680e8
3      949d5b44dbf5de918fe9c16f97b45f8a
7c142cf63193a1473d2e66489a9ae977
4      ad21c59c0840e6cb83a9ceb5573f8159
72632f0f9dd73dfee390c9b22eb56dd6
...                                  ...
...
99436  9c5dedf39a927c1b2549525ed64a053c
6359f309b166b0196dbf7ad2ac62bb5a
99437  63943bddc261676b46f01ca7ac2f7bd8
da62f9e57a76d978d02ab5362c509660
99438  83c1379a015df1e13d02aae0204711ab
737520a9aad80b3fbbdad19b66b37b30
99439  11c177c8e97725db2631073c19f07b62
5097a5312c8b157bb7be58ae360ef43c
99440  66dea50a8b16d9b4dee7af250b4be1a5
60350aa974b26ff12caad89e55993bd6

      order_status        order_date order_approved_date
pickup_date  \
0        delivered   10/2/2017 10:56     10/2/2017 11:07   10/4/2017
19:55
1        delivered   7/24/2018 20:41      7/26/2018 3:24   7/26/2018
14:31
2        delivered     8/8/2018 8:38       8/8/2018 8:55    8/8/2018
13:50
3        delivered  11/18/2017 19:28    11/18/2017 19:45  11/22/2017
13:39
4        delivered   2/13/2018 21:18     2/13/2018 22:20   2/14/2018
19:46
...            ...               ...                 ...
...
99436    delivered     3/9/2017 9:54       3/9/2017 9:54   3/10/2017
11:18
99437    delivered    2/6/2018 12:58      2/6/2018 13:10    2/7/2018
23:22
99438    delivered   8/27/2017 14:46     8/27/2017 15:04   8/28/2017
20:52
99439    delivered    1/8/2018 21:28      1/8/2018 21:36   1/12/2018
15:35
99440    delivered    3/8/2018 20:57      3/9/2018 11:20    3/9/2018
22:11
```

```
         delivered_date estimated_time_delivery
0      10/10/2017 21:25         10/18/2017 0:00
1        8/7/2018 15:27          8/13/2018 0:00
2       8/17/2018 18:06           9/4/2018 0:00
3        12/2/2017 0:28         12/15/2017 0:00
4       2/16/2018 18:17          2/26/2018 0:00
...                 ...                    ...
99436   3/17/2017 15:08          3/28/2017 0:00
99437   2/28/2018 17:37           3/2/2018 0:00
99438   9/21/2017 11:24          9/27/2017 0:00
99439   1/25/2018 23:32          2/15/2018 0:00
99440   3/16/2018 13:08           4/3/2018 0:00

[99441 rows x 8 columns]>
```

```
# Convert date columns to datetime
date_columns = ['order_date', 'order_approved_date', 'pickup_date',
'delivered_date', 'estimated_time_delivery']
for col in date_columns:
    df[col] = pd.to_datetime(df[col], format='%m/%d/%Y %H:%M',
errors='coerce')

# Remove any rows with missing or invalid rows
df.dropna( inplace=True)

# Save the cleaned dataset
output_file = "cleaned_order_data.csv"
df.to_csv(output_file, index=False)
```