Electric bikes of rental service detailed analysis

First, we will load the data file (bikes.data) and import some relevant libraries. We will use different libraries, such as NumPy and SciPy Pandas, Matplotlib, Seaborn, and Ststatsmodels throughout the analysis process. We will import more libraries later. We will load the file with a default integer index because the first column contains categorical variables with three subcategories.

In [53]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as SA
import statsmodels.stats as SS
import seaborn as sns
df=pd.read_csv("bikes.data")
```

In [54]:
```
# Quickly view the structure and values of the dataset from the first fiv
df.head(5)
```

Out[54]:

	ticket	cost	month	location_from	location_to	duration	distance	ass
0	single	0.35	9	MICROTEKNIA	PUIJONLAAKSO	411.0	2150	
1	single	1.20	5	SATAMA	KEILANKANTA	1411.0	7130	
2	savonia	0.00	9	TASAVALLANKATU	NEULAMÄKI	1308.0	5420	
3	savonia	0.00	10	TORI	KAUPPAKATU	1036.0	1180	
4	single	0.30	9	TORI	TORI	319.0	1120	

In [55]:
```
# Find some basic information about the Rows and Columns of the dataset
print(f"Number of Observations (Rows): {len(df)}")
print(f"Number of Observations (columns): {len(df.columns)}"  "\n\n")
print("Data types of each column:")
print(df.dtypes)
```

```
Number of Observations (Rows): 1774
Number of Observations (columns): 10


Data types of each column:
ticket               object
cost                float64
month                 int64
location_from        object
location_to          object
duration            float64
distance              int64
assistance            int64
energy_used         float64
energy_collected    float64
dtype: object
```

```
In [56]: #Checking variables
         # We want to find out all variables of the dataset so that we can compare
         print("Total number of variables:", df.shape[1])
         print("List of all variables:")
         print(list(df.columns))
```

```
Total number of variables: 10
List of all variables:
['ticket', 'cost', 'month', 'location_from', 'location_to', 'duration', 'd
istance', 'assistance', 'energy_used', 'energy_collected']
```

Findings and conclusion:

All the found variables match the variables given in the documentation. At a glance, it is visible that there are 10 variables, which contain objects, float64, and int64. The dataset has 1774 rows and 10 columns.

Task

Use both visualisations and numerical summaries to explore the variables. Filter out irrelevant records and process invalid values, if any. Explain why the records are irrelevant or the values invalid. Also explain how the modifications you made to the data fix theproblems.

Data Cleaning

To get accurate visualizations and numerical summaries we would first perform the data cleaning step which includes handling missing values and invalid values, dealing with outliers, converting data type, and creating condition for filter out fake trips.

```
In [57]: # Finding Missing Values
         print("Total missing values in the DataFrame:")
         print(df.isna().sum())
```

```
Total missing values in the DataFrame:
ticket              0
cost                0
month               0
location_from       0
location_to         0
duration            0
distance            0
assistance          0
energy_used         0
energy_collected    0
dtype: int64
```

As none of the columns contain any missing values, we don't need to take further action for handling missing value.

```
In [58]: # Checking if there is any negative value in the cplumn of "cost", "dista
         columns_check = ["cost", "distance", "energy_used", "energy_collected"]
         for column in columns_check:
             has_negative_values = (df[column] < 0).any()
             if has_negative_values:
```

```
        print(f"The column '{column}' contains negative values.")
    else:
        pass
```

```
The column 'distance' contains negative values.
```

In [59]:
```
# Find the the number of negative items in the distance column
num_negative_distance = len(df[df['distance'] < 0])
num_negative_distance
```

Out[59]:
```
39
```

In total there are 39 negative values in the distance columns. However, the maintenance team reported that they had quite a few technical problems with the bikes. Which might have caused to recorad of the GPS distance as negative value. Therefore instead of dropping the negative distance I simply converted the negative distance to positive value.

In [60]:
```
# Ignoring the negative values we are converting the negatives into absol
df['distance'] = df['distance'].abs()
```

In [61]:
```
# We can check the total repeated rows
resemble_count = df.duplicated(keep='last').sum()
resemble_count
```

Out[61]:
```
12
```

There are 12 rows that has duplicate value. As all of the trips are considered particular trip and only 12 trips are repeated, we can assume these repetitions happened coincidently. So repeated values are included in the data set and we will not drop the duplicate rows.

In [62]:
```
#Filter out irrelevant records
# It can be assumed that the trip records that have zero values in all th
df = df[~((df['cost'] == 0) & (df['distance'] == 0) & (df['energy_used']
```

Filter out irrelevant records: As stated in the description, the data may have contained irrelevant records of customers who only tried out the app and canceled the trips without actually riding the bike. To determine these unreal or canceled trips, we can assume that the trip records that have zero values in all the columns of cost, distance, and energy_used are canceled trips. We can simply drop these records.

In [63]:
```
#Finding order of magnitude errors in the duration column
#Finding potential erroneous rows considering the value of duration.
mean_duration = df['duration'].mean()
std_duration = df['duration'].std()

threshold = 4 * std_duration

duration_errors = df[abs(df['duration'] - mean_duration) > threshold]
print("Potential Duration Errors:")
duration_errors
```

```
Potential Duration Errors:
```

Out[63]:

	ticket	cost	month	location_from	location_to	duration	distance
100	savonia	100.0	9	NEULAMÄKI	MICROTEKNIA	25614.0	2310
609	savonia	100.0	9	NEULAMÄKI	MICROTEKNIA	25614.0	2310
1231	single	100.0	6	TASAVALLANKATU	PIRTTI	22793.0	14690
1501	single	4.8	6	KAUPPAKATU	KAUPPAKATU	5722.0	3150

After considering the potential duration error with the veriable cost it seems that these rows are acceptable as the value of cost is also very high where the the value of duaration is high. Again the maximum duration is 25614 second which is equivalent to 7.115 hour that is a feasible travel duration.

In [94]:
```
#Finding potential erroneous rows considering the value of distance.
mean_distance = df['distance'].mean()
std_distance = df['distance'].std()
threshold = 4 * std_distance

distance_errors = df[abs(df['distance'] - mean_distance) > threshold]
print("Potential Distance Errors:")
distance_errors.head(5)
```

```
Potential Distance Errors:
```

Out[94]:

	ticket	cost	month	location_from	location_to	duration	distance	ass
432	single	2.00	August	KAUPPAKATU	KAUPPAKATU	2391.0	12200	
555	single	3.65	August	KEILANKANTA	KAUPPAKATU	4345.0	14240	
705	single	3.55	June	PUIJONLAAKSO	PUIJONLAAKSO	4207.0	17410	
1097	single	3.00	July	KAUPPAKATU	KAUPPAKATU	3545.0	13650	
1202	single	4.10	August	NEULAMÄKI	NEULAMÄKI	4911.0	20770	

In [65]:
```
# For a better analysis of the potential distance errors, we should also
print("Mean distance:", df['distance'].mean())
print("Mean cost:", df['cost'].mean())
print("Mean energy_used:", df['energy_used'].mean())
```

```
Mean distance: 2649.438744719372
Mean cost: 0.5259203379601689
Mean energy_used: 18.628062764031384
```

The distance values of the potential distance error columns are very high; however, it is clearly visible that the energy_used value is also high along with these high distance values. The overall mean of energy_used is 18.62, but in these rows, energy_used values are several times higher than the mean energy_used value. Therefore, we can assume these high distance values are vallid rows, and it is not necessary to drop these potential error distance values.

In [66]:
```
print(f"Number of Observations (Rows): {len(df)}")
print(f"Number of Observations (columns): {len(df.columns)}"  "\n\n")
```

```
Number of Observations (Rows): 1657
Number of Observations (columns): 10
```

In [67]:
```
# Find out unique months
print("Months:", df["month"].unique(), "\n\n")
```

```
Months: [ 9  5 10  6  8  7  4]
```

In [68]:
```
# For better visualization we can convert the numerical month values to m
import calendar
df['month'] = df['month'].map(lambda x: calendar.month_name[x])
print("Months:", df["month"].unique(), "\n\n")
```

```
Months: ['September' 'May' 'October' 'June' 'August' 'July' 'April']
```

In [69]:
```
#For better visualization we can arrange the month's name in ascending or
ddf = df.sort_values(by='month', ascending=True)
month_order = [ 'April', 'May', 'June', 'July', 'August', 'September', 'O
```

Findings and Conclusion

Although the dataset does not contain any missing values, there were some erroneous negative values in the distance column. We have converted those negative values into positive ones. We examined the repeated values, but the quantity of the repeated values is not very high, so we kept the repeated values. However, we filtered out 177 rows of irrelevent records, as those records were not real trips. We examined the magnitude error and found some magnitude error in the distance and duration columns, but we kept the records because they are quite realistic while judging from the perspective of other variables. After performing the data cleaning task, the number of rows is now 1657, so in total, we dropped 117 rows that contain invalid data processing.

Numerical and visualization summaries to explore the variables

It is necessary to find numerical and visualization summaries to get statistical information about different variables of the dataset. It will help to understand general patterns of the dataset.

In [70]:
```
#We can quickly view  the distribution and basic statistics of numerical
print("Statistical summaries of numerical variables:")
print(df.describe())
```

```
Statistical summaries of numerical variables:
              cost      duration      distance   assistance  energy_used 
\
count  1657.000000   1657.000000   1657.000000  1657.000000  1657.000000   
mean      0.525920    711.840072   2649.438745     0.920338    18.628063   
std       4.268521   1169.978524   2320.520668     0.270851    17.135873   
min       0.000000      2.000000      0.000000     0.000000     0.000000   
25%       0.000000    300.000000   1000.000000     1.000000     4.600000   
50%       0.100000    590.000000   2190.000000     1.000000    15.800000   
75%       0.550000    904.000000   3870.000000     1.000000    28.000000   
max     100.000000  25614.000000  20770.000000     1.000000   144.900000   

       energy_collected  
count       1657.000000  
mean           6.037477  
std            6.416322  
min            0.000000  
25%            1.200000  
50%            4.500000  
75%            8.700000  
max           56.400000  
```

Comment: From this summary, it is evident that the distance column no longer has the negative value, and prior data cleaning helps to generate accurate summary.

We will explore all categorical variables in different ways, first find all the unique subcategories to understand different variables

In [71]:
```python
# Finding the details of unique values  of the catagorical variable
print("Ticket types:", df["ticket"].unique(), "\n\n")
print("Location_from:", df["location_from"].unique(), "\n\n")
print("Location_to:", df["location_to"].unique(), "\n\n")
print("Assistance:", df["assistance"].unique(), "\n\n")
print("Months:", df["month"].unique(), "\n\n")
```

```
Ticket types: ['single' 'savonia' 'season'] 


Location_from: ['MICROTEKNIA' 'SATAMA' 'TASAVALLANKATU' 'TORI' 'NEULAMÄKI'
'KEILANKANTA'
 'PUIJONLAAKSO' 'KAUPPAKATU' 'KYS' 'PIRTTI'] 


Location_to: ['PUIJONLAAKSO' 'KEILANKANTA' 'NEULAMÄKI' 'KAUPPAKATU' 'TORI'
 'TASAVALLANKATU' 'MICROTEKNIA' 'SATAMA' 'PIRTTI' 'KYS'] 


Assistance: [1 0] 


Months: ['September' 'May' 'October' 'June' 'August' 'July' 'April'] 
```

In [72]:
```python
#In order to quickly view  the distribution of categorical columns find s
print("Statistical summaries of categorical variables:")
print(df.describe(include=['object']))
```

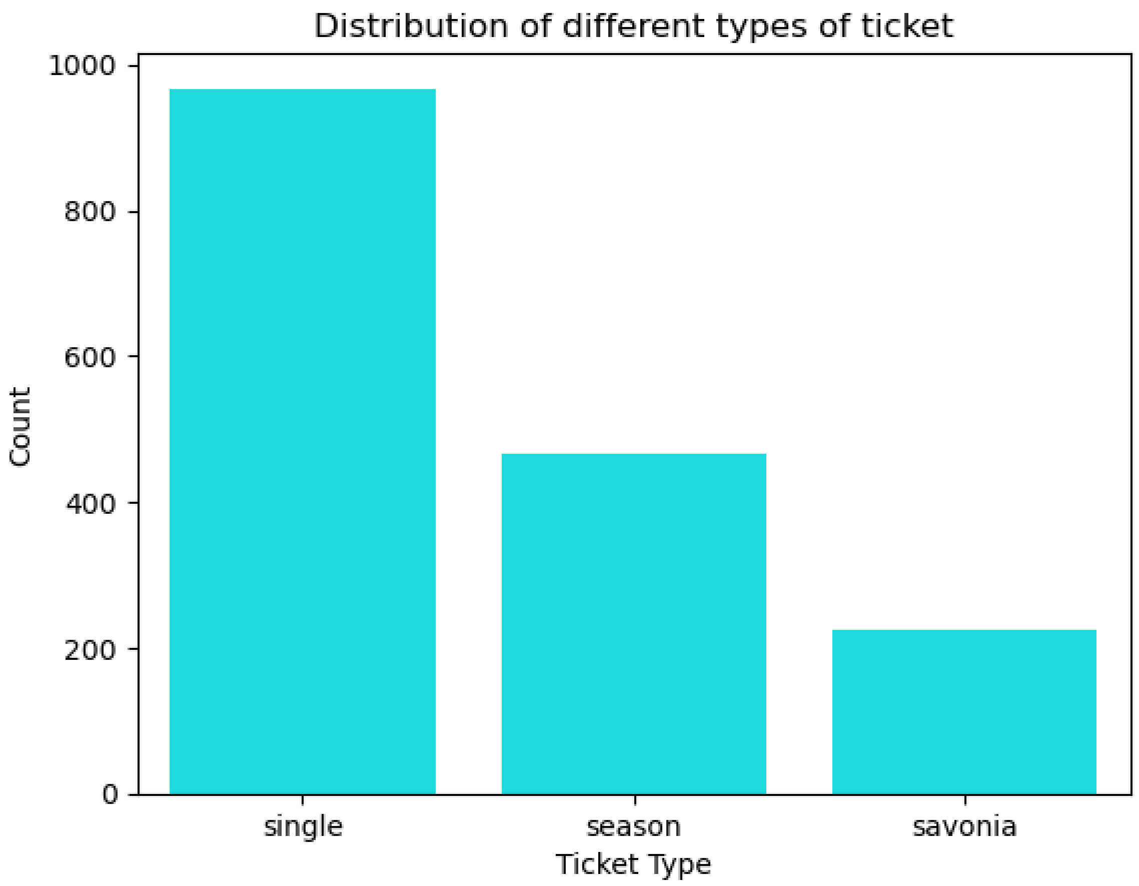
```
Statistical summaries of categorical variables:
        ticket month location_from location_to
count     1657  1657          1657        1657
unique       3     7            10          10
top     single  July          TORI        TORI
freq       967   332           383         368
```

Findings: From the statistical summaries, it is visible that the busiest station is TORI and in the month of July, the highest number of customers used the bike-sharing service. In the given dataset, the information is only available from April to September. The average distance of the trip was 2.649 km, and the average cost was 0.525920 euros. To fully understand the dataset, additional investigation is necessary.

Using visualisations to explore the variables: We will use the Seaborn library for making statistical graphics of different variables. First, we will create bar charts to view the distribution of different tickets, the distribution of bikes departing from different stations, and the distribution of bikes heading towards different stations.

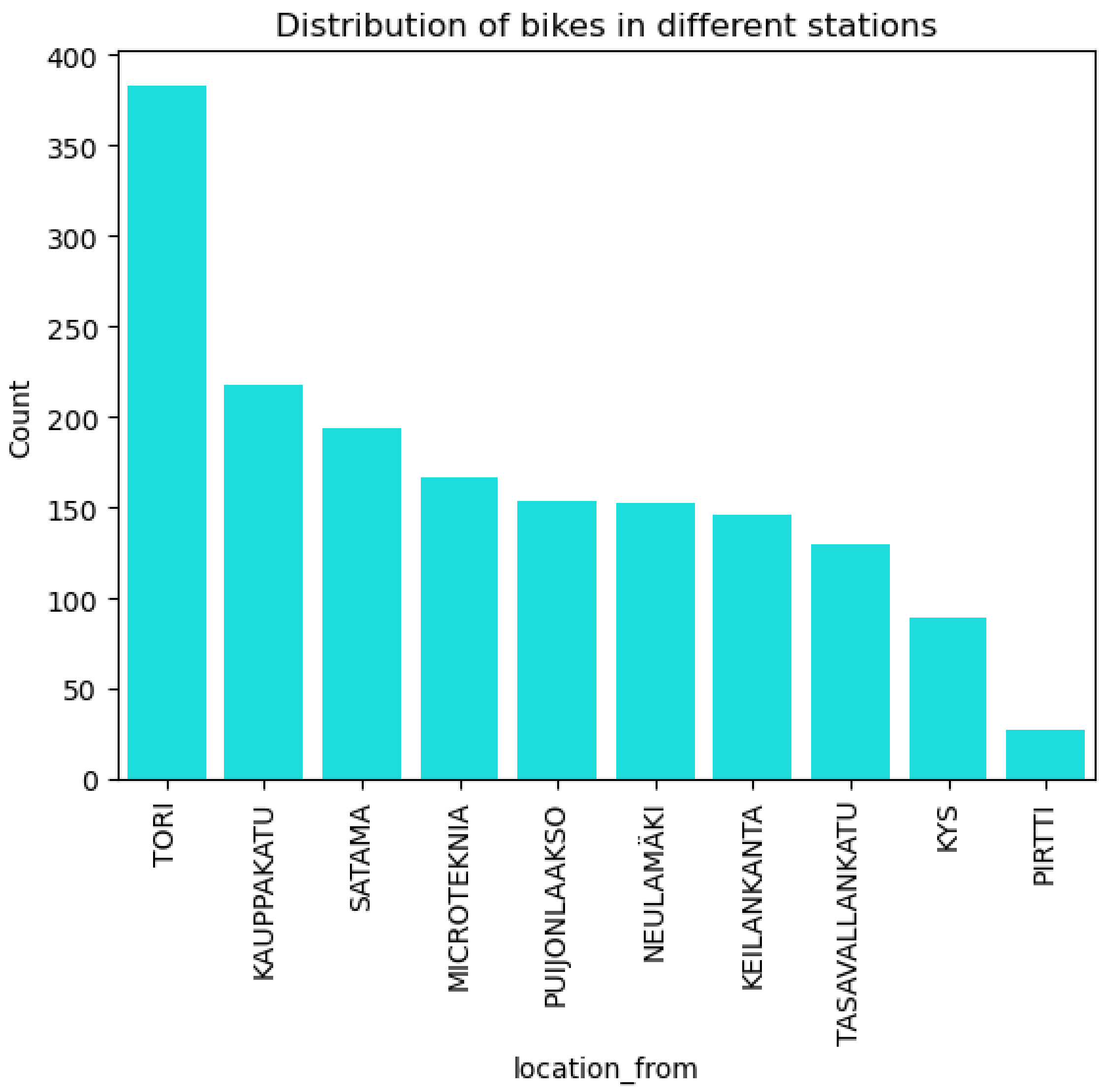
In [73]:
```
# import relevant libraries
import seaborn as sns
import matplotlib.pyplot as plt
```

In [74]:
```
# Plot distribution of different types of ticket
ticket_counts = df['ticket'].value_counts()
sns.barplot(x=df['ticket'].value_counts().index, y=df['ticket'].value_cou
plt.title('Distribution of different types of ticket')
plt.xlabel('Ticket Type')
plt.ylabel('Count')

plt.xticks(rotation=0)
plt.show()
```



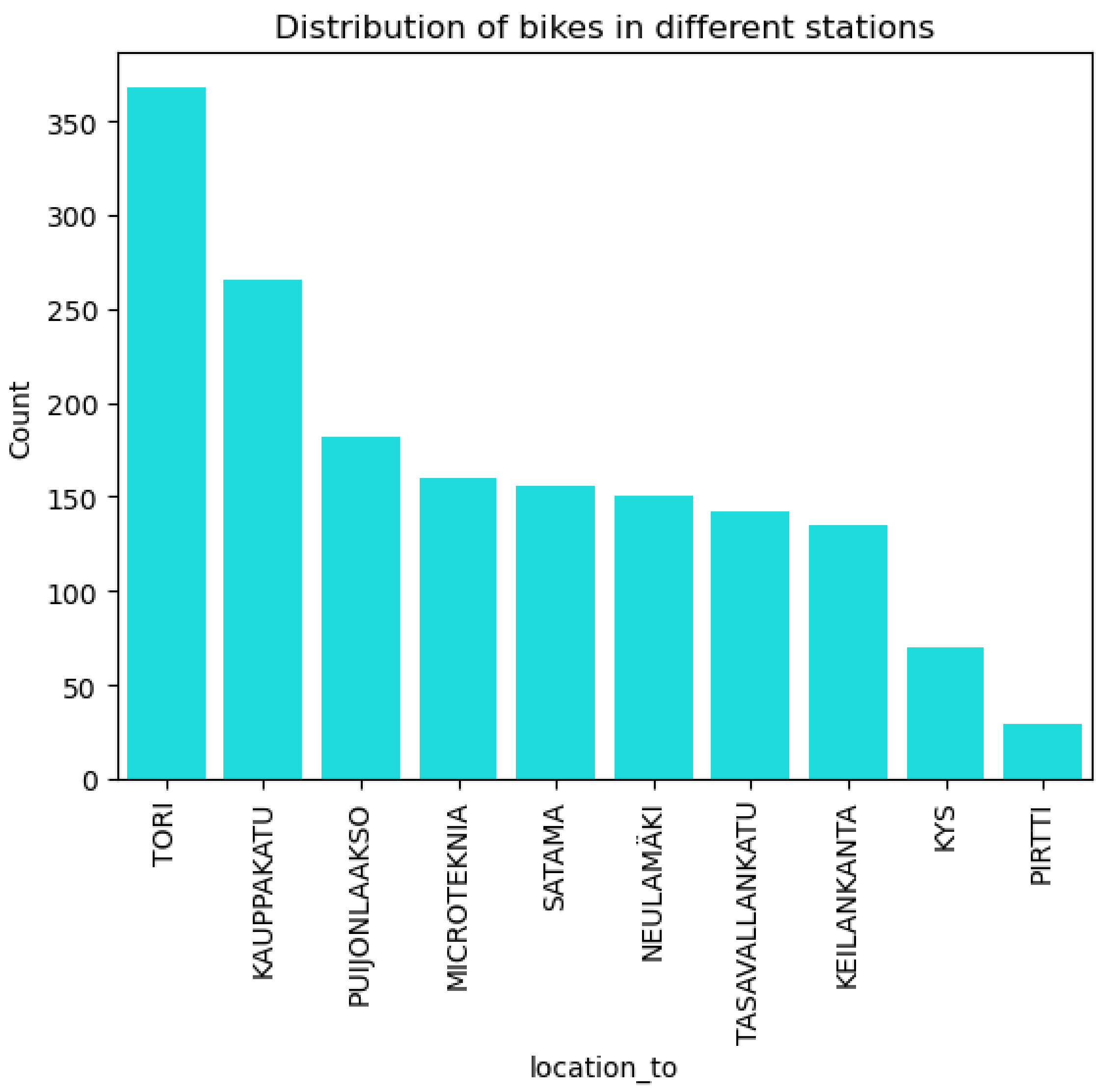
Comments: It is visible that the single type of tickets are used most and savonia tickets were less popular.

In [75]:
```
# Plot distribution of bikes in different departure stations

sns.barplot(x=df['location_from'].value_counts().index, y=df['location_fr
plt.title('Distribution of bikes in different stations')
plt.xlabel('location_from')
plt.ylabel('Count')

plt.xticks(rotation=90)
plt.show()
```



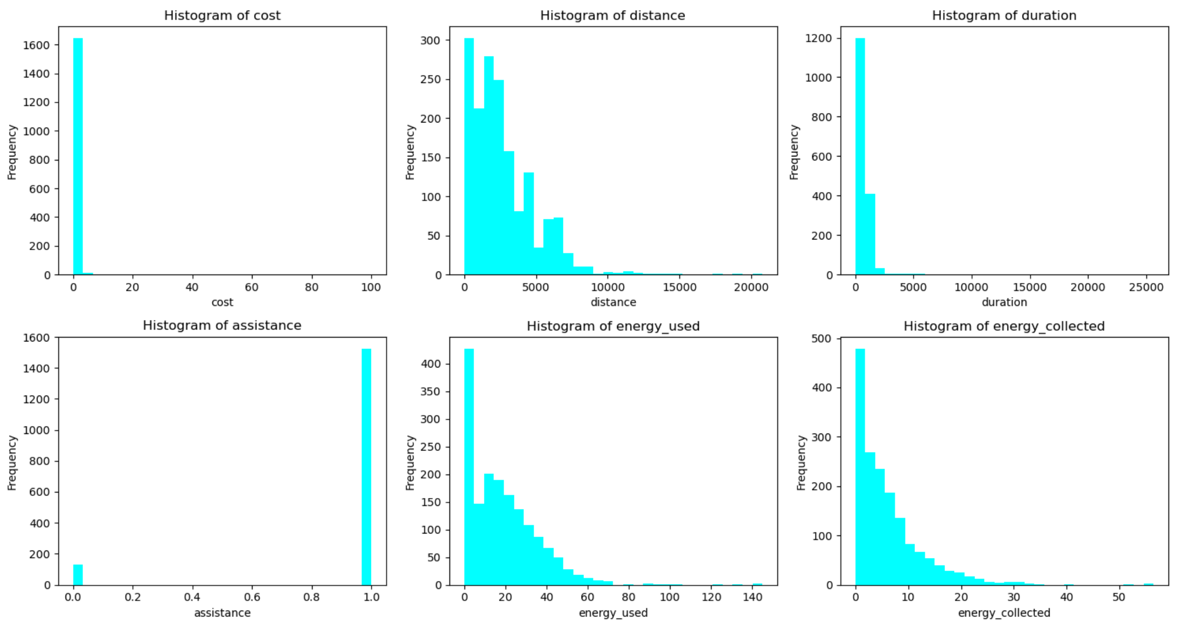
Comments: It is visible that the highest number of bikes departed from station TORI and the least number of bikes departed from station PIRTTI. From the stations, MICROTEKNIA, PUIJONLAAKSO, NEULAMÄKI, and KEILANKANTA almost same number of bikes departed to other stations.

In [76]:
```
# Plot distribution of bikes in different stations
ticket_counts = df['location_to'].value_counts()
sns.barplot(x=df['location_to'].value_counts().index, y=df['location_to']
plt.title('Distribution of bikes in different stations')
plt.xlabel('location_to')
plt.ylabel('Count')

plt.xticks(rotation=90)
plt.show()
```



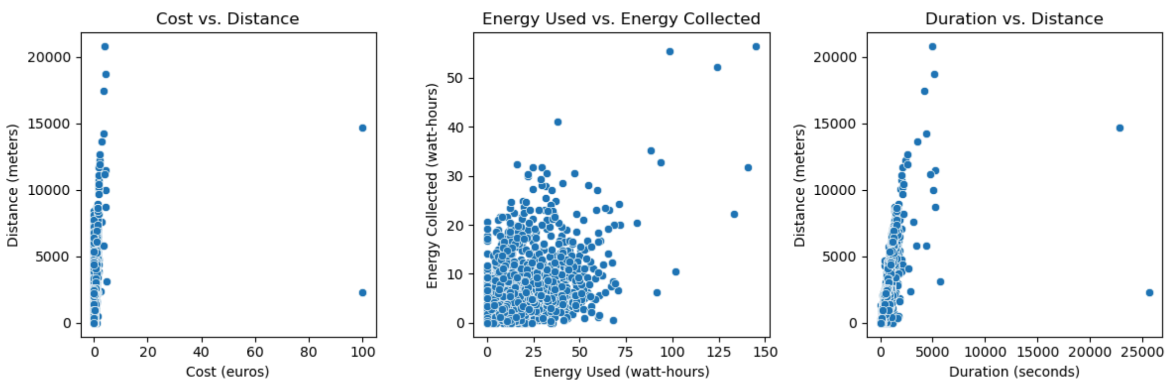
Comments: It is visible that the highest number of bikes heading towards station TORI and less amount of bikes heading towards station PIRTTI. From the stations, MICROTEKNIA, PUIJONLAAKSO, NEULAMÄKI, and KEILANKANTA almost same number of bikes heading towards other stations.

In [77]:
```
# By using Histogram We can visualize the distribution of variables (cost

numerical_columns = ['cost', 'distance', 'duration', 'assistance', 'energ
plt.figure(figsize=(15, 8))

for idx, column in enumerate(numerical_columns, start=1):
    plt.subplot(2, 3, idx)
    plt.hist(df[column], bins=30, color='cyan')
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')

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



Comments: It is visible that most of the cost of tickets was between 0 to 4 euros; however, there are a few tickets whose price was very high. The distance frequency histogram shows that the number of longest distance trips is lower than the number of shortest distance trips. The assistance frequency histogram indicates that customers used assistance on most of the trips. The energy used frequency histogram indicates that in the majority of the trips, the value of the total energy used was o to 20, and the same kind of frequency distribution is also applicable for the energy_collected frequency histogram.

In [78]:
```
# We can use Scatter plot for visiualize the duration vs distance, cost v

fig, axes = plt.subplots(nrows=1, ncols=3, figsize= (12, 4))
fig.subplots_adjust(wspace=0.3)

sns.scatterplot(data=df, x='cost', y='distance', ax=axes[0])
axes[0].set_title('Cost vs. Distance')
axes[0].set_xlabel('Cost (euros)')
axes[0].set_ylabel('Distance (meters)')

sns.scatterplot(data=df, x='energy_used', y='energy_collected', ax=axes[1
axes[1].set_title('Energy Used vs. Energy Collected')
axes[1].set_xlabel('Energy Used (watt-hours)')
axes[1].set_ylabel('Energy Collected (watt-hours)')

sns.scatterplot(data=df, x='duration', y='distance', ax=axes[2])
axes[2].set_title('Duration vs. Distance')
axes[2].set_xlabel('Duration (seconds)')
axes[2].set_ylabel('Distance (meters)')

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



Comment: In the scatter plot of 'Distance vs cost' and the 'Distance vs Duration' most of the dots fall along a straight line, therefore it is evident that there is a strong linear relationship between Cost and Distance as well as Duration and Distance. However, in the scatter plot Energy collected vs Energy Used there is a weak linear relationship between the variables.

Data exploration

Task

For each ticket type, calculate the number of trips made, the total distance travelled, the total time travelled and the total amount of fees paid. Examine your results and make comments about the customers.

In [79]:
```
#We will group the dataframe by ticket type and perform the different agg

grouped_data = df.groupby('ticket')
result = grouped_data.agg({
    'ticket': 'count',              # to find the total number of trips
    'distance': 'sum',              # To find the total distance travelled
    'duration': 'sum',              # To find the total time travelled
    'cost': 'sum'                   # To find the amount of fees paid
})

result.rename(columns={'ticket': 'Number of Trips', 'distance': 'Total Di
                       'duration': 'Total Time', 'cost': 'Total Fees Paid

print("For Each Ticket Type:")
print(result)
```

```
For Each Ticket Type:
         Number of Trips  Total Distance  Total Time  Total Fees Paid
ticket
savonia              225          499880    189776.0           202.00
season               465         1315880    317402.0             3.00
single               967         2574360    672341.0           666.45
```

Result and Comments:

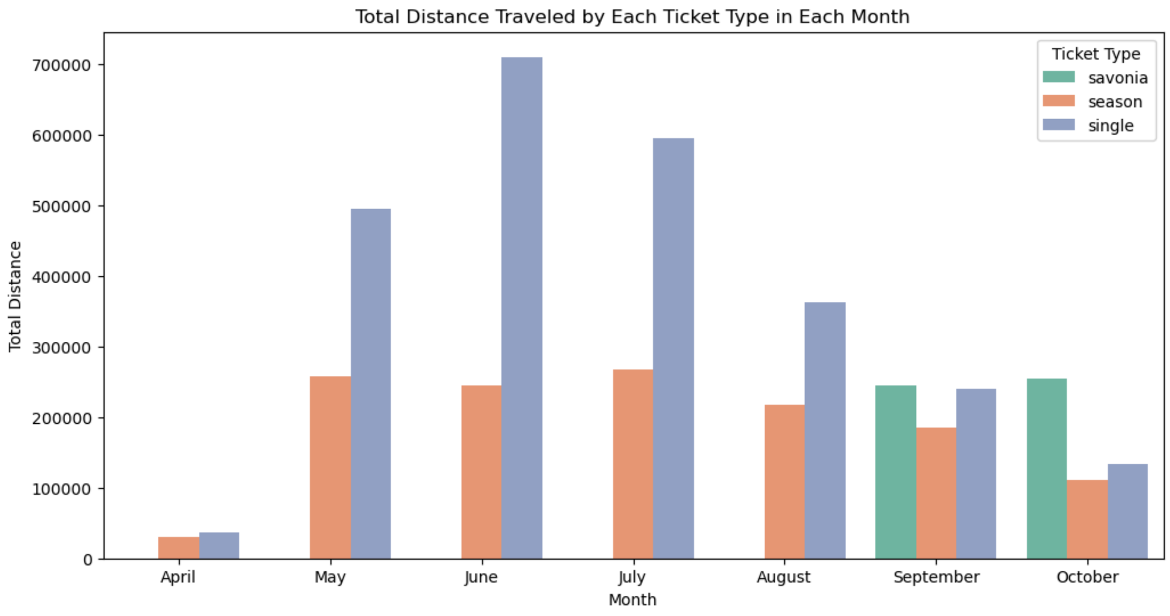
It is evident from the output that customers utilized the single-type ticket more frequently than the other two varieties. The majority of travel is done using a single type ticket, which also covers a larger total distance than a season or savonia type ticket. Customer preference for the single-type ticket is so high. Customers did, however, travel less with the Savonia ticket, and the second most popular ticket is the season ticket. It is unclear why customers paid very low for the season-type ticket. Perhaps there is a policy to pay at a time for the season-type ticket.

Task

For each ticket type, visualise the monthly rental activity in terms of the total distance travelled. Examine your results and makecomments about the development of the rental activity over time.

We will create a bar plot to visualize total distance traveled in each month by each ticket. We will use group data by month and ticket type and calculate the total distance for each group.

In [80]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

df['month'] = pd.Categorical(df['month'], categories=month_order, ordered

# Group data by month and ticket, and calculate total distance traveled
new_variable = df.groupby(['month', 'ticket'])['distance'].sum().reset_in

# Plotting using seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x='month', y='Total Distance', hue='ticket', data=new_variabl
plt.title('Total Distance Traveled by Each Ticket Type in Each Month')
plt.xlabel('Month')
plt.ylabel('Total Distance')
plt.legend(title='Ticket Type', bbox_to_anchor=(1, 1))
plt.show()
```



Comment:

From the bar chart, it is visible that the total distance traveled by the single type ticket is highest in every month except October and in October the total distance traveled by the Savoina type ticket was higher than the other two type ticket. The highest total distance traveled by the single type ticket was in June then in each month the total distance travelled by the single type ticket decreased. The Savonia-type ticket was only used in the last two months of September and October. The total distance traveled by the season-type ticket in the months of May, June, July, August, and September was almost the same.

Task

Find the three stations that have the highest total deficit of bikes (i.e. the largest negative difference between the number of arrived bikes and the number of departured bikes) and the three stations that have the highest total surplus of bikes (i.e. the largest positive difference). Examine your results and make suggestions about how bikes could be relocated.

In [81]:
```
#First we will find the number of arrived bikes and the number of departe
total_arrived_bikes = df['location_to'].value_counts()
total_departed_bikes = df['location_from'].value_counts()

bike_differences = total_arrived_bikes - total_departed_bikes

# Then we will find the three stations that have the highest total difici
print("Three stations that have the highest total deficit of bikes:")
print(bike_differences.nsmallest(3))

# Then we will find the three stations that have the highest total surplu
print("\nThree stations that have the highest total surpluss of bikes:")
print(bike_differences.nlargest(3))
```

```
Three stations that have the highest total deficit of bikes:
SATAMA   -38
KYS      -19
TORI     -15
dtype: int64

Three stations that have the highest total surpluss of bikes:
KAUPPAKATU        47
PUIJONLAAKSO      29
TASAVALLANKATU    13
dtype: int64
```

Result and Suggestion:

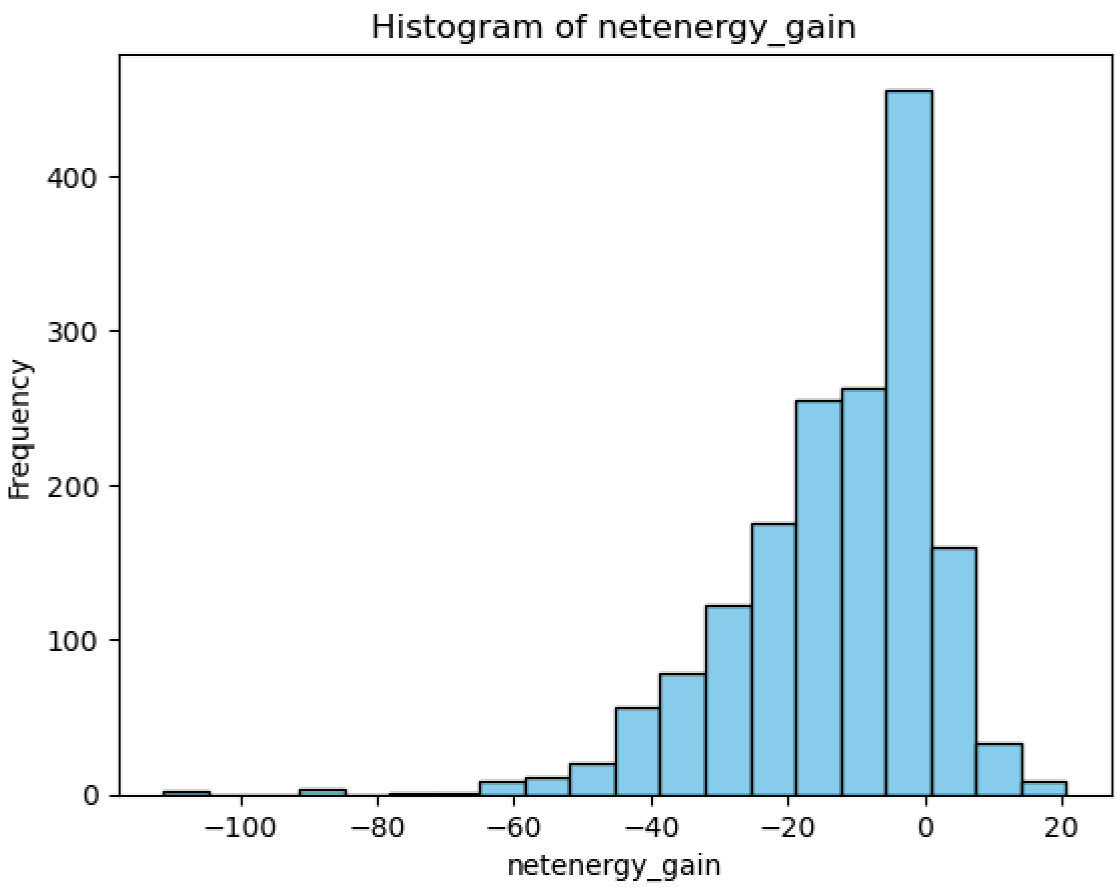
The largest deficit of bikes is visible in the SATAMA station; on the other hand, the KAUPPAKATU station has a good number of surplus bikes. Therefore, we can transfer some bikes from KAUPPAKATU station to SATAMA station. Moving at least 38 bikes from KAUPPAKATU to SATAMA station would be appropriate. The second-largest deficit is at the KYS station. To make up for this deficit, we should move some bikes from PUIJONLAAKSO station to KYS. Even after relocating 20 bikes from the PUIJONLAAKSO station to the KYS station, the PUIJONLAAKSO station will retain 10 surplus bikes; some of these bikes can be moved to the TORI station, and from the surplus bikes at TASAVALLANKATU at least 10 bikes can be relocated to TORI.

Task

For each trip in the data set, calculate its net energy gain, which is defined as the difference between the energy collected and consumed during a trip. Visualise the distribution of this new variable. Make comments about how much battery levels tend to change during trips.

In [82]:
```
# First we will define the new variables as per the given condition and f
df['netenergy_gain'] = df['energy_collected'] - df['energy_used']
```

In [83]:
```
# Now we will use the Histograms plot to visualise the frequency distribu
import matplotlib.pyplot as plt
import seaborn as sns
bins = 20
plt.hist(df['netenergy_gain'], bins=bins, color='skyblue', edgecolor='bla
plt.title('Histogram of netenergy_gain')
plt.xlabel('netenergy_gain')
plt.ylabel('Frequency')
plt.show()
df['netenergy_gain'].describe()
```



Out[83]:
```
count    1657.000000
mean      -12.590585
std        15.126167
min      -111.200000
25%       -20.900000
50%       -10.000000
75%         0.000000
max        20.700000
Name: netenergy_gain, dtype: float64
```

Comments:

Considering the visualization of the frequency distribution of the new variable net energy gain, and from the summary statistics of the net energy gain, it is very evident that the battery levels tend to decrease in most of the trips. In some cases, the battery level change is very high. On average, customers experienced a drop in battery level during the trip. Management should pay attention to increasing the efficacy of the battery so that negative gains can be reduced.

Hypothesis testing

Task

Is there statistical evidence to claim that the travel times tend to be shorter or longer for the single than for the season ticket type. Justify your design choices, interpret the results and use your discoveries to make conclusions about customers.

In [ ]:

Here the Null Hypotheses is that the ticket types has no effect on the travel times or the alternative hypothesis is that the ticket type has effects on the durution of the travel. First try to find out the P value of the duration of single type ticket and the duration of season type ticket so that we can choose the right statistical test.

In [84]:
```
# P value of the duration of single type ticket and the duration of seaso
import scipy.stats as stats
p_value_single = stats.shapiro(df[df['ticket'] == 'single']['duration']).
p_value_season = stats.shapiro(df[df['ticket'] == 'season']['duration']).
print("P-value for 'single' ticket type:", p_value_single)
print("P-value for 'season' ticket type:", p_value_season)
```

```
P-value for 'single' ticket type: 0.0
P-value for 'season' ticket type: 8.494552283799806e-10
```

Due to very low p-values it is very clear that none of the data are not normally distributed therefore I will use Mann-Whitney U test test to compare the durations of single and season tickets as this test does not assume normality.

In [85]:
```
from scipy.stats import mannwhitneyu
duration_single =df[df['ticket'] == 'single']['duration']
duration_season = df[df['ticket'] == 'season']['duration']
stat, p_value = mannwhitneyu(duration_single, duration_season)
stat, p_value
```

Out[85]:
```
(202078.5, 0.001906500076119 57)
```

In [86]:
```
print(f"Mean travel time for single ticket : {duration_single.mean():.2f}
print(f"Mean travel time for season ticket : {duration_season.mean():.2f}
```

```
Mean travel time for single ticket : 695.29 seconds
Mean travel time for season ticket : 682.58 seconds
```

Finding and Conclusion:

The low P_value of Mann–Whitney U test indicate that statistically it is accepted that there is a difference between the duration of travel times of single tickets and the travel times of srason tickets. As a result we assume that the travel times tend to be shorter or longer for the single type ticket than for the season type ticket type and it is statistically valid. For a better insight we can find the mean value of the duration of single type ticket and season type ticket.

From the mean value it is clear that overall, Customers who purchase "single" tickets often have a little longer travel time than those who purchase "season" tickets. Management can use this information for putting more effort to design their services.

Task

Is there statistical evidence to claim that the travel distance positively correlates with the average rate at which electricity is consumed during the trip? Justify your design choices, interpret the results and use your discoveries to make conclusions about customers.

In [87]:
```
# First find the the average rate at which electricity is consumed during
df['electricity_consumption_rate'] = df['energy_used'] / df['distance']
```

In [88]:
```
# As the energy used column have many zero values therefore we should che
df['electricity_consumption_rate'].isna().sum()
```

Out[88]:
```
140
```

In [89]:
```
# we will clean the NaN values by simply droping the rows as we have sitl
df_clean = df.dropna(subset=['electricity_consumption_rate'])
```

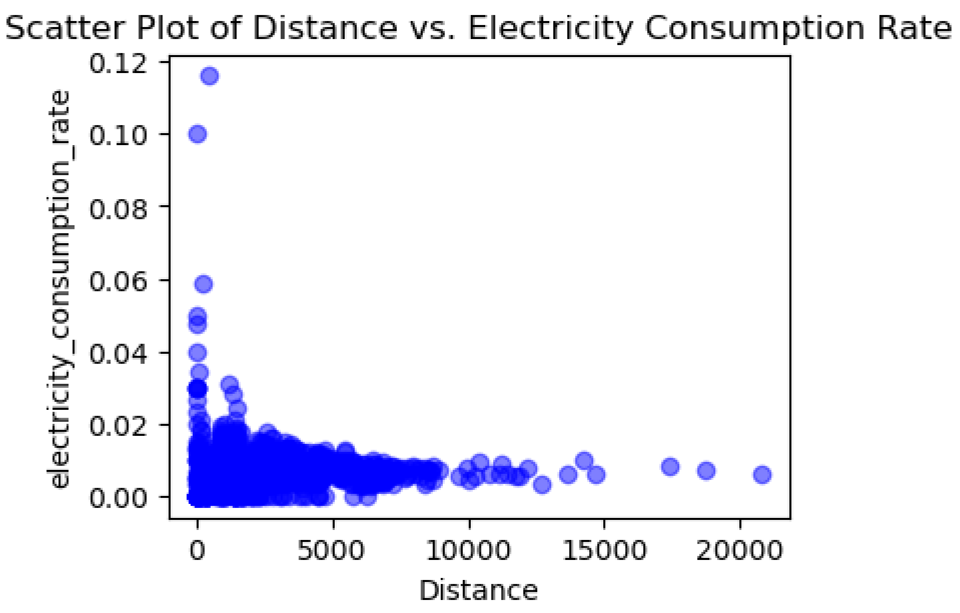
In [90]:
```
# we will use Shapiro-Wilk test to calculate the P value of the two colum
print(" p-value for 'distance' column:", stats.shapiro(df_clean['distance
print(" p-value for 'electricity_consumption_rate' column:", stats.shapir
```

```
 p-value for 'distance' column: ShapiroResult(statistic=0.875376045703887
9, pvalue=3.496672883071994e-33)
 p-value for 'electricity_consumption_rate' column: ShapiroResult(statisti
c=0.6207758188247681, pvalue=0.0)
```

In [91]:
```
# As both of the columns are not normally distributed we will use Spearma
from scipy.stats import spearmanr
correlation_coefficient, p_value = spearmanr(df_clean['distance'], df_cle
correlation_coefficient, p_value
```

Out[91]:
```
(-0.03055834512365383, 0.2342413895173393)
```

In [92]:
```
# For getting more insight we can simply draw a scatter plot of distance

import matplotlib.pyplot as plt
plt.figure(figsize=(4, 3))
plt.scatter(df_clean['distance'], df_clean['electricity_consumption_rate'
plt.title('Scatter Plot of Distance vs. Electricity Consumption Rate')
plt.xlabel('Distance')
plt.ylabel('electricity_consumption_rate')
plt.show()
```



Result Discoveries and Conclusions

As the correlation coefficient is negative and the p-value is higher than 0.05, we can assume there is no correlation between the energy consumption rate and distance. From the scatter plot, we can also assume that there might be distinct patterns or subgroups within trip types, but these subgroups are small as most of the dots are concentrated randomly without following any line.

Customers are more likely do not feel any experience that the energy consumption rate increases or decreases more rapidly or slowly while increasing the distance. However, in certain situations, there might be a correlation between the energy consumption rate and distance according to the plot diagram.

Task

Is there statistical evidence to claim that the savonia ticket type differs from the others with respect to how often the electric assistance is used? Justify your design choices, interpret the results and use your discoveries to make conclusions about customers.

Design Choice

Here, the null hypothesis is that there is no relation between the usage of electric assistance and the ticket types, and the alternative hypothesis is that the ticket type affects the customers usage of electric assistance. As both of the variables are categorical variables, we will use the Chi-square test for calculate the P value.

In [93]:
```
# as the variables ("ticket" type and "electric assistance" status) are c
import pandas as pd
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df['ticket'], df['assistance'])
```

```
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
chi2, p_value
```

```
Out[93]: (24.29650264347632, 5.297628333840502e-06)
```

Result and Conclusion

Here, the p_value is very small and close to zero; therefore, we can assume that there is a strong relationship between ticket type and electric assistance. Therefore, it is statistically likely that customers with savonia ticket types might have a different pattern of using electric assistance than customers with season and single-type tickets.