finalitics-project

July 18, 2024

[ ]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# to read the data
data=pd.read_csv('/content/online_advertising_performance_data.csv')
data
```

[ ]:

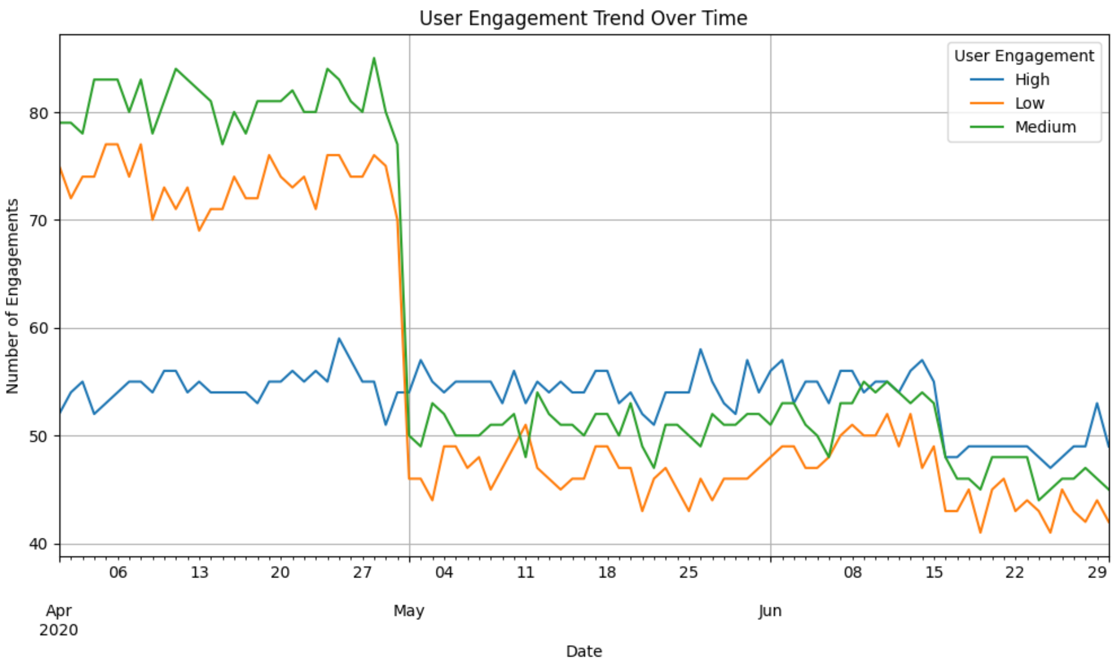
```
       month  day campaign_number user_engagement     banner placement  \
0      April    1          camp 1            High  160 x 600       abc
1      April    1          camp 1            High  160 x 600       def
2      April    1          camp 1            High  160 x 600       ghi
3      April    1          camp 1            High  160 x 600       mno
4      April    1          camp 1             Low  160 x 600       def
…      …  …             …             …        …       …
15403  April    1          camp 1             Low  160 x 600       ghi
15404  April    1          camp 1             Low  160 x 600       mno
15405   June   29          camp 1            High  800 x 250       ghi
15406   June   29          camp 1            High  800 x 250       mno
15407   June   29          camp 3            High  240 x 400       def

       displays      cost  clicks  revenue  post_click_conversions  \
0             4    0.0060       0   0.0000                       0
1         20170   26.7824     158  28.9717                      23
2         14701   27.6304     158  28.9771                      78
3        171259  216.8750    1796 329.4518                     617
4           552    0.0670       1   0.1834                       0
…         …       …     …       …                     …
15403        16    0.0249       0   0.0000                       0
15404      2234    0.4044      10   1.8347                       3
15405         1    0.0157       0   0.0000                       0
15406         4    0.0123       0   0.0000                       0
15407      1209    0.3184       2   0.1115                       3

       post_click_sales_amount  Unnamed: 12  Unnamed: 13
0                       0.0000          NaN          NaN
1                    1972.4602          NaN          NaN
2                    2497.2636          NaN          NaN
```

```
3                   24625.3234          NaN          NaN
4                       0.0000          NaN          NaN
…                        …          …          …
15403                   0.0000          NaN          NaN
15404                 101.7494          NaN          NaN
15405                   0.0000          NaN          NaN
15406                   0.0000          NaN          NaN
15407                 110.4224          NaN          NaN

[15408 rows x 14 columns]
```

[ ]:
```
#1
#data pre-processing
data['Date'] = pd.to_datetime(data['month'] + ' ' + data['day'].astype(str) +␣
 ↪', 2020')
data['Date']
# Group by date and user engagement level, then count occurrences
engagement_trend = data.groupby(['Date', 'user_engagement']).size().
 ↪unstack(fill_value=0)
engagement_trend
```

[ ]:
```
user_engagement  High  Low  Medium
Date
2020-04-01         52   75      79
2020-04-02         54   72      79
2020-04-03         55   74      78
2020-04-04         52   74      83
2020-04-05         53   77      83
…               …  …     …
2020-06-26         48   45      46
2020-06-27         49   43      46
2020-06-28         49   42      47
2020-06-29         53   44      46
2020-06-30         49   42      45

[91 rows x 3 columns]
```

[ ]:
```
#plotting
plt.figure(figsize=(12, 6))
engagement_trend.plot(ax=plt.gca())
plt.title('User Engagement Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Engagements')
plt.legend(title='User Engagement')
plt.grid(True)
plt.show()
```



```
[ ]: #2
     data = data.dropna(subset=['banner', 'clicks'])
     data
```

```
[ ]:        month  day campaign_number user_engagement     banner placement  \
     0      April    1          camp 1            High  160 x 600       abc
     1      April    1          camp 1            High  160 x 600       def
     2      April    1          camp 1            High  160 x 600       ghi
     3      April    1          camp 1            High  160 x 600       mno
     4      April    1          camp 1             Low  160 x 600       def
     …        …   …             …              …        …         …
     15403  April    1          camp 1             Low  160 x 600       ghi
     15404  April    1          camp 1             Low  160 x 600       mno
     15405   June   29          camp 1            High  800 x 250       ghi
     15406   June   29          camp 1            High  800 x 250       mno
     15407   June   29          camp 3            High  240 x 400       def

            displays      cost  clicks   revenue  post_click_conversions  \
     0             4    0.0060       0    0.0000                       0
     1         20170   26.7824     158   28.9717                      23
     2         14701   27.6304     158   28.9771                      78
     3        171259  216.8750    1796  329.4518                     617
     4           552    0.0670       1    0.1834                       0
     …             …       …       …        …                       …
     15403        16    0.0249       0    0.0000                       0
```

```
15404      2234    0.4044      10    1.8347                       3
15405         1    0.0157       0    0.0000                       0
15406         4    0.0123       0    0.0000                       0
15407      1209    0.3184       2    0.1115                       3

       post_click_sales_amount  Unnamed: 12  Unnamed: 13       Date
0                       0.0000          NaN          NaN 2020-04-01
1                    1972.4602          NaN          NaN 2020-04-01
2                    2497.2636          NaN          NaN 2020-04-01
3                   24625.3234          NaN          NaN 2020-04-01
4                       0.0000          NaN          NaN 2020-04-01
…                        …          …          …        …
15403                   0.0000          NaN          NaN 2020-04-01
15404                 101.7494          NaN          NaN 2020-04-01
15405                   0.0000          NaN          NaN 2020-06-29
15406                   0.0000          NaN          NaN 2020-06-29
15407                 110.4224          NaN          NaN 2020-06-29

[15408 rows x 15 columns]
```

```
[ ]: data.isnull().sum()
```

```
[ ]: month                          0
     day                            0
     campaign_number                0
     user_engagement                0
     banner                         0
     placement                    413
     displays                       0
     cost                           0
     clicks                         0
     revenue                        0
     post_click_conversions         0
     post_click_sales_amount        0
     Unnamed: 12                15408
     Unnamed: 13                15408
     Date                           0
     dtype: int64
```

```
[ ]: #grouping the data
     banner_clicks=data.groupby('banner')['clicks'].mean()
     banner_clicks
```

```
[ ]: banner
     160 x 600    132.725762
     240 x 400    459.074639
     300 x 250    145.820567
```

```
468 x 60       0.681938
580 x 400    199.143564
670 x 90      15.817602
728 x 90     181.287715
800 x 250      0.033426
Name: clicks, dtype: float64
```

[ ]:

```
#sorting the values
banner_sort=banner_clicks.sort_values(ascending=False)
banner_sort
```

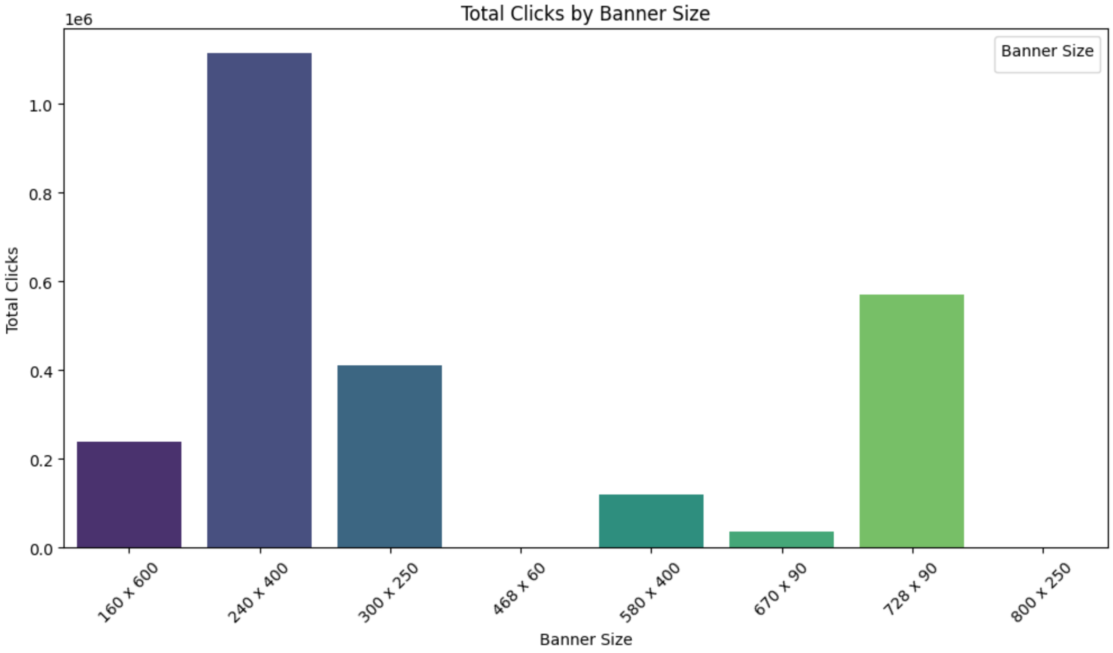
[ ]:

```
banner
240 x 400    459.074639
580 x 400    199.143564
728 x 90     181.287715
300 x 250    145.820567
160 x 600    132.725762
670 x 90      15.817602
468 x 60       0.681938
800 x 250      0.033426
Name: clicks, dtype: float64
```

[ ]:

```
import matplotlib.pyplot as plt
# Aggregate clicks by banner size
banner_sort = data.groupby('banner')['clicks'].sum().reset_index()
# Add a column for hue
banner_sort['banner_category'] = banner_sort['banner']
# Plot the data with a specific color palette and hue
plt.figure(figsize=(12, 6))
sns.barplot(x='banner', y='clicks', data=banner_sort, hue='banner_category',␣
 ↪palette='viridis')
plt.title('Total Clicks by Banner Size')
plt.xlabel('Banner Size')
plt.ylabel('Total Clicks')
plt.xticks(rotation=45)  # Rotate x labels for better readability
plt.legend(title='Banner Size')
plt.show()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
```



```
[ ]: #3
     data = data.dropna(subset=['placement', 'displays', 'clicks'])
     data
```

```
[ ]:        month  day campaign_number user_engagement     banner placement  \
     0      April    1          camp 1            High  160 x 600       abc
     1      April    1          camp 1            High  160 x 600       def
     2      April    1          camp 1            High  160 x 600       ghi
     3      April    1          camp 1            High  160 x 600       mno
     4      April    1          camp 1             Low  160 x 600       def
     …        …   …             …              …       …         …
     15403  April    1          camp 1             Low  160 x 600       ghi
     15404  April    1          camp 1             Low  160 x 600       mno
     15405   June   29          camp 1            High  800 x 250       ghi
     15406   June   29          camp 1            High  800 x 250       mno
     15407   June   29          camp 3            High  240 x 400       def

            displays      cost  clicks   revenue  post_click_conversions  \
     0             4    0.0060       0    0.0000                       0
     1         20170   26.7824     158   28.9717                      23
     2         14701   27.6304     158   28.9771                      78
     3        171259  216.8750    1796  329.4518                     617
     4           552    0.0670       1    0.1834                       0
     …             …       …       …        …                      …
     15403        16    0.0249       0    0.0000                       0
     15404      2234    0.4044      10    1.8347                       3
```

```
15405         1    0.0157       0    0.0000                       0
15406         4    0.0123       0    0.0000                       0
15407      1209    0.3184       2    0.1115                       3

       post_click_sales_amount  Unnamed: 12  Unnamed: 13       Date
0                       0.0000          NaN          NaN 2020-04-01
1                    1972.4602          NaN          NaN 2020-04-01
2                    2497.2636          NaN          NaN 2020-04-01
3                   24625.3234          NaN          NaN 2020-04-01
4                       0.0000          NaN          NaN 2020-04-01
…                        …          …          …        …
15403                   0.0000          NaN          NaN 2020-04-01
15404                 101.7494          NaN          NaN 2020-04-01
15405                   0.0000          NaN          NaN 2020-06-29
15406                   0.0000          NaN          NaN 2020-06-29
15407                 110.4224          NaN          NaN 2020-06-29

[14995 rows x 15 columns]
```

```
[ ]: placement_performance = data.groupby('placement')[['displays', 'clicks']].sum().
      ↪reset_index()
     placement_performance
```

```
[ ]:   placement   displays   clicks
     0       abc     242142     1584
     1       def   28177492   176097
     2       ghi   59740415  1247049
     3       jkl    7692732    75063
     4       mno  143161775   993039
```

```
[ ]:
```

```
[ ]: placement_sort=placement_performance.sort_values(by='clicks',ascending=False)
     placement_sort
     placement_display_sort=placement_performance.
      ↪sort_values(by='displays',ascending=False)
     placement_display_sort
```

```
[ ]:   placement   displays   clicks
     4       mno  143161775   993039
     2       ghi   59740415  1247049
     1       def   28177492   176097
     3       jkl    7692732    75063
     0       abc     242142     1584
```

```
[ ]: #to display the top placement of clicks and displays
     placement_sort.head()
```

```
placement_display_sort.head()
```

[ ]:
```
  placement   displays   clicks
4       mno  143161775   993039
2       ghi   59740415  1247049
1       def   28177492   176097
3       jkl    7692732    75063
0       abc     242142     1584
```

[ ]:
```
#4
correlation = data['cost'].corr(data['revenue'])
correlation
```

[ ]:
```
0.760258117132741
```

[ ]:
```
#5 avg revenue
total_revenue=data['revenue'].sum()
total_revenue
```

[ ]:
```
276264.26670000004
```

[ ]:
```
total_clicks=data['clicks'].sum()
total_clicks
```

[ ]:
```
2492832
```

[ ]:
```
avg_revenue_perclicks=total_revenue/total_clicks
avg_revenue_perclicks
```

[ ]:
```
0.11082345970366235
```

[ ]:
```
#6
campaign_performance = data.groupby('campaign_number')[['clicks',␣
 ↪'post_click_conversions']].sum().reset_index()
campaign_performance['conversion_rate'] =␣
 ↪campaign_performance['post_click_conversions'] /␣
 ↪campaign_performance['clicks']
campaign_performance_sorted = campaign_performance.
 ↪sort_values(by='conversion_rate', ascending=False)
campaign_performance_sorted[['campaign_number', 'conversion_rate']]
```

[ ]:
```
  campaign_number  conversion_rate
0          camp 1         0.449271
2          camp 3         0.024272
1          camp 2         0.015624
```

[ ]:
```
#7
# Combine the 'month' and 'day' columns into a single datetime column
data['date'] = pd.to_datetime(data['month'] + ' ' + data['day'].astype(str) +␣
 ↪', 2020')
data['date']
```

```
<ipython-input-20-69bdc4f0a807>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['date'] = pd.to_datetime(data['month'] + ' ' + data['day'].astype(str) +
', 2020')
```

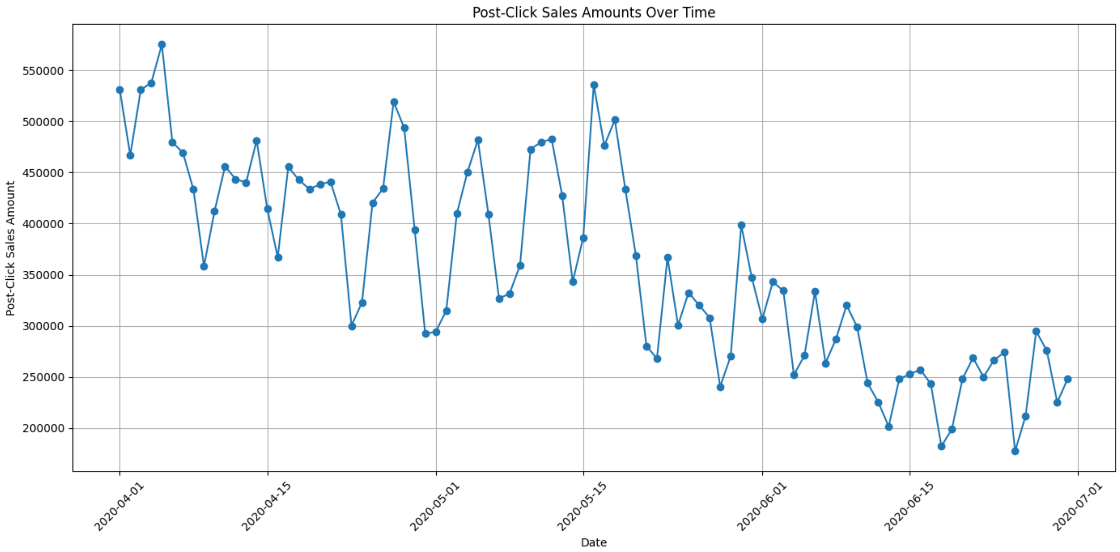
[ ]:
```
0        2020-04-01
1        2020-04-01
2        2020-04-01
3        2020-04-01
4        2020-04-01
            …
15403    2020-04-01
15404    2020-04-01
15405    2020-06-29
15406    2020-06-29
15407    2020-06-29
Name: date, Length: 14995, dtype: datetime64[ns]
```

[ ]:
```
daily_sales = data.groupby('date')['post_click_sales_amount'].sum().
 ↪reset_index()
daily_sales
```

[ ]:
```
          date  post_click_sales_amount
0   2020-04-01              531410.5466
1   2020-04-02              466908.9690
2   2020-04-03              530984.9128
3   2020-04-04              537908.7395
4   2020-04-05              575301.4028
..         …                        …
86  2020-06-26              211969.3014
87  2020-06-27              294567.5051
88  2020-06-28              275890.7343
89  2020-06-29              225088.4150
90  2020-06-30              248297.3469

[91 rows x 2 columns]
```

[ ]:

```python
plt.figure(figsize=(14, 7))
plt.plot(daily_sales['date'], daily_sales['post_click_sales_amount'], marker='o')
plt.title('Post-Click Sales Amounts Over Time')
plt.xlabel('Date')
plt.ylabel('Post-Click Sales Amount')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:

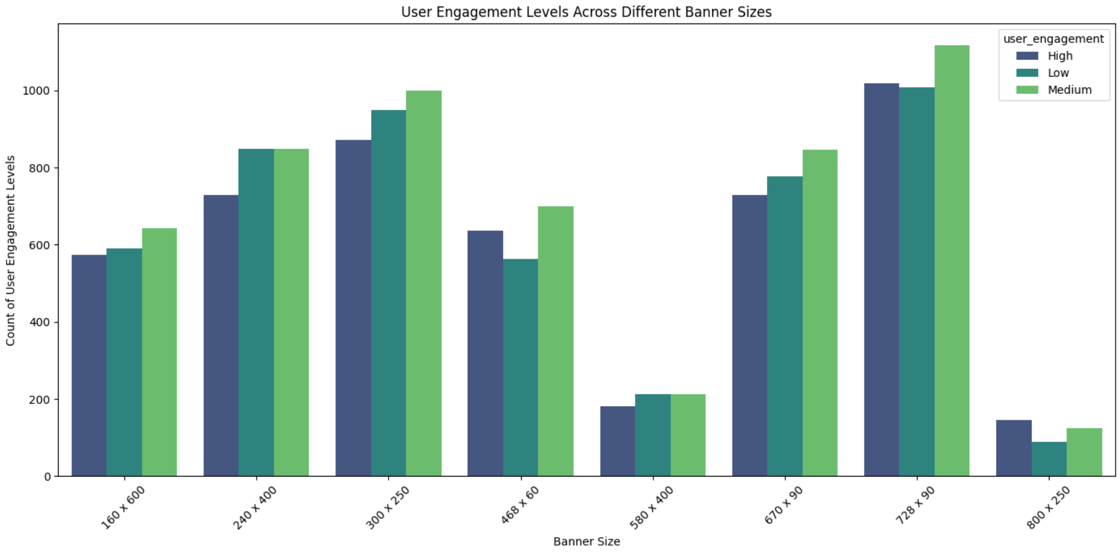
```python
#8
#How does the level of user engagement vary across different banner sizes?
user_engagement_by_banner = data.groupby(['banner', 'user_engagement']).size().reset_index(name='count')
user_engagement_by_banner
```

[ ]:

```
      banner user_engagement  count
0  160 x 600            High    573
1  160 x 600             Low    590
2  160 x 600          Medium    642
3  240 x 400            High    729
4  240 x 400             Low    848
5  240 x 400          Medium    848
6  300 x 250            High    872
7  300 x 250             Low    949
8  300 x 250          Medium    999
9   468 x 60            High    637
```

```
10   468 x 60          Low    563
11   468 x 60       Medium    699
12  580 x 400         High    182
13  580 x 400          Low    212
14  580 x 400       Medium    212
15   670 x 90         High    728
16   670 x 90          Low    777
17   670 x 90       Medium    847
18   728 x 90         High   1018
19   728 x 90          Low   1007
20   728 x 90       Medium   1117
21  800 x 250         High    145
22  800 x 250          Low     89
23  800 x 250       Medium    125
```

[ ]:
```
#plotting using bargraph
plt.figure(figsize=(14, 7))
sns.barplot(x='banner', y='count', hue='user_engagement',␣
 ↪data=user_engagement_by_banner, palette='viridis')
plt.title('User Engagement Levels Across Different Banner Sizes')
plt.xlabel('Banner Size')
plt.ylabel('Count of User Engagement Levels')
plt.xticks(rotation=45)
plt.tight_layout()
```



[ ]:
```
#9
#Which placement types result in the highest post-click conversion rates?
```

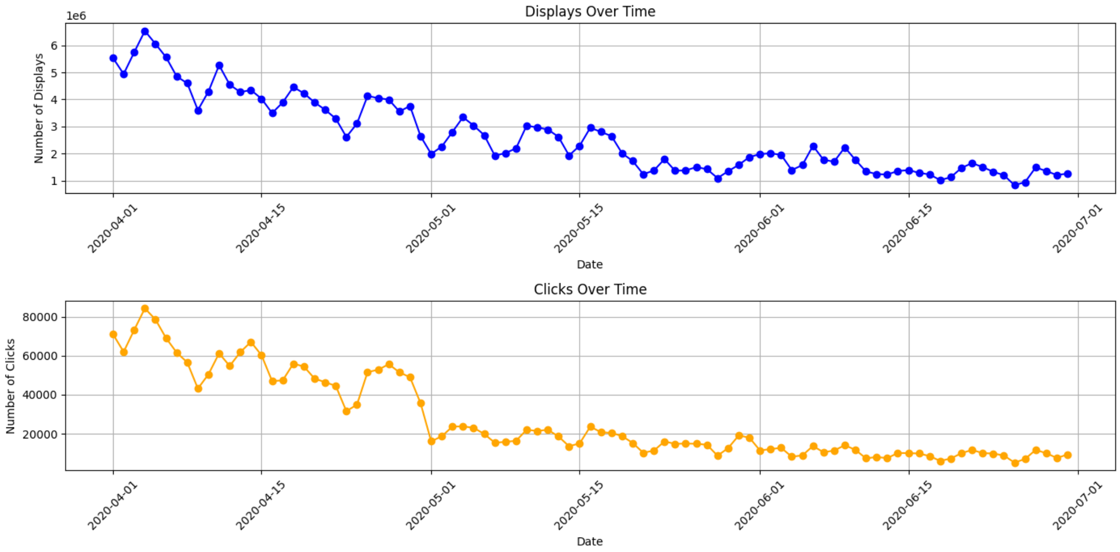
```
placement_performance = data.groupby('placement')[['clicks',␣
 ↪'post_click_conversions']].sum().reset_index()
#calculate the conversation rate
placement_performance['conversion_rate'] =␣
 ↪placement_performance['post_click_conversions'] /␣
 ↪placement_performance['clicks']
#sort the placement performance
placement_performance_sorted = placement_performance.
 ↪sort_values(by='conversion_rate', ascending=False)
placement_performance_sorted
#displaying top placement post-click conversation rates
print(placement_performance_sorted[['placement', 'conversion_rate']])
```

```
  placement  conversion_rate
0       abc         0.520202
3       jkl         0.277807
2       ghi         0.270288
4       mno         0.265015
1       def         0.169543
```

[ ]:
```
#10
#Can we identify any seasonal patterns or fluctuations in displays and clicks␣
 ↪throughout the campaign period?
# Aggregate displays and clicks by date
daily_stats = data.groupby('date')[['displays', 'clicks']].sum().reset_index()
daily_stats
```

[ ]:
```
          date  displays  clicks
0   2020-04-01   5529025   70959
1   2020-04-02   4938863   61968
2   2020-04-03   5745603   73219
3   2020-04-04   6531564   84224
4   2020-04-05   6053536   78538
..         …         …       …
86  2020-06-26    943082    7080
87  2020-06-27   1488976   11738
88  2020-06-28   1353066   10032
89  2020-06-29   1207018    7591
90  2020-06-30   1260038    9366

[91 rows x 3 columns]
```

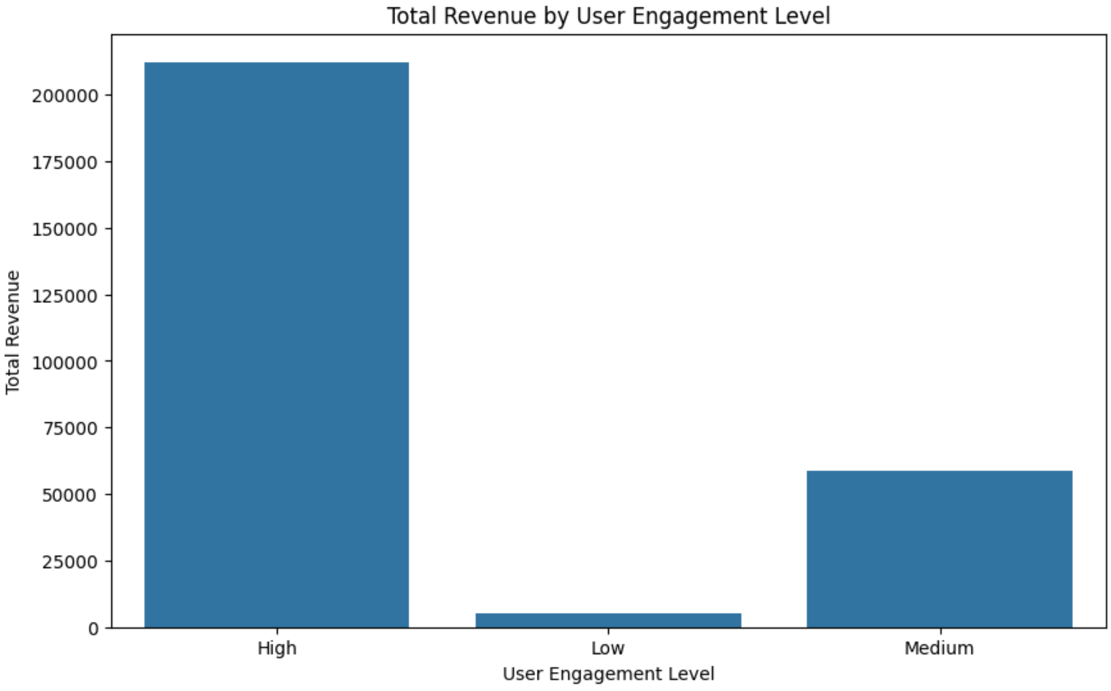
[ ]:
```
#plotting the trends
plt.figure(figsize=(14, 7))
# Plot for displays
plt.subplot(2, 1, 1)
plt.plot(daily_stats['date'], daily_stats['displays'], marker='o', color='blue')
plt.title('Displays Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Displays')
plt.grid(True)
plt.xticks(rotation=45)
# Plot for clicks
plt.subplot(2, 1, 2)
plt.plot(daily_stats['date'], daily_stats['clicks'], marker='o', color='orange')
plt.title('Clicks Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Clicks')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:

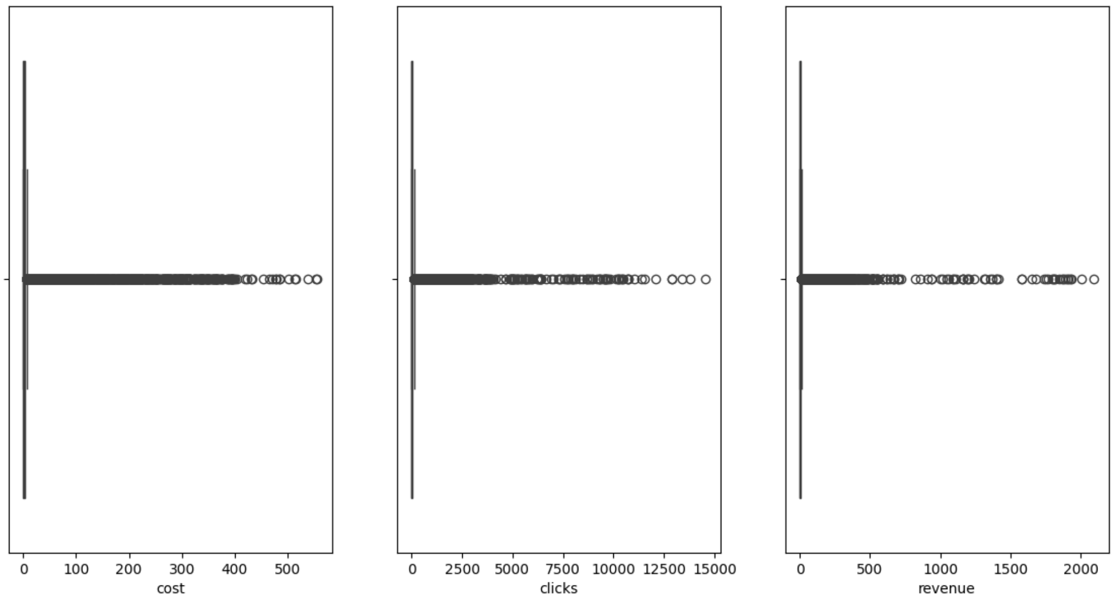
```
#11
#Is there a correlation between user engagement levels and the revenue generated?
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

# Drop rows with null values in 'user_engagement' or 'revenue'
data = data.dropna(subset=['user_engagement', 'revenue'])

# Aggregate revenue by user engagement level
```

```
engagement_revenue = data.groupby('user_engagement')['revenue'].sum().
 ↪reset_index()

# Visualize the relationship using a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x='user_engagement', y='revenue', data=engagement_revenue)
plt.title('Total Revenue by User Engagement Level')
plt.xlabel('User Engagement Level')
plt.ylabel('Total Revenue')
plt.show()

# Map user engagement levels to numerical values
engagement_mapping = {'Low': 1, 'Medium': 2, 'High': 3}
data['user_engagement_num'] = data['user_engagement'].map(engagement_mapping)

# Drop rows where the mapping could not be applied
data = data.dropna(subset=['user_engagement_num'])

# Calculate the correlation coefficient between user engagement levels and␣
 ↪revenue
correlation_coefficient, p_value = pearsonr(data['user_engagement_num'],␣
 ↪data['revenue'])

print(f'Correlation coefficient between user engagement levels and revenue:␣
 ↪{correlation_coefficient}')
print(f'P-value: {p_value}')
```



```
Correlation coefficient between user engagement levels and revenue:
0.17936885915272663
P-value: 1.2343946349454289e-108
```

```
[ ]: #12
     #Are there any outliers in terms of cost, clicks, or revenue that warrant␣
      ↪further investigation?
     # Specify the columns to plot
     import seaborn as sns
     import matplotlib.pyplot as plt
     columns_to_plot = ['cost', 'clicks', 'revenue']
     columns_to_plot
     # Plot box plots to visualize outliers
     plt.figure(figsize=(14, 7))
     for column in columns_to_plot:
         plt.subplot(1, 3, columns_to_plot.index(column) + 1)
         sns.boxplot(x=column, data=data)
```



```
[ ]: # Function to detect outliers using IQR
     def detect_outliers(df, col):
         Q1 = df[col].quantile(0.25)
         Q3 = df[col].quantile(0.75)
         IQR = Q3 - Q1
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR
```

```python
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    return outliersw

# Detect and print outliers for each column
outliers = {}
for col in columns_to_plot:
    outliers[col] = detect_outliers(data, col)
    if not outliers[col].empty:
        print(f"Outliers for {col}:")
        print(outliers[col])
        print("\n")
```

```
Outliers for cost:
       month  day campaign_number user_engagement     banner placement  \
1      April    1          camp 1            High  160 x 600       def   
2      April    1          camp 1            High  160 x 600       ghi   
3      April    1          camp 1            High  160 x 600       mno   
9      April    1          camp 1          Medium  160 x 600       mno   
10     April    1          camp 1            High  240 x 400       def   
…      …  …             …             …       …       …              
15320   June   30          camp 1            High   728 x 90       ghi   
15322   June   30          camp 1            High   728 x 90       mno   
15331   June   30          camp 1          Medium   728 x 90       ghi   
15337   June   30          camp 3            High  240 x 400       ghi   
15352   June   30          camp 3            High  300 x 250       mno   

       displays      cost  clicks   revenue  post_click_conversions  \
1         20170   26.7824     158   28.9717                      23   
2         14701   27.6304     158   28.9771                      78   
3        171259  216.8750    1796  329.4518                     617   
9         20152   11.1678     185   33.9397                      13   
10        56499   50.5157     309   56.6775                     105   
…         …       …     …       …                     …             
15320    117364  208.0751    1235  139.0000                     789   
15322    147455  105.7007     649   73.0000                     424   
15331      4792    8.3755     113   12.7235                      11   
15337      6556    6.6968      65    3.6596                      15   
15352     27927    9.0831      80    4.5038                       7   

       post_click_sales_amount  Unnamed: 12  Unnamed: 13        Date  \
1                    1972.4602          NaN          NaN  2020-04-01   
2                    2497.2636          NaN          NaN  2020-04-01   
3                   24625.3234          NaN          NaN  2020-04-01   
9                     653.1896          NaN          NaN  2020-04-01   
10                   4288.6699          NaN          NaN  2020-04-01   
…                         …           …          …         …          
15320               37919.1960          NaN          NaN  2020-06-30   
```

```
15322               17025.8546          NaN          NaN 2020-06-30
15331                 653.6581          NaN          NaN 2020-06-30
15337                 607.8665          NaN          NaN 2020-06-30
15352                 690.0245          NaN          NaN 2020-06-30

            date  user_engagement_num
1     2020-04-01                    3
2     2020-04-01                    3
3     2020-04-01                    3
9     2020-04-01                    2
10    2020-04-01                    3
…          …                  …
15320 2020-06-30                    3
15322 2020-06-30                    3
15331 2020-06-30                    2
15337 2020-06-30                    3
15352 2020-06-30                    3

[2515 rows x 17 columns]


Outliers for clicks:
       month  day campaign_number user_engagement     banner placement  \
1      April    1          camp 1            High  160 x 600       def
2      April    1          camp 1            High  160 x 600       ghi
3      April    1          camp 1            High  160 x 600       mno
9      April    1          camp 1          Medium  160 x 600       mno
10     April    1          camp 1            High  240 x 400       def
…      …  …             …             …        …       …
15304   June   30          camp 1            High  580 x 400       mno
15320   June   30          camp 1            High   728 x 90       ghi
15322   June   30          camp 1            High   728 x 90       mno
15362   June   30          camp 3          Medium  300 x 250       mno
15401   June   30          camp 3          Medium   728 x 90       mno

       displays       cost  clicks   revenue  post_click_conversions  \
1         20170    26.7824     158   28.9717                      23
2         14701    27.6304     158   28.9771                      78
3        171259   216.8750    1796  329.4518                     617
9         20152    11.1678     185   33.9397                      13
10        56499    50.5157     309   56.6775                     105
…         …       …     …       …                     …
15304     27059    45.4395     229   25.0000                     316
15320    117364   208.0751    1235  139.0000                     789
15322    147455   105.7007     649   73.0000                     424
15362     49675     4.8145     182   10.2462                       0
15401     37790     2.6023     195   10.9785                       0
```

```
       post_click_sales_amount  Unnamed: 12  Unnamed: 13       Date  \
1                    1972.4602          NaN          NaN 2020-04-01   
2                    2497.2636          NaN          NaN 2020-04-01   
3                   24625.3234          NaN          NaN 2020-04-01   
9                     653.1896          NaN          NaN 2020-04-01   
10                   4288.6699          NaN          NaN 2020-04-01   
…                           …          …          …        …   
15304               15489.0316          NaN          NaN 2020-06-30   
15320               37919.1960          NaN          NaN 2020-06-30   
15322               17025.8546          NaN          NaN 2020-06-30   
15362                   0.0000          NaN          NaN 2020-06-30   
15401                   0.0000          NaN          NaN 2020-06-30   

            date  user_engagement_num  
1     2020-04-01                    3  
2     2020-04-01                    3  
3     2020-04-01                    3  
9     2020-04-01                    2  
10    2020-04-01                    3  
…            …                  …  
15304 2020-06-30                    3  
15320 2020-06-30                    3  
15322 2020-06-30                    3  
15362 2020-06-30                    2  
15401 2020-06-30                    2  

[2325 rows x 17 columns]


Outliers for revenue:
       month  day campaign_number user_engagement     banner placement  \
1      April    1          camp 1            High  160 x 600       def   
2      April    1          camp 1            High  160 x 600       ghi   
3      April    1          camp 1            High  160 x 600       mno   
9      April    1          camp 1          Medium  160 x 600       mno   
10     April    1          camp 1            High  240 x 400       def   
…        …  …               …              …        …       …   
15320   June   30          camp 1            High   728 x 90       ghi   
15322   June   30          camp 1            High   728 x 90       mno   
15331   June   30          camp 1          Medium   728 x 90       ghi   
15362   June   30          camp 3          Medium  300 x 250       mno   
15401   June   30          camp 3          Medium   728 x 90       mno   

       displays       cost  clicks   revenue  post_click_conversions  \
1         20170   26.7824     158   28.9717                      23   
2         14701   27.6304     158   28.9771                      78   
3        171259  216.8750    1796  329.4518                     617   
9         20152   11.1678     185   33.9397                      13   
```

```
10        56499   50.5157     309   56.6775                     105
…         …       …     …       …                     …
15320    117364  208.0751    1235  139.0000                     789
15322    147455  105.7007     649   73.0000                     424
15331      4792    8.3755     113   12.7235                      11
15362     49675    4.8145     182   10.2462                       0
15401     37790    2.6023     195   10.9785                       0

       post_click_sales_amount  Unnamed: 12  Unnamed: 13       Date  \
1                    1972.4602          NaN          NaN 2020-04-01
2                    2497.2636          NaN          NaN 2020-04-01
3                   24625.3234          NaN          NaN 2020-04-01
9                     653.1896          NaN          NaN 2020-04-01
10                   4288.6699          NaN          NaN 2020-04-01
…                        …          …          …        …
15320               37919.1960          NaN          NaN 2020-06-30
15322               17025.8546          NaN          NaN 2020-06-30
15331                 653.6581          NaN          NaN 2020-06-30
15362                   0.0000          NaN          NaN 2020-06-30
15401                   0.0000          NaN          NaN 2020-06-30

            date  user_engagement_num
1     2020-04-01                    3
2     2020-04-01                    3
3     2020-04-01                    3
9     2020-04-01                    2
10    2020-04-01                    3
…          …                  …
15320 2020-06-30                    3
15322 2020-06-30                    3
15331 2020-06-30                    2
15362 2020-06-30                    2
15401 2020-06-30                    2

[2512 rows x 17 columns]
```

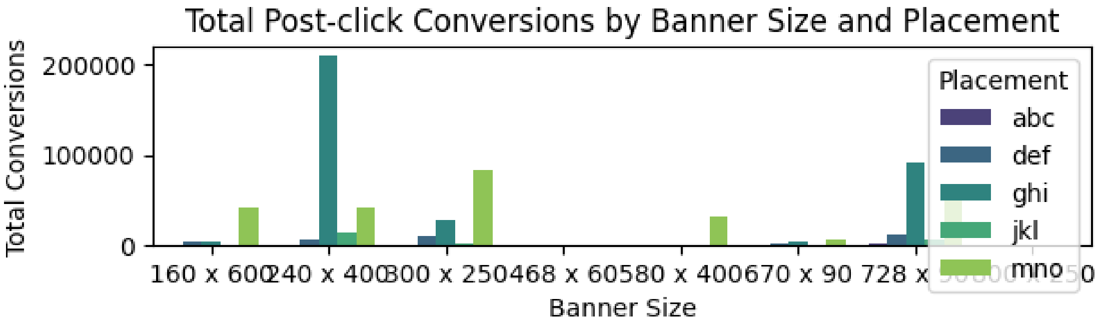
[ ]:

```
#13
#How does the effectiveness of campaigns vary based on the size of the ad and␣
 ↪placement type?
# Aggregate metrics by banner and placement
agg_data = data.groupby(['banner', 'placement']).agg({
    'clicks': 'sum',
    'revenue': 'sum',
    'post_click_conversions': 'sum'
}).reset_index()
```

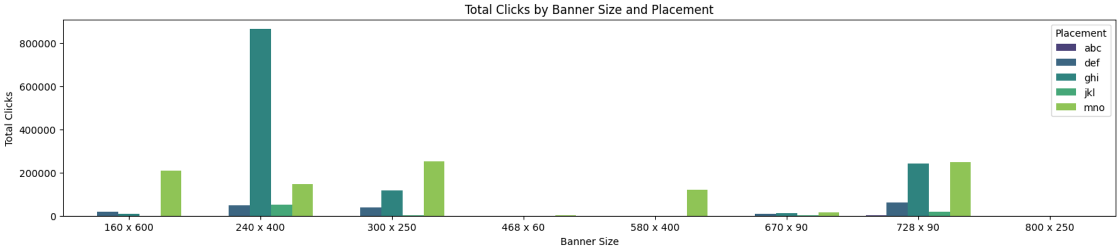
```
agg_data
```

```
[ ]:       banner placement  clicks      revenue  post_click_conversions
     0   160 x 600       abc       3       0.1227                       0
     1   160 x 600       def   20257    1939.9884                    2525
     2   160 x 600       ghi    9799    1374.6526                    4021
     3   160 x 600       jkl       0       0.0000                       0
     4   160 x 600       mno  209511   20201.3655                   42239
     5   240 x 400       def   48452    4009.4231                    5376
     6   240 x 400       ghi  866275  103894.5633                  210014
     7   240 x 400       jkl   52580    5719.2669                   13978
     8   240 x 400       mno  145949   16306.9931                   40497
     9   300 x 250       abc     270      16.3239                     140
     10  300 x 250       def   38932    3432.5302                    8450
     11  300 x 250       ghi  117586   12609.7854                   26609
     12  300 x 250       jkl    2538     317.1134                     877
     13  300 x 250       mno  251888   26795.6003                   83714
     14   468 x 60       def     436      26.6465                     118
     15   468 x 60       ghi      97       5.2563                      95
     16   468 x 60       jkl       4       0.5807                       1
     17   468 x 60       mno     758      55.1012                     336
     18  580 x 400       mno  120681   11193.1428                   31759
     19   670 x 90       def    7763     740.1972                    1627
     20   670 x 90       ghi   11525    1477.2974                    4247
     21   670 x 90       jkl     781      45.3751                     143
     22   670 x 90       mno   17134    1759.9921                    4602
     23   728 x 90       abc    1311     123.8138                     684
     24   728 x 90       def   60257    5712.4502                   11760
     25   728 x 90       ghi  241767   32186.5470                   92077
     26   728 x 90       jkl   19160    2172.1528                    5854
     27   728 x 90       mno  247106   24146.9022                   60020
     28  800 x 250       ghi       0       0.0000                       0
     29  800 x 250       mno      12       1.0826                       3
```

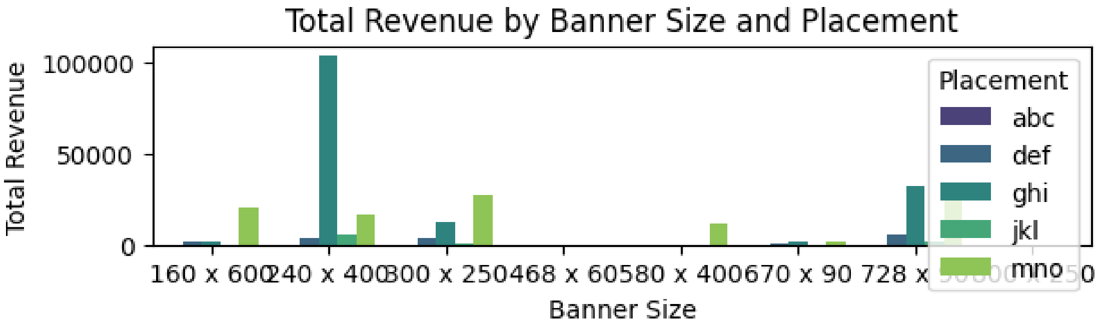
```
[ ]: # Plot for Post-click Conversions
     plt.subplot(3, 1, 3)
     sns.barplot(x='banner', y='post_click_conversions', hue='placement',␣
      ↪data=agg_data, palette='viridis')
     plt.title('Total Post-click Conversions by Banner Size and Placement')
     plt.ylabel('Total Conversions')
     plt.xlabel('Banner Size')
     plt.legend(title='Placement', loc='upper right')

     plt.tight_layout()
     plt.show()
```



```
[ ]: # Plotting effectiveness metrics based on banner and placement
     plt.figure(figsize=(16, 10))

     # Plot for Clicks
     plt.subplot(3, 1, 1)
     sns.barplot(x='banner', y='clicks', hue='placement', data=agg_data,␣
      ↪palette='viridis')
     plt.title('Total Clicks by Banner Size and Placement')
     plt.ylabel('Total Clicks')
     plt.xlabel('Banner Size')
     plt.legend(title='Placement', loc='upper right')
     plt.tight_layout()
     plt.show()
```



```
[ ]: # Plot for Revenue
     plt.subplot(3, 1, 2)
     sns.barplot(x='banner', y='revenue', hue='placement', data=agg_data,␣
      ↪palette='viridis')
     plt.title('Total Revenue by Banner Size and Placement')
     plt.ylabel('Total Revenue')
     plt.xlabel('Banner Size')
     plt.legend(title='Placement', loc='upper right')
     plt.tight_layout()
     plt.show()
```



[ ]:

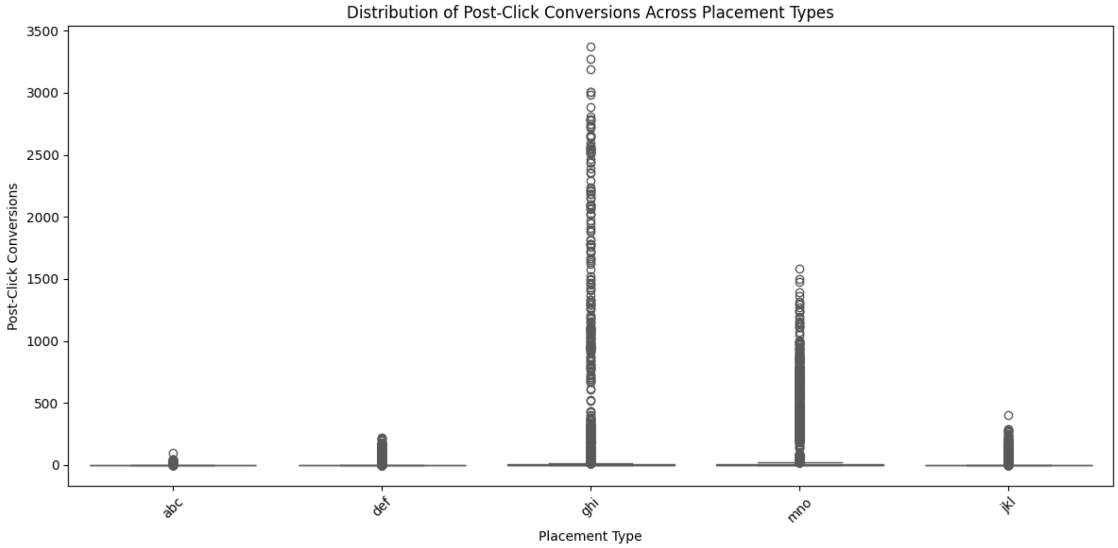
```
#14
#Are there any specific campaigns or banner sizes that consistently outperform␣
 ↪others in terms of ROI?
# Calculate ROI for each campaign and banner size
data['ROI'] = data['revenue'] / data['cost']
data['ROI']
avg_roi = data.groupby(['campaign_number', 'banner'])['ROI'].mean().
 ↪reset_index()
avg_roi
# Identify campaigns or banner sizes with highest average ROI
high_roi_campaigns = avg_roi.sort_values(by='ROI', ascending=False).head(10)
print("Top 10 Campaigns/Banner Sizes with Highest Average ROI:")
print(high_roi_campaigns)
```

```
Top 10 Campaigns/Banner Sizes with Highest Average ROI:
   campaign_number     banner       ROI
16          camp 3  160 x 600  4.096757
0           camp 1  160 x 600  3.931911
12          camp 2  580 x 400  3.113678
20          camp 3  580 x 400  3.004255
1           camp 1  240 x 400  2.531187
2           camp 1  300 x 250  2.171725
4           camp 1  580 x 400  2.070334
9           camp 2  240 x 400  1.906611
13          camp 2   670 x 90  1.851741
21          camp 3   670 x 90  1.803126
```

[ ]:

```
#15
#What is the distribution of post-click conversions across different placement␣
 ↪types?
#Plotting distribution of post-click conversions by placement type



plt.figure(figsize=(12, 6))
```

```
sns.boxplot(x='placement', y='post_click_conversions', data=data,␣
↪hue='placement', palette='Set2')
plt.title('Distribution of Post-Click Conversions Across Placement Types')
plt.xlabel('Placement Type')
plt.ylabel('Post-Click Conversions')

plt.xticks(rotation=45)
plt.show()
```



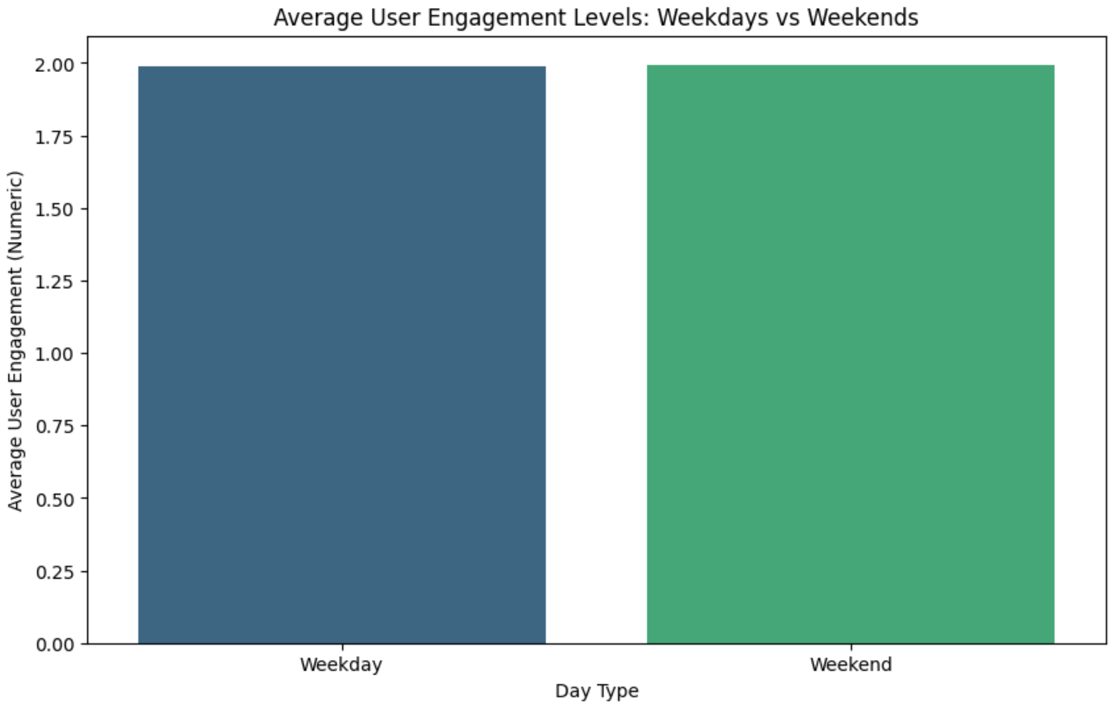
```
[ ]: #16
     #Are there any noticeable differences in user engagement levels between␣
     ↪weekdays and weekends?
     # Convert 'date' column to datetime format
     import datetime as dt
     data['date'] = pd.to_datetime(data['date'])
     data['date']
```

```
[ ]: 0        2020-04-01
     1        2020-04-01
     2        2020-04-01
     3        2020-04-01
     4        2020-04-01
                 …
     15403    2020-04-01
     15404    2020-04-01
     15405    2020-06-29
     15406    2020-06-29
     15407    2020-06-29
```

```
Name: date, Length: 15408, dtype: datetime64[ns]
```

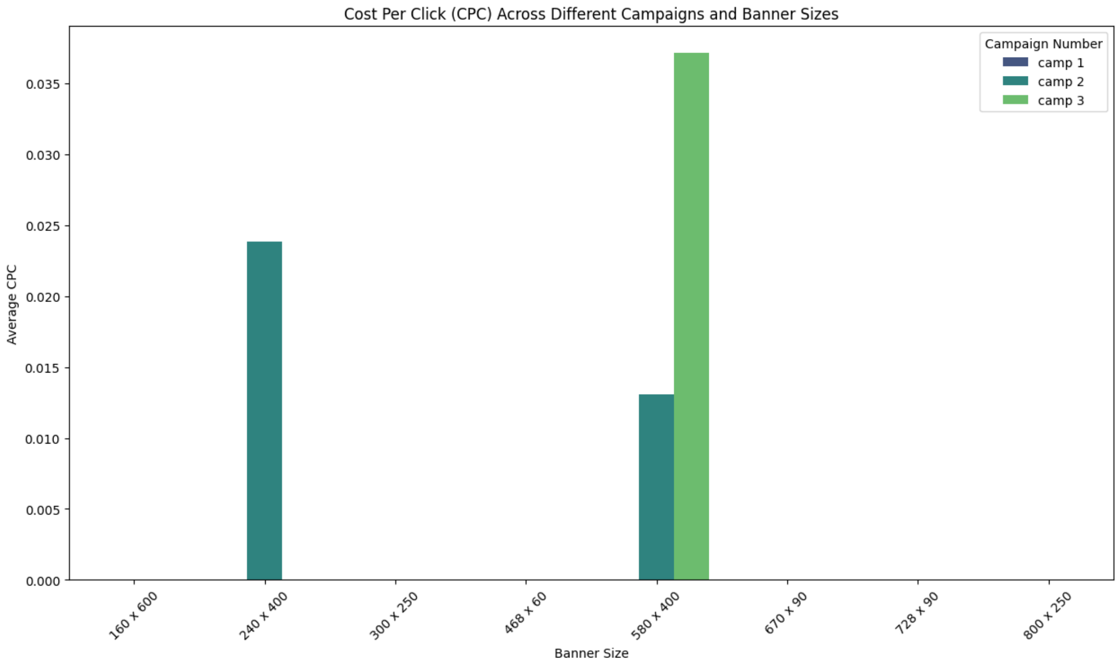
[ ]:

```python
# Combine 'month' and 'day' into a single 'date' column
data['date'] = pd.to_datetime(data['month'] + ' ' + data['day'].astype(str) +␣
 ↪', 2020', format='%B %d, %Y')

# Extract day of the week (Monday=0, Sunday=6)
data['day_of_week'] = data['date'].dt.dayofweek

# Map day of week to weekday or weekend
data['day_type'] = data['day_of_week'].apply(lambda x: 'Weekend' if x >= 5 else␣
 ↪'Weekday')

# Assuming 'user_engagement' column exists and contains categorical values␣
 ↪('Low', 'Medium', 'High')
# Map the engagement levels to numeric values
engagement_mapping = {'Low': 1, 'Medium': 2, 'High': 3}
data['user_engagement_numeric'] = data['user_engagement'].
 ↪map(engagement_mapping)

# Calculate average user engagement levels for weekdays and weekends
avg_engagement = data.groupby('day_type')['user_engagement_numeric'].mean().
 ↪reset_index()

# Plotting the results
plt.figure(figsize=(10, 6))
sns.barplot(x='day_type', y='user_engagement_numeric', data=avg_engagement,␣
 ↪palette='viridis', hue='day_type', dodge=False)
plt.title('Average User Engagement Levels: Weekdays vs Weekends')
plt.xlabel('Day Type')
plt.ylabel('Average User Engagement (Numeric)')
plt.legend([],[], frameon=False)
plt.show()
```



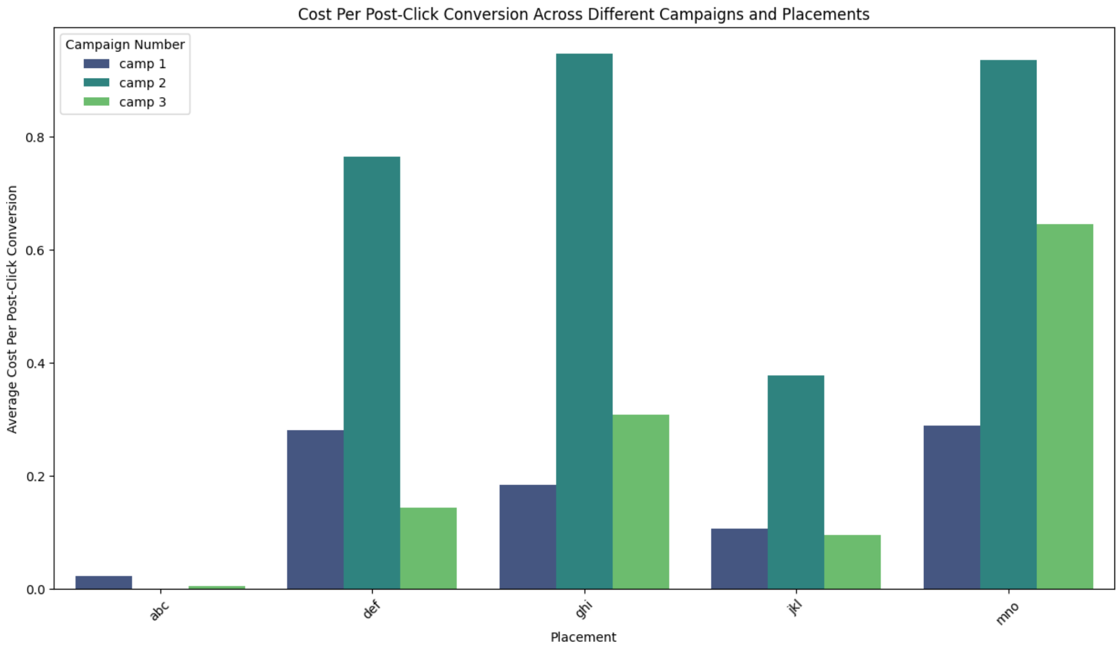
[ ]:

```
#17
#How does the cost per click (CPC) vary across different campaigns and banner␣
↪sizes?
# Calculate Cost Per Click (CPC)
data['CPC'] = data['cost'] / data['clicks']
data['CPC'].fillna(0, inplace=True)

# Group the data by campaign and banner size, then calculate the mean CPC
cpc_data = data.groupby(['campaign_number', 'banner'])['CPC'].mean().
↪reset_index()
cpc_data

# Plotting the results
plt.figure(figsize=(15, 8))
sns.barplot(x='banner', y='CPC', hue='campaign_number', data=cpc_data,␣
↪palette='viridis')
plt.title('Cost Per Click (CPC) Across Different Campaigns and Banner Sizes')
plt.xlabel('Banner Size')
plt.ylabel('Average CPC')
plt.xticks(rotation=45)
plt.legend(title='Campaign Number')
plt.show()
```



[ ]:

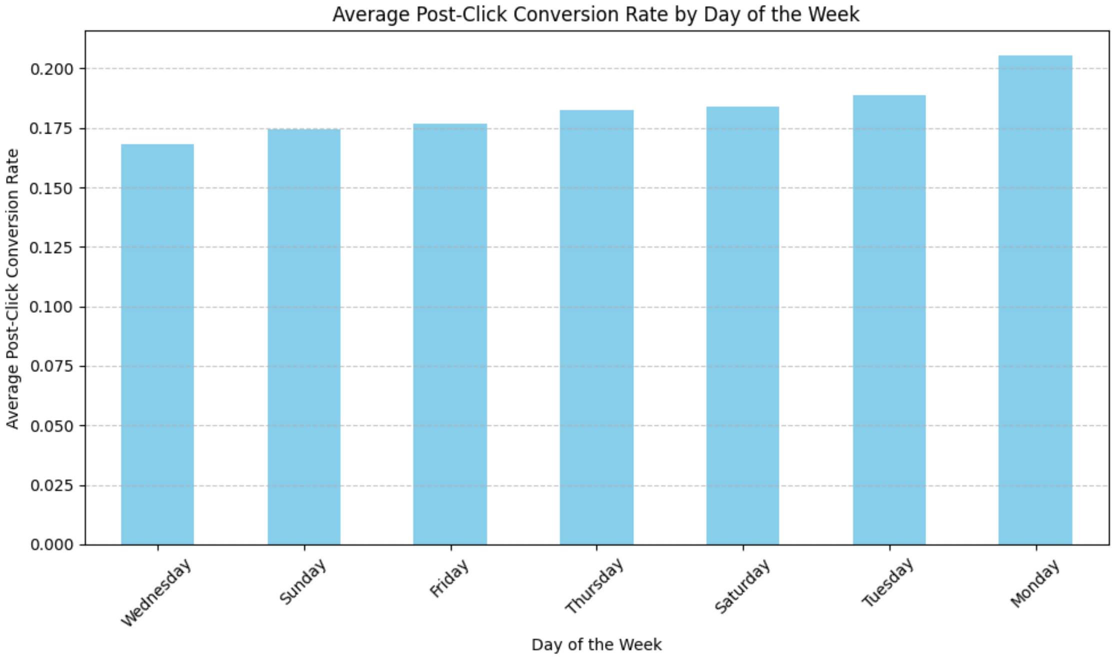
```
#18
#Are there any campaigns or placements that are particularly cost-effective in␣
↪terms of generating post-click conversions?
import numpy as np

# Calculate Cost Per Post-Click Conversion
data['Cost_Per_Post_Click_Conversion'] = data['cost'] /␣
↪data['post_click_conversions']
data['Cost_Per_Post_Click_Conversion'].replace([np.inf, -np.inf], np.nan,␣
↪inplace=True)
data['Cost_Per_Post_Click_Conversion'].fillna(0, inplace=True)  # Replace NaN␣
↪values with 0
# Group the data by campaign and placement, then calculate the mean Cost Per␣
↪Post-Click Conversion
cost_effectiveness = data.groupby(['campaign_number',␣
↪'placement'])['Cost_Per_Post_Click_Conversion'].mean().reset_index()
cost_effectiveness

# Plotting the results
plt.figure(figsize=(15, 8))
sns.barplot(x='placement', y='Cost_Per_Post_Click_Conversion',␣
↪hue='campaign_number', data=cost_effectiveness, palette='viridis')
plt.title('Cost Per Post-Click Conversion Across Different Campaigns and␣
↪Placements')
plt.xlabel('Placement')
plt.ylabel('Average Cost Per Post-Click Conversion')
plt.xticks(rotation=45)
plt.legend(title='Campaign Number')
plt.show()
```



[ ]:

```
#19
#Can we identify any trends or patterns in post-click conversion rates based on␣
 ↪the day of the week?

import pandas as pd
import matplotlib.pyplot as plt

# Converting the 'Month' and 'Day' columns to a date format and extract the day␣
 ↪of the week
data['Date'] = pd.to_datetime(data['month'] + ' ' + data['day'].astype(str),␣
 ↪format='%B %d')
data['Day_of_Week'] = data['Date'].dt.day_name()

# Calculate post-click conversion rate
data['Post_Click_Conversion_Rate'] = data['post_click_conversions'] /␣
 ↪data['clicks']
data['Post_Click_Conversion_Rate'].replace([float('inf'), -float('inf')], 0,␣
 ↪inplace=True)  # Handle infinite values

#  Group the data by day of the week and calculate average conversion rates
```

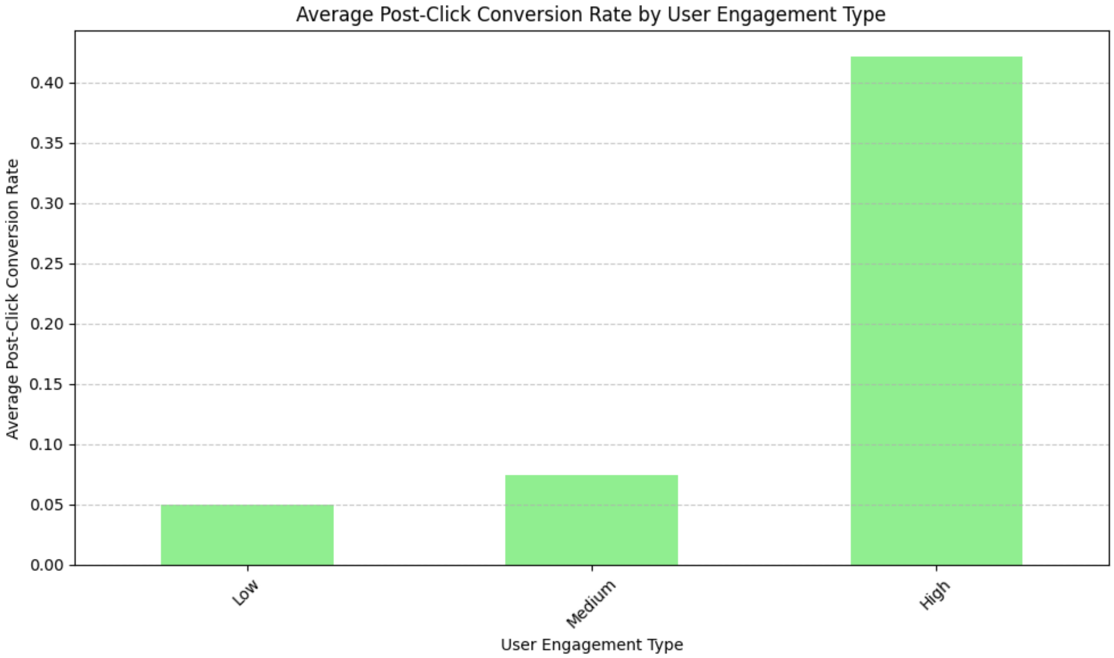
```
grouped_data = data.groupby('Day_of_Week')['Post_Click_Conversion_Rate'].mean().
 ↪sort_values()

# Plot the trends
plt.figure(figsize=(10, 6))
grouped_data.plot(kind='bar', color='skyblue')
plt.title('Average Post-Click Conversion Rate by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Post-Click Conversion Rate')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



[ ]:

```
#20
#How does the effectiveness of campaigns vary throughout different user␣
 ↪engagement types in terms of post-click conversions?
#  Calculate post-click conversion rate
data['Post_Click_Conversion_Rate'] = data['post_click_conversions'] /␣
 ↪data['clicks']
data['Post_Click_Conversion_Rate'].replace([float('inf'), -float('inf')], 0,␣
 ↪inplace=True)  # Handle infinite values

#  Group the data by user engagement type and calculate average post-click␣
 ↪conversion rates
```

```
grouped_data = data.groupby('user_engagement')['Post_Click_Conversion_Rate'].
 ↪mean().sort_values()

# Plot the trends
plt.figure(figsize=(10, 6))
grouped_data.plot(kind='bar', color='lightgreen')
plt.title('Average Post-Click Conversion Rate by User Engagement Type')
plt.xlabel('User Engagement Type')
plt.ylabel('Average Post-Click Conversion Rate')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```


Average Post-Click Conversion Rate by User Engagement Type
Average Post-Click Conversion Rate
Low
Medium
High
User Engagement Type