In [36]:

```
# import libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

%matplotlib inline

sns.set_style('darkgrid')
```

In [37]:

```
# read data set

df = pd.read_csv('Coffee Shop Sales.csv')
df
```

Out[37]:

	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	u
0	1	01/01/2023	07:06:11	2	5	Lower Manhattan	32	
1	2	01/01/2023	07:08:56	2	5	Lower Manhattan	57	
2	3	01/01/2023	07:14:04	2	5	Lower Manhattan	59	
3	4	01/01/2023	07:20:24	1	5	Lower Manhattan	22	
4	5	01/01/2023	07:22:41	2	5	Lower Manhattan	57	
...	...	...	...	...	...	...	...	
149111	149452	30/06/2023	20:18:41	2	8	Hell's Kitchen	44	
149112	149453	30/06/2023	20:25:10	2	8	Hell's Kitchen	49	
149113	149454	30/06/2023	20:31:34	1	8	Hell's Kitchen	45	
149114	149455	30/06/2023	20:57:19	1	8	Hell's Kitchen	40	
149115	149456	30/06/2023	20:57:19	2	8	Hell's Kitchen	64	

149116 rows × 11 columns

In [38]:

```
# see top 5 rows

df.head()
```

Out[38]:

	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_pri
0	1	01/01/2023	07:06:11	2	5	Lower Manhattan	32	3

1	2	01/01/2023	07:08:56	2	5	Lower Manhattan	57	3
2	3	01/01/2023	07:14:04	2	5	Lower Manhattan	59	4
3	4	01/01/2023	07:20:24	1	5	Lower Manhattan	22	2
4	5	01/01/2023	07:22:41	2	5	Lower Manhattan	57	3

In [39]:

```
# see numbers of rows and columns

df.shape
```

Out[39]:

```
(149116, 11)
```

In [40]:

```
# check missing values

df.isna().sum()
```

Out[40]:

```
transaction_id      0
transaction_date    0
transaction_time    0
transaction_qty     0
store_id            0
store_location      0
product_id          0
unit_price          0
product_category    0
product_type        0
product_detail      0
dtype: int64
```

In [41]:

```
# see quick info

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149116 entries, 0 to 149115
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   transaction_id    149116 non-null  int64  
 1   transaction_date  149116 non-null  object 
 2   transaction_time  149116 non-null  object 
 3   transaction_qty   149116 non-null  int64  
 4   store_id          149116 non-null  int64  
 5   store_location    149116 non-null  object 
 6   product_id        149116 non-null  int64  
 7   unit_price        149116 non-null  float64
 8   product_category  149116 non-null  object 
 9   product_type      149116 non-null  object 
 10  product_detail    149116 non-null  object 
dtypes: float64(1), int64(4), object(6)
memory usage: 12.5+ MB
```

In [42]:

```
# check duplicated rows

df.duplicated().any()
```

Out[42]:

```
False
```

In [43]:

```
# see unique values in each column
# 1- creat new data frame with number of unique value in each column
columnValue = df.nunique().reset_index()

# 2- rename column name
columnValue.rename(columns={'index':'Column_name', 0:'Unique values'}, inplace = True)

# 3- see columns and number of unique values of each
columnValue
```

Out[43]:

	Column_name	Unique values
0	transaction_id	149116
1	transaction_date	181
2	transaction_time	25762
3	transaction_qty	6
4	store_id	3
5	store_location	3
6	product_id	80
7	unit_price	41
8	product_category	9
9	product_type	29
10	product_detail	80

In [44]:

```
# See quick info of numeric data

df.describe()
```

Out[44]:

	transaction_id	transaction_qty	store_id	product_id	unit_price
count	149116.000000	149116.000000	149116.000000	149116.000000	149116.000000
mean	74737.371872	1.438276	5.342063	47.918607	3.382219
std	43153.600016	0.542509	2.074241	17.930020	2.658723
min	1.000000	1.000000	3.000000	1.000000	0.800000
25%	37335.750000	1.000000	3.000000	33.000000	2.500000
50%	74727.500000	1.000000	5.000000	47.000000	3.000000
75%	112094.250000	2.000000	8.000000	60.000000	3.750000
max	149456.000000	8.000000	8.000000	87.000000	45.000000

In [45]:

```
# see quick info of categorical data

df.describe(include = object)
```

Out[45]:

	transaction_date	transaction_time	store_location	product_category	product_type	product_detail
count	149116	149116	149116	149116	149116	149116
unique	181	25762	3	9	29	80
top	19/06/2023	09:31:15	Hell's Kitchen	Coffee	Brewed Chai	Chocolate

					tea	Croissant
freq	1343	41	50735	58416	17183	3076

In [46]:
```
# we don't use the id columns in our data ,so i will remove it

df.drop(columns = ["transaction_id", "store_id", "product_id"], inplace = True)
```

In [47]:
```
df.head()
```

Out[47]:

	transaction_date	transaction_time	transaction_qty	store_location	unit_price	product_category	product_type
0	01/01/2023	07:06:11	2	Lower Manhattan	3.0	Coffee	Gourmet brewed coffee
1	01/01/2023	07:08:56	2	Lower Manhattan	3.1	Tea	Brewed Chai tea
2	01/01/2023	07:14:04	2	Lower Manhattan	4.5	Drinking Chocolate	Hot chocolate
3	01/01/2023	07:20:24	1	Lower Manhattan	2.0	Coffee	Drip coffee
4	01/01/2023	07:22:41	2	Lower Manhattan	3.1	Tea	Brewed Chai tea

In [48]:
```
# 1. Convert 'transaction_time' to strings (if necessary)
if not pd.api.types.is_string_dtype(df['transaction_time']):
    df['transaction_time'] = df['transaction_time'].astype(str)
```

In [49]:
```
df['sales'] = df['transaction_qty'] * df['unit_price']
```

In [50]:
```
df['datetime'] = df['transaction_date'] + df['transaction_time']
df.head()
```

Out[50]:

	transaction_date	transaction_time	transaction_qty	store_location	unit_price	product_category	product_type
0	01/01/2023	07:06:11	2	Lower Manhattan	3.0	Coffee	Gourmet brewed coffee
1	01/01/2023	07:08:56	2	Lower Manhattan	3.1	Tea	Brewed Chai tea
2	01/01/2023	07:14:04	2	Lower Manhattan	4.5	Drinking Chocolate	Hot chocolate
3	01/01/2023	07:20:24	1	Lower Manhattan	2.0	Coffee	Drip coffee
4	01/01/2023	07:22:41	2	Lower Manhattan	3.1	Tea	Brewed Chai tea

In [51]:
```
daily_sales_by_location = df.groupby(['transaction_date', 'store_location'])['sales'].su
daily_sales_by_location
```

Out[51]:

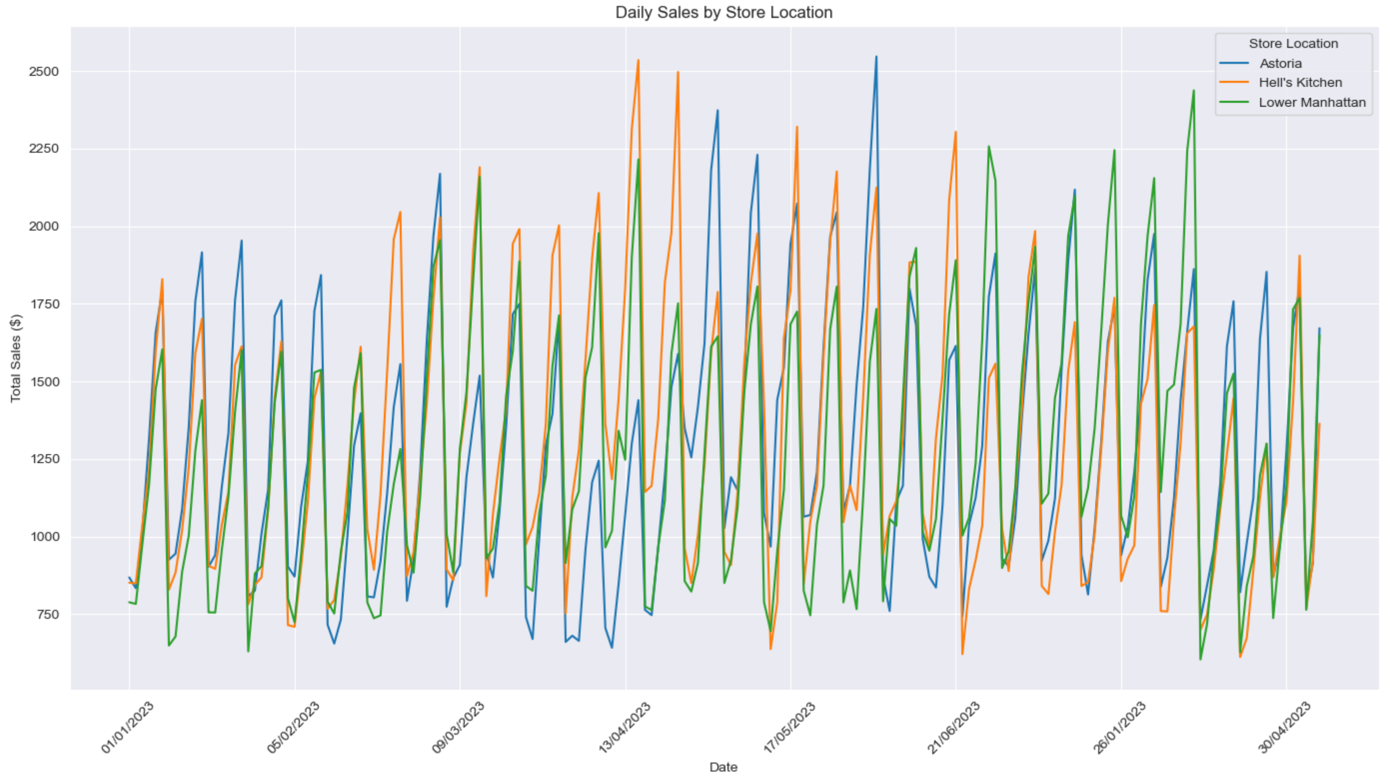
store_location	Astoria	Hell's Kitchen	Lower Manhattan

transaction_date			
01/01/2023	868.40	851.45	788.35
01/02/2023	833.70	849.40	783.20
01/03/2023	1021.10	1040.45	978.70
01/04/2023	1316.50	1215.35	1168.05
01/05/2023	1657.65	1598.40	1475.40
...	...	...	...
30/05/2023	1670.95	1432.50	1732.03
30/06/2023	1807.65	1904.93	1768.74
31/01/2023	801.50	768.40	764.23
31/03/2023	915.15	923.40	1049.53
31/05/2023	1671.11	1363.75	1649.27

181 rows × 3 columns

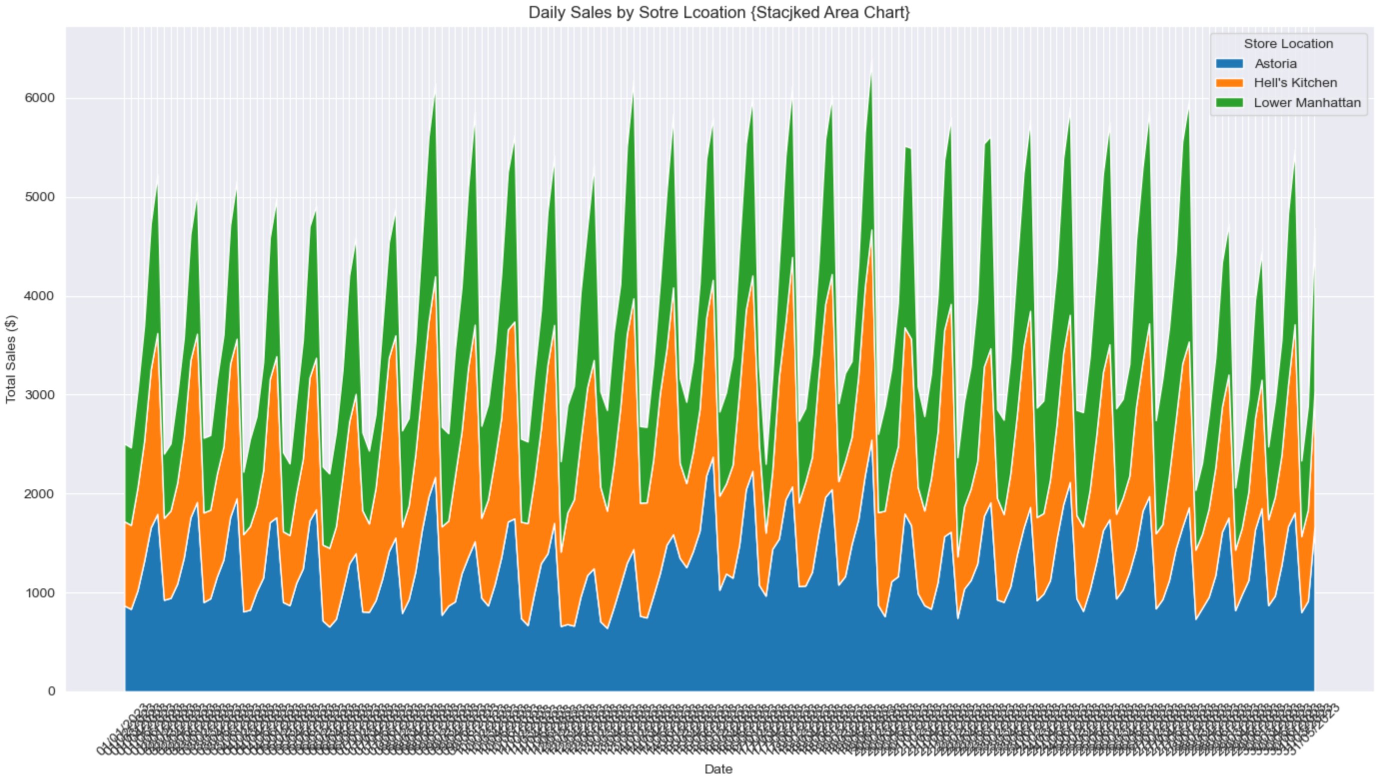
In [52]:

```python
daily_sales_by_location.plot(figsize=(14,8),title='Daily Sales by Store Location')

plt.xlabel('Date')
plt.ylabel('Total Sales ($)')
plt.legend(title='Store Location')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

#show the plot
plt.show()
```



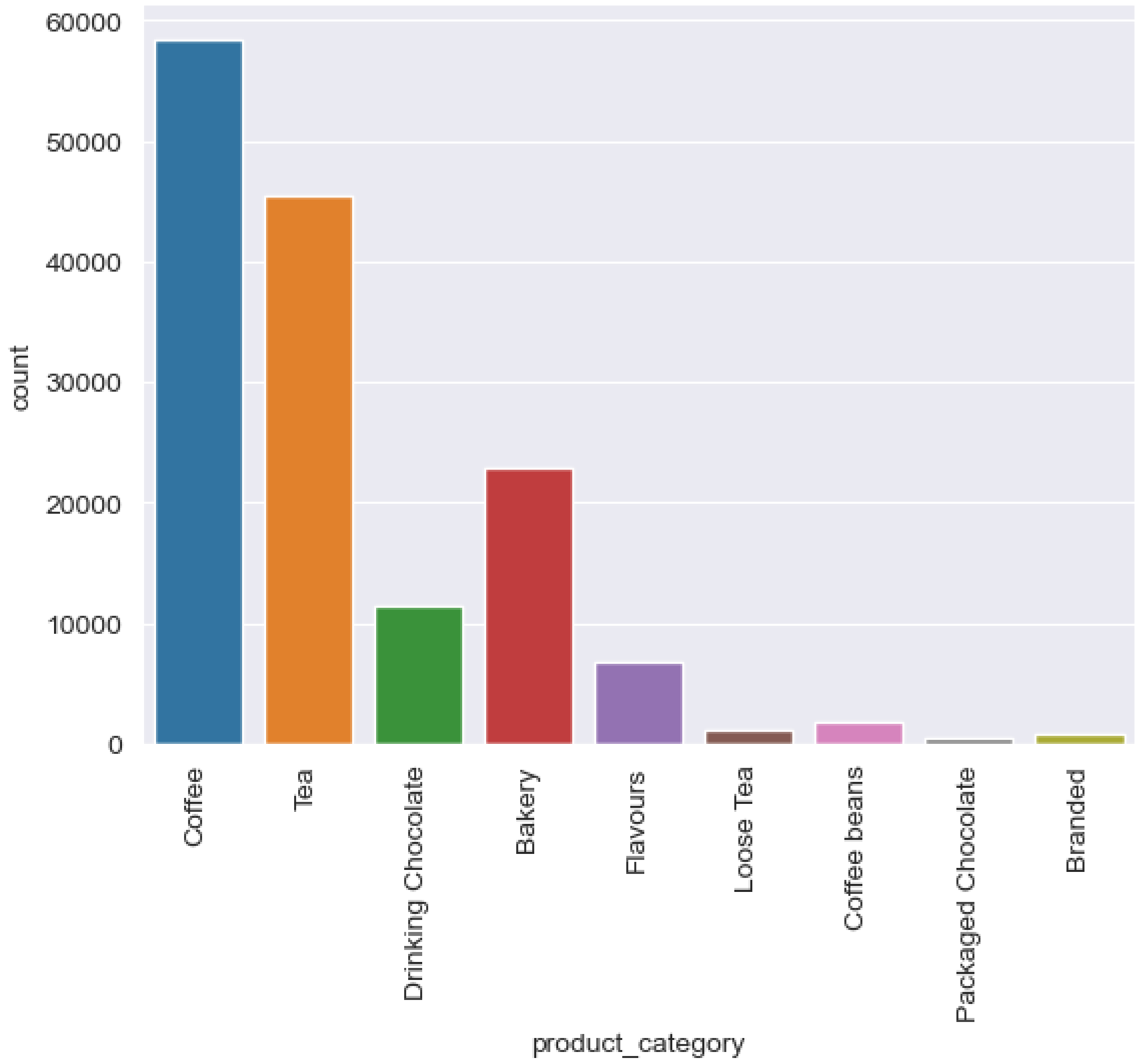
In [53]:

```python
plt.figure(figsize=(14,8))

plt.stackplot(daily_sales_by_location.index,daily_sales_by_location.T,labels=daily_sales
plt.title('Daily Sales by Sotre Lcoation {Stacjked Area Chart}')
plt.xlabel('Date')
plt.ylabel('Total Sales ($)')
plt.legend(title='Store Location')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

#show the plot
plt.show()
```



In [54]:

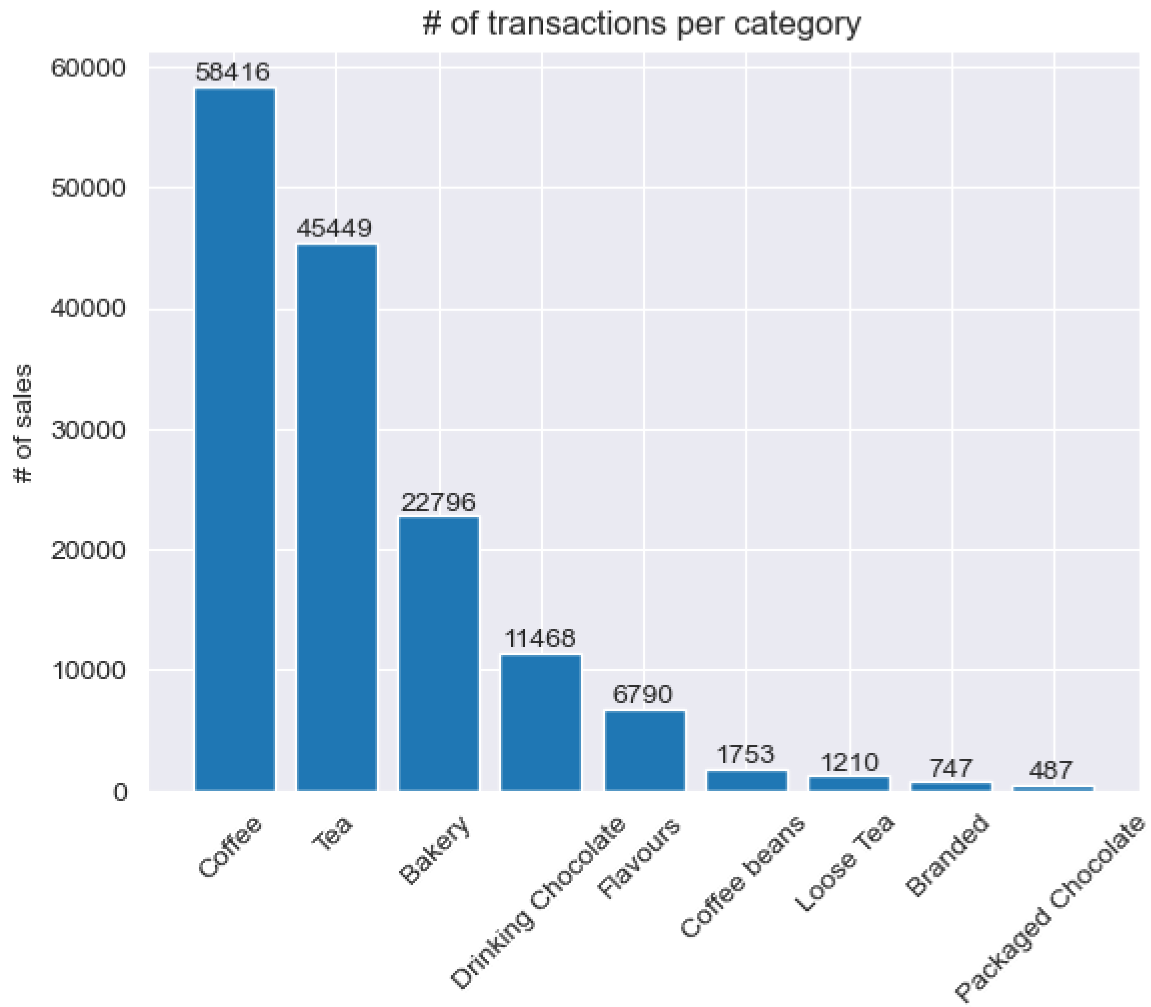
```
# the most wanted category from customers

sns.countplot(x= 'product_category' , data = df )
plt.xticks(rotation = 90)
plt.show()
```



In [55]:

```
# Plot out # of transactions per category

fig, ax = plt.subplots()
value_counts = df['product_category'].value_counts()
bar_container = ax.bar(value_counts.index, value_counts.values)
ax.set(ylabel = '# of sales', title = '# of transactions per category')
ax.set_xticklabels(value_counts.index, rotation = 45)
ax.bar_label(bar_container)
print(value_counts)
```

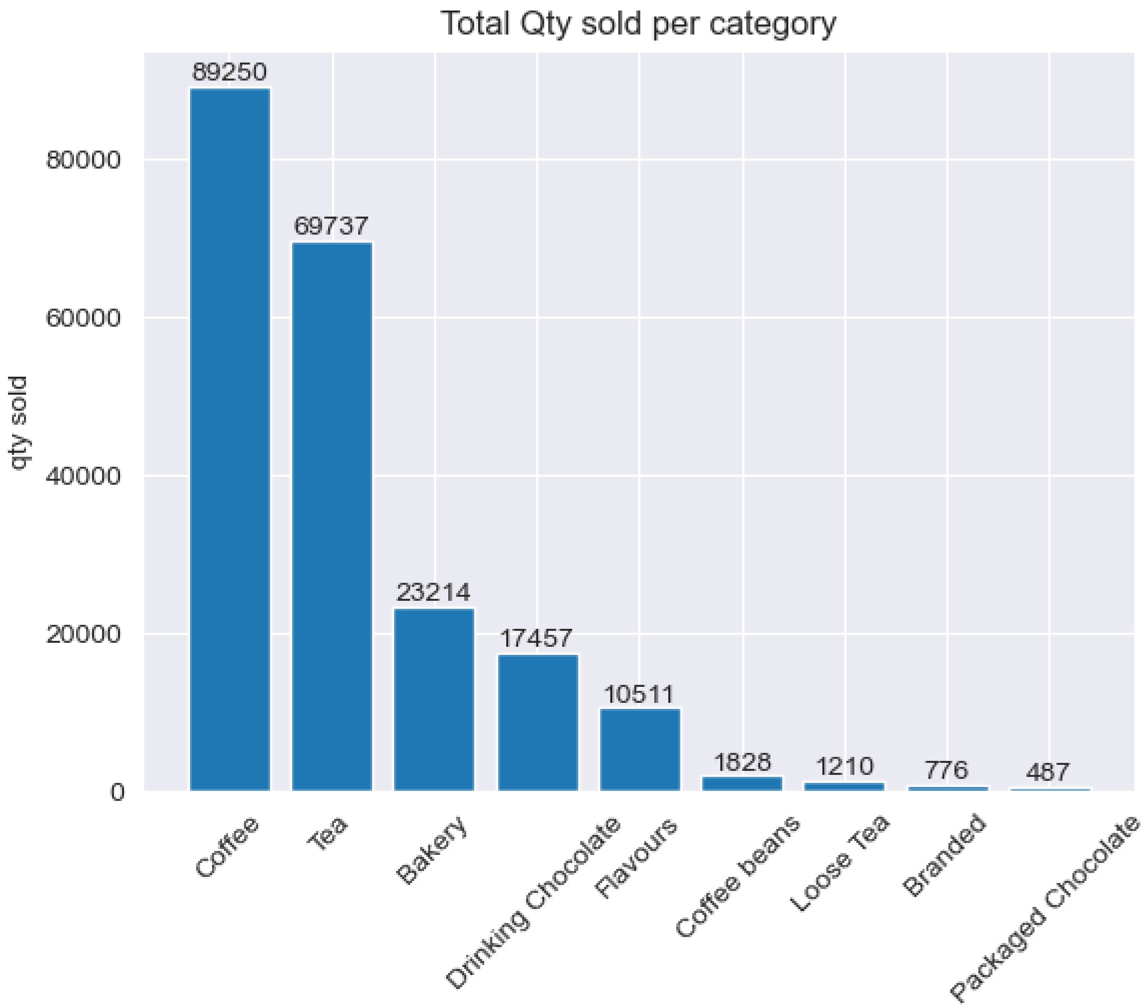
```
C:\Users\ilham_7t2frur\AppData\Local\Temp\ipykernel_20092\1905731253.py:7: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(value_counts.index, rotation = 45)
product_category
Coffee                58416
Tea                   45449
Bakery                22796
Drinking Chocolate    11468
Flavours               6790
Coffee beans           1753
Loose Tea              1210
Branded                 747
Packaged Chocolate      487
Name: count, dtype: int64
```



In [56]:

```
# However, quantities per transaction can vary, so have to add up transaction_qty per ca

fig1, ax1 = plt.subplots()
sum_quantity_counts = df.groupby('product_category')['transaction_qty'].sum()
sum_quantity_counts = sum_quantity_counts.sort_values(ascending= False)
bar_container1 = ax1.bar(sum_quantity_counts.index, sum_quantity_counts.values)
ax1.set(ylabel = 'qty sold', title = 'Total Qty sold per category')
ax1.set_xticklabels(sum_quantity_counts.index, rotation = 45)
ax1.bar_label(bar_container1)
print(sum_quantity_counts)
```

```
C:\Users\ilham_7t2frur\AppData\Local\Temp\ipykernel_20092\2257508552.py:8: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(sum_quantity_counts.index, rotation = 45)
product_category
Coffee                89250
Tea                   69737
Bakery                23214
Drinking Chocolate    17457
Flavours              10511
Coffee beans           1828
Loose Tea              1210
Branded                 776
Packaged Chocolate      487
Name: transaction_qty, dtype: int64
```



In [57]:

```
# What category has contributed the most to the overall revenue?
# Create 'transaction_total' column to calculate total of transaction, multiplies transa

df['transaction_total'] = df['transaction_qty'] * df['unit_price']
df.head()
```

Out[57]:

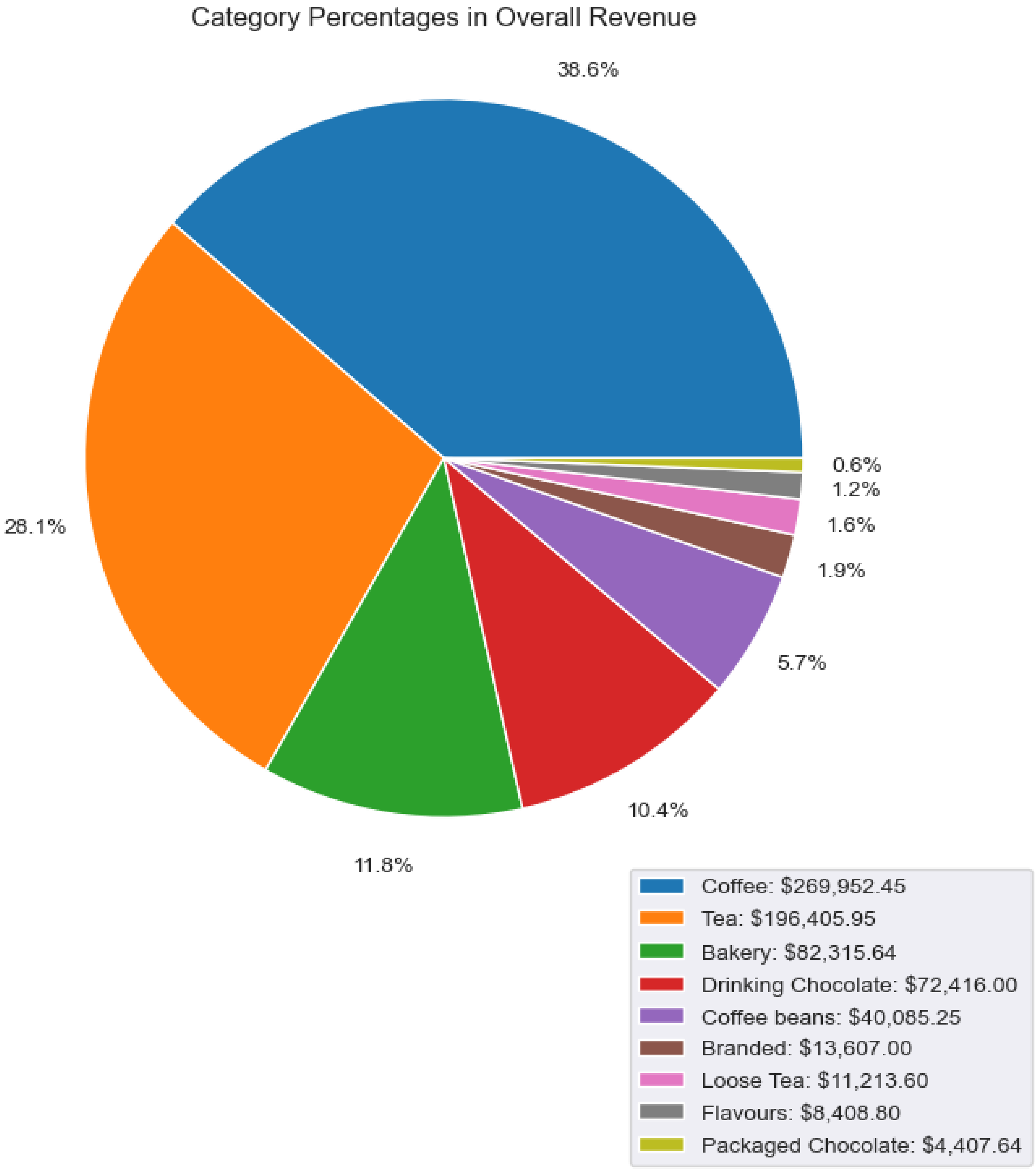
	transaction_date	transaction_time	transaction_qty	store_location	unit_price	product_category	product_type
0	01/01/2023	07:06:11	2	Lower Manhattan	3.0	Coffee	Gourmet brewed coffee
1	01/01/2023	07:08:56	2	Lower Manhattan	3.1	Tea	Brewed Chai tea
2	01/01/2023	07:14:04	2	Lower Manhattan	4.5	Drinking Chocolate	Hot chocolate
3	01/01/2023	07:20:24	1	Lower Manhattan	2.0	Coffee	Drip coffee
4	01/01/2023	07:22:41	2	Lower Manhattan	3.1	Tea	Brewed Chai tea

In [58]:

```
# Use a pie chart to see what category has contributed the most

fig3, ax3 = plt.subplots()

# Find the sums of transaction_total of each category, then sort by descending.
category_sums = df.groupby('product_category')['transaction_total'].sum().\
        sort_values(ascending=False)

# create labels for legends formatted as: CATEGORY: $###,###,###.##
leg_labels = [f"{category}: ${total_sum:,.2f}" for \
              category, total_sum in zip(category_sums.index, category_sums.values)]

# create pie chart, and adjust parameters for clarity
ax3.pie(category_sums.values, autopct= '%1.1f%%', \
        pctdistance=1.15, radius=1.5, labeldistance= 1)

# create legend and place in best area manually
plt.title("Category Percentages in Overall Revenue", y = 1.2)
plt.legend(leg_labels, loc = 'lower right', bbox_to_anchor = (1.5,-.7))
```

Out[58]:

```
<matplotlib.legend.Legend at 0x2ddb112f910>
```



In [ ]: