Problem Description : Project analyzes flight advertisement data to improve ad performance. It focuses on bidding strategies, CPC, impressions, and clicks to help advertisers optimize marketing and reduce costs.

Problem Statement : Airlines face high competition, unclear returns, and budget inefficiencies in ads. This project analyzes ad performance to provide data-driven solutions for better engagement and cost optimization.

Desire outcome : The project will identify top-performing ads, optimize bidding strategies, improve ROI, and predict future trends to enhance ad effectiveness. Findings will be presented through visual reports for better decision-making.

importing Libraries.

In [2]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

load the data

In [3]:
```
File1 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [4]:
```
File1.head()
```

Out[4]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank
0	FlyDealFarePanelM_US_FPCMP2	XNA	MAA	XNA	MAA	1.00	2.00
1	FlyDealFarePanelM_US_FPCMP2	WAS	DEL	WAS	DEL	1.00	1.22
2	FlyDealFarePanelM_US_FPCMP2	WAS	AMD	WAS	AMD	1.00	1.00
3	FlyDealFarePanelM_US_FPCMP2	VPS	BOM	VPS	BOM	1.00	2.00
4	FlyDealFarePanelM_US_FPCMP2	VGA	SFO	VGA	SFO	0.45	5.00

In [5]:
```
File2 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [6]:
```
File2.head()
```

Out[6]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Aver R
0	FlyDealFarePanelD_US_FPCMP2	SFO	BLR	SFO	BLR	0.95	
1	FlyDealFarePanelM_US_FPCMP2	DEL	BOS	DEL	BOS	0.45	
2	FlyDealFarePanelDOW_US_FPCMP2	CCJ	ORD	CCJ	ORD	0.30	
3	FlyDealFarePanelMOW_US_FPCMP2	BDQ	JFK	BDQ	JFK	0.25	
4	FlyDealFarePanelM_US_FPCMP2	TUL	BOM	TUL	BOM	1.00	

In [7]:
```
File3 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [8]:
```
File3.head()
```

Out[8]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank
0	FlyDealFarePanelD_US_FPCMP2	LAX	BOM	LAX	BOM	0.95	5.57
1	FlyDealFarePanelD_US_FPCMP2	PDK	BOM	PDK	BOM	0.95	4.00
2	FlyDealFarePanelM_US_FPCMP2	JAX	AMD	JAX	AMD	1.00	2.00
3	FlyDealFarePanelM_US_FPCMP2	PHL	HYD	PHL	HYD	1.00	2.00
4	FlyDealFarePanelD_US_FPCMP2	IAD	BDQ	IAD	BDQ	0.95	4.00

In [9]:
```
File4 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [10]:
```
File4.head()
```

Out[10]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Aver R
0	FlyDealFarePanelM_US_FPCMP2	SEA	ATQ	SEA	ATQ	1.00	
1	FlyDealFarePanelMOW_US_FPCMP2	AMD	CLT	AMD	CLT	0.25	
2	FlyDealFarePanelD_US_FPCMP2	ORD	TRV	ORD	TRV	1.17	
3	FlyDealFarePanelD_US_FPCMP2	ORD	DEL	ORD	DEL	0.95	
4	FlyDealFarePanelD_US_FPCMP2	DFW	HYD	DFW	HYD	0.95	

In [11]:
```
File5 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [12]:
```
File5.head()
```

Out[12]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Aver R
0	FlyDealFarePanelMOW_US_FPCMP2	BOS	BOM	BOS	BOM	1.21	
1	FlyDealFarePanelM_US_FPCMP2	NYC	GAU	NYC	GAU	1.00	
2	FlyDealFarePanelM_US_FPCMP2	DEL	USA	DEL	USA	0.45	
3	FlyDealFarePanelD_US_FPCMP2	BOM	LAS	BOM	LAS	0.55	
4	FlyDealFarePanelM_US_FPCMP2	ATL	MAA	ATL	MAA	1.00	

In [13]:
```
File6 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [14]:
```
File6.head()
```

Out[14]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank
0	FlyDealFarePanelM_US_FPCMP2	WAS	MAA	WAS	MAA	1.0	1.0
1	FlyDealFarePanelM_US_FPCMP2	WAS	ATQ	WAS	ATQ	1.0	1.0
2	FlyDealFarePanelM_US_FPCMP2	WAS	AMD	WAS	AMD	1.0	2.0
3	FlyDealFarePanelM_US_FPCMP2	USA	HYD	USA	HYD	1.0	2.0
4	FlyDealFarePanelM_US_FPCMP2	TYS	BOM	TYS	BOM	1.0	2.0

In [15]:
```
File7 = pd.read_csv("C:/Users/sande/Downloads/Last Days Report/Last Days Repor
```

In [16]:
```
File7.head()
```

Out[16]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank
0	FlyDealFarePanelM_US_FPCMP2	XNA	MAA	XNA	MAA	1.00	3.00
1	FlyDealFarePanelM_US_FPCMP2	WAS	MAA	WAS	MAA	1.00	1.62
2	FlyDealFarePanelM_US_FPCMP2	WAS	DEL	WAS	DEL	1.00	1.13
3	FlyDealFarePanelM_US_FPCMP2	VGA	PHL	VGA	PHL	0.45	3.00
4	FlyDealFarePanelM_US_FPCMP2	VCV	HYD	VCV	HYD	1.00	1.00

In [ ]:

In [17]:
```
print(File1.shape)
print(File2.shape)
print(File3.shape)
print(File4.shape)
print(File5.shape)
print(File6.shape)
print(File7.shape)
```

```
(1239, 14)
(1210, 14)
(1151, 14)
(11, 14)
(1120, 14)
(1051, 14)
(1155, 14)
```

merge the data

In [18]:
```
merged = pd.concat([File1,File2,File3,File4,File5,File6,File7], ignore_index=T
```

In [19]:
```
merged.to_csv("merged_file.csv", index=False)
```

In [20]:
```
df = pd.read_csv("merged_file.csv")
```

In [70]:
```
os.getcwd()
```

Out[70]:
```
'C:\\Users\\sande'
```

In [21]:
```
df
```

Out[21]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	A
0	FlyDealFarePanelM_US_FPCMP2	XNA	MAA	XNA	MAA	1.00	
1	FlyDealFarePanelM_US_FPCMP2	WAS	DEL	WAS	DEL	1.00	
2	FlyDealFarePanelM_US_FPCMP2	WAS	AMD	WAS	AMD	1.00	
3	FlyDealFarePanelM_US_FPCMP2	VPS	BOM	VPS	BOM	1.00	
4	FlyDealFarePanelM_US_FPCMP2	VGA	SFO	VGA	SFO	0.45	
...	...	...	...	...	...	...	
6932	FlyDealFarePanelDOW_US_FPCMP2	BLR	LAX	BLR	LAX	0.30	
6933	FlyDealFarePanelDOW_US_FPCMP2	BLR	DFW	BLR	DFW	0.30	
6934	FlyDealFarePanelDOW_US_FPCMP2	BLR	BOS	BLR	BOS	0.30	
6935	FlyDealFarePanelDOW_US_FPCMP2	AUS	DEL	AUS	DEL	1.20	
6936	FlyDealFarePanelDOW_US_FPCMP2	ATL	BLR	ATL	BLR	1.20	

6937 rows × 14 columns

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6937 entries, 0 to 6936
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Placement               6937 non-null   object 
 1   Origin                  6937 non-null   object 
 2   Destination             6937 non-null   object 
 3   Advertiser Origin       6937 non-null   object 
 4   Advertiser Destination  6937 non-null   object 
 5   Average CPC (USD)       6937 non-null   float64
 6   Average Rank            6937 non-null   float64
 7   First Rank Bid (USD)    6937 non-null   float64
 8   Third Rank Bid (USD)    6937 non-null   float64
 9   6th Rank Bid (USD)      6937 non-null   float64
 10  9th Rank Bid (USD)      6937 non-null   float64
 11  Est. Clicks             6937 non-null   int64  
 12  Est. Impressions        6937 non-null   int64  
 13  Est. Spend (USD)        6937 non-null   float64
dtypes: float64(7), int64(2), object(5)
memory usage: 758.9+ KB
```

In [23]:
```
df.shape
```

Out[23]: (6937, 14)

In [24]:
```
df.head()
```

Out[24]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank
0	FlyDealFarePanelM_US_FPCMP2	XNA	MAA	XNA	MAA	1.00	2.00
1	FlyDealFarePanelM_US_FPCMP2	WAS	DEL	WAS	DEL	1.00	1.22
2	FlyDealFarePanelM_US_FPCMP2	WAS	AMD	WAS	AMD	1.00	1.00
3	FlyDealFarePanelM_US_FPCMP2	VPS	BOM	VPS	BOM	1.00	2.00
4	FlyDealFarePanelM_US_FPCMP2	VGA	SFO	VGA	SFO	0.45	5.00

In [25]:
```
df.columns
```

Out[25]:
```
Index(['Placement', 'Origin', 'Destination', 'Advertiser Origin',
       'Advertiser Destination', 'Average CPC (USD)', 'Average Rank',
       'First Rank Bid (USD)', 'Third Rank Bid (USD)', '6th Rank Bid (USD)',
       '9th Rank Bid (USD)', 'Est. Clicks', 'Est. Impressions',
       'Est. Spend (USD)'],
      dtype='object')
```

Data Preprocessing

In [26]:
```
df.isna().sum()
```

Out[26]:
```
Placement                 0
Origin                    0
Destination               0
Advertiser Origin         0
Advertiser Destination    0
Average CPC (USD)         0
Average Rank              0
First Rank Bid (USD)      0
Third Rank Bid (USD)      0
6th Rank Bid (USD)        0
9th Rank Bid (USD)        0
Est. Clicks               0
Est. Impressions          0
Est. Spend (USD)          0
dtype: int64
```

Statistical summary

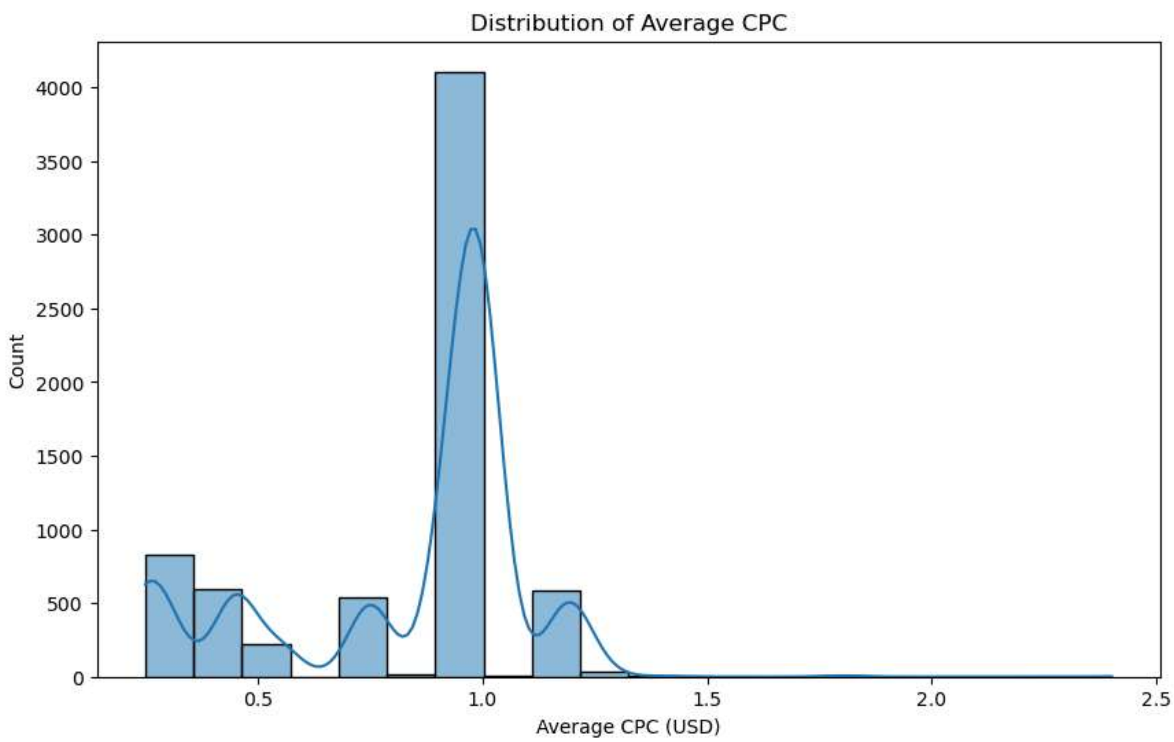
In [27]:
```
df.describe().T
```

Out[27]:

	count	mean	std	min	25%	50%	75%	max
Average CPC (USD)	6937.0	0.835876	0.282954	0.25	0.75	0.95	1.00	2.40
Average Rank	6937.0	2.709693	1.221168	1.00	2.00	2.00	3.00	10.00
First Rank Bid (USD)	6937.0	1.307314	0.459999	0.25	1.22	1.22	1.22	4.10
Third Rank Bid (USD)	6937.0	0.888505	0.287122	0.26	0.76	1.01	1.01	2.41
6th Rank Bid (USD)	6937.0	0.848429	0.281937	0.26	0.76	0.96	1.01	2.41
9th Rank Bid (USD)	6937.0	0.845918	0.282953	0.26	0.76	0.96	1.01	2.41
Est. Clicks	6937.0	0.334583	0.717675	0.00	0.00	0.00	1.00	10.00
Est. Impressions	6937.0	3.075537	5.442984	0.00	1.00	2.00	3.00	108.00
Est. Spend (USD)	6937.0	0.266150	0.640340	0.00	0.00	0.00	0.15	10.00

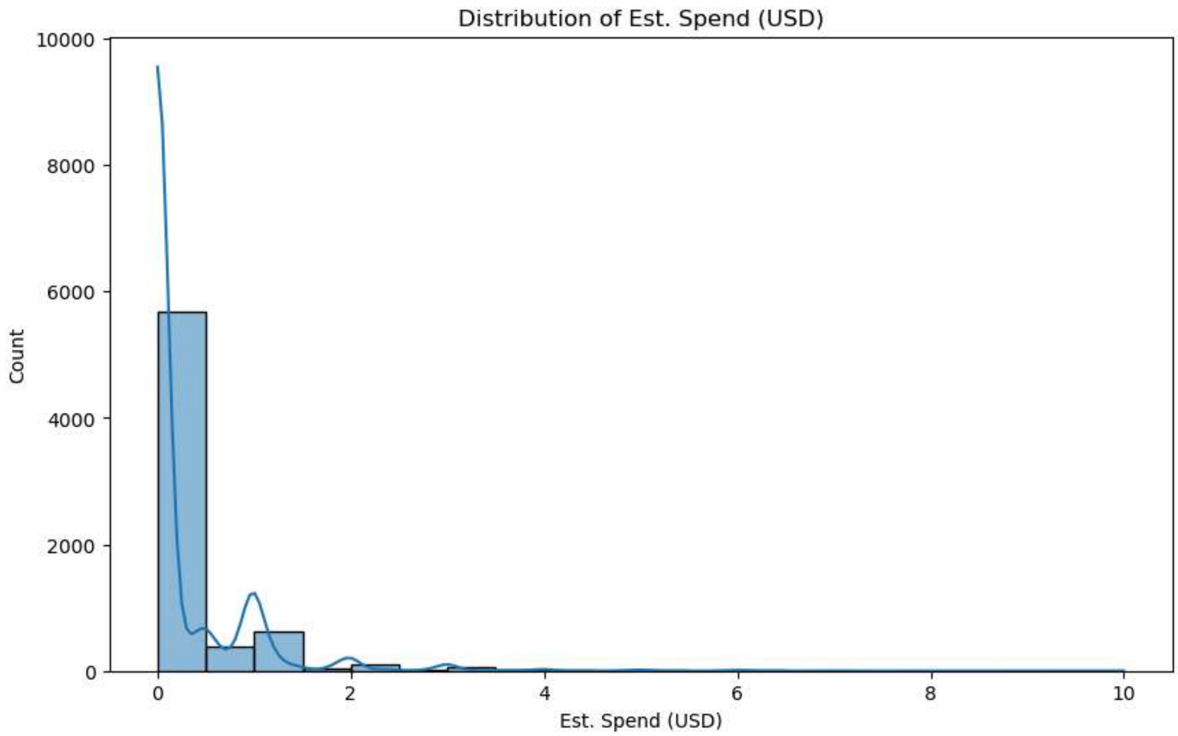
Visualization

In [28]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['Average CPC (USD)'], bins=20, kde=True)
plt.title('Distribution of Average CPC')
plt.show()
```

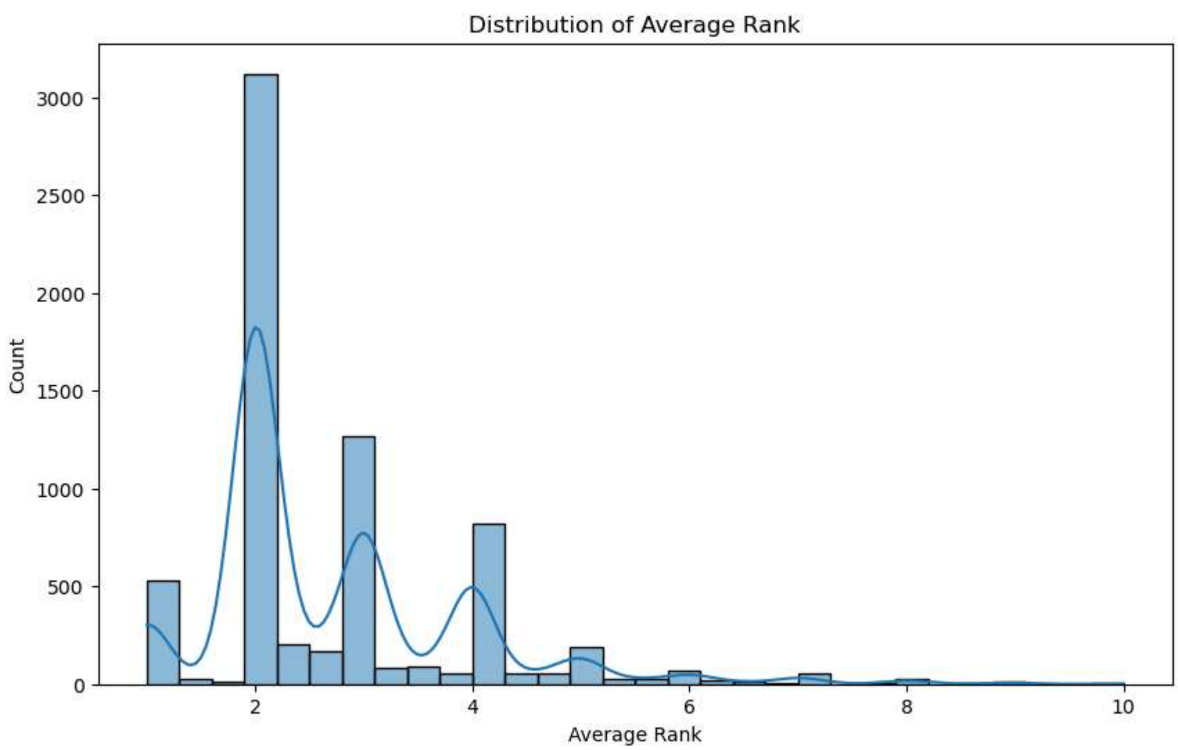


In [ ]:
```
# it seems that most values are clustered around a particular cpc near to 1.0
```

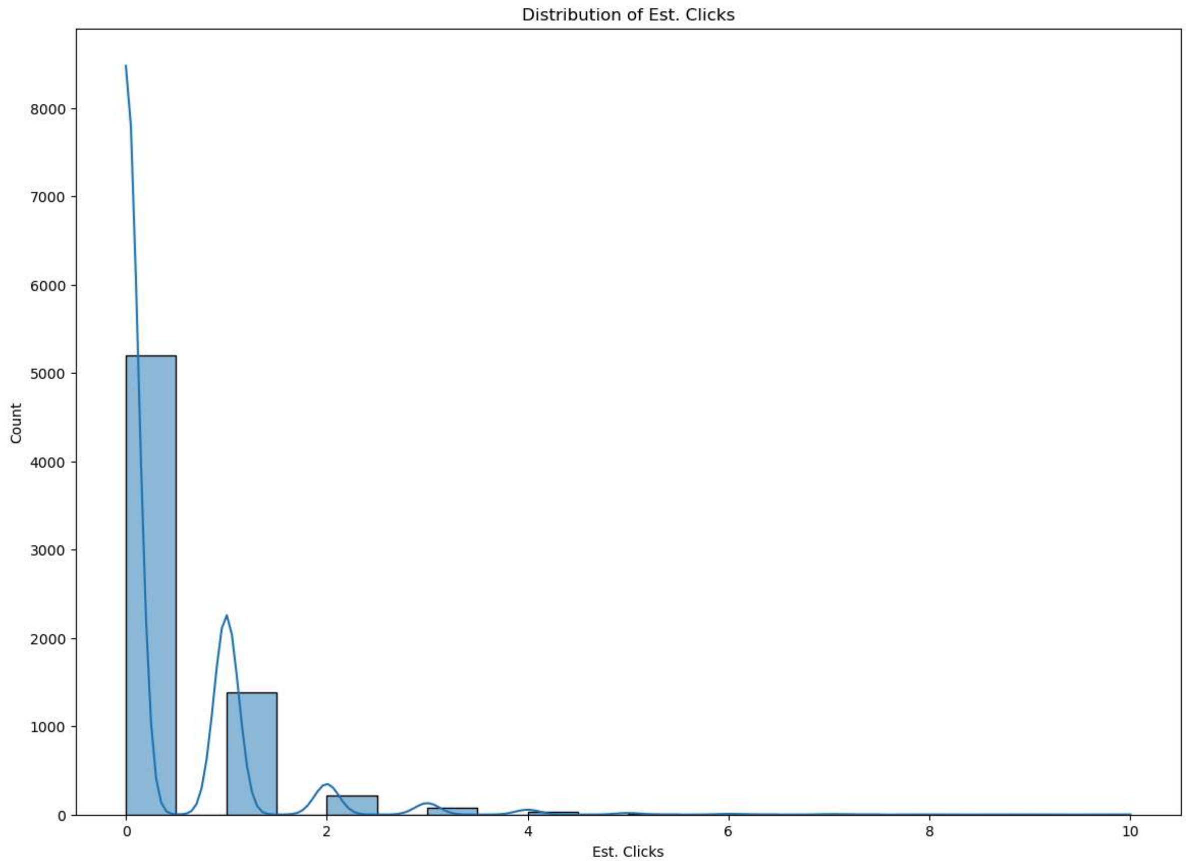
In [29]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(df['Est. Spend (USD)'], bins=20, kde=True)
plt.title('Distribution of Est. Spend (USD)')
plt.show()
```



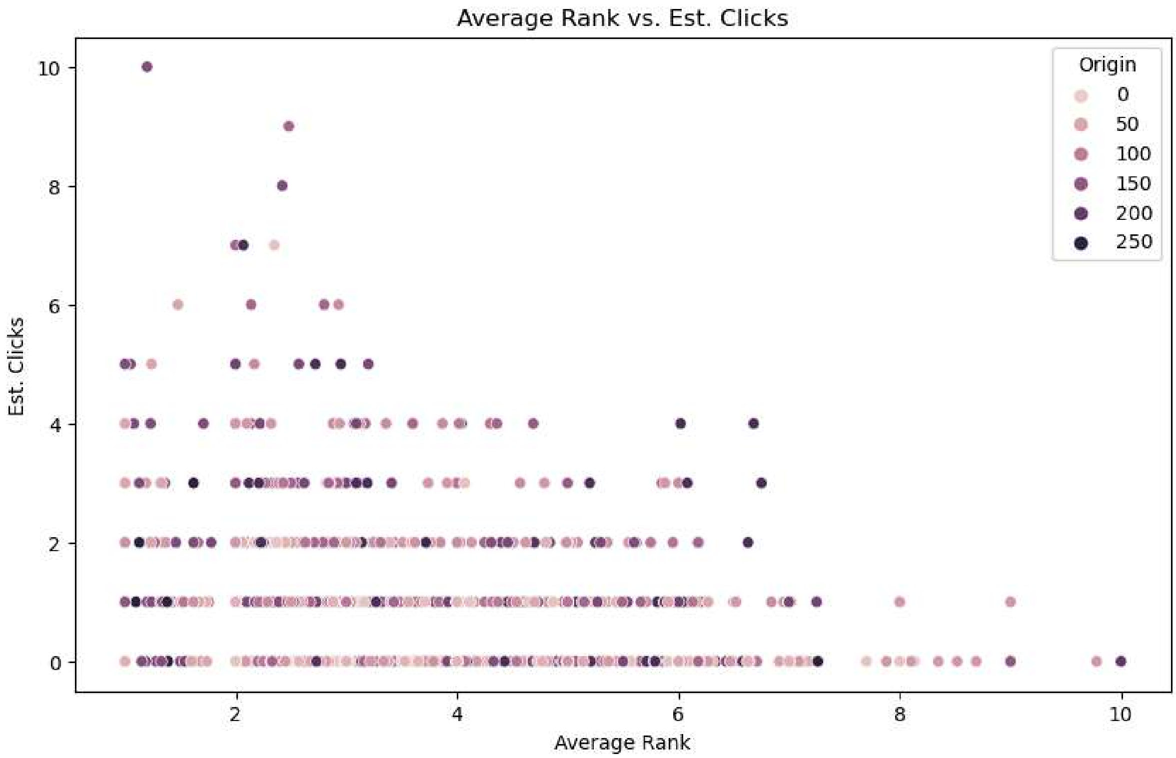
In [114]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(df['Average Rank'], bins=30, kde=True)
plt.title('Distribution of Average Rank')
plt.show()
```



In [117]:
```
plt.figure(figsize=(14, 10))
sns.histplot(df['Est. Clicks'], bins=20, kde=True)
plt.title('Distribution of Est. Clicks')
plt.show()
```



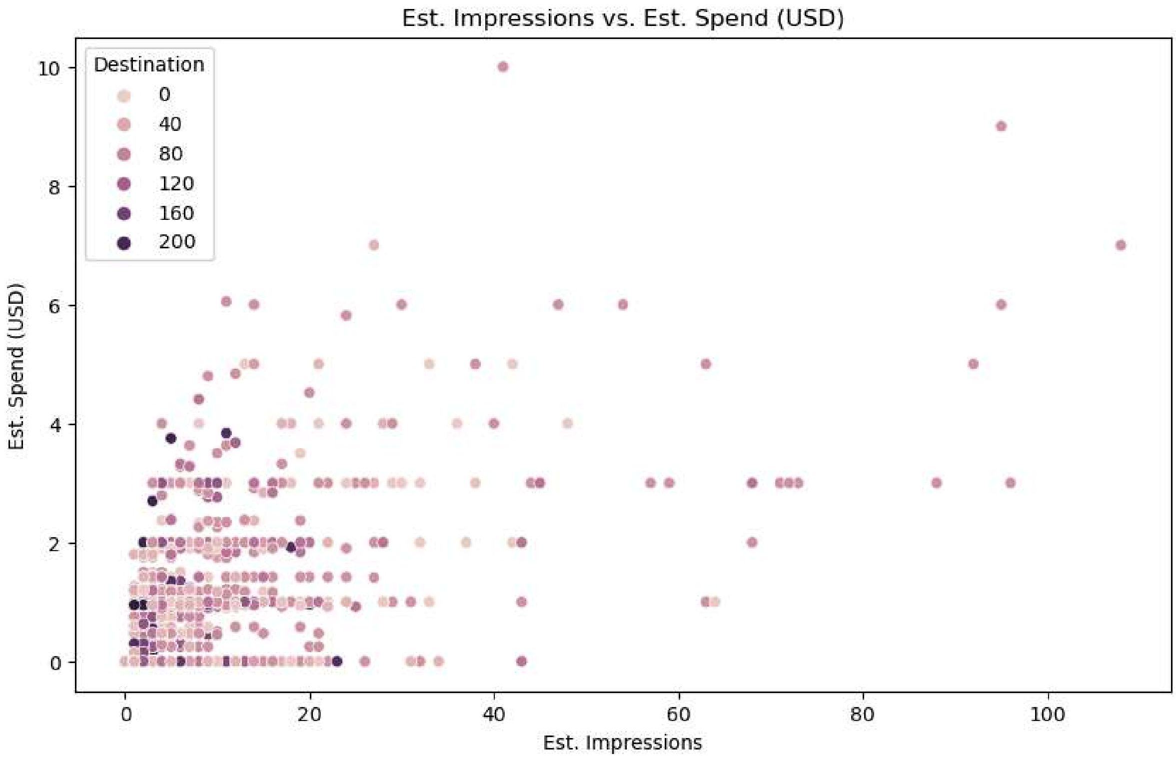
In [107]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Average Rank', y='Est. Clicks', data=df, hue='Origin')
plt.title('Average Rank vs. Est. Clicks')
plt.xlabel('Average Rank')
plt.ylabel('Est. Clicks')
plt.show()
```



In [ ]:
```
# in Average rank and estimated clicks plotting we can observe that average ra
```

In [112]:

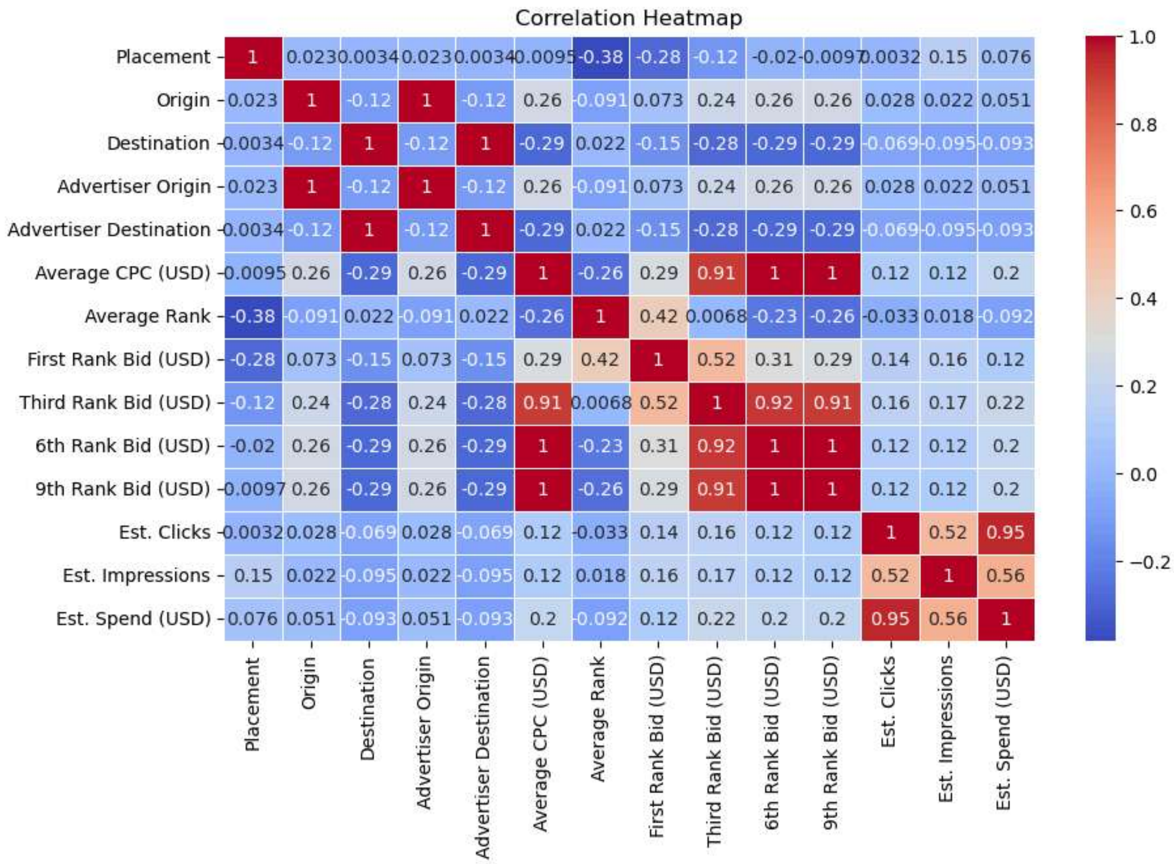
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Est. Impressions', y='Est. Spend (USD)', data=df, hue='Dest
plt.title('Est. Impressions vs. Est. Spend (USD)')
plt.xlabel('Est. Impressions')
plt.ylabel('Est. Spend (USD)')
plt.show()
```



In [ ]:

```
# it can be seem that estimated spend is more for the estimated impression bet
```

```
In [111]: plt.figure(figsize=(10, 6))
          sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
          plt.title('Correlation Heatmap')
          plt.show()
```




```
In [ ]: # it can be seen that all the numerical independent variables are correlated p
```

```
In [30]: from sklearn.preprocessing import LabelEncoder
```

```
In [31]: le = LabelEncoder()
```

```
In [32]: df['Placement'] = le.fit_transform(df['Placement'])
```

```
In [33]: df['Origin'] = le.fit_transform(df['Origin'])
```

```
In [34]: df['Destination'] = le.fit_transform(df['Destination'])
```

```
In [35]: df['Advertiser Origin'] = le.fit_transform(df['Advertiser Origin'])
```

In [36]:
```python
df['Advertiser Destination'] = le.fit_transform(df['Advertiser Destination'])
```

In [37]:
```python
df
```

Out[37]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank	First Rank Bid (USD)	Third Rank Bid (USD)
0	3	258	135	258	135	1.00	2.00	1.22	1.01
1	3	257	61	257	61	1.00	1.22	1.01	1.01
2	3	257	8	257	8	1.00	1.00	1.00	1.01
3	3	255	28	255	28	1.00	2.00	1.22	1.01
4	3	253	190	253	190	0.45	5.00	1.22	0.61
...	...	...	...	...	...	...	...	...	...
6932	0	33	125	33	125	0.30	3.53	1.02	0.56
6933	0	33	64	33	64	0.30	4.00	1.02	0.65
6934	0	33	29	33	29	0.30	4.00	1.02	0.65
6935	0	19	61	19	61	1.20	2.00	1.28	1.21
6936	0	15	25	15	25	1.20	2.00	1.66	1.21

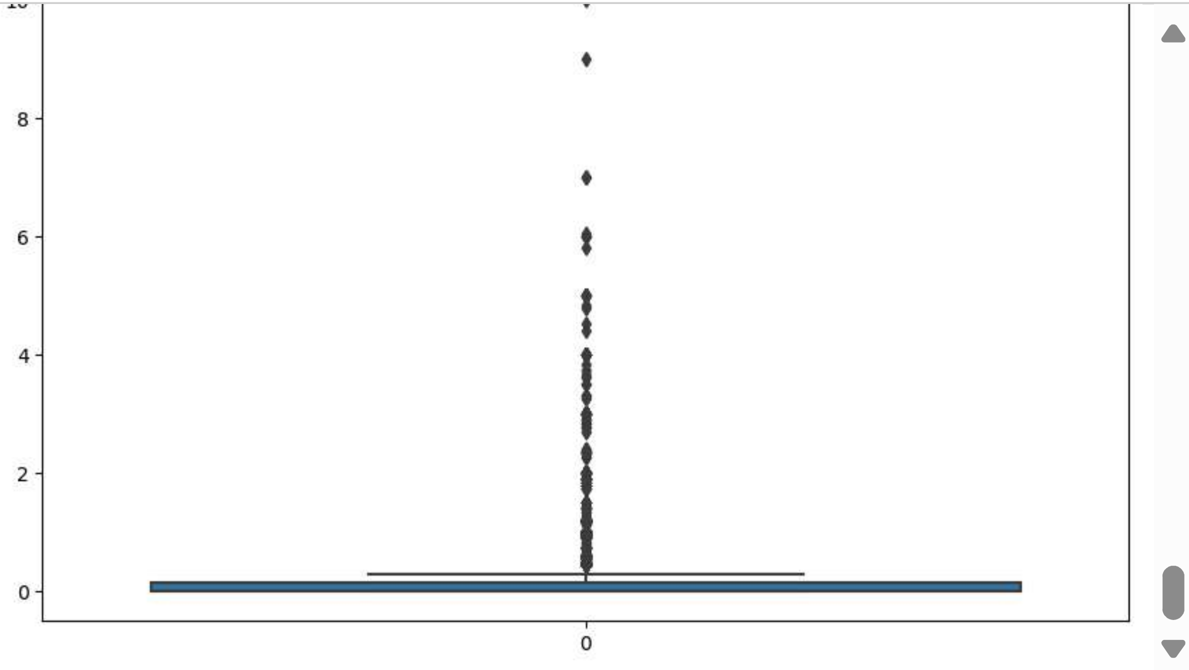
6937 rows × 14 columns

In [39]:
```python
df['Est. Spend (USD)'].value_counts()
```

Out[39]:
```
Est. Spend (USD)
0.00     5202
1.00      479
0.47      199
0.95      107
2.00       93
         ...
2.88        1
9.00        1
2.32        1
10.00       1
1.19        1
Name: count, Length: 84, dtype: int64
```

Outliers removing

In [72]:
```
# Boxplot to detect outliers
for column in df.select_dtypes(include=['float64', 'int64']).columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(df[column])
    plt.title(f'Boxplot of {column}')
    plt.show()
```



In [73]:
```
# Remove outliers using Z-score method or IQR method
from scipy import stats

df_no_outliers = df[(np.abs(stats.zscore(df.select_dtypes(include=['float64',
```

In [74]:
```
df_no_outliers
```

Out[74]:

	Placement	Origin	Destination	Advertiser Origin	Advertiser Destination	Average CPC (USD)	Average Rank	First Rank Bid (USD)	Third Rank Bid (USD)
0	3	258	135	258	135	1.00	2.00	1.22	1.01
1	3	257	61	257	61	1.00	1.22	1.01	1.01
2	3	257	8	257	8	1.00	1.00	1.00	1.01
3	3	255	28	255	28	1.00	2.00	1.22	1.01
4	3	253	190	253	190	0.45	5.00	1.22	0.61
...	...	...	...	...	...	...	...	...	...
6932	0	33	125	33	125	0.30	3.53	1.02	0.56
6933	0	33	64	33	64	0.30	4.00	1.02	0.65
6934	0	33	29	33	29	0.30	4.00	1.02	0.65
6935	0	19	61	19	61	1.20	2.00	1.28	1.21
6936	0	15	25	15	25	1.20	2.00	1.66	1.21

6484 rows × 14 columns

Standardization

In [56]:
```
from sklearn.preprocessing import StandardScaler
```

In [57]:
```
scaler = StandardScaler()
```

In [58]:
```
df_scaled = scaler.fit_transform(df[['Average CPC (USD)', 'Average Rank', 'Fir
```

In [59]:
```
df_scaled
```

Out[59]:
```
array([[ 0.58008102, -0.58120085, -0.18982796, ...,  0.92725219,
        -0.19761487,  1.14611492],
       [ 0.58008102, -1.21997944, -0.64638346, ...,  2.32074189,
         0.9048012 ,  2.70789777],
       [ 0.58008102, -1.40014776, -0.6681242 , ...,  0.92725219,
        -0.38135089,  1.14611492],
       ...,
       [-1.89399996,  1.05669297, -0.62464273, ...,  0.92725219,
        -0.38135089, -0.1814005 ],
       [ 1.28696129, -0.58120085, -0.05938352, ..., -0.46623751,
        -0.38135089, -0.41566793],
       [ 1.28696129, -0.58120085,  0.76676454, ..., -0.46623751,
        -0.38135089, -0.41566793]])
```

Data modeling

In [62]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [76]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [77]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [78]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[78]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [79]:
```
y_pred = model.predict(X_test)
```

Model Evaluation

In [80]:
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mse ** 0.5
r2 = r2_score(y_test, y_pred)
```

In [81]:
```
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R²): {r2}")
```

```
Mean Absolute Error (MAE): 0.11316998048865708
Mean Squared Error (MSE): 0.03248948761793285
Root Mean Squared Error (RMSE): 0.18024840531314792
R-squared (R²): 0.925797710525115
```

Report on Regression Model Performance:

Model Accuracy:

The high R-squared (0.9258) indicates that the model has a strong ability to explain the variance in the data. This suggests that the model is well-suited for the dataset and that most of the target variable’s behavior is being captured effectively.

Error Metrics:

The MAE (0.1132) is relatively low, which means that, on average, the model's predictions are close to the actual values. MAE is particularly useful as it is easy to understand and does not exaggerate the impact of outliers. The MSE (0.0325) and RMSE (0.1802) are also low, which further indicates that the model has good predictive performance. The RMSE, in particular, shows that the model's average prediction error is just under 0.2 units on the same scale as the data.

Model Robustness:

The low MSE and RMSE suggest that the model's predictions are generally accurate, with

Overall Evaluation:

This model is performing very well, as indicated by the combination of high R-squared and low error metrics. The results suggest that the model is reliable and can be used with confidence for predictive purposes in similar contexts.