Online Advertising Performance Data - Basic Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/online_advertising_performance_data.csv')
```

```
df['date_str'] = df['day'].astype(str) + '-' + df['month'].astype(str) + '-2024'
df['datetime'] = pd.to_datetime(df['date_str'], format='%d-%B-%Y')
```

Double-click (or enter) to edit

```
df.head()
```

	month	day	campaign_number	user_engagement	banner	placement	displays	
0	April	1	camp 1	High	160 x 600	abc	4	
1	April	1	camp 1	High	160 x 600	def	20170	
2	April	1	camp 1	High	160 x 600	ghi	14701	
3	April	1	camp 1	High	160 x 600	mno	171259	2
4	April	1	camp 1	Low	160 x 600	def	552	

Next steps: Generate code with df View recommended plots

```
df.to_csv('updated_file.csv', index=False)
```

```
df_new=pd.read_csv('updated_file.csv')
```

```
df_new.head()
```

	month	day	campaign_number	user_engagement	banner	placement	displays	
0	April	1	camp 1	High	160 x 600	abc	4	
1	April	1	camp 1	High	160 x 600	def	20170	
2	April	1	camp 1	High	160 x 600	ghi	14701	
3	April	1	camp 1	High	160 x 600	mno	171259	2
4	April	1	camp 1	Low	160 x 600	def	552	

Next steps: Generate code with df_new View recommended plots

What is the overall trend in user engagement throughout the campaign period?

```
df_new['user_engagement'] = df_new['user_engagement'].replace({'High': 1, 'Low': 0,'Medium':0.5})
# ordinal mapping
```

```
df_new.head()
```

	month	day	campaign_number	user_engagement	banner	placement	displays	
0	April	1	camp 1	1.0	160 x 600	abc	4	
1	April	1	camp 1	1.0	160 x 600	def	20170	
2	April	1	camp 1	1.0	160 x 600	ghi	14701	
3	April	1	camp 1	1.0	160 x 600	mno	171259	2
4	April	1	camp 1	0.0	160 x 600	def	552	

Next steps: Generate code with df_new View recommended plots

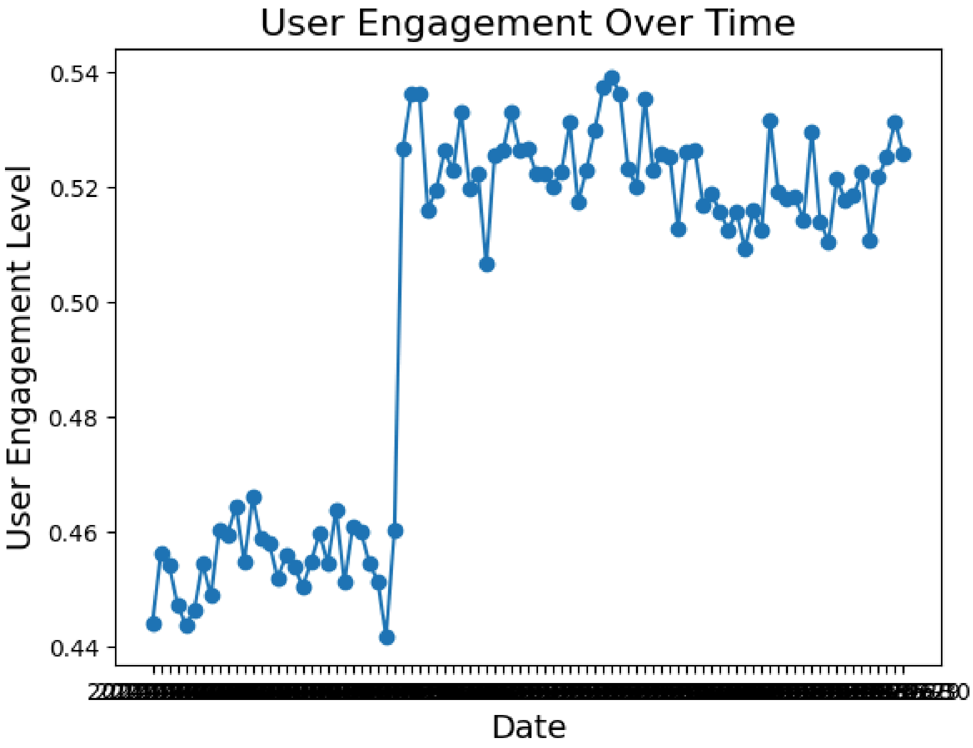
```
engagement_trend = df_new.groupby('datetime')['user_engagement'].mean()
```

```
engagement_trend
```

```
datetime
2024-04-01    0.444175
2024-04-02    0.456098
2024-04-03    0.454106
2024-04-04    0.447368
2024-04-05    0.443662
                ...
2024-06-26    0.510791
2024-06-27    0.521739
2024-06-28    0.525362
2024-06-29    0.531469
2024-06-30    0.525735
Name: user_engagement, Length: 91, dtype: float64
```

```
plt.plot(engagement_trend, marker='o')
plt.xlabel('Date', fontsize=14)
plt.ylabel('User Engagement Level', fontsize=14)
plt.title('User Engagement Over Time', fontsize=16)
plt.xticks
```

```
matplotlib.pyplot.xticks
def xticks(ticks=None, labels=None, *, minor=False, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Get or set the current tick locations and labels of the x-axis.

Pass no arguments to return the current values without modifying them.

Parameters
```



```
engagement_trend.idxmax()
```

```
'2024-05-26'
```

```
engagement_trend[engagement_trend.idxmax()]
```

```
0.5392156862745098
```

```
mean_engagement = engagement_trend.mean()
median_engagement = engagement_trend.median()
mode_engagement = engagement_trend.mode()[0]

# Description of overall trend
print(f"Mean engagement level: {mean_engagement:.2f}")
print(f"Median engagement level: {median_engagement}")
print(f"Mode engagement level: {mode_engagement}")
```

```
Mean engagement level: 0.50
Median engagement level: 0.5177304964539007
Mode engagement level: 0.45454545454545453
```

```
rolling_window = 3  # Choose a suitable window size
engagement_trend['rolling_mean'] = engagement_trend.rolling(window=rolling_window).mean()
```

Double-click (or enter) to edit

```
engagement_trend['rolling_mean']
```

```
datetime
2024-04-01         NaN
2024-04-02         NaN
2024-04-03    0.451460
2024-04-04    0.452524
2024-04-05    0.448379
                ...
2024-06-26    0.517289
2024-06-27    0.518362
2024-06-28    0.519298
2024-06-29    0.526190
2024-06-30    0.527522
Name: user_engagement, Length: 91, dtype: float64
```

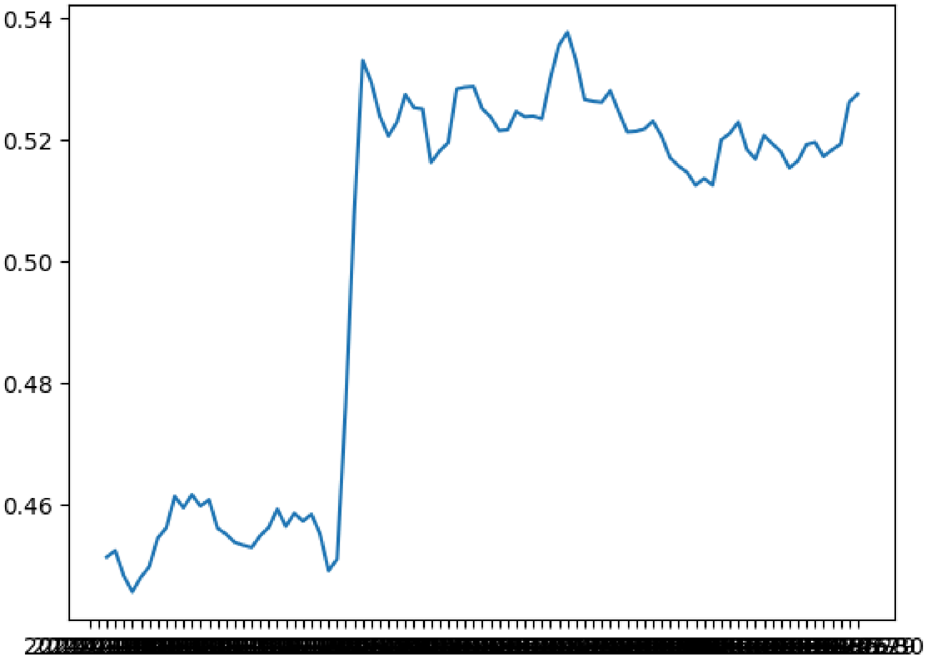
```
Start coding or generate with AI.
```

```
engagement_trend
```

```
datetime
2024-04-01                                            0.444175
2024-04-02                                            0.456098
2024-04-03                                            0.454106
2024-04-04                                            0.447368
2024-04-05                                            0.443662
                                      ...
2024-06-27                                            0.521739
2024-06-28                                            0.525362
2024-06-29                                            0.531469
2024-06-30                                            0.525735
rolling_mean    datetime
2024-04-01         NaN
2024-04-02    ...
Name: user_engagement, Length: 92, dtype: object
```

```
plt.plot(engagement_trend['rolling_mean'])
```

```
[<matplotlib.lines.Line2D at 0x7e6a5d0c6b00>]
```

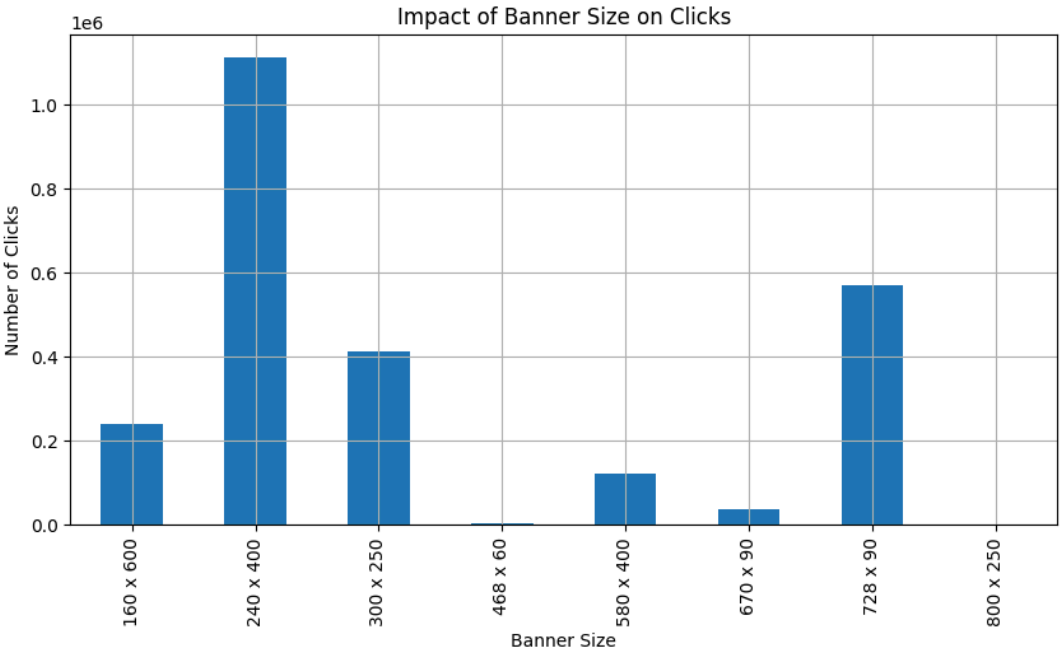


2 How does the size of the ad (banner) impact the number of clicks generated?

```
banner_clicks = df.groupby('banner')['clicks'].sum()
banner_clicks
```

```
banner
160 x 600     239570
240 x 400    1113256
300 x 250     411214
468 x 60        1295
580 x 400     120681
670 x 90       37203
728 x 90      569606
800 x 250         12
Name: clicks, dtype: int64
```

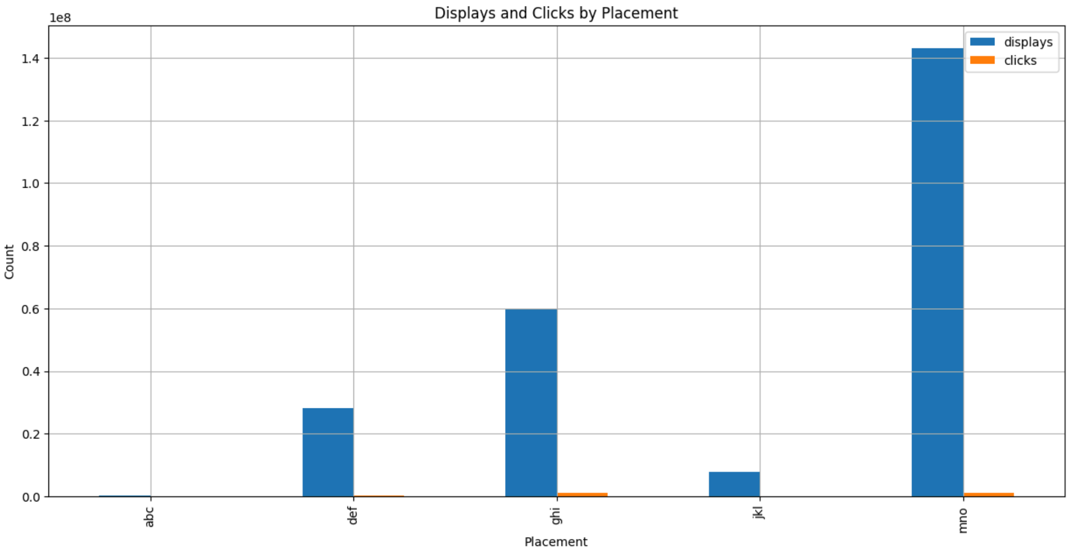
```
banner_clicks.plot(kind='bar', figsize=(10, 5))
plt.title('Impact of Banner Size on Clicks')
plt.xlabel('Banner Size')
plt.ylabel('Number of Clicks')
plt.grid(True)
plt.show()
```



Which publisher spaces (placements) yielded the highest number of displays and clicks?

```
placement_performance = df.groupby('placement')[['displays', 'clicks']].sum()
```

```
placement_performance.plot(kind='bar', figsize=(15, 7))
plt.title('Displays and Clicks by Placement')
plt.xlabel('Placement')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



```
placement_performance
```

placement	displays	clicks
abc	242142	1584
def	28177492	176097
ghi	59740415	1247049
jkl	7692732	75063
mno	143161775	993039

Start coding or generate with AI.

```
max_values = placement_performance.max()
print(max_values)
```

```
displays    143161775
clicks        1247049
dtype: int64
```

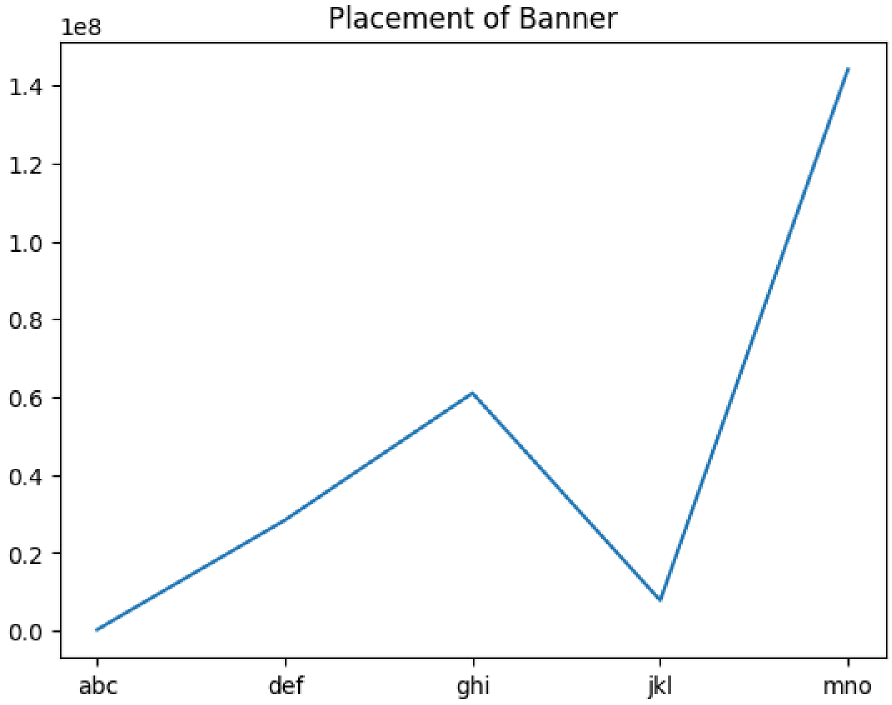
```
placement_performance.idxmax()
```

```
displays    mno
clicks      ghi
dtype: object
```

```
placement_performance['sum'] = placement_performance['clicks'] + placement_performance['displays']
```

```
plt.plot(placement_performance['sum'])
plt.title('Placement of Banner')
```

```
Text(0.5, 1.0, 'Placement of Banner')
```



```
placement_performance['displays'].mean()
```

```
47802911.2
```

```
placement_performance['clicks'].mean()
```

```
498566.4
```

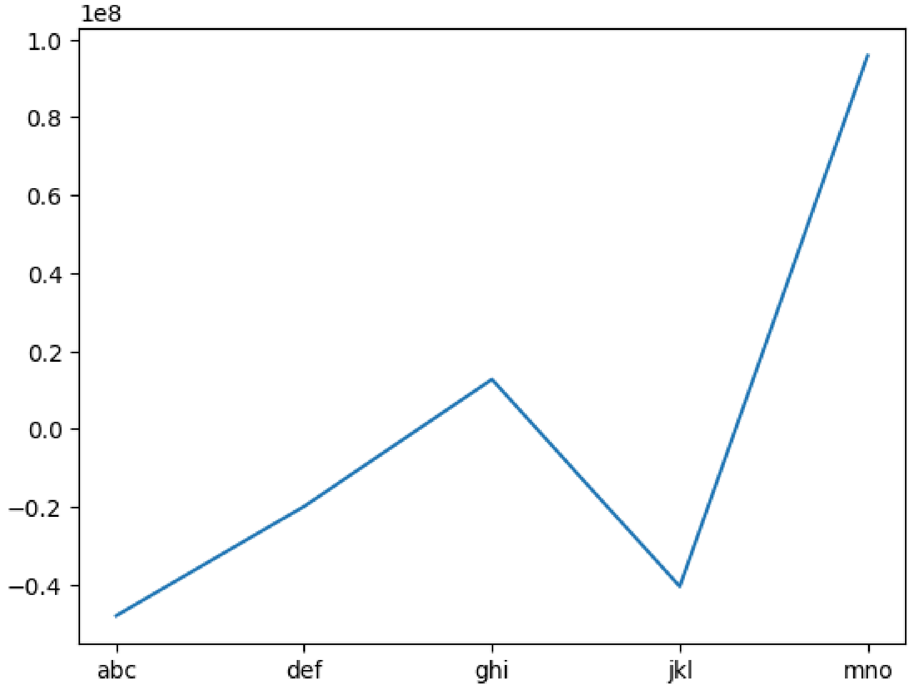
```
placement_performance['sum']-=placement_performance['clicks'].mean()+placement_performance['displa
```

```
placement_performance
```

	displays	clicks	sum
placement			
abc	242142	1584	-48057751.6
def	28177492	176097	-19947888.6
ghi	59740415	1247049	12685986.4
jkl	7692732	75063	-40533682.6
mno	143161775	993039	95853336.4

```
plt.plot(placement_performance['sum'])
```

```
[<matplotlib.lines.Line2D at 0x7e048750d510>]
```



```
placement_performance['sum'].max()
```

```
95853336.4
```

```
placement_performance['sum'].idxmax()
```

```
'mno'
```

```
placement_performance['displays'].min()
```

```
242142
```

```
placement_performance['displays'].max()
```

```
143161775
```

```
placement_performance['clicks'].min()
```

```
1584
```

```
placement_performance['clicks'].max()
```

```
1247049
```

```
placement_performance_norm = (placement_performance-placement_performance.min())/(placement_perfor
```

```
placement_performance_norm
```

	displays	clicks	sum
placement			
abc	0.000000	0.000000	0.000000
def	0.195462	0.140119	0.195328
ghi	0.416306	1.000000	0.422092
jkl	0.052131	0.058997	0.052283
mno	1.000000	0.796052	1.000000

Start coding or generate with AI.

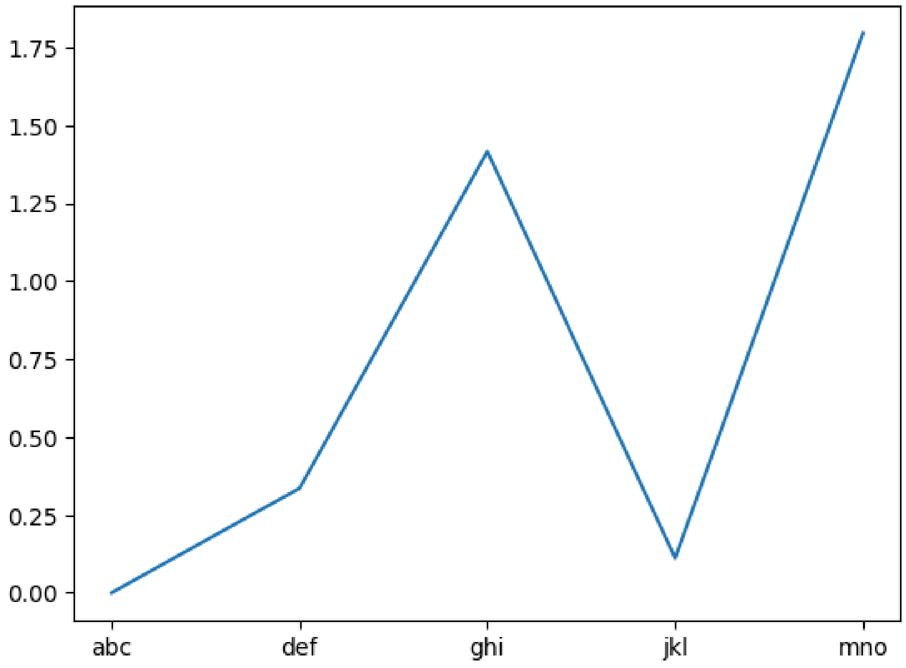
```
placement_performance_norm['sum']=placement_performance_norm['displays']+placement_performance_nor
```

```
placement_performance_norm
```

	displays	clicks	sum
placement			
abc	0.000000	0.000000	0.000000
def	0.195462	0.140119	0.335581
ghi	0.416306	1.000000	1.416306
jkl	0.052131	0.058997	0.111129
mno	1.000000	0.796052	1.796052

```
plt.plot(placement_performance_norm['sum'])
```

```
[<matplotlib.lines.Line2D at 0x7e048755f4f0>]
```



```
placement_performance_norm['sum'].idxmax()
```

```
'mno'
```

Double-click (or enter) to edit

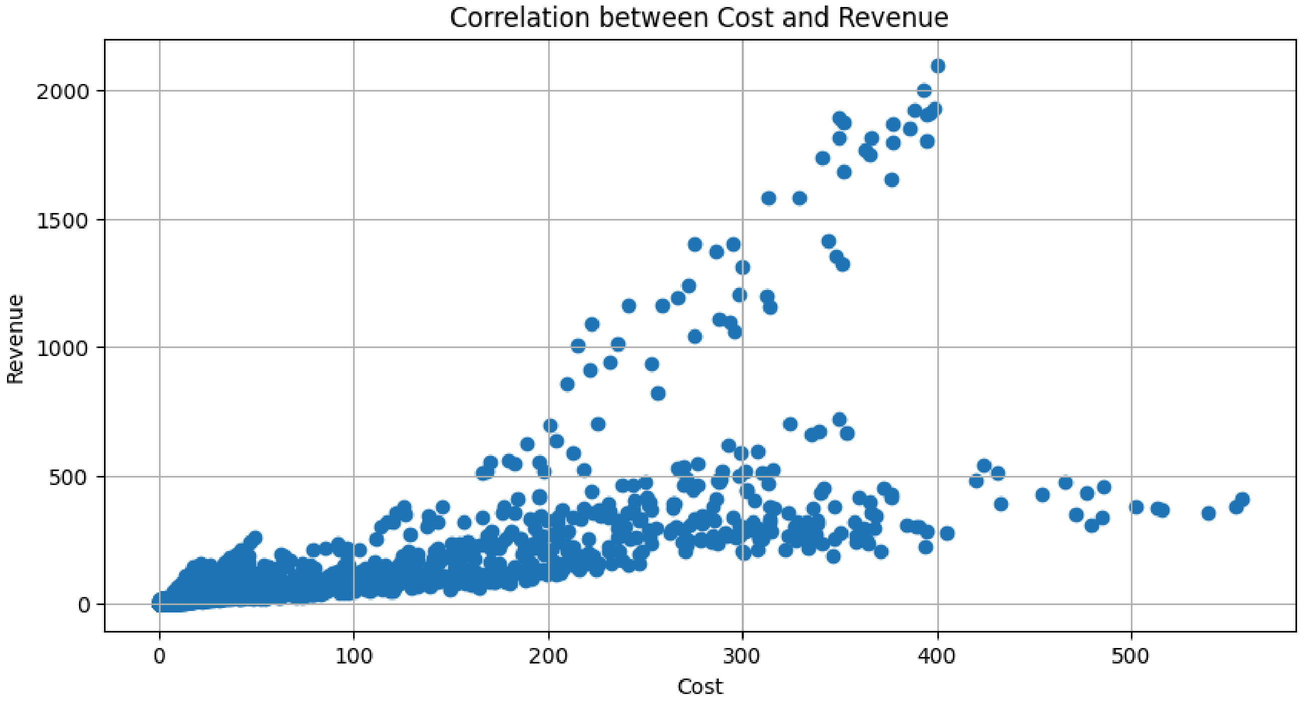
```
placement_performance.loc['mno']
```

```
displays    143161775.0
clicks         993039.0
sum          95853336.4
Name: mno, dtype: float64
```

Is there a correlation between the cost of serving ads and the revenue generated from clicks?

```
correlation = df_new['cost'].corr(df_new['revenue'])

print(f'Correlation between Cost and Revenue: {correlation}')

plt.figure(figsize=(10, 5))
plt.scatter(df_new['cost'], df_new['revenue'])
plt.title('Correlation between Cost and Revenue')
plt.xlabel('Cost')
plt.ylabel('Revenue')
plt.grid(True)
plt.show()
```

```
Correlation between Cost and Revenue: 0.7605199343382271
```



```
correlation
```

```
0.7605199343382271
```

Double-click (or enter) to edit

What is the average revenue generated per click for Company X during the campaign period?

```
average_revenue_per_click = df_new['revenue'].sum() / df_new['clicks'].sum()

print(f'Average Revenue Per Click: ${average_revenue_per_click:.2f}')
```

```
Average Revenue Per Click: $0.11
```

Which campaigns had the highest post-click conversion rates?

```
df_new['Conversion Rate'] = df_new['post_click_conversions'] / df_new['clicks']
```

```
df_new
```

	month	day	campaign_number	user_engagement	banner	placement	displays	cost	clicks
0	April	1	camp 1	1.0	160 x 600	abc	4	0.0060	C
1	April	1	camp 1	1.0	160 x 600	def	20170	26.7824	158
2	April	1	camp 1	1.0	160 x 600	ghi	14701	27.6304	158
3	April	1	camp 1	1.0	160 x 600	mno	171259	216.8750	1796
4	April	1	camp 1	0.0	160 x 600	def	552	0.0670	1
...	...	...	...	...	...	...	...	...	...
15403	April	1	camp 1	0.0	160 x 600	ghi	16	0.0249	C
15404	April	1	camp 1	0.0	160 x 600	mno	2234	0.4044	1C
15405	June	29	camp 1	1.0	800 x 250	ghi	1	0.0157	C
15406	June	29	camp 1	1.0	800 x 250	mno	4	0.0123	C
15407	June	29	camp 3	1.0	240 x 400	def	1209	0.3184	2

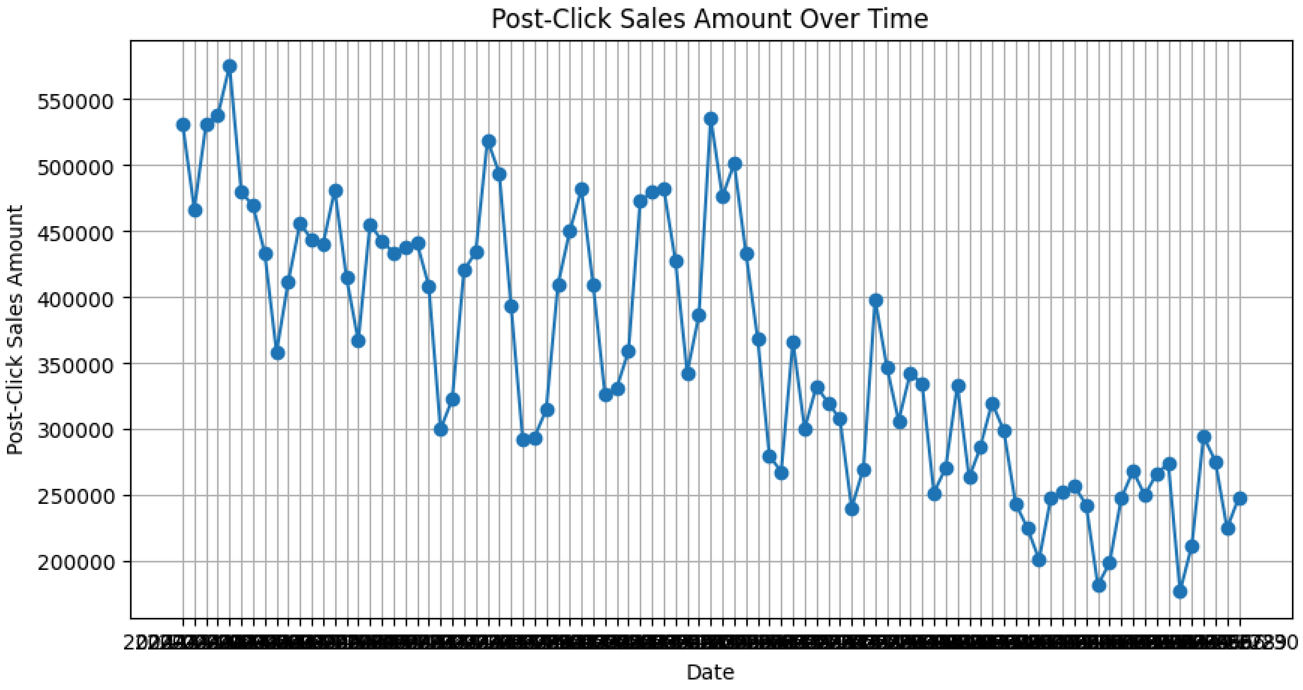
15408 rows × 17 columns

```
top_campaigns = df_new.groupby('campaign_number')['Conversion Rate'].mean().sort_values(ascending=

print(top_campaigns)
```

```
campaign_number
camp 3    0.045453
camp 2    0.020079
camp 1         NaN
Name: Conversion Rate, dtype: float64
```

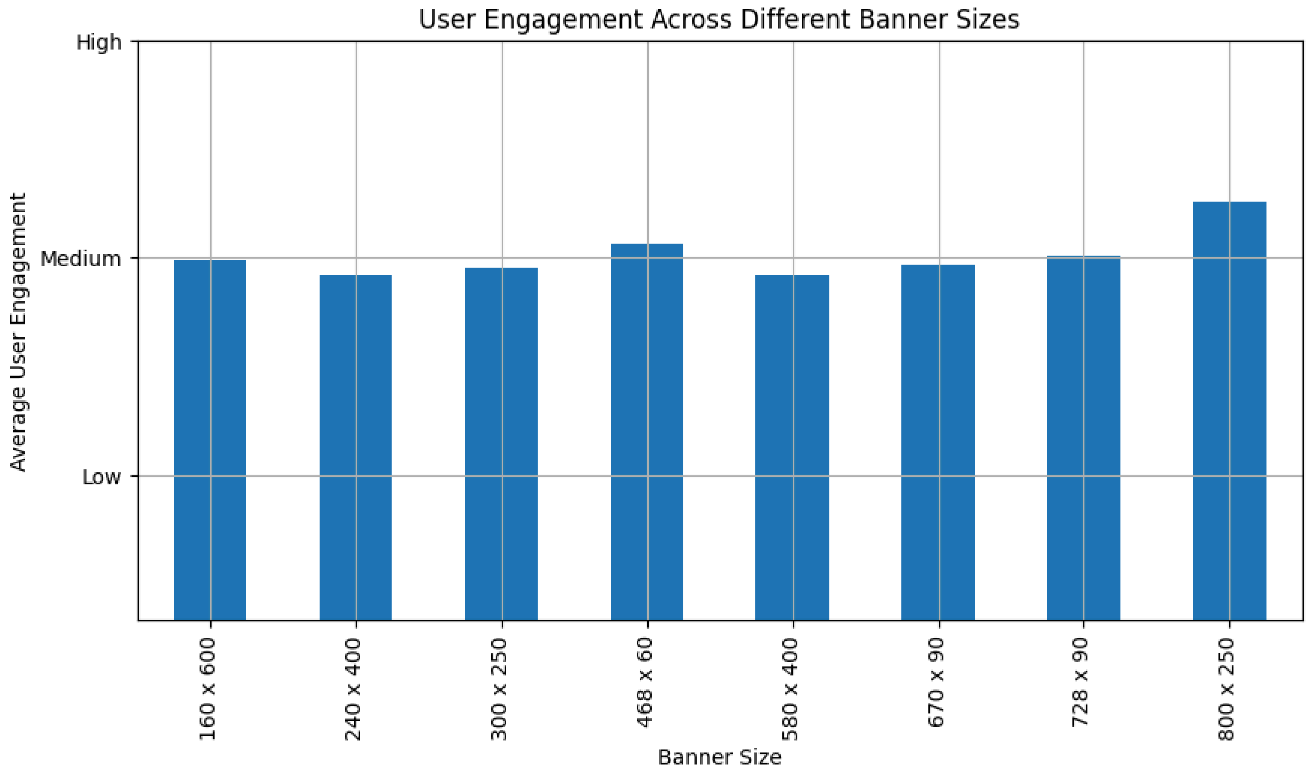
Are there any specific trends or patterns in post-click sales amounts over time?

```
sales_trend = df_new.groupby('datetime')['post_click_sales_amount'].sum()


plt.figure(figsize=(10, 5))
plt.plot(sales_trend, marker='o')
plt.title('Post-Click Sales Amount Over Time')
plt.xlabel('Date')
plt.ylabel('Post-Click Sales Amount')
plt.grid(True)
plt.show()
```



How does the level of user engagement vary across different banner sizes?

```
banner_engagement = df_new.groupby('banner')['user_engagement'].mean()

engagement_labels = {
    'High': banner_engagement > 0.7,
    'Medium': (banner_engagement >= 0.4) & (banner_engagement <= 0.7),
    'Low': banner_engagement < 0.4
}



ax = banner_engagement.plot(kind='bar', figsize=(10, 5))
plt.title('User Engagement Across Different Banner Sizes')
plt.xlabel('Banner Size')
plt.ylabel('Average User Engagement')
plt.grid(True)


ax.set_yticks([0.2, 0.5, 0.8])
ax.set_yticklabels(['Low', 'Medium', 'High'])

plt.show()
```



Which placement types result in the highest post-click conversion rates?

```
placement_conversion_rate = df.groupby('placement')['post_click_conversions'].sum() / df.groupby('

top_placements = placement_conversion_rate.sort_values(ascending=False).head()

print(top_placements)
```

```
placement
abc    0.520202
jkl    0.277807
ghi    0.270288
mno    0.265015
def    0.169543
dtype: float64
```

```
df_new.head()
```

	month	day	campaign_number	user_engagement	banner	placement	displays	cost	clicks	re
0	April	1	camp 1	1.0	160 x 600	abc	4	0.0060	0	
1	April	1	camp 1	1.0	160 x 600	def	20170	26.7824	158	2
2	April	1	camp 1	1.0	160 x 600	ghi	14701	27.6304	158	2
3	April	1	camp 1	1.0	160 x 600	mno	171259	216.8750	1796	32
4	April	1	camp 1	0.0	160 x 600	def	552	0.0670	1	

```
df_new['datetime'] = pd.to_datetime(df_new['datetime'])

df_new['Month'] = df_new['datetime'].dt.to_period('M')

print(df_new)
```

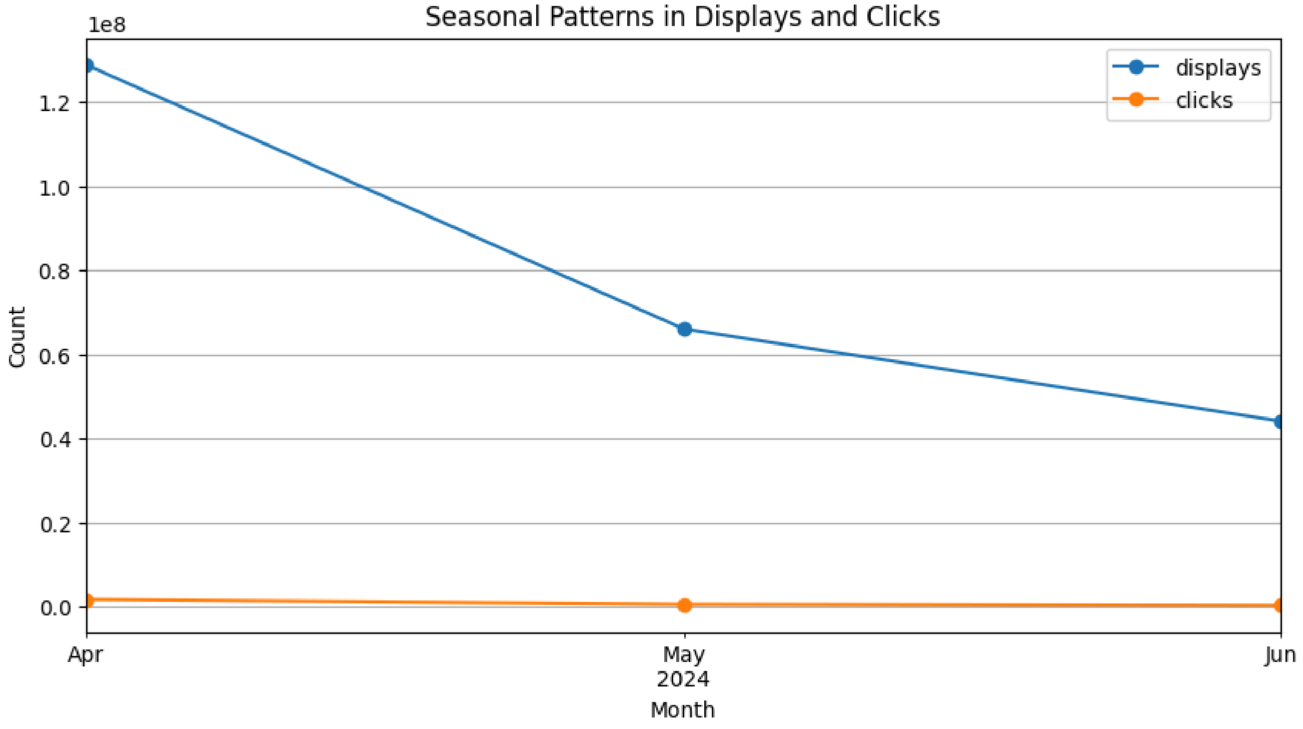
```
       month  day campaign_number  user_engagement     banner placement  \
0      April    1          camp 1              1.0  160 x 600       abc   
1      April    1          camp 1              1.0  160 x 600       def   
2      April    1          camp 1              1.0  160 x 600       ghi   
3      April    1          camp 1              1.0  160 x 600       mno   
4      April    1          camp 1              0.0  160 x 600       def   
...      ...  ...             ...              ...        ...       ...   
15403  April    1          camp 1              0.0  160 x 600       ghi   
15404  April    1          camp 1              0.0  160 x 600       mno   
15405   June   29          camp 1              1.0  800 x 250       ghi   
15406   June   29          camp 1              1.0  800 x 250       mno   
15407   June   29          camp 3              1.0  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_str  \
0                       0.0000          NaN          NaN  1-April-2024   
1                    1972.4602          NaN          NaN  1-April-2024   
2                    2497.2636          NaN          NaN  1-April-2024   
3                   24625.3234          NaN          NaN  1-April-2024   
4                       0.0000          NaN          NaN  1-April-2024   
...                        ...          ...          ...           ...   
15403                   0.0000          NaN          NaN  1-April-2024   
15404                 101.7494          NaN          NaN  1-April-2024   
15405                   0.0000          NaN          NaN  29-June-2024   
15406                   0.0000          NaN          NaN  29-June-2024   
15407                 110.4224          NaN          NaN  29-June-2024   

        datetime  Conversion Rate    Month  
0     2024-04-01              NaN  2024-04  
1     2024-04-01         0.145570  2024-04  
2     2024-04-01         0.493671  2024-04  
3     2024-04-01         0.343541  2024-04  
4     2024-04-01         0.000000  2024-04  
...          ...              ...      ...  
15403 2024-04-01              NaN  2024-04  
15404 2024-04-01         0.300000  2024-04  
15405 2024-06-29              NaN  2024-06  
15406 2024-06-29              NaN  2024-06  
15407 2024-06-29         1.500000  2024-06  

[15408 rows x 18 columns]
```

```
df_new.head()
```

	month	day	campaign_number	user_engagement	banner	placement	displays	cost	clicks	re
0	April	1	camp 1	1.0	160 x 600	abc	4	0.0060	0	
1	April	1	camp 1	1.0	160 x 600	def	20170	26.7824	158	2
2	April	1	camp 1	1.0	160 x 600	ghi	14701	27.6304	158	2
3	April	1	camp 1	1.0	160 x 600	mno	171259	216.8750	1796	32
4	April	1	camp 1	0.0	160 x 600	def	552	0.0670	1	

Can we identify any seasonal patterns or fluctuations in displays and clicks throughout the campaign period?

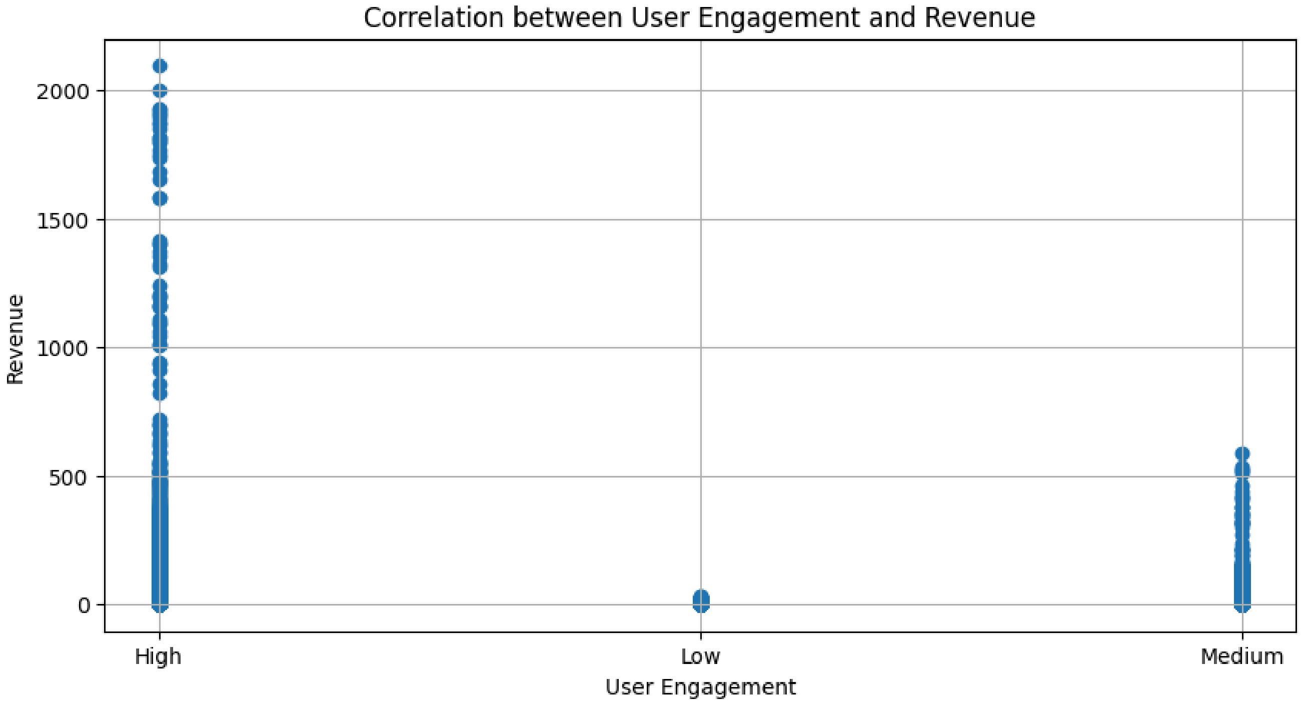
```
#10
monthly_performance = df_new.groupby('Month')[['displays', 'clicks']].sum()


monthly_performance.plot(kind='line', marker='o', figsize=(10, 5))
plt.title('Seasonal Patterns in Displays and Clicks')
plt.xlabel('Month')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



Is there a correlation between user engagement levels and the revenue generated?

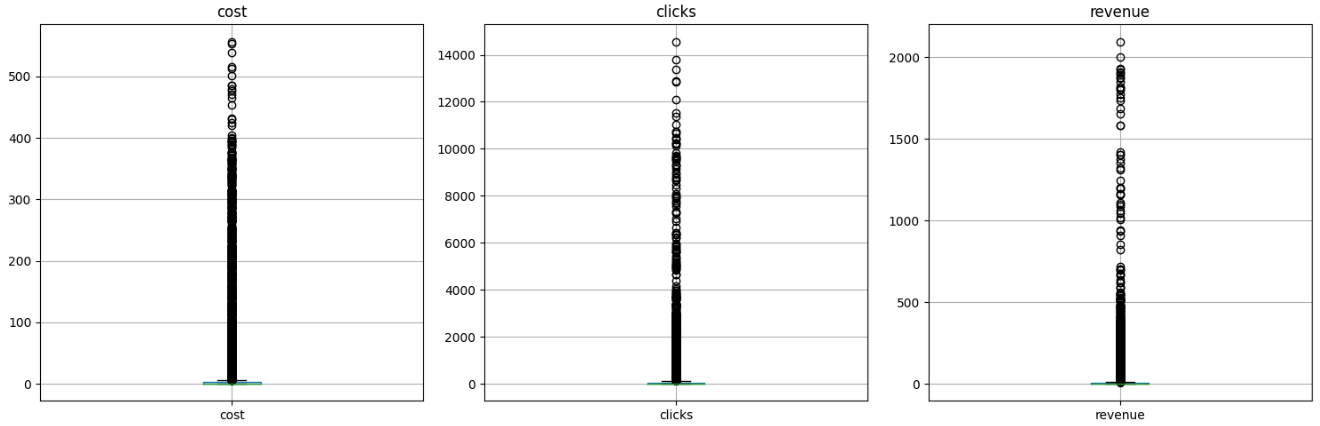
```
correlation = df_new['user_engagement'].corr(df_new['revenue'])

print(f'Correlation between User Engagement and Revenue: {correlation}')


plt.figure(figsize=(10, 5))
plt.scatter(df['user_engagement'], df['revenue'])
plt.title('Correlation between User Engagement and Revenue')
plt.xlabel('User Engagement')
plt.ylabel('Revenue')
plt.grid(True)
plt.show()
```

```
Correlation between User Engagement and Revenue: 0.1753892426950314
```



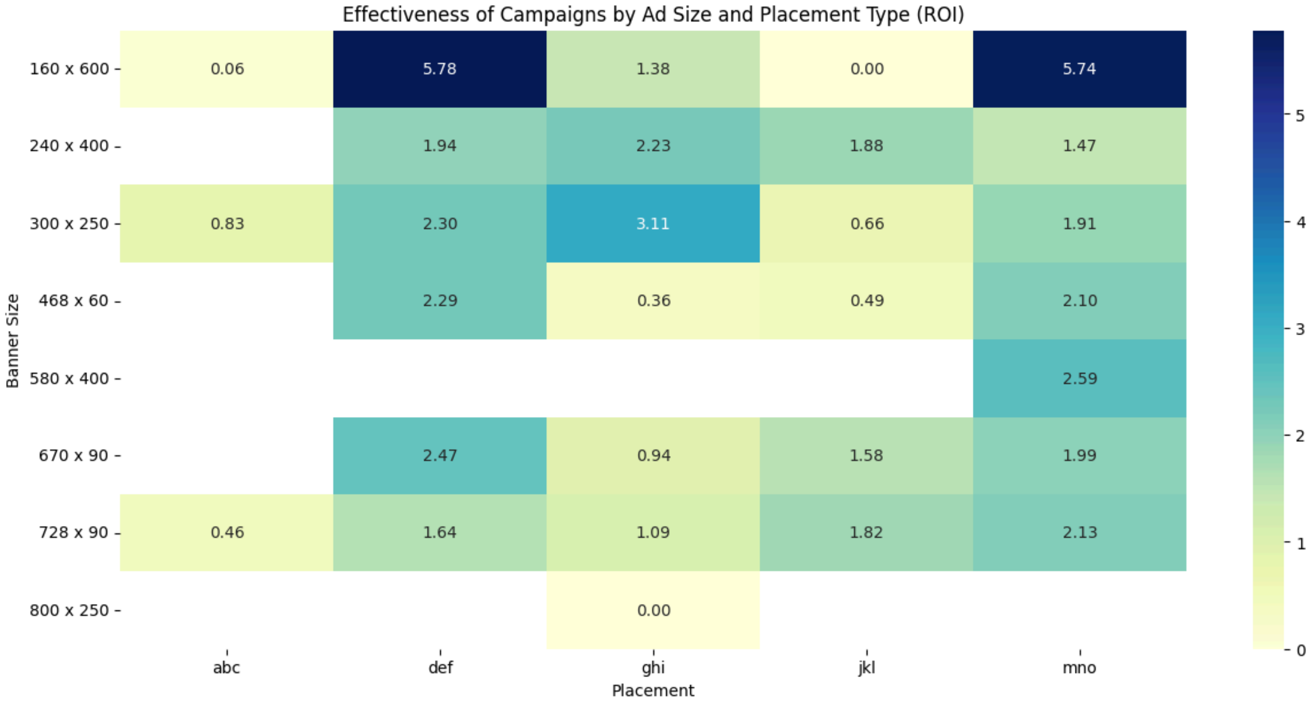
Are there any outliers in terms of cost, clicks, or revenue that warrant further investigation?

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
df.boxplot(column='cost', ax=axes[0])
axes[0].set_title('cost')
df.boxplot(column='clicks', ax=axes[1])
axes[1].set_title('clicks')
df.boxplot(column='revenue', ax=axes[2])
axes[2].set_title('revenue')

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



How does the effectiveness of campaigns vary based on the size of the ad and placement type?

```
df_new['ROI'] = df_new['revenue'] / df_new['cost']


roi_analysis = df_new.groupby(['banner', 'placement'])['ROI'].mean().unstack()

plt.figure(figsize=(15, 7))
sns.heatmap(roi_analysis, annot=True, fmt='.2f', cmap='YlGnBu')
plt.title('Effectiveness of Campaigns by Ad Size and Placement Type (ROI)')
plt.xlabel('Placement')
plt.ylabel('Banner Size')
plt.show()
```



Are there any specific campaigns or banner sizes that consistently outperform others in terms of ROI

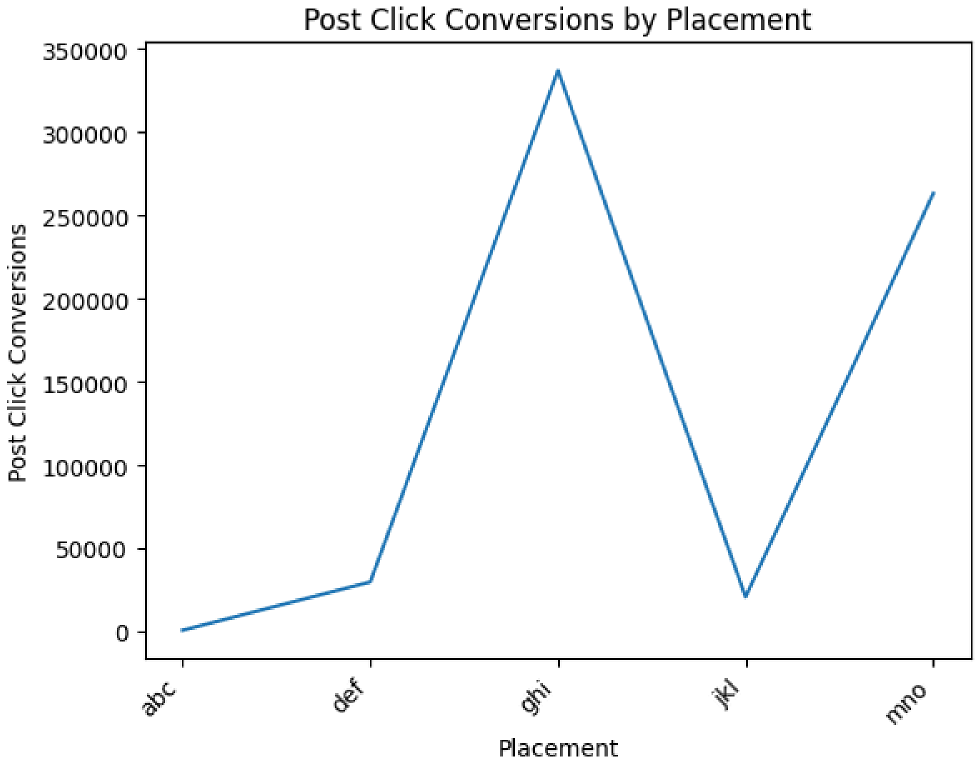
```
#14
campaign_roi = df_new.groupby(['campaign_number', 'banner'])['ROI'].mean().sort_values(ascending=F

print(campaign_roi)
```

```
campaign_number  banner
camp 3           160 x 600    4.096757
camp 1           160 x 600    3.931911
camp 2           580 x 400    3.113678
camp 3           580 x 400    3.004255
camp 1           240 x 400    2.531187
Name: ROI, dtype: float64
```

What is the distribution of post-click conversions across different placement types?

```
#15
conversionsByPlacement = df_new.groupby('placement')['post_click_conversions'].sum()
plt.plot(conversionsByPlacement.index, conversionsByPlacement.values)
plt.xlabel('Placement')
plt.xticks(rotation=45, ha = 'right')
plt.ylabel('Post Click Conversions')
plt.title('Post Click Conversions by Placement')
```

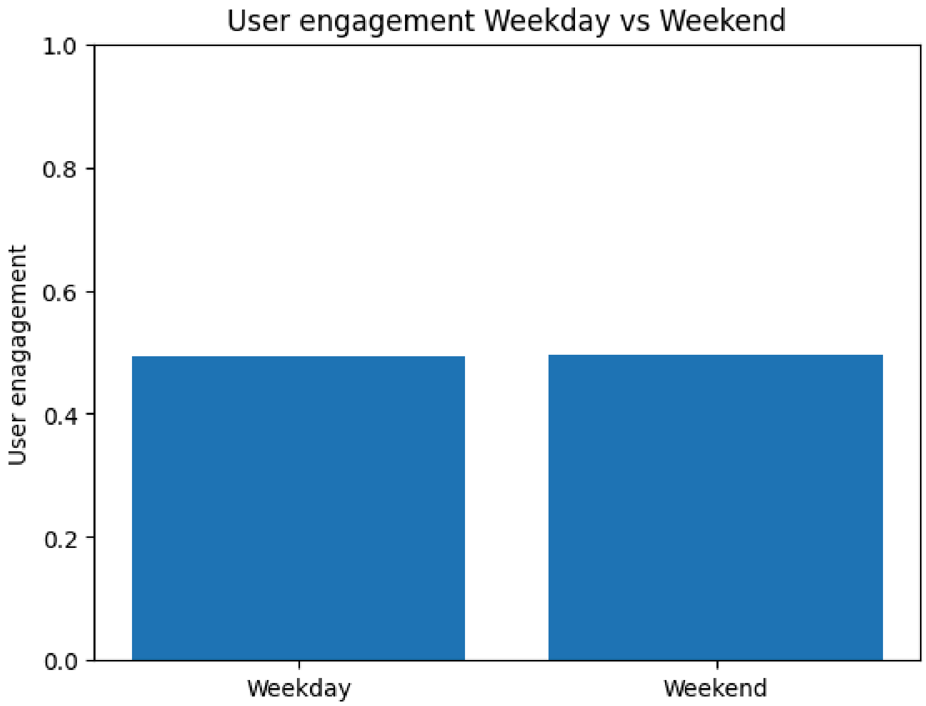
```
Text(0.5, 1.0, 'Post Click Conversions by Placement')
```



Are there any noticeable differences in user engagement levels between weekdays and weekends?

```
#16
df_new['DayType'] = df_new['datetime'].dt.dayofweek.apply(lambda x: 'Weekend' if x >= 5 else 'Week
engagementWeekday = df_new.groupby('DayType')['user_engagement'].mean()
plt.bar(engagementWeekday.index, engagementWeekday.values)
plt.ylim(0,1)
plt.ylabel('User enagagement')
plt.title('User engagement Weekday vs Weekend')
```

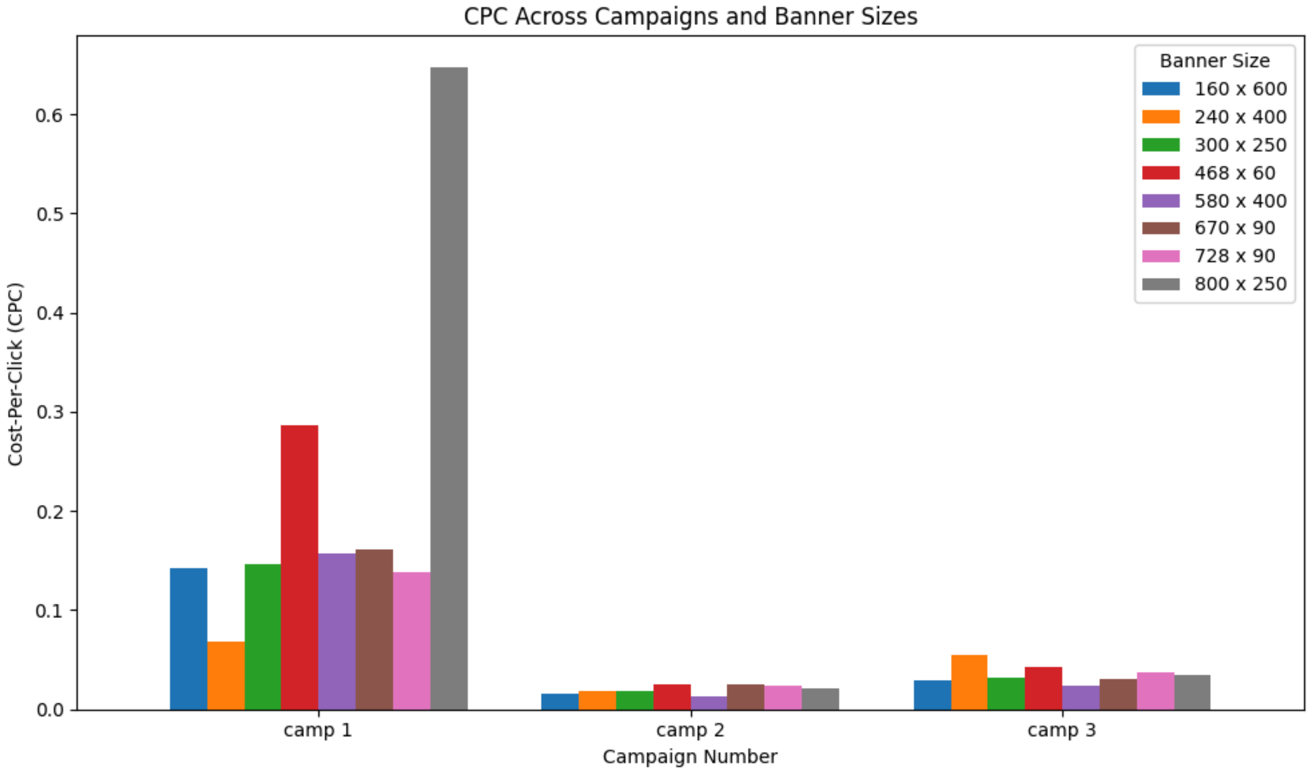
```
Text(0.5, 1.0, 'User engagement Weekday vs Weekend')
```



How does the cost per click (CPC) vary across different campaigns and banner sizes?

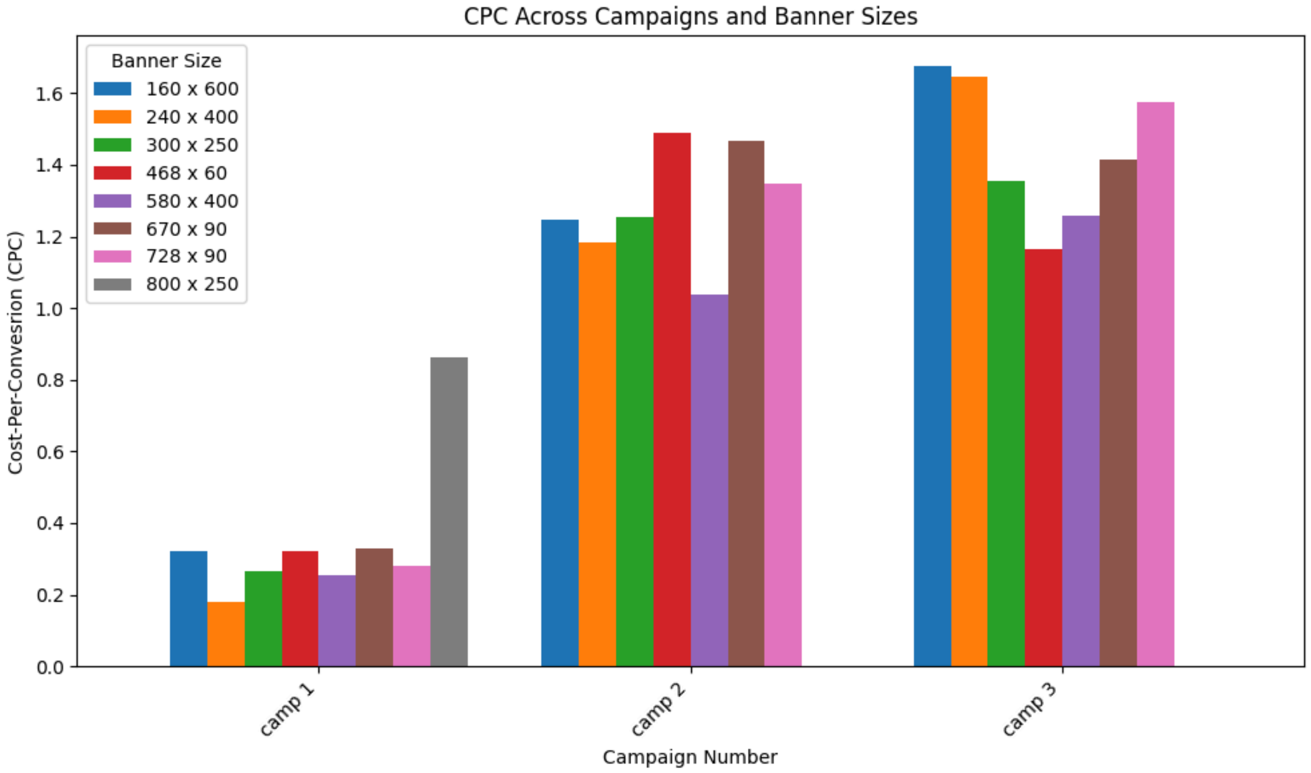
```
clicksCost = df_new.groupby(['campaign_number', 'banner'])['cost'].sum()
clicksTotal = df_new.groupby(['campaign_number', 'banner'])['clicks'].sum()

CPCdataset = clicksCost / clicksTotal

CPC_df = CPCdataset.reset_index()
CPC_df.columns = ['Campaign Number', 'Banner Size', 'CPC']

CPC_pivot = CPC_df.pivot(index='Campaign Number', columns='Banner Size', values='CPC')

CPC_pivot.plot(kind='bar', figsize=(10, 6), width=0.8)


plt.xlabel('Campaign Number')
plt.ylabel('Cost-Per-Click (CPC)')
plt.title('CPC Across Campaigns and Banner Sizes')
plt.xticks(rotation=0)
plt.legend(title='Banner Size')
plt.tight_layout()
print(CPC_pivot)
```

```
Banner Size      160 x 600  240 x 400  300 x 250  468 x 60  580 x 400  \
Campaign Number
camp 1            0.141828   0.068478   0.145982  0.286150   0.157525
camp 2            0.015196   0.018840   0.018667  0.025294   0.013562
camp 3            0.028731   0.055356   0.032140  0.042840   0.023580

Banner Size      670 x 90  728 x 90  800 x 250
Campaign Number
camp 1           0.161551  0.137835    0.64665
camp 2           0.025209  0.024304    0.02164
camp 3           0.030377  0.036951    0.03520
```



Are there any campaigns or placements that are particularly cost-effective in terms of generating post-click conversions?

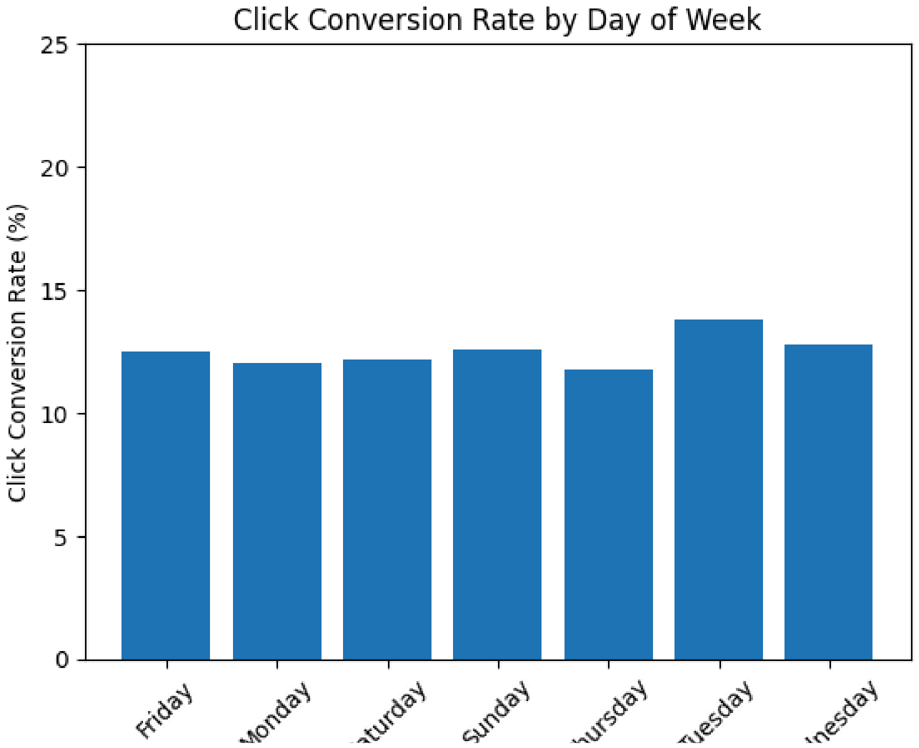
```
#18
conversionCost = df_new.groupby(['campaign_number', 'banner'])['cost'].sum()
conversionTotal = df_new.groupby(['campaign_number', 'banner'])['post_click_conversions'].sum()

CPCdataset = conversionCost / conversionTotal

CPC_df = CPCdataset.reset_index()
CPC_df.columns = ['Campaign Number', 'Banner Size', 'CPC']

CPC_pivot = CPC_df.pivot(index='Campaign Number', columns='Banner Size', values='CPC')

CPC_pivot.plot(kind='bar', figsize=(10, 6), width=0.8)

plt.xlabel('Campaign Number')
plt.ylabel('Cost-Per-Convesrion (CPC)')
plt.title('CPC Across Campaigns and Banner Sizes')
plt.xticks(rotation=45, ha = 'right')
plt.legend(title='Banner Size')
plt.tight_layout()
```



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

```
#19
import numpy as np
df_new['DayOfWeek'] = df_new['datetime'].dt.day_name()
df_new['click_conversion_rate'] = np.where(df_new['clicks'] > 0, ((df_new['post_click_conversions'
conversionByDayOFWeek = df_new.groupby('DayOfWeek')['click_conversion_rate'].mean()
plt.bar(conversionByDayOFWeek.index, conversionByDayOFWeek.values)
plt.xlabel('Day of Week')
plt.xticks(rotation = 45)
plt.ylabel('Click Conversion Rate (%)')
plt.ylim(0, 25)
plt.title('Click Conversion Rate by Day of Week')
```

```
Text(0.5, 1.0, 'Click Conversion Rate by Day of Week')
```


Click Conversion Rate by Day of Week
Click Conversion Rate (%)
Friday
Monday
aturday
Sunday
hursday
Tuesday
inesday