import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

*# Load the dataset*

df = pd.read\_csv('/kaggle/input/aviator/aviator\_payouts.csv')

*# Display the first few rows of the dataset*

print(df.head())

created\_at app payout

0 2024-02-03 10:41:29.066595+00 WINPESA 2.07

1 2024-02-03 10:41:28.237897+00 ODIBETS 2.04

2 2024-02-03 10:41:18.705345+00 ODIBETS 1.63

3 2024-02-03 10:41:08.531921+00 BETIKA 18.40

4 2024-02-03 10:40:48.961445+00 WINPESA 5.03

In [6]:

*# Convert 'created\_at' column to datetime format*

df['created\_at'] = pd.to\_datetime(df['created\_at'])

*# Count the occurrences of each second in the 'created\_at' column*

second\_counts = df['created\_at'].dt.floor('s').value\_counts()

*# Identify duplicates where 5 or more payouts have the same second*

duplicates\_mask = df['created\_at'].dt.floor('s').map(second\_counts) >= 5

*# Filter out the duplicates*

df = df[~duplicates\_mask]

In [7]:

*# Summary statistics*

print(df.describe())

*# Time series plot of payouts over time*

plt.figure(figsize=(10, 6))

payouts\_by\_date = df.groupby(df['created\_at'].dt.date)['payout'].sum()

plt.plot(payouts\_by\_date.index, payouts\_by\_date.values, marker='o', linestyle='-')

plt.title('Payouts Over Time')

plt.xlabel('Date')

plt.ylabel('Total Payout')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()

*# Boxplot of payouts by betting company*

plt.figure(figsize=(10, 6))

sns.boxplot(x='app', y='payout', data=df)

plt.title('Boxplot of Payouts by Betting Company')

plt.xlabel('Betting Company')

plt.ylabel('Payout')

plt.grid(True)

plt.show()

payout

count 2.551000e+05

mean 1.997357e+01

std 4.122334e+03

min 1.000000e+00

25% 1.290000e+00

50% 1.940000e+00

75% 3.880000e+00

max 2.071590e+06

In [8]:

print(df.isnull().sum())

created\_at 0

app 0

payout 0

dtype: int64

In [9]:

*# Remove outliers in the 'payout' column using z-score*

from scipy import stats

z\_scores = stats.zscore(df['payout'])

abs\_z\_scores = abs(z\_scores)

filtered\_entries = (abs\_z\_scores < 3)

df = df[filtered\_entries]

In [11]:

*# Total payouts by betting company*

plt.figure(figsize=(10, 6))

total\_payouts\_by\_company = df.groupby('app')['payout'].sum().sort\_values(ascending=False)

total\_payouts\_by\_company.plot(kind='bar')

plt.title('Total Payouts by Betting Company')

plt.xlabel('Betting Company')

plt.ylabel('Total Payout')

plt.grid(axis='y')

plt.xticks(rotation=45)

plt.show()

In [13]:

*# Pie chart of payouts distribution by betting company*

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

payouts\_by\_company = df.groupby('app')['payout'].sum()

plt.pie(payouts\_by\_company, labels=payouts\_by\_company.index, autopct='**%1.1f%%**', startangle=140)

plt.title('Payouts Distribution by Betting Company')

plt.show()

In [14]:

*# Detect and analyze outliers*

outliers = df[df['payout'] > df['payout'].quantile(0.95)]

In [15]:

*# Time series analysis of payouts by hour of the day*

df['hour'] = df['created\_at'].dt.hour

hourly\_payouts = df.groupby('hour')['payout'].mean()

hourly\_payouts.plot(kind='bar', figsize=(10, 6))

plt.title('Average Payouts by Hour of the Day')

plt.xlabel('Hour of the Day')

plt.ylabel('Average Payout')

plt.xticks(rotation=0)

plt.grid(axis='y')

plt.show()

*# Comparison of average payouts between weekdays and weekends*

df['day\_of\_week'] = df['created\_at'].dt.dayofweek

df['is\_weekend'] = df['day\_of\_week'].isin([5, 6])

avg\_payouts\_weekend = df[df['is\_weekend']]['payout'].mean()

avg\_payouts\_weekday = df[~df['is\_weekend']]['payout'].mean()

print("Average Payout on Weekends:", avg\_payouts\_weekend)

print("Average Payout on Weekdays:", avg\_payouts\_weekday)

*# Distribution of payouts over different months*

df['month'] = df['created\_at'].dt.month

monthly\_payouts = df.groupby('month')['payout'].sum()

monthly\_payouts.plot(kind='bar', figsize=(10, 6))

plt.title('Total Payouts by Month')

plt.xlabel('Month')

plt.ylabel('Total Payout')

plt.xticks(rotation=0)

plt.grid(axis='y')

plt.show()

Average Payout on Weekends: 9.276150661027476

Average Payout on Weekdays: 9.292056620683034

In [18]:

*# Convert 'payout' column to numeric type*

df['payout'] = pd.to\_numeric(df['payout'], errors='coerce')

*# Check for unique values in the 'app' column*

print(df['app'].unique())

*# If inconsistencies are found, consider standardizing naming conventions*

['WINPESA' 'ODIBETS' 'BETIKA' 'BETGR8']

In [19]:

*# Distribution of payouts over different days of the week*

weekday\_payouts = df.groupby(df['created\_at'].dt.dayofweek)['payout'].mean()

weekday\_payouts.index = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

weekday\_payouts.plot(kind='bar', figsize=(10, 6))

plt.title('Average Payouts by Day of the Week')

plt.xlabel('Day of the Week')

plt.ylabel('Average Payout')

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.show()

*# Relationship between payout amounts and time elapsed since the previous payout*

df['time\_since\_previous\_payout'] = df['created\_at'].diff().dt.total\_seconds() / 60 *# Time elapsed in minutes*

plt.figure(figsize=(10, 6))

plt.scatter(df['time\_since\_previous\_payout'], df['payout'], alpha=0.5)

plt.title('Relationship between Payout and Time Since Previous Payout')

plt.xlabel('Time Since Previous Payout (minutes)')

plt.ylabel('Payout')

plt.grid(True)

plt.show()

In [21]:

*# Distribution of payouts by hour of the day for each betting company*

plt.figure(figsize=(12, 8))

sns.violinplot(x='hour', y='payout', hue='app', data=df)

plt.title('Payouts Distribution by Hour of the Day and Betting Company')

plt.xlabel('Hour of the Day')

plt.ylabel('Payout')

plt.legend(title='Betting Company')

plt.grid(True)

plt.show()

In [22]:

*# Distribution of payouts by hour of the day for each betting company*

plt.figure(figsize=(12, 8))

sns.catplot(x='hour', y='payout', hue='app', kind='box', data=df, height=6, aspect=2)

plt.title('Payouts Distribution by Hour of the Day and Betting Company')

plt.xlabel('Hour of the Day')

plt.ylabel('Payout')

plt.grid(True)

plt.show()

<Figure size 1200x800 with 0 Axes>

In [23]:

*# Relationship between payout amounts and the day of the week*

weekday\_payouts = df.groupby(df['created\_at'].dt.dayofweek)['payout'].mean()

weekday\_payouts.index = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

weekday\_payouts.plot(kind='bar', figsize=(10, 6))

plt.title('Average Payouts by Day of the Week')

plt.xlabel('Day of the Week')

plt.ylabel('Average Payout')

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.show()

In [25]:

*# Calculate the Interquartile Range (IQR)*

Q1 = df['payout'].quantile(0.25)

Q3 = df['payout'].quantile(0.75)

IQR = Q3 - Q1

*# Identify outliers using the IQR method*

outliers = df[(df['payout'] < Q1 - 1.5 \* IQR) | (df['payout'] > Q3 + 1.5 \* IQR)]

print("Outliers:**\n**", outliers)

*# Normalize the 'payout' column*

df['normalized\_payout'] = (df['payout'] - df['payout'].mean()) / df['payout'].std()

Outliers:

created\_at app payout hour day\_of\_week \

3 2024-02-03 10:41:08.531921+00:00 BETIKA 18.40 10 5

12 2024-02-03 10:39:04.882000+00:00 BETIKA 13.02 10 5

196 2024-02-03 09:52:28.572610+00:00 BETIKA 9.60 9 5

208 2024-02-03 09:50:25.323566+00:00 WINPESA 14.85 9 5

211 2024-02-03 09:49:55.963231+00:00 ODIBETS 9.98 9 5

... ... ... ... ... ...

299970 2024-01-01 09:32:01.290760+00:00 BETGR8 13.43 9 0

299975 2024-01-01 09:30:47.024244+00:00 BETIKA 62.83 9 0

299988 2024-01-01 09:27:12.327310+00:00 BETGR8 14.15 9 0

299993 2024-01-01 09:25:54.024233+00:00 BETIKA 16.34 9 0

299994 2024-01-01 09:25:41.050928+00:00 BETGR8 12.25 9 0

is\_weekend month time\_since\_previous\_payout

3 True 2 -0.169557

12 True 2 -0.152883

196 True 2 -0.028715

208 True 2 -0.018464

211 True 2 -0.002440

... ... ... ...

299970 False 1 -0.211697

299975 False 1 -0.231705

299988 False 1 -0.041992

299993 False 1 -0.125747

299994 False 1 -0.216222

[32013 rows x 8 columns]

In [26]:

*# Distribution of payouts by month*

monthly\_payouts = df.groupby(df['created\_at'].dt.month)['payout'].sum()

monthly\_payouts.plot(kind='bar', figsize=(10, 6))

plt.title('Total Payouts by Month')

plt.xlabel('Month')

plt.ylabel('Total Payout')

plt.xticks(rotation=0)

plt.grid(axis='y')

plt.show()

*# Relationship between payout amounts and time elapsed since the previous payout*

plt.figure(figsize=(10, 6))

plt.scatter(df['time\_since\_previous\_payout'], df['payout'], alpha=0.5)

plt.title('Relationship between Payout and Time Since Previous Payout')

plt.xlabel('Time Since Previous Payout (minutes)')

plt.ylabel('Payout')

plt.grid(True)

plt.show()

In [29]:

*# Distribution of normalized payouts by betting company*

plt.figure(figsize=(10, 6))

sns.boxplot(x='app', y='normalized\_payout', data=df)

plt.title('Distribution of Normalized Payouts by Betting Company')

plt.xlabel('Betting Company')

plt.ylabel('Normalized Payout')

plt.grid(True)

plt.show()

In [30]:

linkcode

*# Check for unique values in the 'app' column*

print(df['app'].unique())

*# Handle inconsistencies in 'app' column if found*

df['app'] = df['app'].str.lower().str.strip() *# Convert to lowercase and remove leading/trailing spaces*

*# Explore and handle issues with 'created\_at' column*

print("Number of missing timestamps:", df['created\_at'].isnull().sum())

print("Number of duplicate timestamps:", df['created\_at'].duplicated().sum())

['WINPESA' 'ODIBETS' 'BETIKA' 'BETGR8']

Number of missing timestamps: 0

Number of duplicate timestamps: 19