Apple Store Reviews

In [24]:

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

In [25]:

```
df=pd.read_csv('Apple_Store_Reviews.csv')
df
```

Out[25]:

	Review_ID	App_Name	User_Age	Review_Date	Rating	Review_Text	Likes	Device_'
0	1	Candy Crush Saga	21	2023-01-16	4	Great game, but too many in-game purchases.	70	iPhor
1	2	Spotify	57	2024-02-01	1	Good, but has connection issues sometimes.	49	iPhor
2	3	TikTok	33	2023-11-30	5	Awesome app! Best entertainment content.	98	iPhor
3	4	Audible	40	2023-04-03	5	Great app, but it's a bit pricey.	74	iPhor
4	5	Spotify	44	2023-05-01	1	Good, but has connection issues sometimes.	47	iPhor
...	...	...	...	...	...	...	...	
995	996	Headspace	30	2023-11-15	3	Good, but the premium content is expensive.	65	iPhor
996	997	Duolingo	19	2024-09-27	1	Disappointing. Hard to follow and buggy.	4	iPhor
997	998	Duolingo	38	2023-06-07	5	Excellent for learning new skills!	85	iPhor
998	999	Instagram	52	2024-03-04	4	Great app, but sometimes it lags.	55	iPhor
999	1000	Audible	25	2024-02-20	2	Terrible. Very limited selection of books.	7	iPhor

1000 rows × 12 columns

In [26]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Review_ID        1000 non-null   int64  
 1   App_Name         1000 non-null   object 
 2   User_Age         1000 non-null   int64  
 3   Review_Date      1000 non-null   object 
 4   Rating           1000 non-null   int64  
 5   Review_Text      1000 non-null   object 
 6   Likes            1000 non-null   int64  
 7   Device_Type      1000 non-null   object 
 8   Version_Used     1000 non-null   object 
 9   Country          1000 non-null   object 
 10  Purchase_Amount  1000 non-null   float64
 11  Category         1000 non-null   object 
dtypes: float64(1), int64(4), object(7)
memory usage: 93.9+ KB
```

In [27]:

```
df.describe()
```

Out[27]:

	Review_ID	User_Age	Rating	Likes	Purchase_Amount
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	39.211000	2.869000	44.776000	5.361120
std	288.819436	11.908917	1.467649	28.685444	5.755652
min	1.000000	18.000000	1.000000	0.000000	0.000000
25%	250.750000	30.000000	1.000000	17.000000	0.000000
50%	500.500000	39.000000	3.000000	42.500000	4.995000
75%	750.250000	49.000000	4.000000	71.000000	10.192500
max	1000.000000	60.000000	5.000000	100.000000	19.970000

1. Calculate the mean, median, and mode of the app ratings in the dataset. Which measure (mean, median, or mode) best represents the central tendency of the ratings?

In [28]:

```
#calculating mean of rating column in dataset.
rating_mean = round(float(df['Rating'].mean()),2)
rating_mean
```

Out[28]:

```
2.87
```

In [29]:

```
#calculating median of rating column in dataset.
rating_median = round(float(df['Rating'].median()),2)
rating_median
```

Out[29]:

```
3.0
```

In [30]:

```
#calculating mode of rating column in dataset.
rating_mode = float(df['Rating'].mode()[0])
rating_mode
```

Out[30]:

```
1.0
```

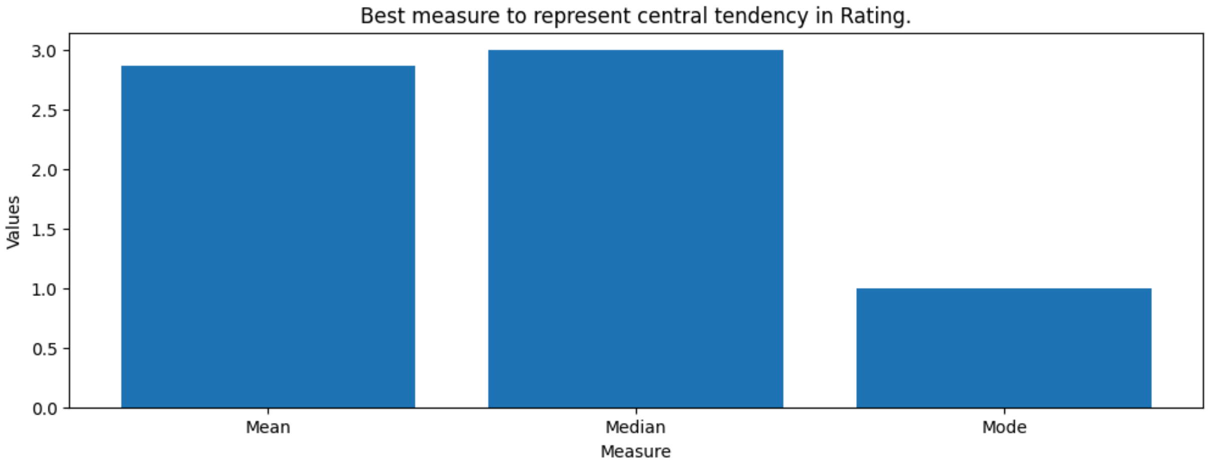
In [31]:

```
print(f"Mean of rating : {rating_mean} .")
print(f"Median of rating : {rating_median} .")
print(f"Mode of rating : {rating_mode} .")
```

```
Mean of rating : 2.87 .
Median of rating : 3.0 .
Mode of rating : 1.0 .
```

In [32]:

```
plt.figure(figsize=(12,4))
plt.bar(x=["Mean","Median","Mode"],height=[rating_mean,rating_median,rating_mode])
plt.title("Best measure to represent central tendency in Rating.")
plt.xlabel("Measure")
plt.ylabel("Values")
plt.show()
```



As we can see from above analysis, Median of 'Rating' column tends to have greater value than compare to others. So we can say that 'Median' represents the best central tendency for Rating column.

2. Find the range and interquartile range (IQR) of the Purchase_Amount in the dataset. How do these values help in understanding the spread of the data?

In [33]:

```
#Max value in Purchase_Amount column.
max_purchase_amount = float(df['Purchase_Amount'].max())
max_purchase_amount
```

Out[33]:

```
19.97
```

In [34]:

```
#Min value in Purchase_Amount column.
min_purchase_amount = float(df['Purchase_Amount'].min())
min_purchase_amount
```

Out[34]: 0.0

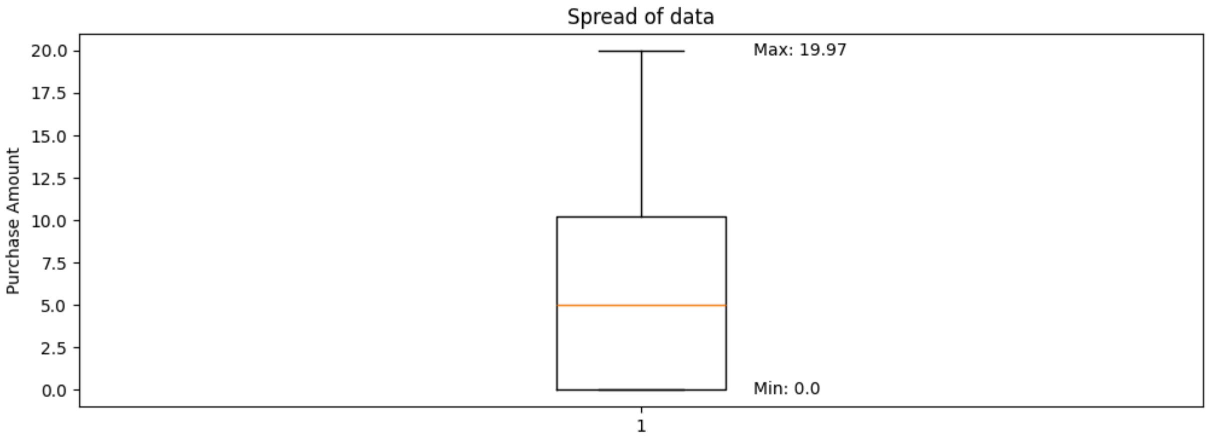
In [35]:
```
#Range of Purchase_Amount column.
range_purchase_amount = max_purchase_amount-min_purchase_amount
print(f"Range of Purchase_Amount column in {range_purchase_amount} .")
```

```
Range of Purchase_Amount column in 19.97 .
```

In [36]:
```
plt.figure(figsize=(12,4))
plt.boxplot(df['Purchase_Amount'])
plt.title("Spread of data")
plt.ylabel("Purchase Amount")

# Add text labels for min and max
plt.text(1.1, min_purchase_amount, f'Min: {min_purchase_amount}', va='center')
plt.text(1.1, max_purchase_amount, f'Max: {max_purchase_amount}', va='center')

plt.show()
```



3. Calculate the variance and standard deviation for the number of likes received on reviews. What does the standard deviation indicate about the spread of the data?

In [39]:
```
variance_likes = float(round(df['Likes'].var(),2))
variance_likes
```

Out[39]: 822.85

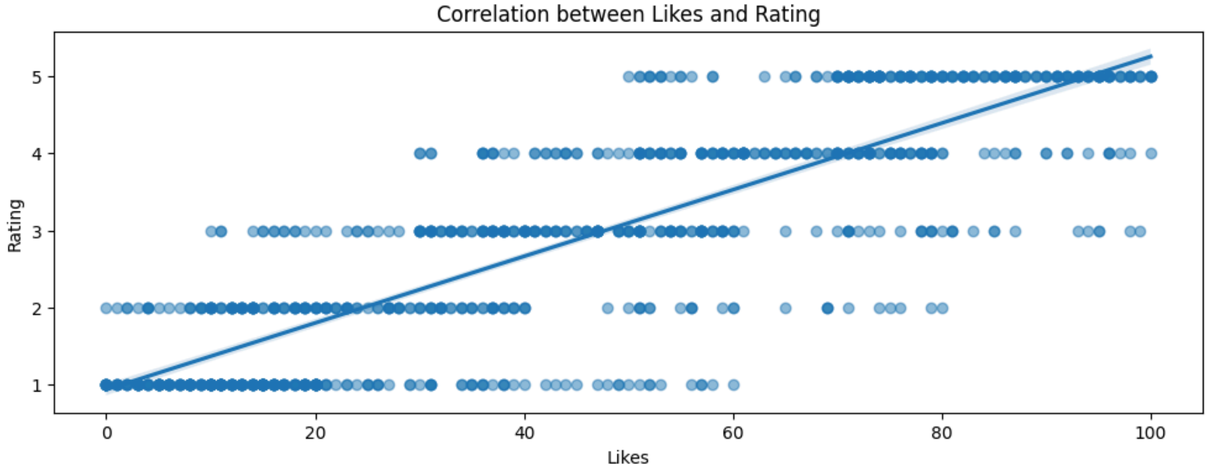
In [38]:
```
standard_deviation_likes = float(round(df['Likes'].std(),2))
standard_deviation_likes
```

Out[38]: 28.69

Observation: As we can observe, the standard deviation is **28.69**, which is a relatively high value. This indicates that the number of likes varies widely between reviews.

4. Determine the correlation between the likes and the rating given. Is there a positive, negative, or no correlation between these variables?

In [46]:
```
plt.figure(figsize=(12,4))
sns.regplot(data=df,x='Likes',y='Rating', scatter_kws={'alpha':0.5})
plt.title('Correlation between Likes and Rating')
plt.show()
```



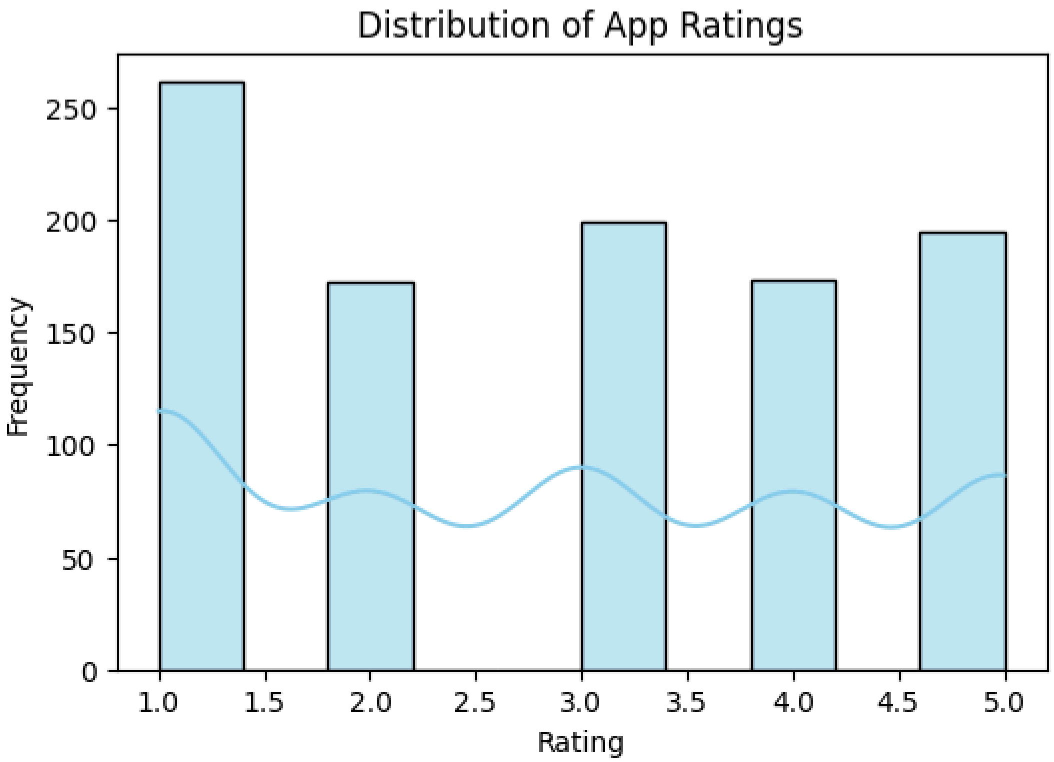
In [53]:
```
# Calculate correlation value
correlation = round(df['Likes'].corr(df['Rating']),2)

if(correlation>0):
   print(f"Correlation coefficient have Positive relationship b/w Likes and Rating
elif(correlation<0):
    print(f"Correlation coefficient have Negative relationship b/w Likes and Rating
else:
    print(f"Correlation coefficient have no relationship b/w Likes and Rating with
```

```
Correlation coefficient have Positive relationship b/w Likes and Rating with value o
f 0.84 .
```

5. Plot the distribution of the app ratings. Is the distribution positively or negatively skewed? What does this indicate about user satisfaction?

In [61]:
```
plt.figure(figsize=(6,4))
sns.histplot(df['Rating'], bins=10, kde=True, color='skyblue')
plt.title('Distribution of App Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



In [68]:

```
skew_value = round(df['Rating'].skew(),2)
#print("Skewness:", skew_value)
if skew_value >0:
    print(f"There is Right Skewed/Positive Skewness. This means Mean and Median > M
elif skew_value < 0:
    print(f"There is Left Skewed/Negative Skewness. This means Mean and Median < Mo
else:
    print(f"There is Normal Destribution. This means Mean=Median=Mode")
```

```
There is Right Skewed/Positive Skewness. This means Mean and Median > Mode.
```

6. Perform a hypothesis test to determine if the average rating for Instagram is significantly higher than the average rating for WhatsApp. Use a 95% confidence level.

In [71]:

```
insta_ratings = df[df['App_Name'] == 'Instagram']['Rating']
whatsapp_ratings = df[df['App_Name'] == 'WhatsApp']['Rating']
```

In [74]:

```
#pip install scipy
```

In [75]:

```
from scipy import stats

t_stat, p_value = stats.ttest_ind(insta_ratings, whatsapp_ratings, alternative='gre
print("t-statistic:", t_stat)
print("p-value:", p_value)
```

```
t-statistic: -0.79674231444911
p-value: 0.786764229580496
```

In [76]:

```
if(p_value < 0.05):
    print(f"Reject H₀ -> Instagram’s average rating is significantly higher.")
```

```
else:
    #p_value ≥ 0.05
    print(f"Fail to reject H₀ -> No significant evidence that Instagram’s average r
```

```
Fail to reject H₀ -> No significant evidence that Instagram’s average rating is high
er.
```

7. Take random samples of ratings from the dataset and calculate their means. Create a sampling distribution and explain how this relates to the Central Limit Theorem.

In [81]:

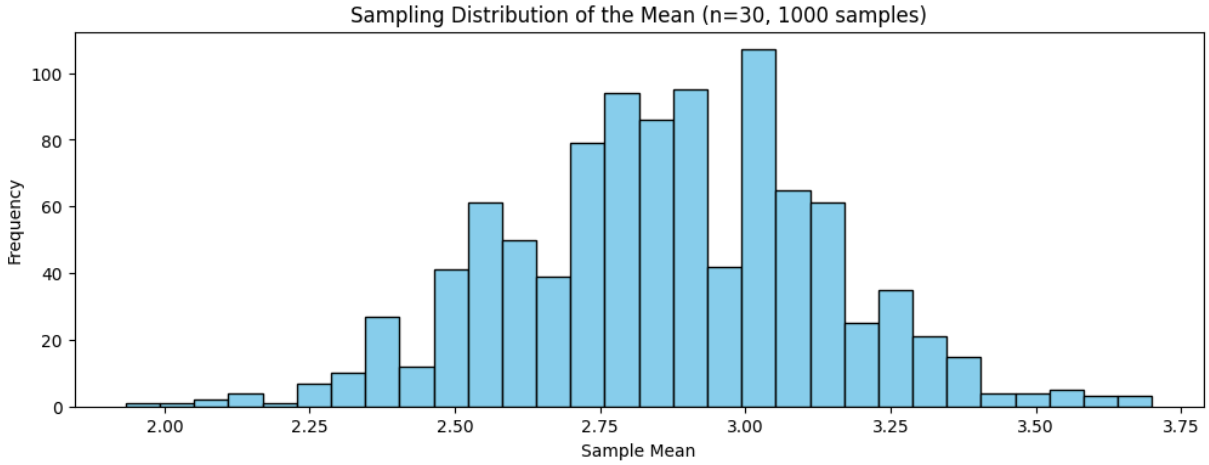
```
sample_means = []

for i in range(1000):
    sample = df['Rating'].sample(n=30, replace=True)
    sample_means.append(sample.mean())

sample_means = np.array(sample_means)
```

In [82]:

```
plt.figure(figsize=(12,4))
plt.hist(sample_means, bins=30, color='skyblue', edgecolor='black')
plt.title('Sampling Distribution of the Mean (n=30, 1000 samples)')
plt.xlabel('Sample Mean')
plt.ylabel('Frequency')
plt.show()
```



In [ ]:

In [ ]:

In [ ]: