Importing required libraries

In [23]:
```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)

# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)

# to scale the data using z-score
from sklearn.preprocessing import StandardScaler

# to perform k-means clustering and compute silhouette scores
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# to create dendrograms
from scipy.cluster.hierarchy import dendrogram, linkage

# to suppress warnings
import warnings

warnings.filterwarnings("ignore")
```

Problem Statement:

Clustering:

Digital Ads Data:

The ads24x7 is a Digital Marketing company which has now got seed funding of $10 Million. They are expanding their wings in Marketing Analytics. They collected data from their Marketing Intelligence team and now wants you (their newly appointed data analyst) to segment type of ads based on the features provided. Use Clustering procedure to segment ads into homogeneous groups.

The following three features are commonly used in digital marketing:

CPM = (Total Campaign Spend / Number of Impressions) * 1,000. Note that the Total Campaign Spend refers to the 'Spend' Column in the dataset and the Number of Impressions refers to the 'Impressions' Column in the dataset.

CPC = Total Cost (spend) / Number of Clicks. Note that the Total Cost (spend) refers to the 'Spend' Column in the dataset and the Number of Clicks refers to the 'Clicks' Column in the dataset.

CTR = Total Measured Clicks / Total Measured Ad Impressions x 100. Note that the Total Measured Clicks refers to the 'Clicks' Column in the dataset and the Total Measured Ad Impressions refers to the 'Impressions' Column in the dataset.

Data Dictionary

Timestamp: The Timestamp of the particular Advertisement.
InventoryType: The Inventory Type of the particular Advertisement. Format 1 to 7. This is a Categorical Variable.
Ad - Length: The Length Dimension of the particular Advertisement.
Ad- Width: The Width Dimension of the particular Advertisement.
Ad Size: The Overall Size of the particular Advertisement. Length*Width.
Ad Type: The type of the particular Advertisement. This is a Categorical Variable.
Platform: The platform in which the particular Advertisement is displayed. Web, Video or App. This is a Categorical Variable.
Device Type: The type of the device which supports the particular Advertisement. This is a Categorical Variable.
Format: The Format in which the Advertisement is displayed. This is a Categorical Variable.
Available_Impressions: How often the particular Advertisement is shown. An impression is counted each time an Advertisement is shown on a search result page or other site on a Network.
Matched_Queries: Matched search queries data is pulled from Advertising Platform and consists of the exact searches typed into the search Engine that generated clicks for the particular Advertisement.
Impressions: The impression count of the particular Advertisement out of the total available impressions.
Clicks: It is a marketing metric that counts the number of times users have clicked on the particular advertisement to reach an online property.
Spend: It is the amount of money spent on specific ad variations within a specific campaign or ad set. This metric helps regulate ad performance.
Fee: The percentage of the Advertising Fees payable by Franchise Entities.
Revenue: It is the income that has been earned from the particular advertisement.
CTR: CTR stands for "Click through rate". CTR is the number of clicks that your ad receives divided by the number of times your ad is shown. Formula used here is CTR = Total Measured Clicks / Total Measured Ad Impressions x 100. Note that the Total Measured Clicks refers to the 'Clicks' Column and the Total Measured Ad Impressions refers to the 'Impressions' Column.
CPM: CPM stands for "cost per 1000 impressions." Formula used here is CPM = (Total Campaign Spend / Number of Impressions) * 1,000. Note that the Total Campaign Spend

refers to the 'Spend' Column and the Number of Impressions refers to the 'Impressions' Column.

CPC: CPC stands for "Cost-per-click". Cost-per-click (CPC) bidding means that you pay for each click on your ads. The Formula used here is CPC = Total Cost (spend) / Number of Clicks. Note that the Total Cost (spend) refers to the 'Spend' Column and the Number of Clicks refers to the 'Clicks' Column.

Understanding the structure of data

In [24]:
```
df_clust = pd.read_excel('Clustering+Clean+Ads_Data.xlsx')
```

In [25]:
```
df_clust.head() # Returns first 5 rows
```

Out[25]:

	Timestamp	InventoryType	Ad - Length	Ad- Width	Ad Size	Ad Type	Platform	Device Type	Format
0	2020-9-2-17	Format1	300	250	75000	Inter222	Video	Desktop	Display
1	2020-9-2-10	Format1	300	250	75000	Inter227	App	Mobile	Video
2	2020-9-1-22	Format1	300	250	75000	Inter222	Video	Desktop	Display
3	2020-9-3-20	Format1	300	250	75000	Inter228	Video	Mobile	Video
4	2020-9-4-15	Format1	300	250	75000	Inter217	Web	Desktop	Video

In [26]:
```
df_clust.tail() # Returns last 5 rows
```

Out[26]:

	Timestamp	InventoryType	Ad - Length	Ad- Width	Ad Size	Ad Type	Platform	Device Type	Fo
23061	2020-9-13-7	Format5	720	300	216000	Inter220	Web	Mobile	
23062	2020-11-2-7	Format5	720	300	216000	Inter224	Web	Desktop	
23063	2020-9-14-22	Format5	720	300	216000	Inter218	App	Mobile	
23064	2020-11-18-2	Format4	120	600	72000	inter230	Video	Mobile	
23065	2020-9-14-0	Format5	720	300	216000	Inter221	App	Mobile	

Number of rows and columns in the dataset

```
In [27]: # checking shape of the data

rows = str(df_clust.shape[0])
columns = str(df_clust.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 23066 rows and 19 columns in the dataset.
```

Datatypes of the different columns in the dataset

```
In [28]: df_clust.info() # Concise summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23066 entries, 0 to 23065
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Timestamp              23066 non-null  object 
 1   InventoryType          23066 non-null  object 
 2   Ad - Length            23066 non-null  int64  
 3   Ad- Width              23066 non-null  int64  
 4   Ad Size                23066 non-null  int64  
 5   Ad Type                23066 non-null  object 
 6   Platform               23066 non-null  object 
 7   Device Type            23066 non-null  object 
 8   Format                 23066 non-null  object 
 9   Available_Impressions  23066 non-null  int64  
 10  Matched_Queries        23066 non-null  int64  
 11  Impressions            23066 non-null  int64  
 12  Clicks                 23066 non-null  int64  
 13  Spend                  23066 non-null  float64
 14  Fee                    23066 non-null  float64
 15  Revenue                23066 non-null  float64
 16  CTR                    18330 non-null  float64
 17  CPM                    18330 non-null  float64
 18  CPC                    18330 non-null  float64
dtypes: float64(6), int64(7), object(6)
memory usage: 3.3+ MB
```

There are 19 columns in the dataset. Out of which 6 have float data type, 7 have integer data type and 6 have object data type.

Finding missing values in the dataset

```
In [29]: df_clust.isna().sum()  # Count NaN values in all columns of dataset
```

```
Out[29]: Timestamp                    0
         InventoryType                0
         Ad - Length                  0
         Ad- Width                    0
         Ad Size                      0
         Ad Type                      0
         Platform                     0
         Device Type                  0
         Format                       0
         Available_Impressions        0
         Matched_Queries              0
         Impressions                  0
         Clicks                       0
         Spend                        0
         Fee                          0
         Revenue                      0
         CTR                       4736
         CPM                       4736
         CPC                       4736
         dtype: int64
```

CTR, CPM and CPC columns have NaN values in 4736 rows.

Treating missing values in the dataset

```
In [30]: # Use of fillna method to treat missing values in STR, CPM and CPC columns

         df_clust['CTR'].fillna(df_clust['Clicks'] / df_clust['Impressions'] * 100,inplace=T
         df_clust['CPM'].fillna(df_clust['Spend'] / df_clust['Impressions'] * 1000,inplace=T
         df_clust['CPC'].fillna(df_clust['Spend'] / df_clust['Clicks'],inplace=True) # Repla
```

```
In [31]: df_clust.isna().sum()  # Count NaN values in all columns of dataset
```

```
Out[31]: Timestamp                0
         InventoryType            0
         Ad - Length              0
         Ad- Width                0
         Ad Size                  0
         Ad Type                  0
         Platform                 0
         Device Type              0
         Format                   0
         Available_Impressions    0
         Matched_Queries          0
         Impressions              0
         Clicks                   0
         Spend                    0
         Fee                      0
         Revenue                  0
         CTR                      0
         CPM                      0
         CPC                      0
         dtype: int64
```

We can see from above list that there are no NaN values in CTR, CPM and CPC columns.

Checking for Duplicates

In [32]:
```
df_clust.duplicated().sum()
```

Out[32]:
```
0
```

There are no duplicate rows in the dataset.

Checking Summary Statistic

In [33]:
```
df_clust.describe(include='all').T
```

Out[33]:

	count	unique	top	freq	mean	std
Timestamp	23066	2018	2020-11-13-22	13	NaN	NaN
InventoryType	23066	7	Format4	7165	NaN	NaN
Ad - Length	23066.0	NaN	NaN	NaN	385.163097	233.651434
Ad- Width	23066.0	NaN	NaN	NaN	337.896037	203.092885
Ad Size	23066.0	NaN	NaN	NaN	96674.468048	61538.329557
Ad Type	23066	14	Inter224	1658	NaN	NaN
Platform	23066	3	Video	9873	NaN	NaN
Device Type	23066	2	Mobile	14806	NaN	NaN
Format	23066	2	Video	11552	NaN	NaN
Available_Impressions	23066.0	NaN	NaN	NaN	2432043.665872	4742887.764666
Matched_Queries	23066.0	NaN	NaN	NaN	1295099.143241	2512969.861258
Impressions	23066.0	NaN	NaN	NaN	1241519.518859	2429399.961091
Clicks	23066.0	NaN	NaN	NaN	10678.518816	17353.409363
Spend	23066.0	NaN	NaN	NaN	2706.625689	4067.927273
Fee	23066.0	NaN	NaN	NaN	0.335123	0.031963
Revenue	23066.0	NaN	NaN	NaN	1924.252331	3105.23841
CTR	23066.0	NaN	NaN	NaN	2.614863	7.853405
CPM	23066.0	NaN	NaN	NaN	8.39673	9.057082
CPC	23066.0	NaN	NaN	NaN	0.336652	0.341231

Observations and Insights:

1. Most Inventory Type is Format4.
2. Most used Platform is Video to watch Digital Ads.
3. Most used Device Type is Mobile to watch Digital Ads.
4. Most used Format is Video to watch Digital Ads (same as Platform).
5. Ad - Length and Ad- Width mean is close to each other. Same is applicable for Matched_Queries and Impressions as well.

Categorial variables in the dataset

In [34]:
```
df_clust['InventoryType'].value_counts().sort_values() # Frequency of each distinct
```

Out[34]:
```
InventoryType
Format7     659
Format2    1789
Format6    1850
Format3    3540
Format1    3814
Format5    4249
Format4    7165
Name: count, dtype: int64
```

There are 7 Inventory Type with Format4 having the maximum count.

In [35]:
```
df_clust['Ad Type'].value_counts().sort_values() # Frequency of each distinct value
```

Out[35]:
```
Ad Type
Inter228    1639
Inter226    1640
Inter225    1643
inter230    1644
Inter220    1644
Inter218    1645
Inter227    1647
Inter229    1648
Inter222    1649
Inter219    1650
Inter221    1650
Inter223    1654
Inter217    1655
Inter224    1658
Name: count, dtype: int64
```

There are 14 Ad Type with Inter224 having the maximum count.

In [36]:
```
df_clust['Platform'].value_counts().sort_values() # Frequency of each distinct valu
```

Out[36]:
```
Platform
App       4942
Web       8251
Video     9873
Name: count, dtype: int64
```

There are 3 Platform with Video having the maximum count.

In [37]:
```
df_clust['Device Type'].value_counts().sort_values() # Frequency of each distinct v
```

Out[37]:
```
Device Type
Desktop      8260
Mobile      14806
Name: count, dtype: int64
```

There are 2 Device Type with Mobile having the maximum count.

In [38]:
```
df_clust['Format'].value_counts().sort_values() # Frequency of each distinct value
```

Out[38]:
```
Format
Display    11514
Video      11552
Name: count, dtype: int64
```

There are 2 Format with Video having the maximum count.

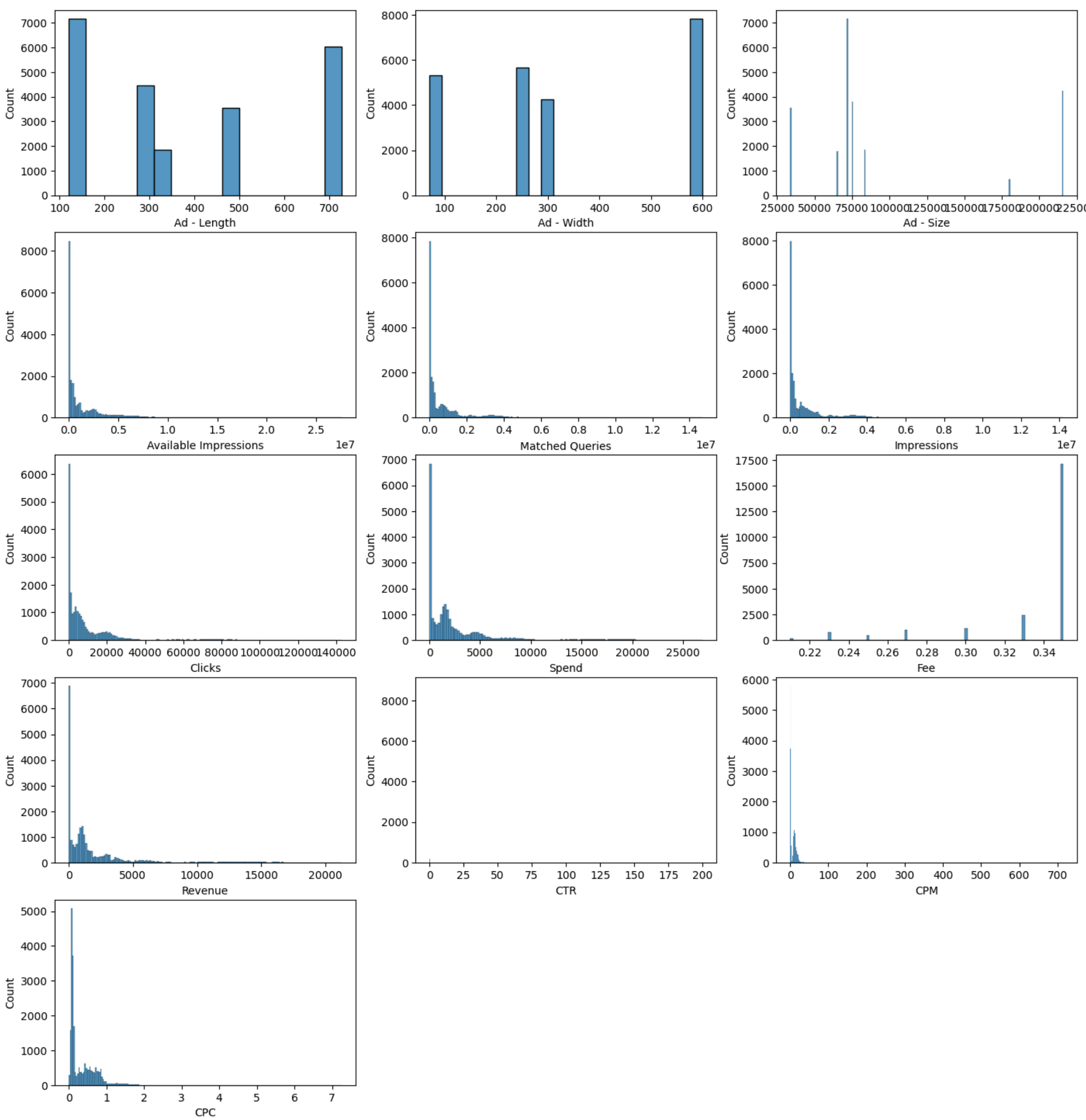
Exploratory Data Analysis (EDA)

Univariate analysis

In [39]:
```
# Hist Plots for Ad - Length, Ad - Width, Ad Size, Available_Impressions, Matched_Q
# Revenue, CTR, CPM, CPC

fig, axes = plt.subplots(5,3, figsize=(17, 18))

sns.histplot(ax=axes[0, 0], data=df_clust, x='Ad - Length')
sns.histplot(ax=axes[0, 1], data=df_clust, x='Ad- Width')
sns.histplot(ax=axes[0, 2], data=df_clust, x='Ad Size')
sns.histplot(ax=axes[1, 0], data=df_clust, x='Available_Impressions')
sns.histplot(ax=axes[1, 1], data=df_clust, x='Matched_Queries')
sns.histplot(ax=axes[1, 2], data=df_clust, x='Impressions')
sns.histplot(ax=axes[2, 0], data=df_clust, x='Clicks')
sns.histplot(ax=axes[2, 1], data=df_clust, x='Spend')
sns.histplot(ax=axes[2, 2], data=df_clust, x='Fee')
sns.histplot(ax=axes[3, 0], data=df_clust, x='Revenue')
sns.histplot(ax=axes[3, 1], data=df_clust, x='CTR')
sns.histplot(ax=axes[3, 2], data=df_clust, x='CPM')
sns.histplot(ax=axes[4, 0], data=df_clust, x='CPC')
axes[4,1].axis("off")
axes[4,2].axis("off")

axes[0,0].set(xlabel='Ad - Length')
axes[0,1].set(xlabel='Ad - Width')
```

```
axes[0,2].set(xlabel='Ad - Size')
axes[1,0].set(xlabel='Available Impressions')
axes[1,1].set(xlabel='Matched Queries')
axes[1,2].set(xlabel='Impressions')
axes[2,0].set(xlabel='Clicks')
axes[2,1].set(xlabel='Spend')
axes[2,2].set(xlabel='Fee')
axes[3,0].set(xlabel='Revenue')
axes[3,1].set(xlabel='CTR')
axes[3,2].set(xlabel='CPM')
axes[4,0].set(xlabel='CPC')

plt.suptitle('Fig 1: Hist Plots: Ad - Length, Ad - Width, Ad - Size, Available Impr

plt.show()
```

Fig 1: Hist Plots: Ad - Length, Ad - Width, Ad - Size, Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CTR, CPM, CPC

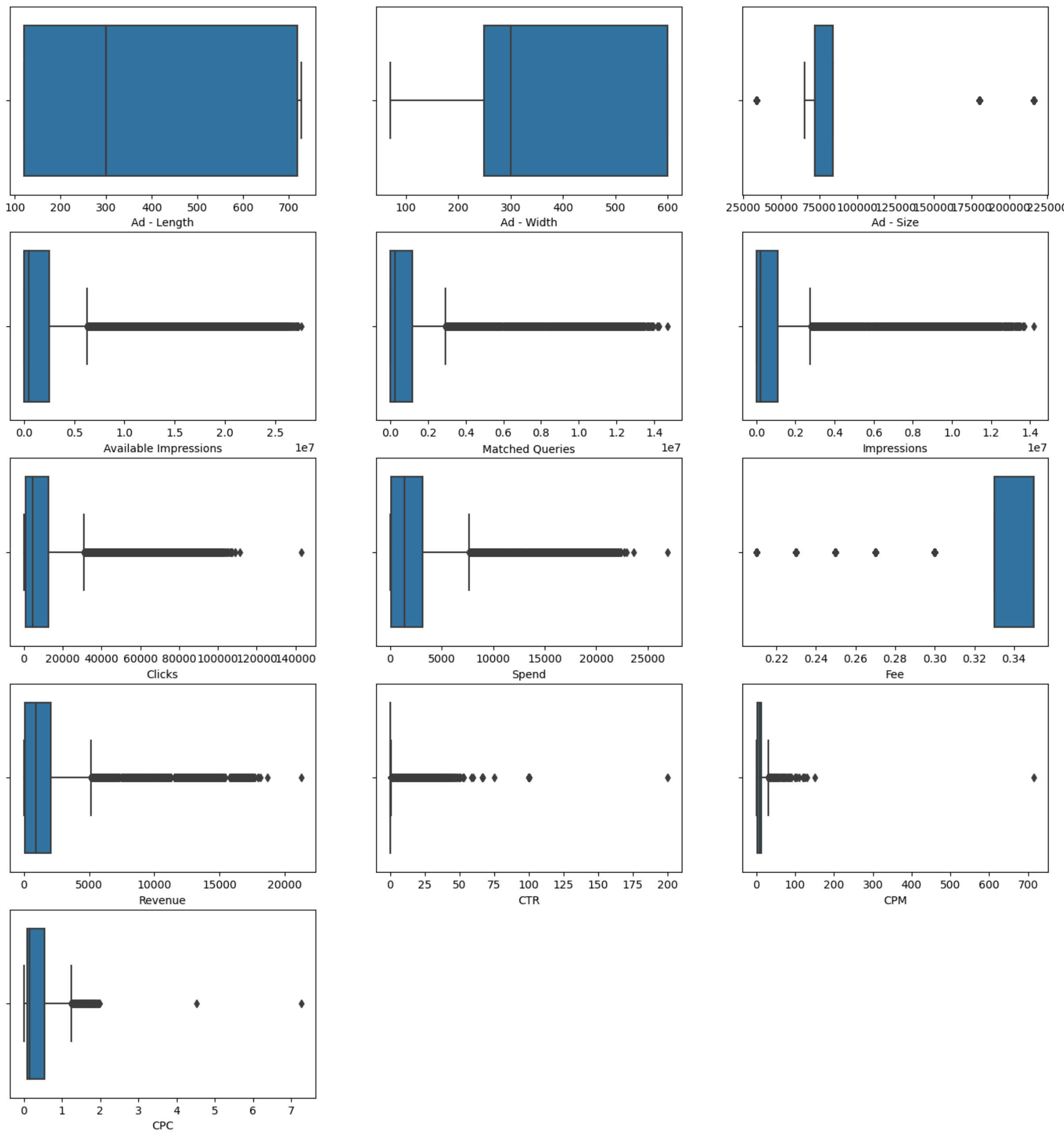


Observations and Insights:

1. No distribution (Ad - Length, Ad - Width, Ad - Size, Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CTR, CPM, CPC) is evenly distributed (symmetric).
2. Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CPM and CPC are Positively Skewed (mean is more than the mode).

In [40]:
```
# Bax Plots for Ad - Length, Ad - Width, Ad Size, Available_Impressions, Matched_Qu
# Revenue, CTR, CPM, CPC

fig, axes = plt.subplots(5,3, figsize=(17, 18))

sns.boxplot(ax=axes[0, 0], data=df_clust, x='Ad - Length')
sns.boxplot(ax=axes[0, 1], data=df_clust, x='Ad- Width')
sns.boxplot(ax=axes[0, 2], data=df_clust, x='Ad Size')
sns.boxplot(ax=axes[1, 0], data=df_clust, x='Available_Impressions')
sns.boxplot(ax=axes[1, 1], data=df_clust, x='Matched_Queries')
sns.boxplot(ax=axes[1, 2], data=df_clust, x='Impressions')
sns.boxplot(ax=axes[2, 0], data=df_clust, x='Clicks')
sns.boxplot(ax=axes[2, 1], data=df_clust, x='Spend')
sns.boxplot(ax=axes[2, 2], data=df_clust, x='Fee')
sns.boxplot(ax=axes[3, 0], data=df_clust, x='Revenue')
sns.boxplot(ax=axes[3, 1], data=df_clust, x='CTR')
sns.boxplot(ax=axes[3, 2], data=df_clust, x='CPM')
sns.boxplot(ax=axes[4, 0], data=df_clust, x='CPC')
axes[4,1].axis("off")
axes[4,2].axis("off")

axes[0,0].set(xlabel='Ad - Length')
axes[0,1].set(xlabel='Ad - Width')
axes[0,2].set(xlabel='Ad - Size')
axes[1,0].set(xlabel='Available Impressions')
axes[1,1].set(xlabel='Matched Queries')
axes[1,2].set(xlabel='Impressions')
axes[2,0].set(xlabel='Clicks')
axes[2,1].set(xlabel='Spend')
axes[2,2].set(xlabel='Fee')
axes[3,0].set(xlabel='Revenue')
axes[3,1].set(xlabel='CTR')
axes[3,2].set(xlabel='CPM')
axes[4,0].set(xlabel='CPC')

plt.suptitle('Fig 2: Box Plots: Ad - Length, Ad - Width, Ad - Size, Available Impre

plt.show()
```

Fig 2: Box Plots: Ad - Length, Ad - Width, Ad - Size, Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CTR, CPM, CPC



Observations and Insights:

1. Ad - Length, Ad - Width do not have outliers.
2. Ad - Size, Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CTR, CPM and CPC are having outliers.

Bivariate Analysis

In [41]:
```
df_clust_num = df_clust.select_dtypes(include='number') # Selecting numerical colum
```

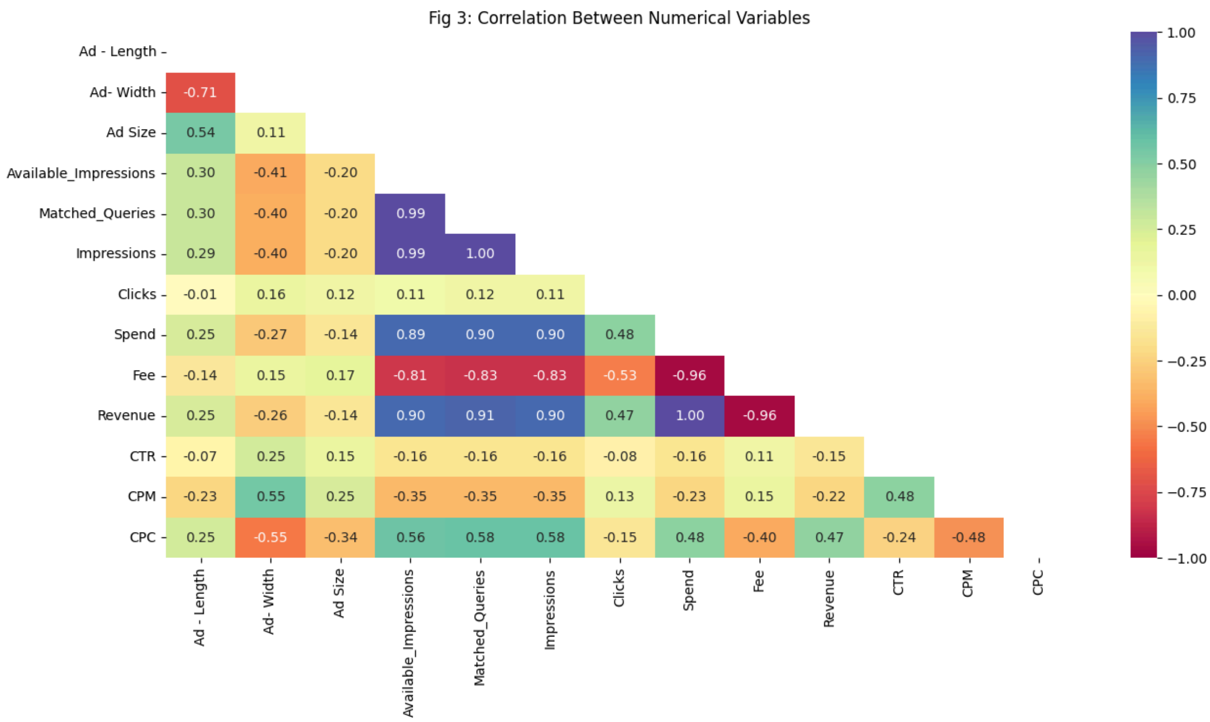
In [42]:
```
df_clust_num.corr()
```

Out[42]:

	Ad - Length	Ad- Width	Ad Size	Available_Impressions	Matched_Quer
Ad - Length	1.000000	-0.705374	0.542391	0.300895	0.2950
Ad- Width	-0.705374	1.000000	0.110318	-0.410493	-0.3977
Ad Size	0.542391	0.110318	1.000000	-0.203853	-0.1970
Available_Impressions	0.300895	-0.410493	-0.203853	1.000000	0.9949
Matched_Queries	0.295007	-0.397779	-0.197089	0.994913	1.0000
Impressions	0.293065	-0.398370	-0.197462	0.994817	0.9999
Clicks	-0.005791	0.157888	0.116659	0.106040	0.1190
Spend	0.248295	-0.274170	-0.144912	0.891942	0.9047
Fee	-0.138311	0.147269	0.169713	-0.814746	-0.8326
Revenue	0.247679	-0.264931	-0.144502	0.896342	0.9082
CTR	-0.069729	0.250371	0.150148	-0.161753	-0.1613
CPM	-0.226355	0.547443	0.246470	-0.354425	-0.3490
CPC	0.250022	-0.550970	-0.335144	0.562747	0.5790

In [43]:

```
# Heatmap to plot correlation between all numerical variables in the dataset

corr = df_clust_num.corr(method='pearson')
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(15, 7))
sns.heatmap(df_clust_num.corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spec
plt.title('Fig 3: Correlation Between Numerical Variables')
plt.show()
```



Observations and Insights:

1. There is strong correlation between Ad Length and Ad Width.
2. There is strong correlation between Ad Length and Ad Size.
3. There is strong correlation between Available Impressions and Matched Queries.
4. There is strong correlation between Available Impressions and Impressions.
5. There is strong correlation between Matched Queries and Impressions.
6. There is strong correlation between Spend and Available Impressions.
7. There is strong correlation between Spend and Matched Queries.
8. There is strong correlation between Spend and Impressions.
9. There is strong correlation between Spend and Fee.
10. There is strong correlation between Spend and Revenue.
11. There is moderate correlation between Spend and Clicks.
12. There is strong correlation between Fee and Available Impressions.
13. There is strong correlation between Fee and Matched Queries.
14. There is strong correlation between Fee and Impressions.
15. There is strong correlation between Fee and Revenue.
16. There is strong correlation between Fee and Clicks.
17. There is strong correlation between Revenue and Available Impressions.
18. There is strong correlation between Revenue and Matched Queries.
19. There is strong correlation between Revenue and Impressions.
20. There is moderate correlation between Revenue and Clicks.
21. There is strong correlation between CPC and Available Impressions.
22. There is strong correlation between CPC and Matched Queries.
23. There is strong correlation between CPC and Impressions.
24. There is moderate correlation between CPC and Spend.

25. There is moderate correlation between CPC and Fee.
26. There is moderate correlation between CPC and Revenue.
27. There is moderate correlation between CPM and Available Impressions.
28. There is moderate correlation between CPM and Matched Queries.
29. There is moderate correlation between CPM and Impressions.
30. There is moderate correlation between CPM and CTR.
31. There is moderate correlation between CPM and CPC.
32. There is moderate correlation between Ad Width and Available Impressions.
33. There is moderate correlation between Ad Width and Matched Queries.
34. There is moderate correlation between Ad Width and Impressions.
35. There is moderate correlation between Ad Width and CPM.
36. There is moderate correlation between Ad Width and CPC.
37. There is moderate correlation between Ad Size and CPC.

Outlier Treatment

In [44]:
```
# User Defined Function (UDF) to treat outliers
def treat_outlier(x):
    # taking 5,25,75 percentile of column
    q5=np.percentile(x,5)
    q25=np.percentile(x,25)
    q75=np.percentile(x,75)
    q95=np.percentile(x,95)
    #calculationg IQR range
    IQR=q75-q25
    #Calculating minimum threshold
    lower_bound=q25-(1.5*IQR)
    upper_bound=q75+(1.5*IQR)
    #Capping outliers
    return x.apply(lambda y: upper_bound if y > upper_bound else y).apply(lambda y:
```

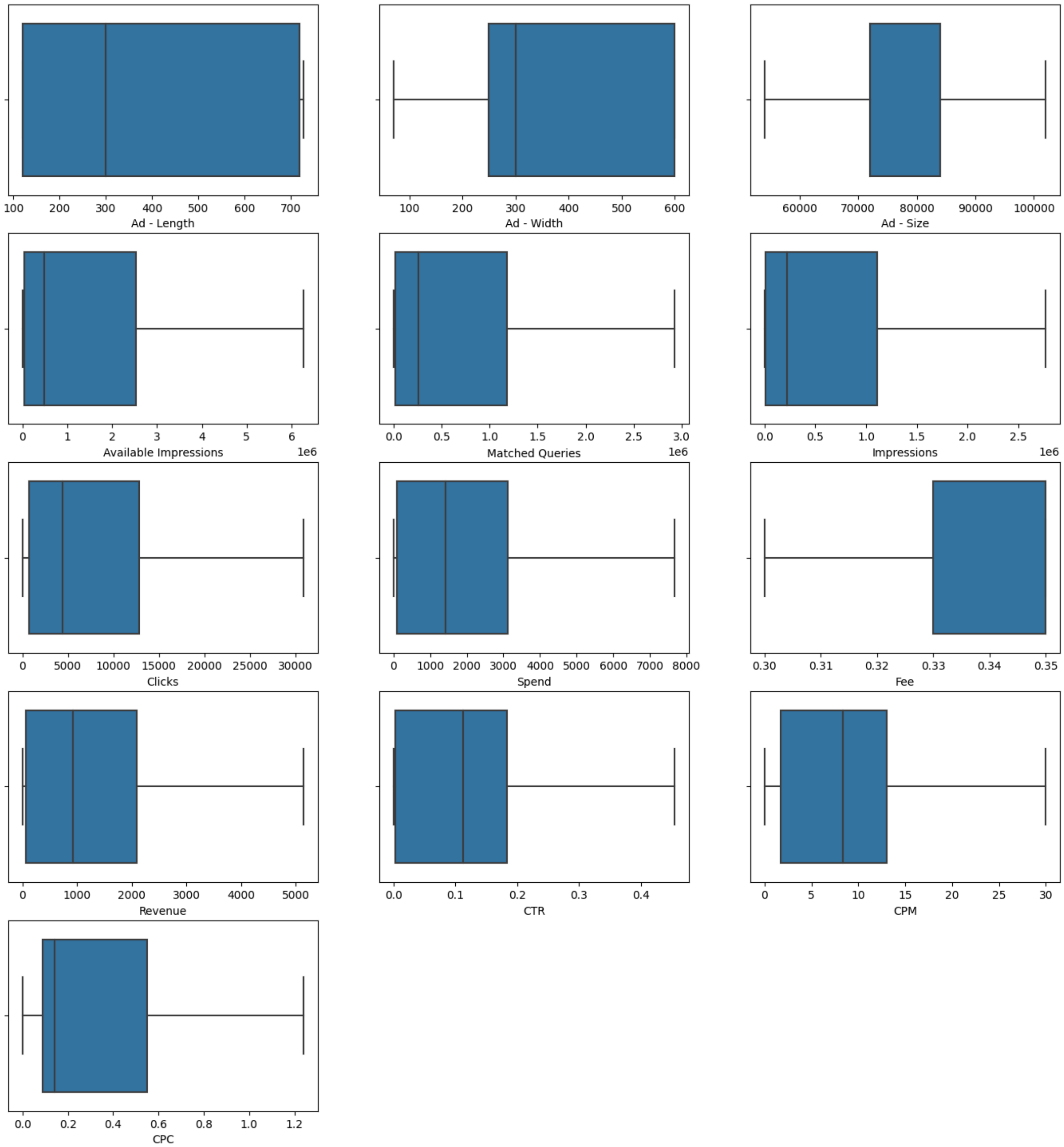
In [45]:
```
no_outlier = ['Ad - Length','Ad - Width'] # Ad - Length and Ad - Width columns do n
outlier_list = [x for x in df_clust_num.columns if x not in no_outlier] # Numerical
```

In [46]:
```
# Using for Loop to iterate over numerical columns and calling treat_outlier UDF to
for i in df_clust_num[outlier_list]:
    df_clust_num[i]=treat_outlier(df_clust_num[i])
```

In [47]:
```
# Bax Plots for Ad - Length, Ad - Width, Ad Size, Available_Impressions, Matched_Qu
# Revenue, CTR, CPM, CPC (after treating the outliers)

fig, axes = plt.subplots(5,3, figsize=(17, 18))

sns.boxplot(ax=axes[0, 0], data=df_clust_num, x='Ad - Length')
sns.boxplot(ax=axes[0, 1], data=df_clust_num, x='Ad- Width')
sns.boxplot(ax=axes[0, 2], data=df_clust_num, x='Ad Size')
sns.boxplot(ax=axes[1, 0], data=df_clust_num, x='Available_Impressions')
sns.boxplot(ax=axes[1, 1], data=df_clust_num, x='Matched_Queries')
sns.boxplot(ax=axes[1, 2], data=df_clust_num, x='Impressions')
sns.boxplot(ax=axes[2, 0], data=df_clust_num, x='Clicks')
```

```
sns.boxplot(ax=axes[2, 1], data=df_clust_num, x='Spend')
sns.boxplot(ax=axes[2, 2], data=df_clust_num, x='Fee')
sns.boxplot(ax=axes[3, 0], data=df_clust_num, x='Revenue')
sns.boxplot(ax=axes[3, 1], data=df_clust_num, x='CTR')
sns.boxplot(ax=axes[3, 2], data=df_clust_num, x='CPM')
sns.boxplot(ax=axes[4, 0], data=df_clust_num, x='CPC')
axes[4,1].axis("off")
axes[4,2].axis("off")

axes[0,0].set(xlabel='Ad - Length')
axes[0,1].set(xlabel='Ad - Width')
axes[0,2].set(xlabel='Ad - Size')
axes[1,0].set(xlabel='Available Impressions')
axes[1,1].set(xlabel='Matched Queries')
axes[1,2].set(xlabel='Impressions')
axes[2,0].set(xlabel='Clicks')
axes[2,1].set(xlabel='Spend')
axes[2,2].set(xlabel='Fee')
axes[3,0].set(xlabel='Revenue')
axes[3,1].set(xlabel='CTR')
axes[3,2].set(xlabel='CPM')
axes[4,0].set(xlabel='CPC')

plt.suptitle('Fig 4: Box Plots: Ad - Length, Ad - Width, Ad - Size, Available Impre

plt.show()
```

Fig 4: Box Plots: Ad - Length, Ad - Width, Ad - Size, Available Impressions, Matched Queries, Impressions, Clicks, Spend, Fee, Revenue, CTR, CPM, CPC



We can observe from above Box Plots that there are no outliers in the numerical columns (to be used for Clustering) after the treatment.

Scaling

In [48]:
```
# scaling the data before clustering
X = StandardScaler()
scaled_df = X.fit_transform(df_clust_num)
```

In [49]:
```
# creating a dataframe of the scaled data
scaled_df_clust = pd.DataFrame(scaled_df, columns=df_clust_num.columns)
```

In [50]:
```python
scaled_df_clust.head() # Returns first 5 rows
```

Out[50]:

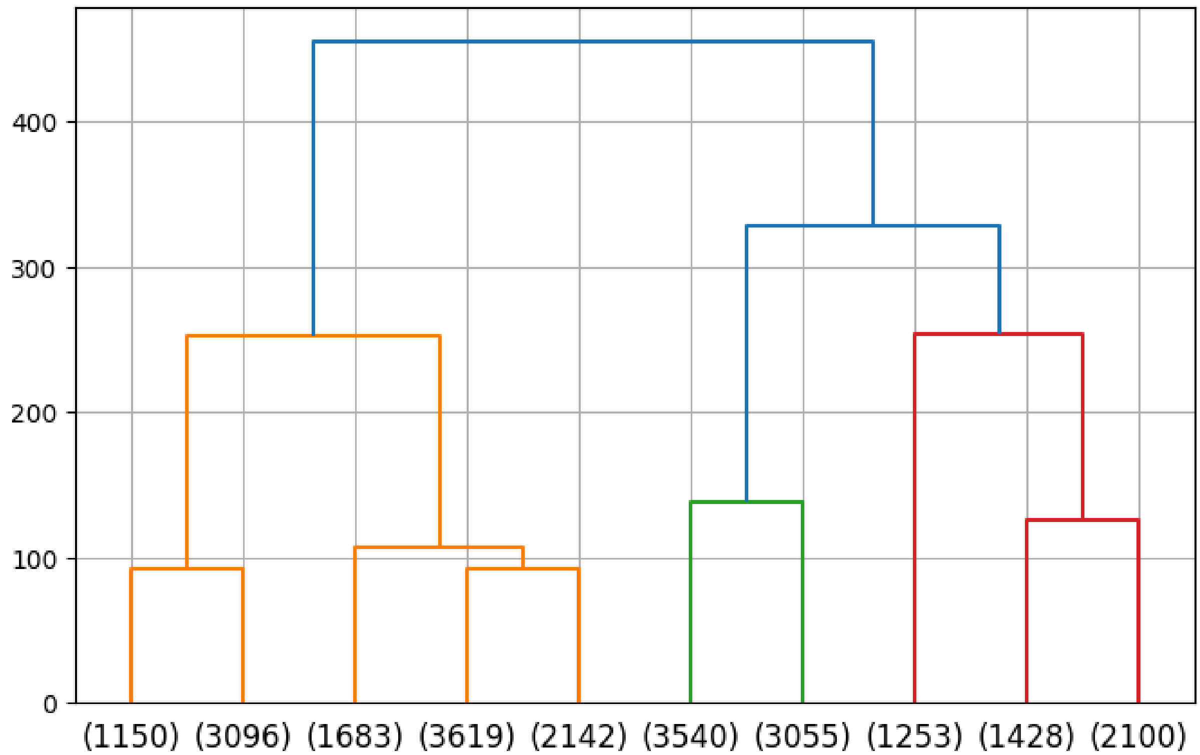
	Ad - Length	Ad- Width	Ad Size	Available_Impressions	Matched_Queries	Impressions	
0	-0.364496	-0.432797	-0.102518	-0.755333	-0.778949	-0.768478	-(
1	-0.364496	-0.432797	-0.102518	-0.755345	-0.778988	-0.768516	-(
2	-0.364496	-0.432797	-0.102518	-0.754900	-0.778919	-0.768445	-(
3	-0.364496	-0.432797	-0.102518	-0.755040	-0.778781	-0.768302	-(
4	-0.364496	-0.432797	-0.102518	-0.755610	-0.779030	-0.768560	-(

Hierarchical Clustering

Dendrogram creation

In [63]:
```python
# Selecting Ward linkage as method and Euclidean distance as metric
linkage_df_clust = linkage(scaled_df_clust, method='ward', metric='euclidean')
```

In [64]:
```python
# Creation of dendrogram

plt.figure(figsize=(8,5))
plt.grid()

dend = dendrogram(linkage_df_clust,
                  truncate_mode='lastp',
                  p = 10,
                  )
```

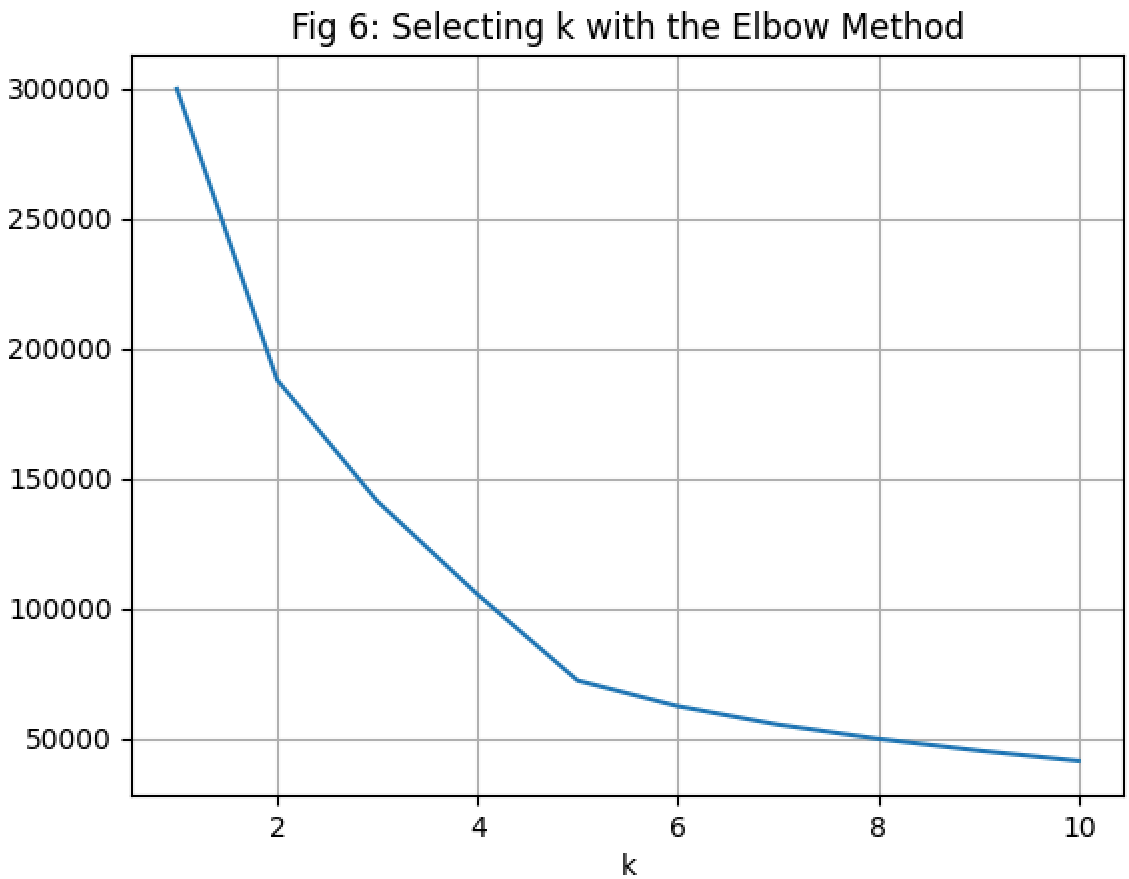


The optimum number of Clusters is 8 as visible in above dendrogram.

K-means Clustering

Elbow Plot creation

In [65]:
```
# Creation of Elbow Plot

wss = []

for i in range(1,11):
    KM = KMeans(n_clusters=i)
    KM.fit(scaled_df)
    wss.append(KM.inertia_)

plt.plot(range(1,11), wss)
plt.xlabel("k")
plt.grid()
plt.title("Fig 6: Selecting k with the Elbow Method")
plt.show()
```



In [66]:
```
# Silhouette Analysis
range_n_clusters=[2,3,4,5,6,7,8,9,10]

for num_clusters in range_n_clusters:

    # initialize K means
    kmeans=KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(scaled_df_clust)
    cluster_labels=kmeans.labels_

    #Silhouette Score
    silhouette_avg = silhouette_score(scaled_df_clust,cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, si
```

```
For n_clusters=2, the silhouette score is 0.40286556236528265
For n_clusters=3, the silhouette score is 0.3454539239336694
For n_clusters=4, the silhouette score is 0.4032921585940855
For n_clusters=5, the silhouette score is 0.48020783078233054
For n_clusters=6, the silhouette score is 0.47613989974053916
For n_clusters=7, the silhouette score is 0.468858103651665
For n_clusters=8, the silhouette score is 0.4323400457749357
For n_clusters=9, the silhouette score is 0.41424479782254425
For n_clusters=10, the silhouette score is 0.4365223863712516
```

The maximum silhouette score is 0.48020783078233054 for 5 Clusters.

Cluster Profiling: K-means Clustering

In [67]:
```python
# creating copy of the original dataset
k_means_clust = df_clust.copy()
```

In [68]:
```python
# adding kmeans cluster labels to the original dataset
k_means = KMeans(n_clusters = 5,random_state=1)
k_means.fit(scaled_df_clust)
labels = k_means.labels_
```

In [69]:
```python
# adding K_Means_Segments column to dataset
k_means_clust["K_Means_Segments"] = labels
```

In [70]:
```python
# finding rows in each Sector in the dataset
k_means_clust.K_Means_Segments.value_counts().sort_index()
```

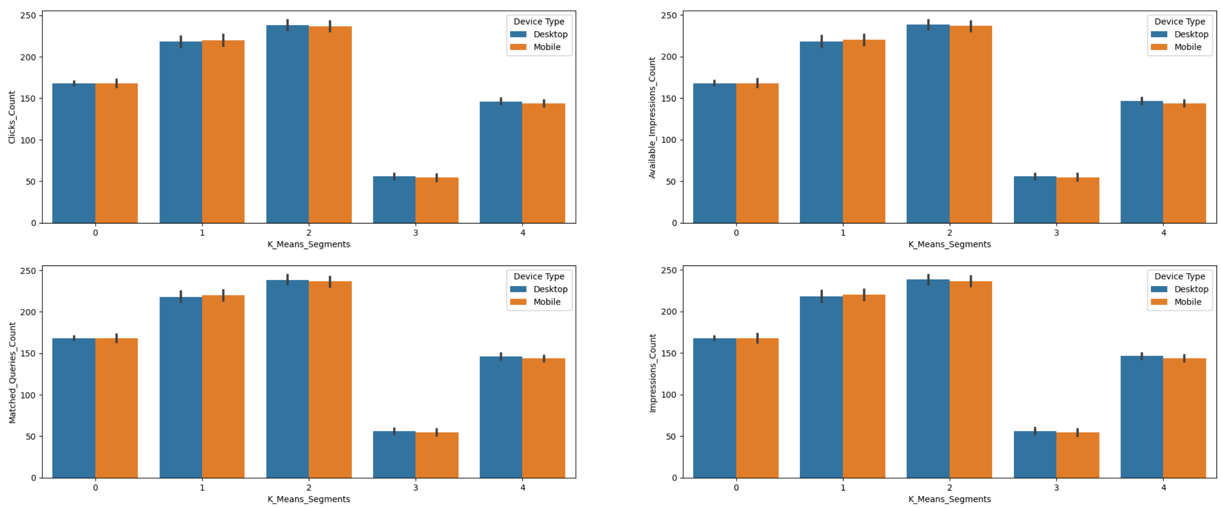
Out[70]:
```
K_Means_Segments
0    4698
1    6140
2    6640
3    1539
4    4049
Name: count, dtype: int64
```

Cluster 3 has highest number of rows followed by Cluster 2, 1, 5 and 3.

In [71]:
```python
km_cluster_profile_count = k_means_clust.groupby(['Ad Type','Device Type','Format',
    Available_Impressions_Count = ('Available_Impressions','count'), Matched_Querie
    Impressions_Count = ('Impressions','count')).sort_values(by=['Ad Type','Device

#km_cluster_profile_count
```

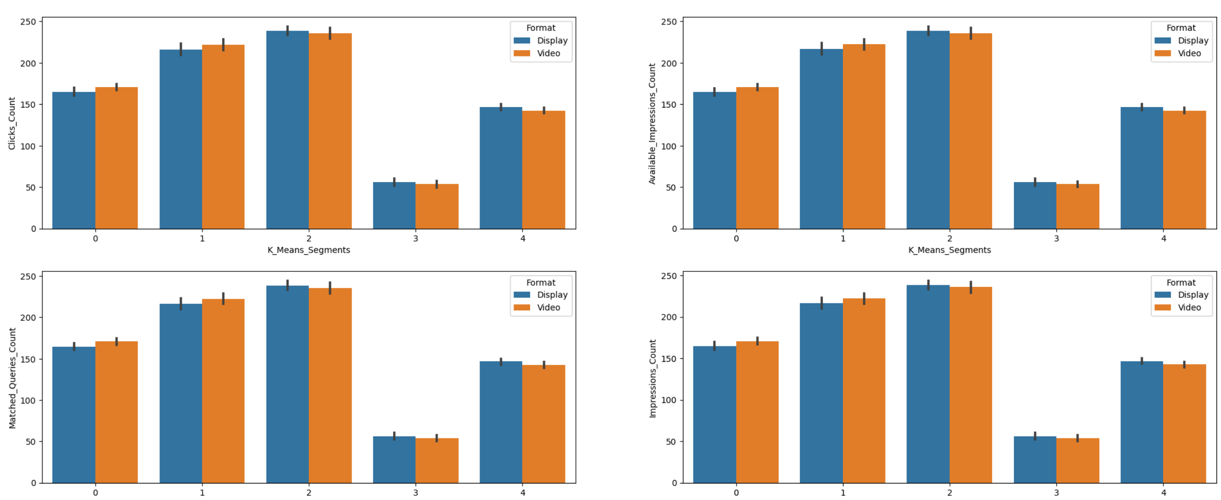
In [88]:
```python
# Bar Plots for Device Type vs Clicks_Count, Available_Impressions_Count, Matched_Q

fig, axes = plt.subplots(2, 2, figsize=(25, 10))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='C
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='A
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='M
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='I

plt.suptitle('Fig 7: Device Type vs Clicks_Count, Available_Impressions_Count, Matc

plt.show()
```

Fig 7: Device Type vs Clicks_Count, Available_Impressions_Count, Matched_Queries_Count and Impressions_Count



In [89]:
```python
# Bar Plots for Format vs Clicks_Count, Available_Impressions_Count, Matched_Querie

fig, axes = plt.subplots(2, 2, figsize=(25, 10))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='C
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='A
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='M
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='I

plt.suptitle('Fig 8: Format vs Clicks_Count, Available_Impressions_Count, Matched_Q

plt.show()
```

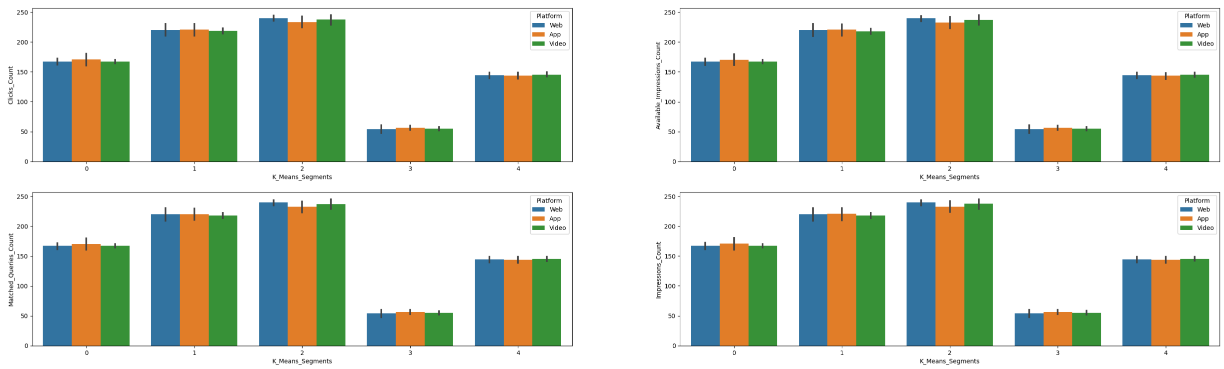
Fig 8: Format vs Clicks_Count, Available_Impressions_Count, Matched_Queries_Count and Impressions_Count



In [90]:
```python
# Bar Plots for Platform vs Clicks_Count, Available_Impressions_Count, Matched_Quer

fig, axes = plt.subplots(2, 2, figsize=(35, 10))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='C
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='A
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_count, x='K_Means_Segments',y='M
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_count, x='K_Means_Segments',y='I
```

```python
plt.suptitle('Fig 9: Platform vs Clicks_Count, Available_Impressions_Count, Matched

plt.show()
```

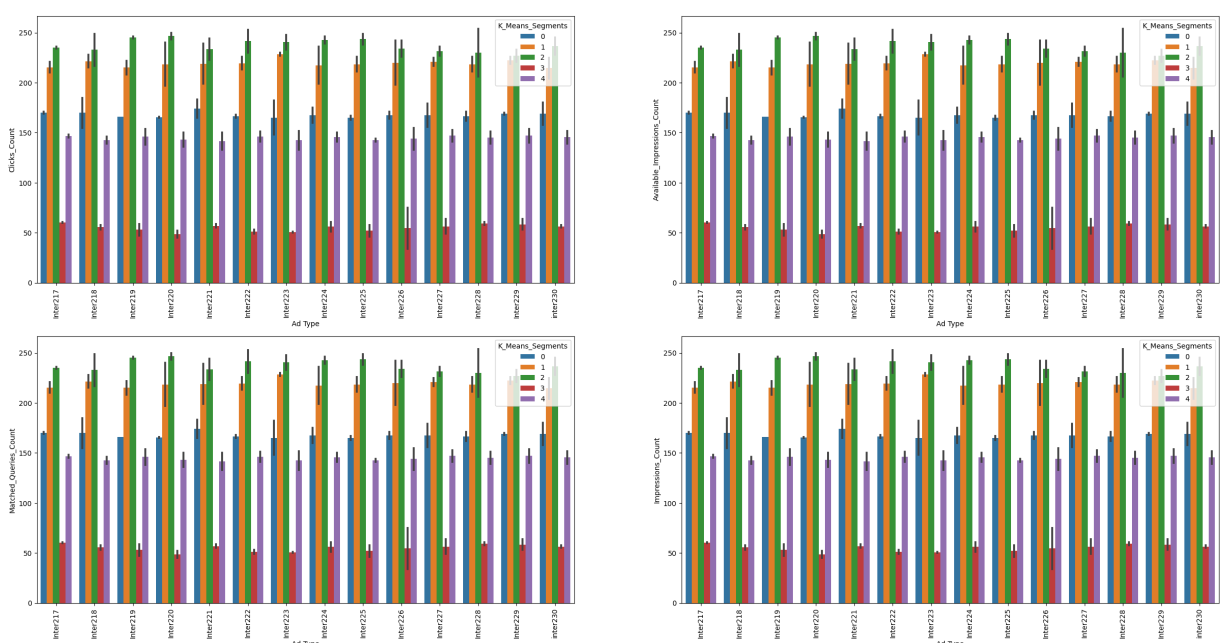
Fig 9: Platform vs Clicks_Count, Available_Impressions_Count, Matched_Queries_Count and Impressions_Count



In [91]:

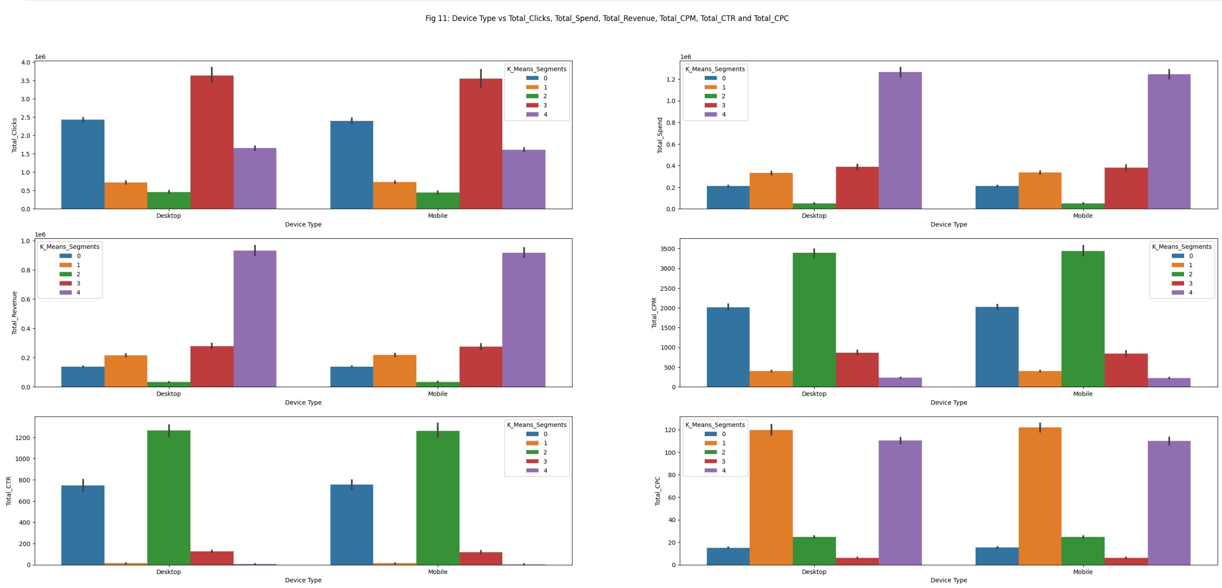
```python
# Bar Plots for Ad Type vs Clicks_Count, Available_Impressions_Count, Matched_Queri

fig, axes = plt.subplots(2, 2, figsize=(30, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_count, x='Ad Type',y='Clicks_Cou
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_count, x='Ad Type',y='Available_
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_count, x='Ad Type',y='Matched_Qu
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_count, x='Ad Type',y='Impression

axes[0,0].tick_params(axis='x', rotation=90)
axes[0,1].tick_params(axis='x', rotation=90)
axes[1,0].tick_params(axis='x', rotation=90)
axes[1,1].tick_params(axis='x', rotation=90)

plt.suptitle('Fig 10: Ad Type vs Clicks_Count, Available_Impressions_Count, Matched

plt.show()
```

Fig 10: Ad Type vs Clicks_Count, Available_Impressions_Count, Matched_Queries_Count and Impressions_Count



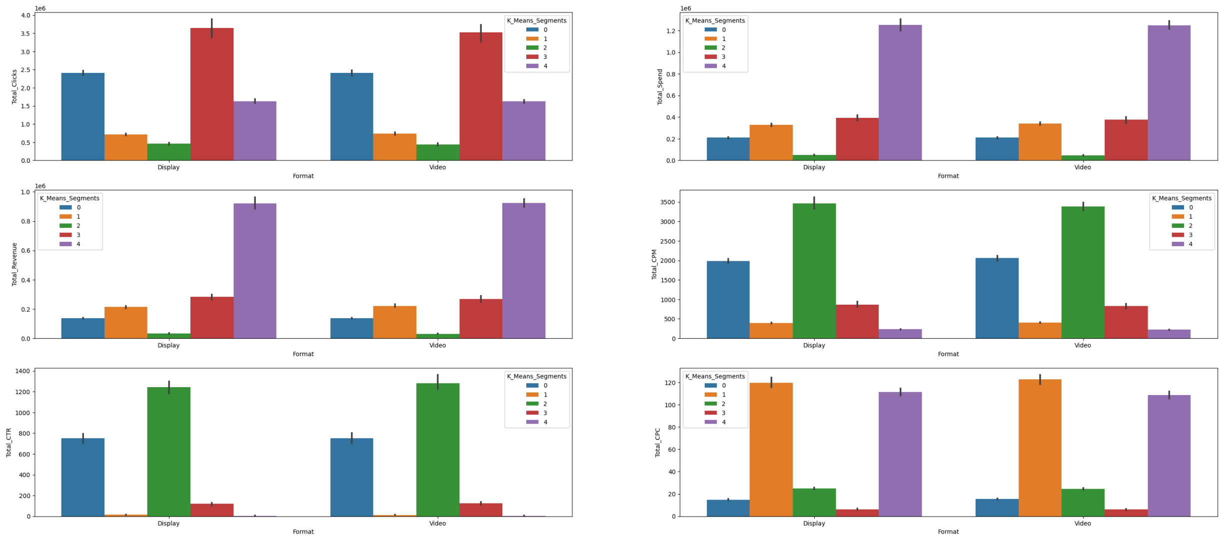
In [92]:
```
km_cluster_profile_sum = k_means_clust.groupby(['Device Type','Format','Platform','
     Total_Spend = ('Spend','sum'), Total_Revenue = ('Revenue','sum'),Total_CPM = (
     Total_CTR = ('CTR','sum'), Total_CPC = ('CPC','sum')).sort_values(by=['Device

#km_cluster_profile_sum
```

In [93]:
```
# Bar Plots for Device Type vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM,

fig, axes = plt.subplots(3, 2, figsize=(35, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_sum, x='Device Type',y='Total_Cl
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_sum, x='Device Type',y='Total_Sp
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_sum, x='Device Type',y='Total_Re
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_sum, x='Device Type',y='Total_CP
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_sum, x='Device Type',y='Total_CT
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_sum, x='Device Type',y='Total_CP

plt.suptitle('Fig 11: Device Type vs Total_Clicks, Total_Spend, Total_Revenue, Tota

plt.show()
```



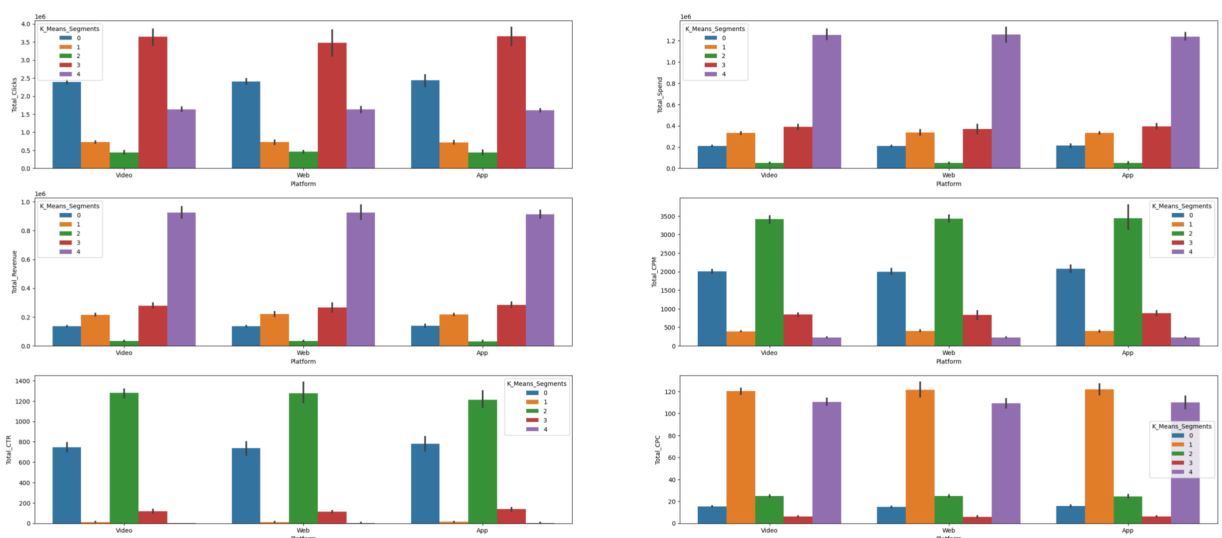
In [94]:
```
# Bar Plots for Format vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM, Tota

fig, axes = plt.subplots(3, 2, figsize=(35, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_sum, x='Format',y='Total_Clicks'
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_sum, x='Format',y='Total_Spend',
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_sum, x='Format',y='Total_Revenue
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_sum, x='Format',y='Total_CPM', h
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_sum, x='Format',y='Total_CTR', h
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_sum, x='Format',y='Total_CPC', h

plt.suptitle('Fig 12: Format vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM

plt.show()
```

Fig 12: Format vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM, Total_CTR and Total_CPC



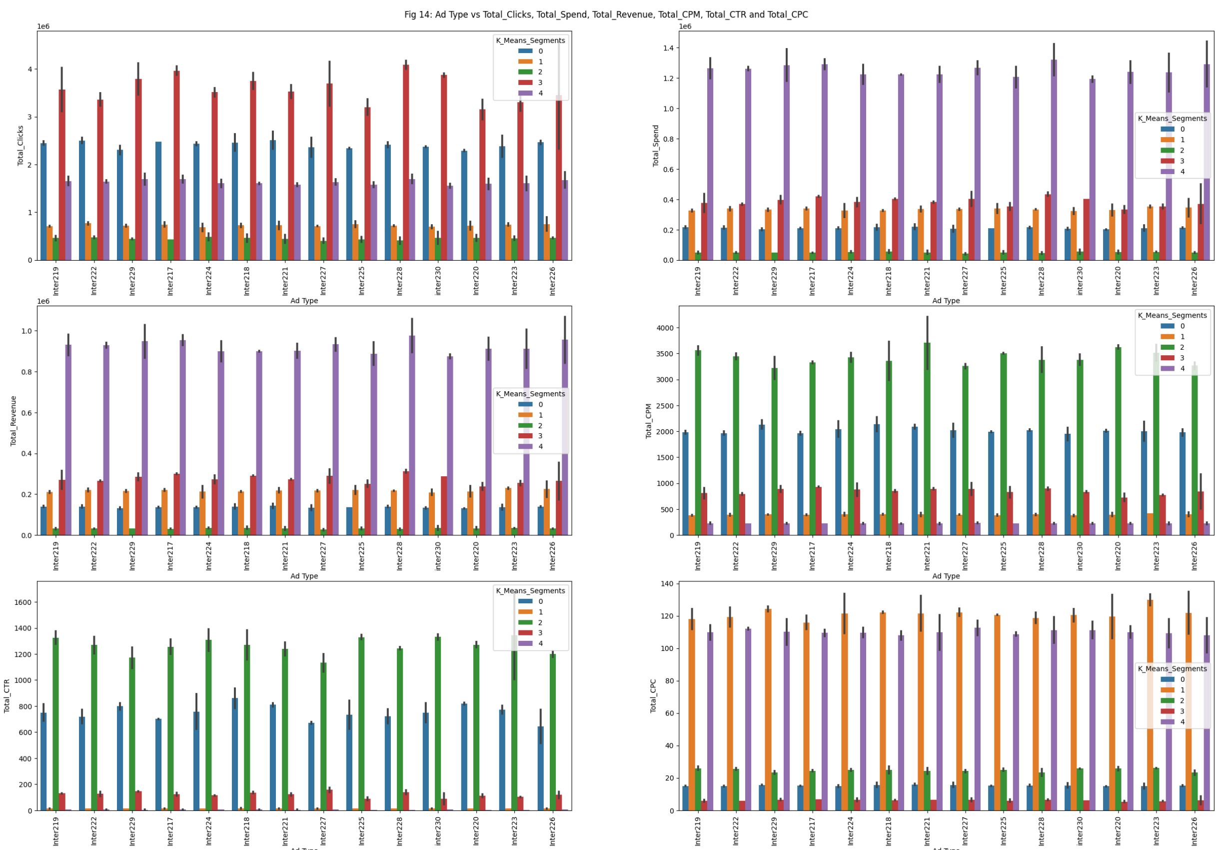
In [95]:
```
# Bar Plots for Platform vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM, To

fig, axes = plt.subplots(3, 2, figsize=(35, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_sum, x='Platform',y='Total_Click
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_sum, x='Platform',y='Total_Spend
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_sum, x='Platform',y='Total_Reven
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_sum, x='Platform',y='Total_CPM',
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_sum, x='Platform',y='Total_CTR',
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_sum, x='Platform',y='Total_CPC',

plt.suptitle('Fig 13: Platform vs Total_Clicks, Total_Spend, Total_Revenue, Total_C

plt.show()
```

Fig 13: Platform vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM, Total_CTR and Total_CPC



In [96]:
```
# Bar Plots for Ad Type vs Total_Clicks, Total_Spend, Total_Revenue, Total_CPM, Tot

fig, axes = plt.subplots(3, 2, figsize=(30, 20))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_sum, x='Ad Type',y='Total_Clicks
```

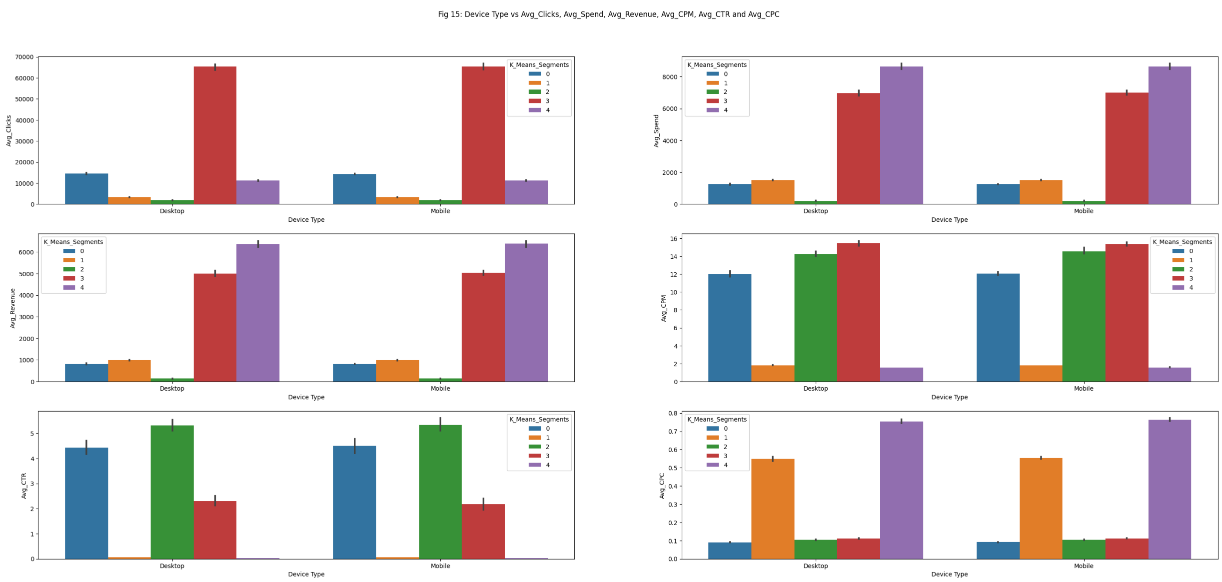
```python
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_sum, x='Ad Type',y='Total_Spend'
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_sum, x='Ad Type',y='Total_Revenu
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_sum, x='Ad Type',y='Total_CPM',
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_sum, x='Ad Type',y='Total_CTR',
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_sum, x='Ad Type',y='Total_CPC',

axes[0,0].tick_params(axis='x', rotation=90)
axes[0,1].tick_params(axis='x', rotation=90)
axes[1,0].tick_params(axis='x', rotation=90)
axes[1,1].tick_params(axis='x', rotation=90)
axes[2,0].tick_params(axis='x', rotation=90)
axes[2,1].tick_params(axis='x', rotation=90)

plt.suptitle('Fig 14: Ad Type vs Total_Clicks, Total_Spend, Total_Revenue, Total_CP

plt.show()
```



In [97]:
```python
km_cluster_profile_mean = k_means_clust.groupby(['Device Type','Format','Platform',
     Avg_Spend = ('Spend','mean'), Avg_Revenue = ('Revenue','mean'),Avg_CPM = ('CPM
     Avg_CTR = ('CTR','mean'), Avg_CPC = ('CPC','mean')).sort_values(by=['Device Ty

#km_cluster_profile_mean
```

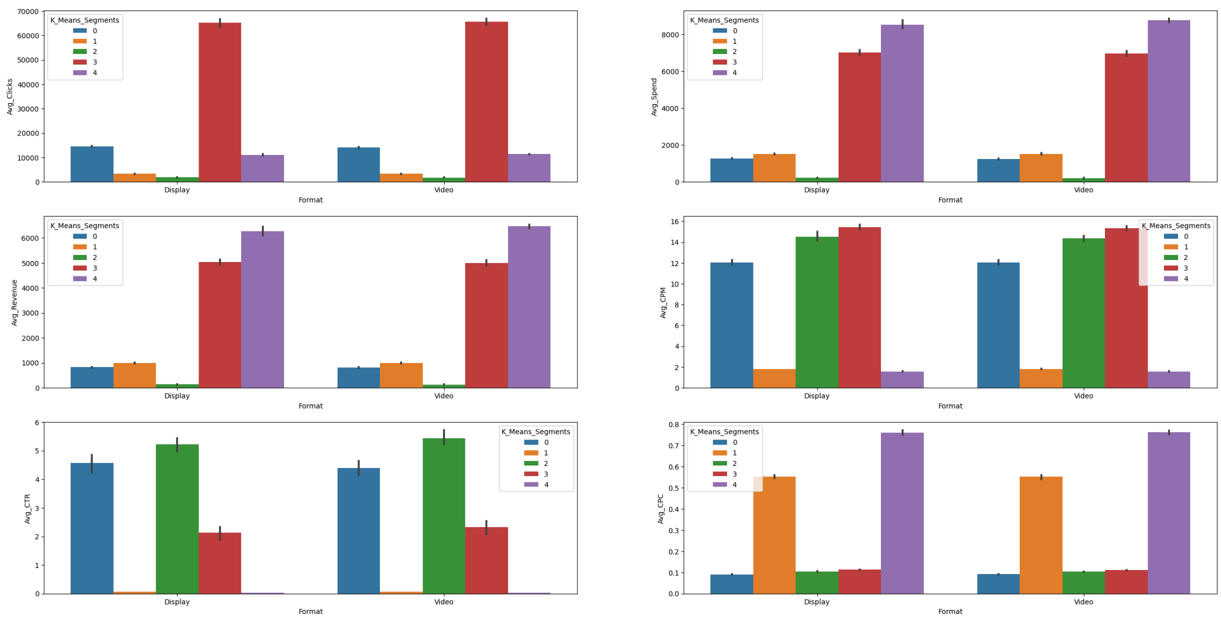
In [98]:
```python
# Bar Plots for Device Type vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR

fig, axes = plt.subplots(3, 2, figsize=(35, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_mean, x='Device Type',y='Avg_Cli
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_mean, x='Device Type',y='Avg_Spe
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_mean, x='Device Type',y='Avg_Rev
```

```
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_mean, x='Device Type',y='Avg_CPM
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_mean, x='Device Type',y='Avg_CTR
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_mean, x='Device Type',y='Avg_CPC

plt.suptitle('Fig 15: Device Type vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, A

plt.show()
```



In [99]:
```
# Bar Plots for Format vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR and

fig, axes = plt.subplots(3, 2, figsize=(30, 15))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_mean, x='Format',y='Avg_Clicks',
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_mean, x='Format',y='Avg_Spend',
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_mean, x='Format',y='Avg_Revenue'
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_mean, x='Format',y='Avg_CPM', hu
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_mean, x='Format',y='Avg_CTR', hu
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_mean, x='Format',y='Avg_CPC', hu

plt.suptitle('Fig 16: Format vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CT

plt.show()
```

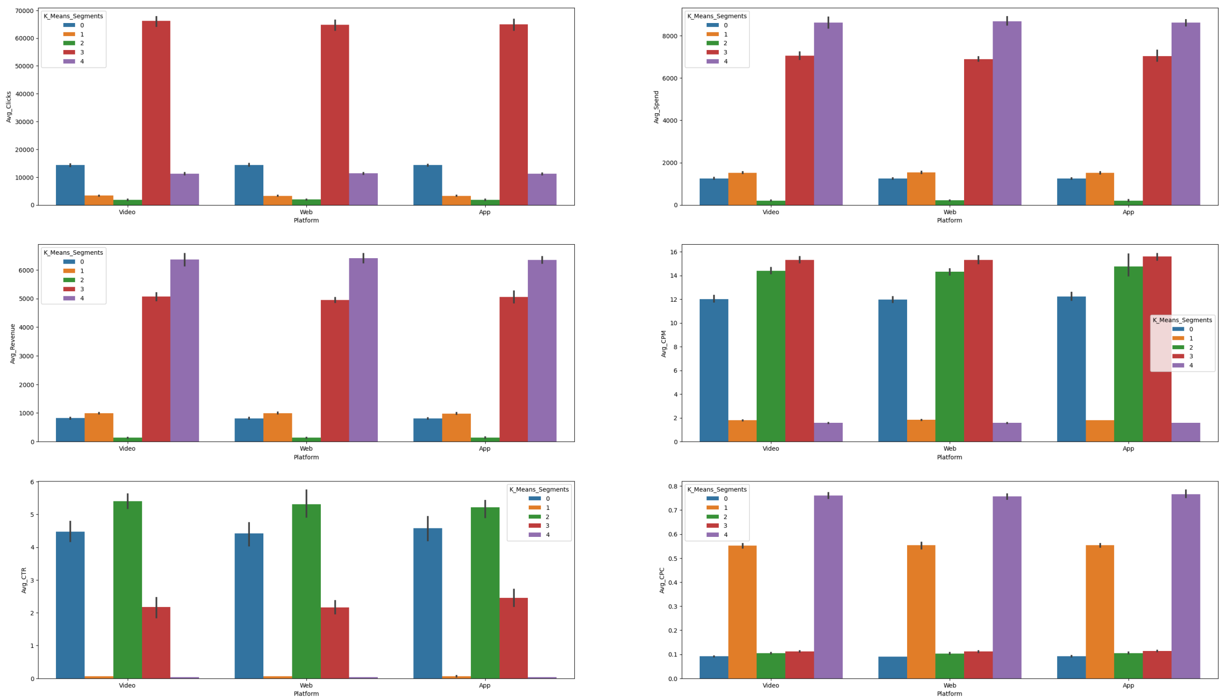
Fig 16: Format vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR and Avg_CPC



In [100…

```
# Bar Plots for Platform vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR an

fig, axes = plt.subplots(3, 2, figsize=(35, 20))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_mean, x='Platform',y='Avg_Clicks
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_mean, x='Platform',y='Avg_Spend'
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_mean, x='Platform',y='Avg_Revenu
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_mean, x='Platform',y='Avg_CPM',
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_mean, x='Platform',y='Avg_CTR',
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_mean, x='Platform',y='Avg_CPC',

plt.suptitle('Fig 17: Platform vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_

plt.show()
```

Fig 17: Platform vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR and Avg_CPC



In [101…

```
# Bar Plots for Ad Type vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR and

fig, axes = plt.subplots(3, 2, figsize=(30, 20))

sns.barplot(ax=axes[0, 0], data=km_cluster_profile_mean, x='Ad Type',y='Avg_Clicks'
sns.barplot(ax=axes[0, 1], data=km_cluster_profile_mean, x='Ad Type',y='Avg_Spend',
sns.barplot(ax=axes[1, 0], data=km_cluster_profile_mean, x='Ad Type',y='Avg_Revenue
sns.barplot(ax=axes[1, 1], data=km_cluster_profile_mean, x='Ad Type',y='Avg_CPM', h
sns.barplot(ax=axes[2, 0], data=km_cluster_profile_mean, x='Ad Type',y='Avg_CTR', h
sns.barplot(ax=axes[2, 1], data=km_cluster_profile_mean, x='Ad Type',y='Avg_CPC', h

axes[0,0].tick_params(axis='x', rotation=90)
axes[1,1].tick_params(axis='x', rotation=90)
axes[1,0].tick_params(axis='x', rotation=90)
axes[1,1].tick_params(axis='x', rotation=90)
axes[2,0].tick_params(axis='x', rotation=90)
axes[2,1].tick_params(axis='x', rotation=90)

plt.suptitle('Fig 18: Ad Type vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_C

plt.show()
```

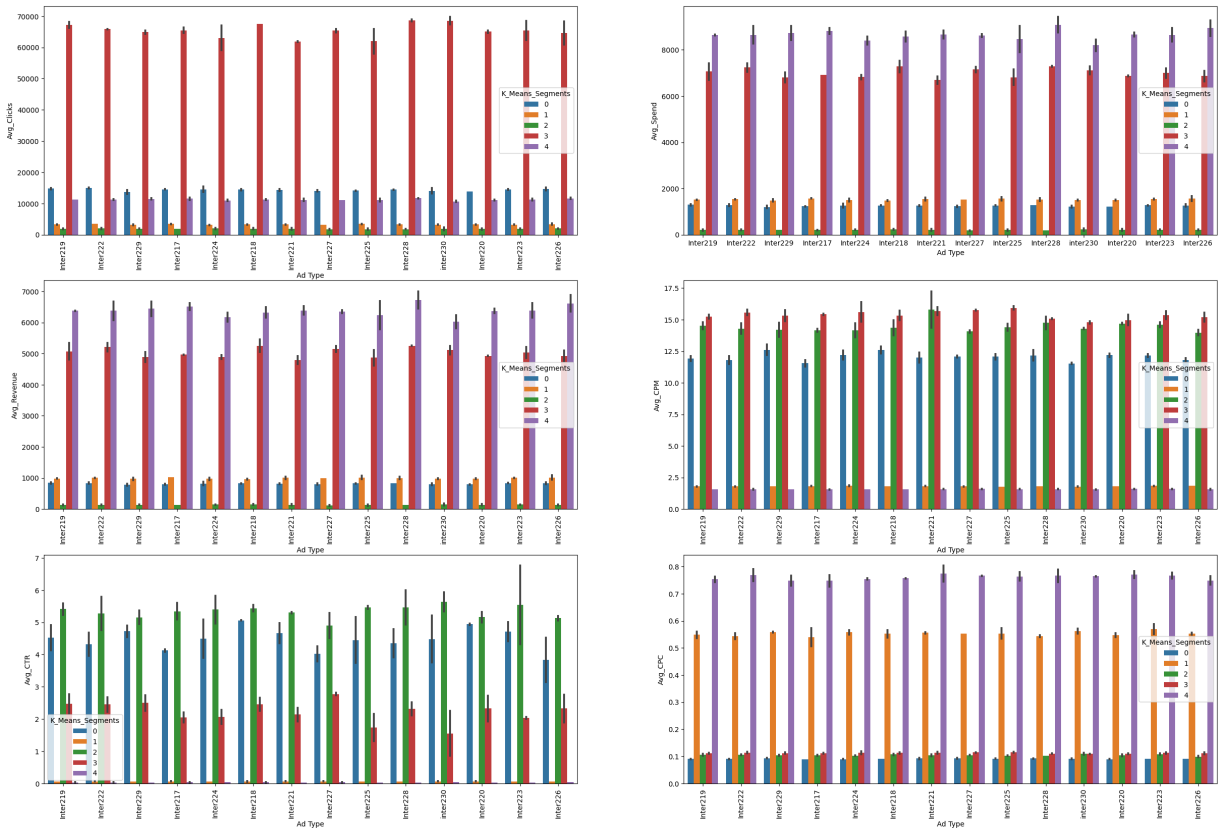


Fig 18: Ad Type vs Avg_Clicks, Avg_Spend, Avg_Revenue, Avg_CPM, Avg_CTR and Avg_CPC

Actionable Insights & Recommendations

Actionable Insights:

1. Most number of Clicks, Available Impressions, Matched Queries and Impressions are falling in Cluster 3rd followed by 2nd, 1st, 5th and 4th for Device Type, Format, Platform and Ad Type.
2. Highest number of Total Clicks is falling in Cluster 4th followed by 1st, 5th, 2nd and 3rd for Device Type, Format, Platform and Ad Type.
3. Total Spending is highest in Cluster 5th followed by 4th, 2nd, 1st and 3rd for Device Type, Format, Platform and Ad Type.
4. Total Revenue is highest in Cluster 5th followed by 4th, 2nd, 1st and 3rd for Device Type, Format, Platform and Ad Type.
5. Total CPM and CTR is highest in Cluster 3rd followed by 1st, 4th, 2nd and 5th Cluster for Device Type, Format, Platform and Ad Type.
6. Total CPC is highest in Cluster 2nd followed by 5th, 3rd, 1st and 4th Cluster for Device Type, Format, Platform and Ad Type.
7. Highest average of Total Clicks is falling in Cluster 4th followed by 1st, 5th, 2nd and 3rd for Device Type, Format, Platform and Ad Type.
8. Average Spending is highest in Cluster 5th followed by 4th, 2nd, 1st and 3rd for Device Type, Format, Platform and Ad Type.

9. Average Revenue is highest in Cluster 5th followed by 4th, 2nd, 1st and 3rd for Device Type, Format, Platform and Ad Type.
10. Average CPM is highest in Cluster 4th followed by 3rd, 1st, 2nd and 5th Cluster for Device Type, Format, Platform and Ad Type.
11. Average CTR is highest in Cluster 3rd followed by 1st, 4th, 2nd and 5th Cluster for Device Type, Format, Platform and Ad Type.
12. Average CPC is highest in Cluster 5th followed by 2nd, 4th, 3rd and 1st Cluster for Device Type, Format, Platform and Ad Type.

Recommendations:

1. Sector 4th, 5th and 1st can be targeted for similar Advertisements (Ad Types) and frequency which are shown in Sector 3rd and 2nd to increase number of Clicks, Available Impressions, Matched Queries and Impressions.
2. Sector 4th, 5th and 1st can be targeted for matching Advertisements (Ad Types) as similar to Sector 3rd and 2nd to increase number of searches for an Advertisement.
3. Sector 3rd, 2nd and 5th can be targeted for similar Advertisements (Ad Types) which are shown in Sector 4th and 1st to increase total number of Clicks.
4. Sector 3rd, 1st and 2nd can be targeted for similar Advertisements (Ad Types) which are shown in Sector 5th and 4th to increase total Spending and Revenue.
5. Sector 5th, 2nd and 4th can be targeted to remove Advertisements (Ad Types) which are shown in Sector 3rd and 1st and having higher CPM (cost per 1000 impressions) to decrease total CPM (cost per 1000 impressions).
6. Sector 5th, 2nd and 4th can be targeted for similar Advertisements (Ad Types) which are shown in Sector 3rd and 1st to increase total CTR (click through rate).
7. Sector 4th, 1st and 3rd can be targeted to remove Advertisements (Ad Types) which are shown in Sector 2nd and 5th and having higher CPC (cost-per-click) to decrease total CPC (cost-per-click).
8. Sector 3rd, 1st and 2nd can be targeted for similar Advertisements (Ad Types) which are shown in Sector 5th and 4th to increase average Spending and Revenue.
9. Sector 5th, 2nd and 1st can be targeted to remove Advertisements (Ad Types) which are shown in Sector 4th and 3rd and having higher CPM (cost per 1000 impressions) to decrease average CPM (cost per 1000 impressions).
10. Sector 5th, 2nd and 4th can be targeted for similar Advertisements (Ad Types) which are shown in Sector 3rd and 1st to increase average CTR (click through rate).
11. Sector 1st, 3rd and 4th can be targeted to remove Advertisements (Ad Types) which are shown in Sector 5th and 2nd and having higher CPC (cost-per-click) to decrease average CPC (cost-per-click).

Above recommendations are applicable for all Device Type, Format and Platform as well.

```
In [ ]:
```