ARDEN
UNIVERSITY

Awarding Body:
Arden University

Programme Name:
Master of Data Science

Module Name (and Part if applicable):
Visualization

Assessment Title:
Optimizing Profitability of UK Pet Stores through Data Visualization

Student Number:
STU221742

Tutor Name:
Mr. Ali Vaisifard

Word Count:

Please refer to the Word Count Policy on your Module Page for guidance

- **Task 1: Implementing Python and Visualization**

- **TASK 2: Analysing the Performance of implementing and Visualization**

Introduction:

In this project, through the presentation of visual elements and interactive insights using the **Python** programming language and Streamlit dashboard, a report on the net profitability of pet stores across the United Kingdom is provided to the Board of Directors. This study focuses on analysing the relationships between key factors such as **revenue**, **expenses**, **store location**, and the **type of animals** sold, in order to calculate the **profitability** of each store. The purpose of this analysis is to offer strategies for reducing costs and increasing revenue, as well as recommendations for modifying or potentially closing down certain branches to minimize losses and improve overall performance. In this study, special attention has been paid to their visualization and its importance in data analysis (Healy, 2018).

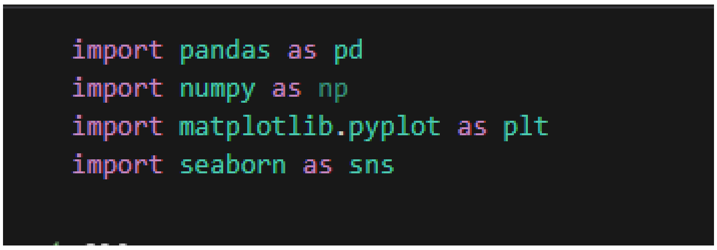
Task 1: Implementation, and visualization based on Python and Streamlit interactive dashboard

Feature Exploration:

Managers First Name:	**string & Nominal**
Managers Surname:	string & Nominal
Area:	string & Nominal
Pet:	string & Nominal
Units Sld:	float & Discrete
Revenue:	float & Continuous
Cost:	float & Continuous
Profit:	float & Continuous Target variable
Date:	Date & Time

Table 1- Features

Importing Libraries:



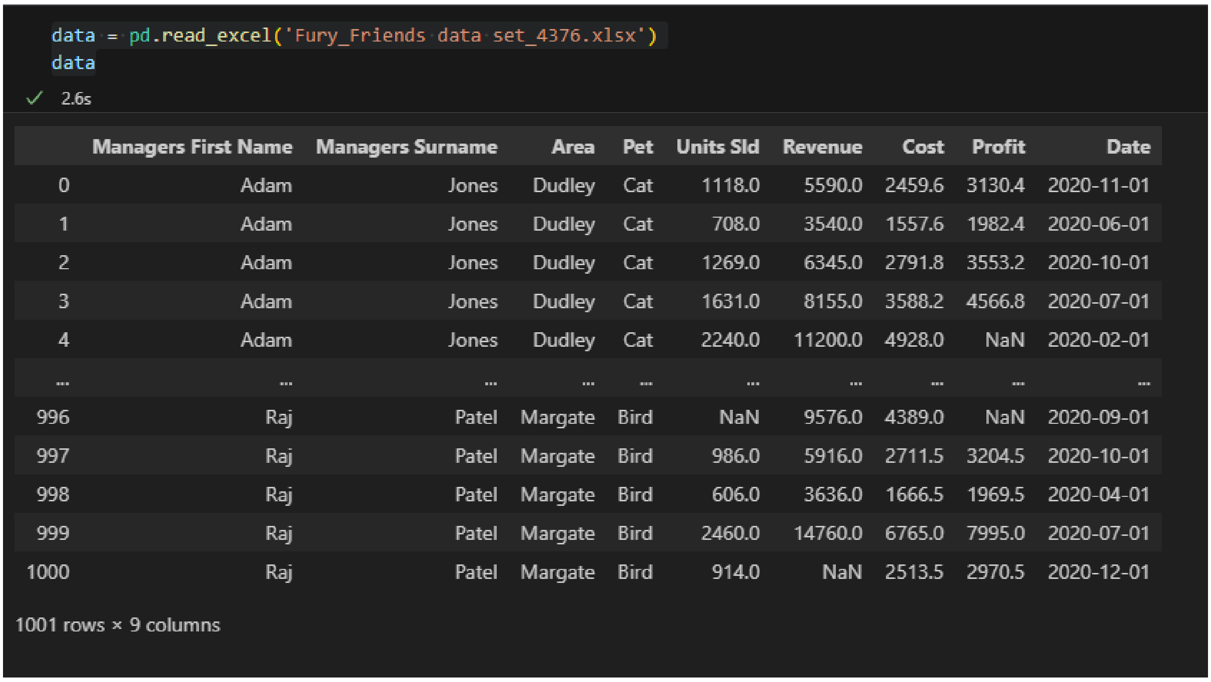
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 1- python codes for importing libraries

Pandas and Numpy: The most important libraries for working with data frames, arrays, and matrixes.

Matplotlib and seaborn: impressive libraries for visualizing and drawing 2 and 3 dimensional diagrams.

Importing Dataset:



```
data = pd.read_excel('Fury_Friends data set_4376.xlsx')
data
```

✓ 2.6s

	Managers First Name	Managers Surname	Area	Pet	Units Sld	Revenue	Cost	Profit	Date
0	Adam	Jones	Dudley	Cat	1118.0	5590.0	2459.6	3130.4	2020-11-01
1	Adam	Jones	Dudley	Cat	708.0	3540.0	1557.6	1982.4	2020-06-01
2	Adam	Jones	Dudley	Cat	1269.0	6345.0	2791.8	3553.2	2020-10-01
3	Adam	Jones	Dudley	Cat	1631.0	8155.0	3588.2	4566.8	2020-07-01
4	Adam	Jones	Dudley	Cat	2240.0	11200.0	4928.0	NaN	2020-02-01
...	...	...	...	...	...	...	...	...	...
996	Raj	Patel	Margate	Bird	NaN	9576.0	4389.0	NaN	2020-09-01
997	Raj	Patel	Margate	Bird	986.0	5916.0	2711.5	3204.5	2020-10-01
998	Raj	Patel	Margate	Bird	606.0	3636.0	1666.5	1969.5	2020-04-01
999	Raj	Patel	Margate	Bird	2460.0	14760.0	6765.0	7995.0	2020-07-01
1000	Raj	Patel	Margate	Bird	914.0	NaN	2513.5	2970.5	2020-12-01

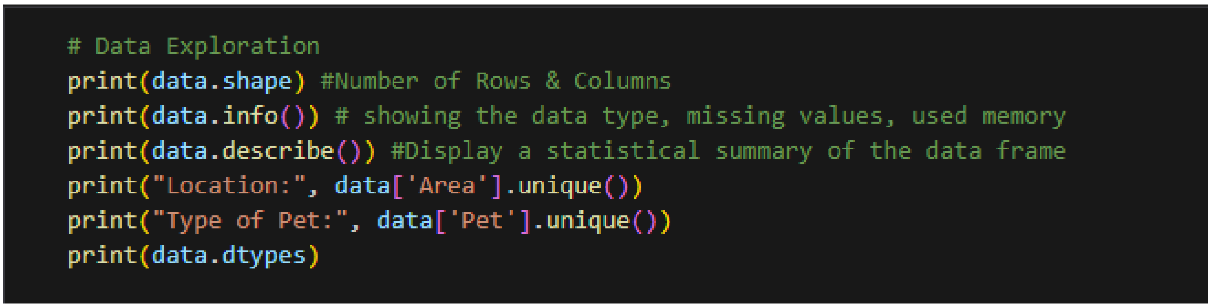
1001 rows × 9 columns

Figure 2- Preview of Data Set

Dataset contain 1001 records and 9 features about the Animal Products Stores.

Data Exploration:

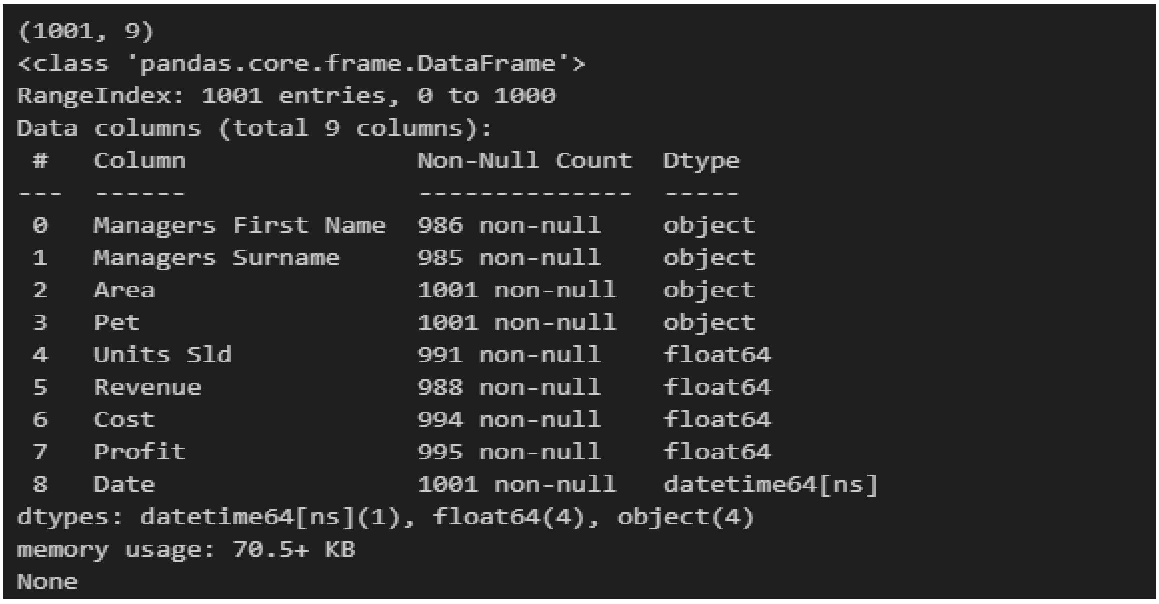
Basic knowledge of dataset leads to a better mastery of features and a greater ability to control or understand relationships and their impact on each other.



```
# Data Exploration
print(data.shape) #Number of Rows & Columns
print(data.info()) # showing the data type, missing values, used memory
print(data.describe()) #Display a statistical summary of the data frame
print("Location:", data['Area'].unique())
print("Type of Pet:", data['Pet'].unique())
print(data.dtypes)
```

Figure 3- python codes for Data Exploration

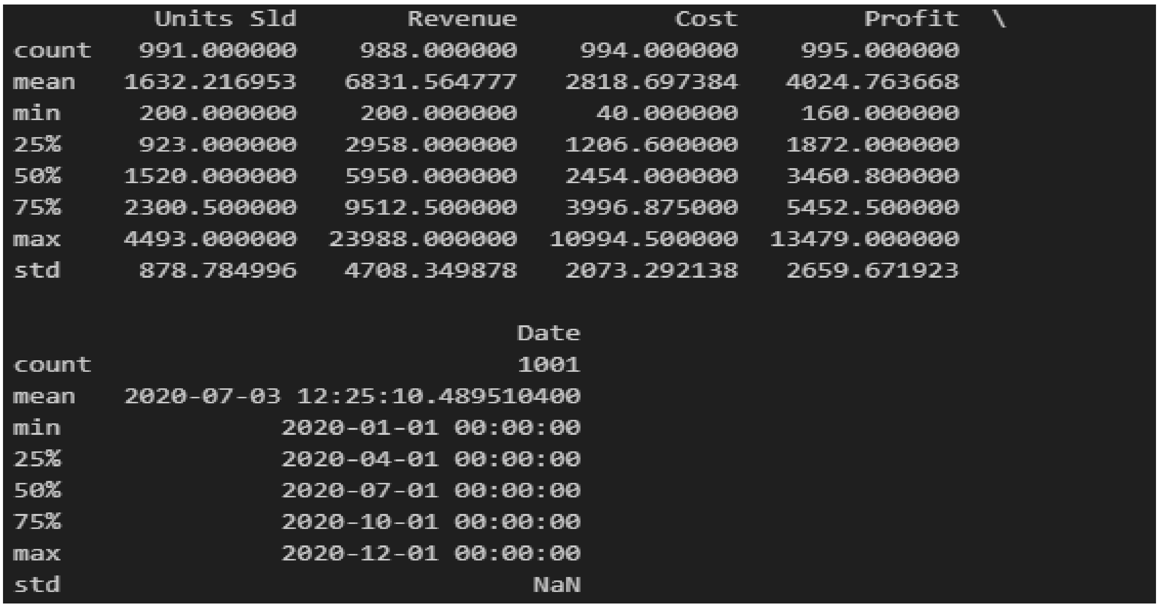
After obtaining information about the number of rows and columns, and descriptive statistics from the dataset, we will display an array of all types of animals, as well as the name of the location of the stores.



```
(1001, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Managers First Name  986 non-null    object
 1   Managers Surname     985 non-null    object
 2   Area                 1001 non-null   object
 3   Pet                  1001 non-null   object
 4   Units Sld            991 non-null    float64
 5   Revenue              988 non-null    float64
 6   Cost                 994 non-null    float64
 7   Profit               995 non-null    float64
 8   Date                 1001 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(4), object(4)
memory usage: 70.5+ KB
None
```

Figure 4- Data set Information

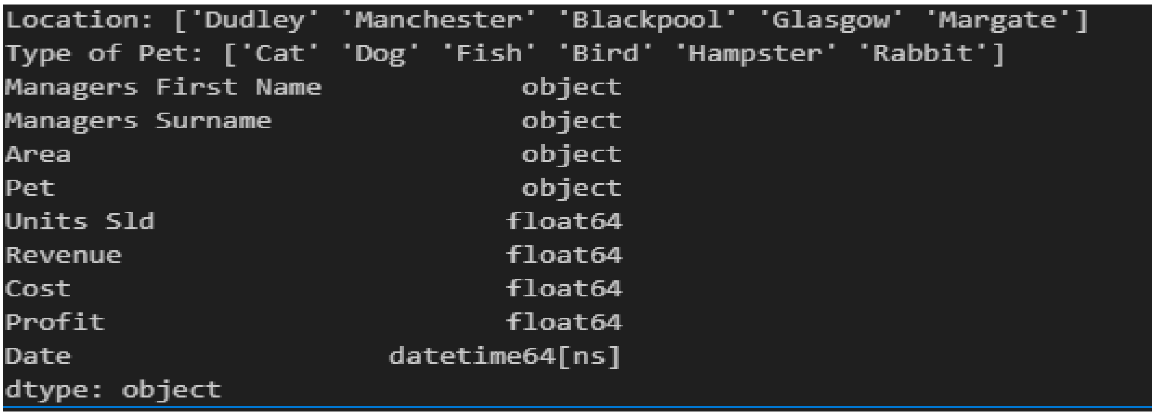
Data set contain 4 object type features, 4 float64, and 1 datetime64 variables.



```
         Units Sld       Revenue          Cost        Profit  \
count   991.000000    988.000000    994.000000    995.000000
mean   1632.216953   6831.564777   2818.697384   4024.763668
min     200.000000    200.000000     40.000000    160.000000
25%     923.000000   2958.000000   1206.600000   1872.000000
50%    1520.000000   5950.000000   2454.000000   3460.800000
75%    2300.500000   9512.500000   3996.875000   5452.500000
max    4493.000000  23988.000000  10994.500000  13479.000000
std     878.784996   4708.349878   2073.292138   2659.671923

                                Date
count                           1001
mean   2020-07-03 12:25:10.489510400
min              2020-01-01 00:00:00
25%              2020-04-01 00:00:00
50%              2020-07-01 00:00:00
75%              2020-10-01 00:00:00
max              2020-12-01 00:00:00
std                              NaN
```

Figure 5- Statistical Descriptive

Descriptive information from the dataset.



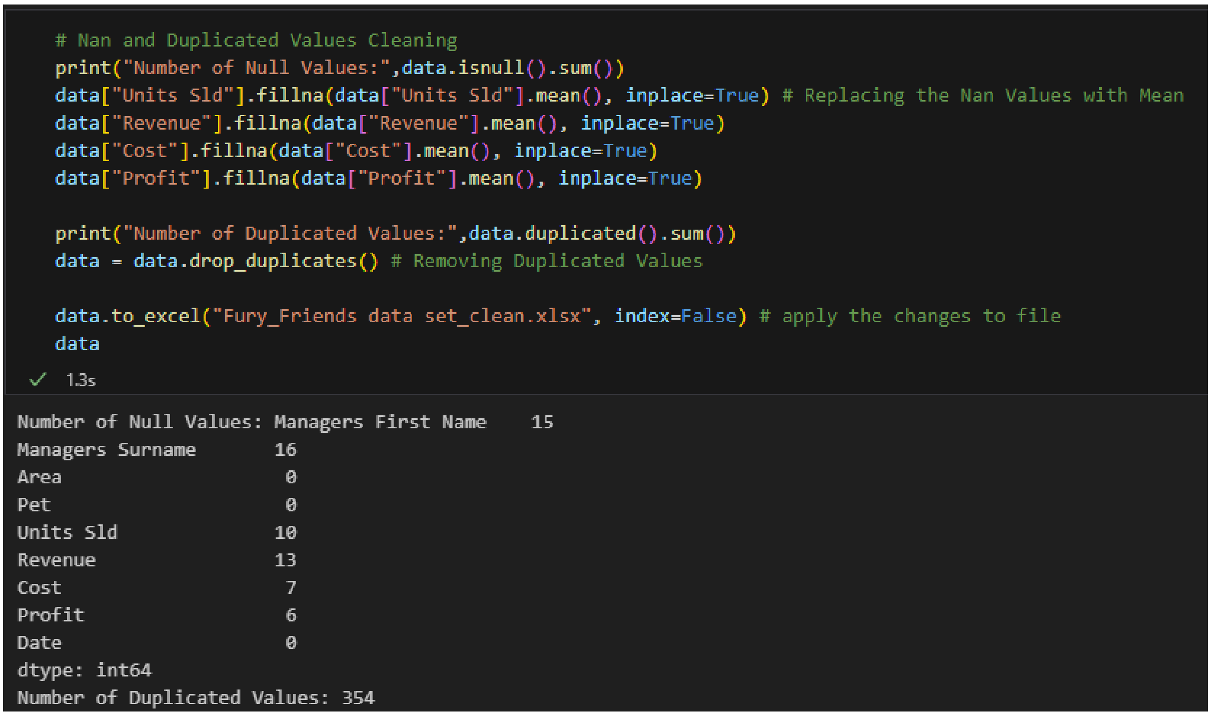
```
Location: ['Dudley' 'Manchester' 'Blackpool' 'Glasgow' 'Margate']
Type of Pet: ['Cat' 'Dog' 'Fish' 'Bird' 'Hampster' 'Rabbit']
Managers First Name             object
Managers Surname                object
Area                            object
Pet                             object
Units Sld                      float64
Revenue                        float64
Cost                           float64
Profit                         float64
Date                    datetime64[ns]
dtype: object
```

Figure 6- Data set information

A representation of two arrays including the name of the pets and the name of the position of the stores.

Data preparation

Exploring and Removing Nan values and Duplicated Rows:



```
# Nan and Duplicated Values Cleaning
print("Number of Null Values:",data.isnull().sum())
data["Units Sld"].fillna(data["Units Sld"].mean(), inplace=True) # Replacing the Nan Values with Mean
data["Revenue"].fillna(data["Revenue"].mean(), inplace=True)
data["Cost"].fillna(data["Cost"].mean(), inplace=True)
data["Profit"].fillna(data["Profit"].mean(), inplace=True)

print("Number of Duplicated Values:",data.duplicated().sum())
data = data.drop_duplicates() # Removing Duplicated Values

data.to_excel("Fury_Friends data set_clean.xlsx", index=False) # apply the changes to file
data
```

✓ 1.3s

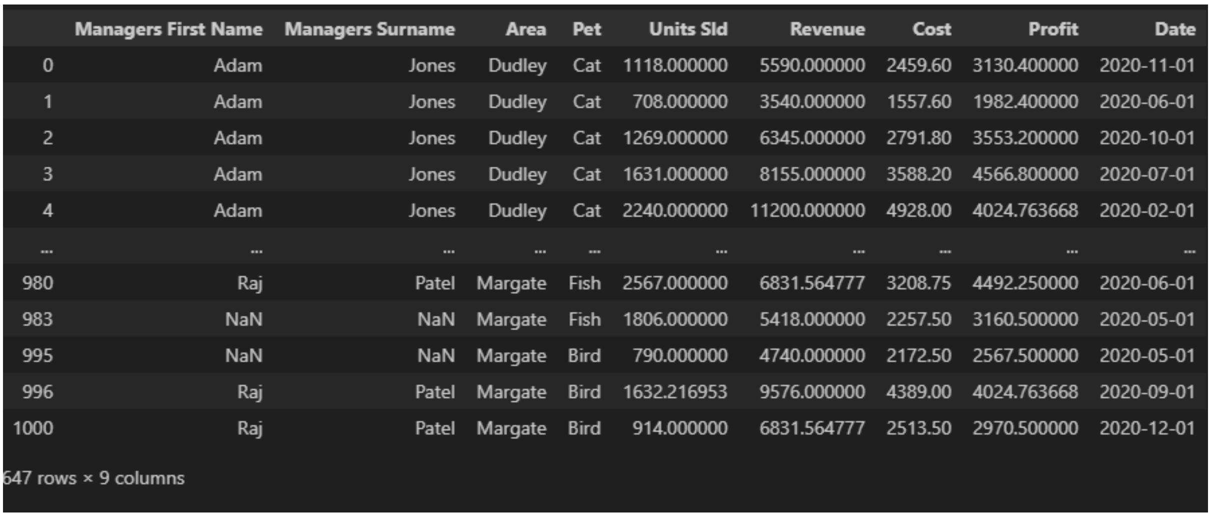
```
Number of Null Values: Managers First Name    15
Managers Surname       16
Area                    0
Pet                     0
Units Sld              10
Revenue                13
Cost                    7
Profit                  6
Date                    0
dtype: int64
Number of Duplicated Values: 354
```

Figure 7- Python Codes for Finding,Filling, and removing Nan and duplicated Values

About 67 Nan values are found in dataset which are replaced with the mean of their columns.

354 duplicated rows are eliminated, and Cleaned dataset are saved in a new dataset.

Preview of cleaned Dataset:

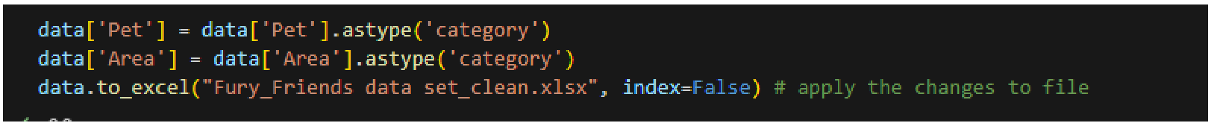


	Managers First Name	Managers Surname	Area	Pet	Units Sld	Revenue	Cost	Profit	Date
0	Adam	Jones	Dudley	Cat	1118.000000	5590.000000	2459.60	3130.400000	2020-11-01
1	Adam	Jones	Dudley	Cat	708.000000	3540.000000	1557.60	1982.400000	2020-06-01
2	Adam	Jones	Dudley	Cat	1269.000000	6345.000000	2791.80	3553.200000	2020-10-01
3	Adam	Jones	Dudley	Cat	1631.000000	8155.000000	3588.20	4566.800000	2020-07-01
4	Adam	Jones	Dudley	Cat	2240.000000	11200.000000	4928.00	4024.763668	2020-02-01
...	...	...	...	...	...	...	...	...	...
980	Raj	Patel	Margate	Fish	2567.000000	6831.564777	3208.75	4492.250000	2020-06-01
983	NaN	NaN	Margate	Fish	1806.000000	5418.000000	2257.50	3160.500000	2020-05-01
995	NaN	NaN	Margate	Bird	790.000000	4740.000000	2172.50	2567.500000	2020-05-01
996	Raj	Patel	Margate	Bird	1632.216953	9576.000000	4389.00	4024.763668	2020-09-01
1000	Raj	Patel	Margate	Bird	914.000000	6831.564777	2513.50	2970.500000	2020-12-01

647 rows × 9 columns

Figure 8- Preview of cleaned data set

Categorization:



```
data['Pet'] = data['Pet'].astype('category')
data['Area'] = data['Area'].astype('category')
data.to_excel("Fury_Friends data set_clean.xlsx", index=False) # apply the changes to file
```

Figure 9- Python Codes for Categorization

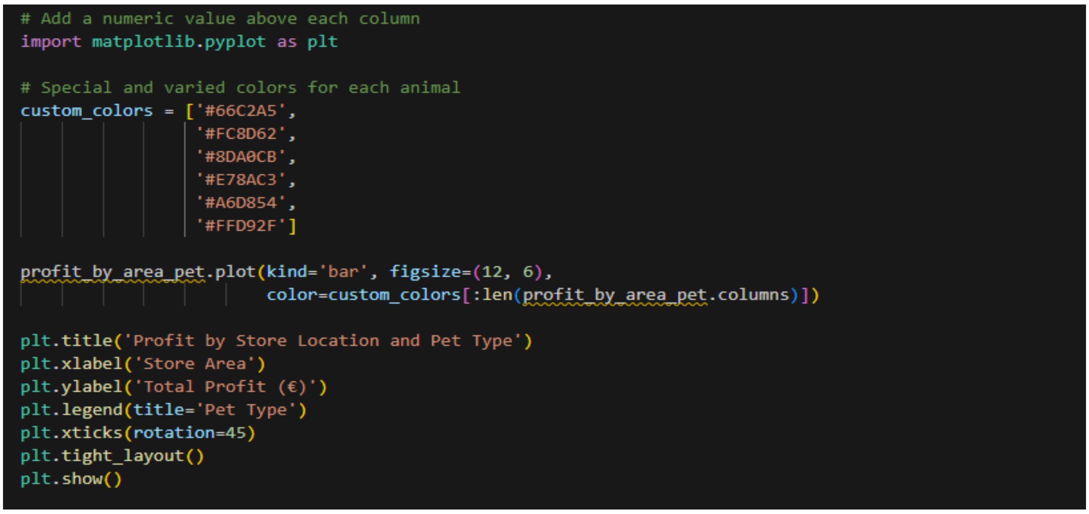
astype('category'): a method of converting string data that has duplicate values into categorized data. When data is stored as Category, it reduces memory consumption and Python processes the data faster.

Note:

Since the objective of this analysis is to explore, compare, and visualize the data rather than performing advanced modeling or clustering, there is no need for standardization and transformation for symmetry. The intended analyses can be effectively conducted using the raw, unaltered data.

Explorer Data Analysis (EDA)

Analyzing profit by store location (Area) and pet type to identify which segments are the most profitable:



```
# Add a numeric value above each column
import matplotlib.pyplot as plt

# Special and varied colors for each animal
custom_colors = ['#66C2A5',
                 '#FC8D62',
                 '#8DA0CB',
                 '#E78AC3',
                 '#A6D854',
                 '#FFD92F']

profit_by_area_pet.plot(kind='bar', figsize=(12, 6),
                        color=custom_colors[:len(profit_by_area_pet.columns)])

plt.title('Profit by Store Location and Pet Type')
plt.xlabel('Store Area')
plt.ylabel('Total Profit (€)')
plt.legend(title='Pet Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figure 10- Python Codes for analyzing profit on store and pet

It groups the data based on the two characteristics of **Area** and **Pet** and calculates the amount of **profit** for each combination.

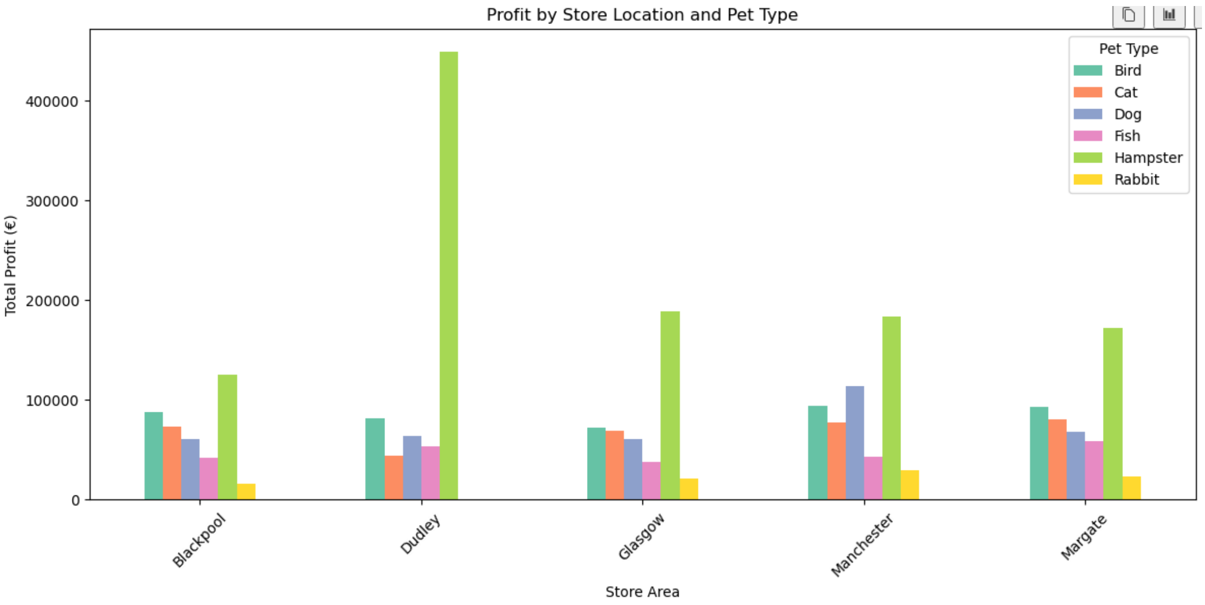
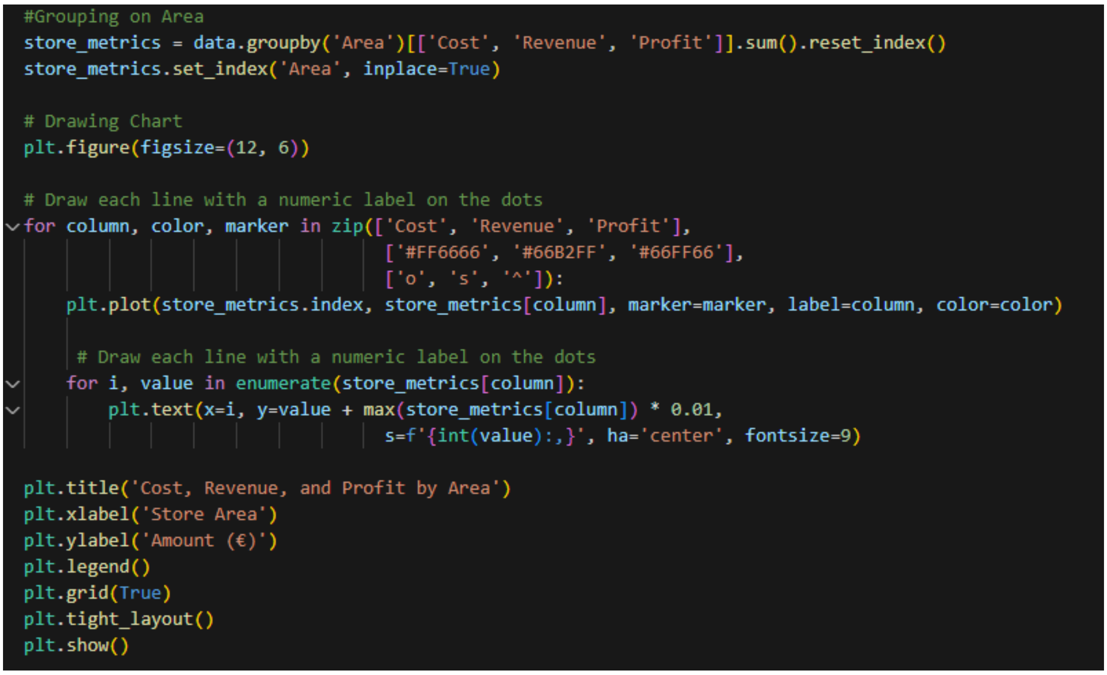


Figure 11- Preview of Bar chart

The highest profit among all stores belongs to the **Dudley** branch. **Hamsters** were the most in-demand products across the other store branches as well. It is evident that rabbits hold the smallest share of the market. Additionally, the **Blackpool** branch has the lowest profitability among all store locations.

Investigating the Effect of Area on Cost, Revenue, and Profit:



```
#Grouping on Area
store_metrics = data.groupby('Area')[['Cost', 'Revenue', 'Profit']].sum().reset_index()
store_metrics.set_index('Area', inplace=True)

# Drawing Chart
plt.figure(figsize=(12, 6))

# Draw each line with a numeric label on the dots
for column, color, marker in zip(['Cost', 'Revenue', 'Profit'],
                                 ['#FF6666', '#66B2FF', '#66FF66'],
                                 ['o', 's', '^']):
    plt.plot(store_metrics.index, store_metrics[column], marker=marker, label=column, color=color)

     # Draw each line with a numeric label on the dots
    for i, value in enumerate(store_metrics[column]):
        plt.text(x=i, y=value + max(store_metrics[column]) * 0.01,
                                 s=f'{int(value):,}', ha='center', fontsize=9)

plt.title('Cost, Revenue, and Profit by Area')
plt.xlabel('Store Area')
plt.ylabel('Amount (€)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 12- Python Codes for analyzing Area on cost, Revenue, and profit

It groups the regions based on income, expenditure, and profit.

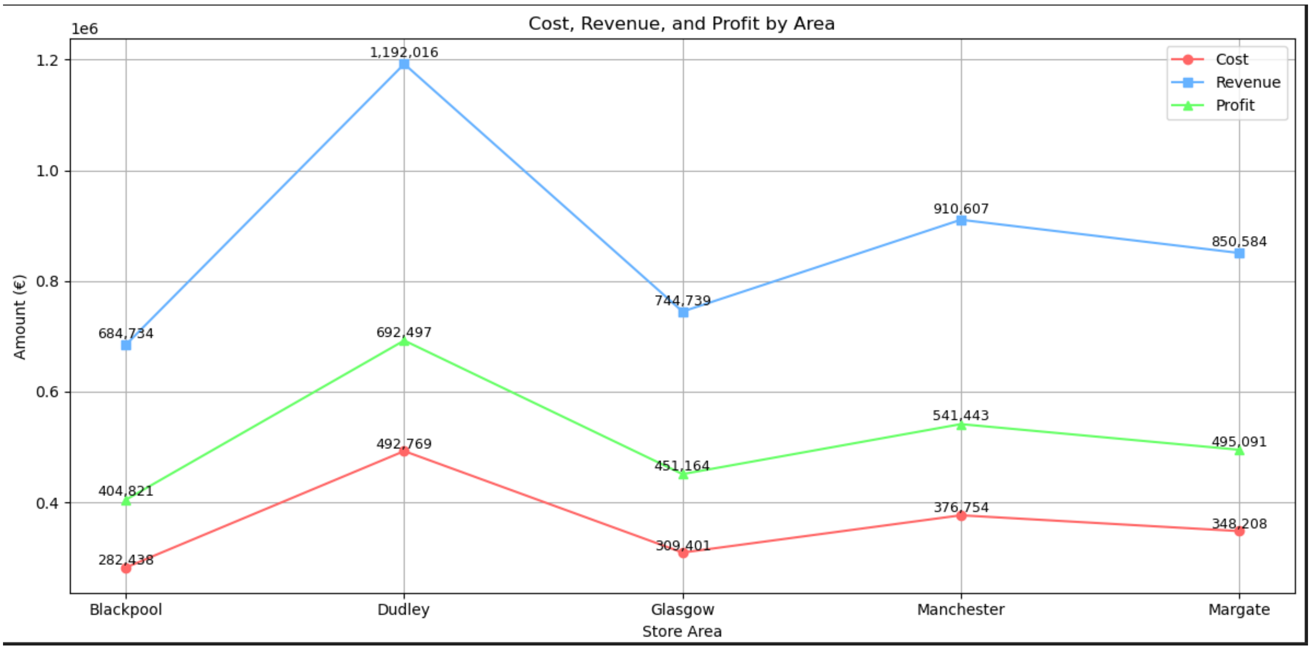
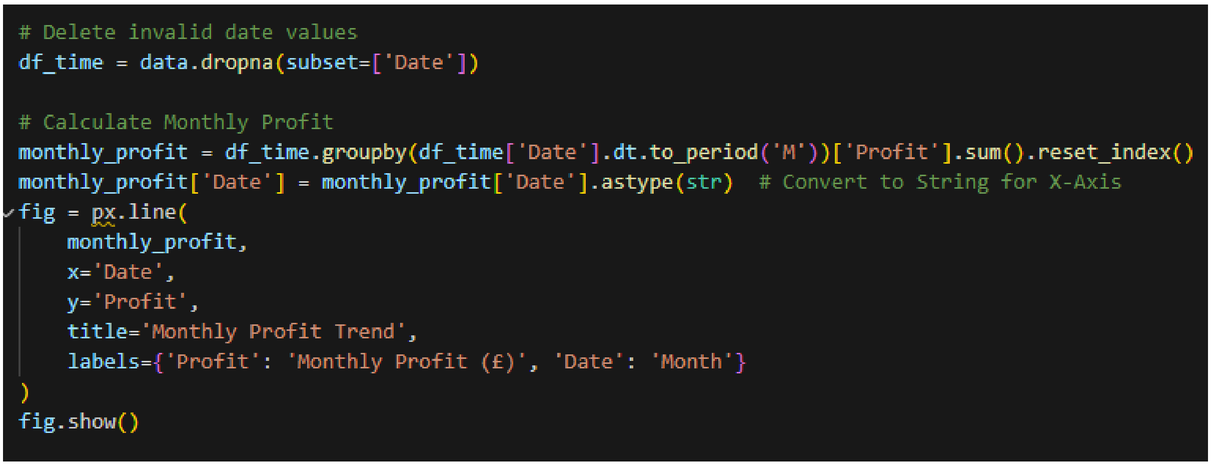


Figure 13- Preview of plot

- **Revenue** reached its highest and lowest values in the **Dudley** and **Blackpool** areas
- **Costs** are low across all regions, it reached their highest and lowest levels in the **Dudley** and **Blackpool**.
- highest **profit** is related to **Dudley**, while the lowest profit belongs to **Blackpool**

Calculating Monthly Profit:



```
# Delete invalid date values
df_time = data.dropna(subset=['Date'])

# Calculate Monthly Profit
monthly_profit = df_time.groupby(df_time['Date'].dt.to_period('M'))['Profit'].sum().reset_index()
monthly_profit['Date'] = monthly_profit['Date'].astype(str)  # Convert to String for X-Axis
fig = px.line(
    monthly_profit,
    x='Date',
    y='Profit',
    title='Monthly Profit Trend',
    labels={'Profit': 'Monthly Profit (£)', 'Date': 'Month'}
)
fig.show()
```

Figure 14- Python Codes for Monthly Profit

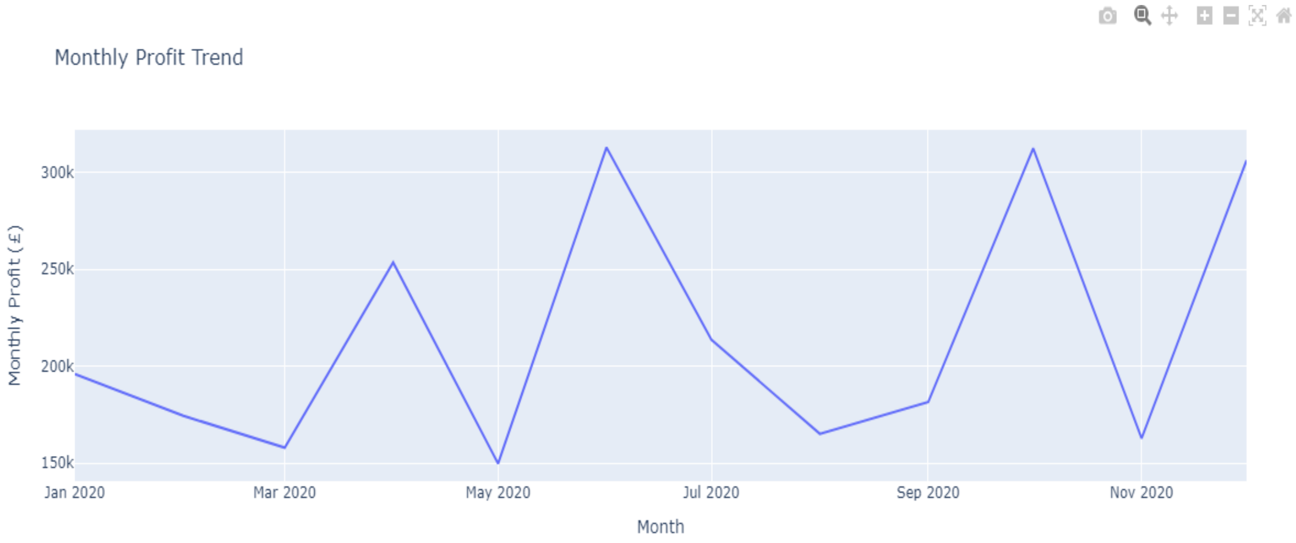
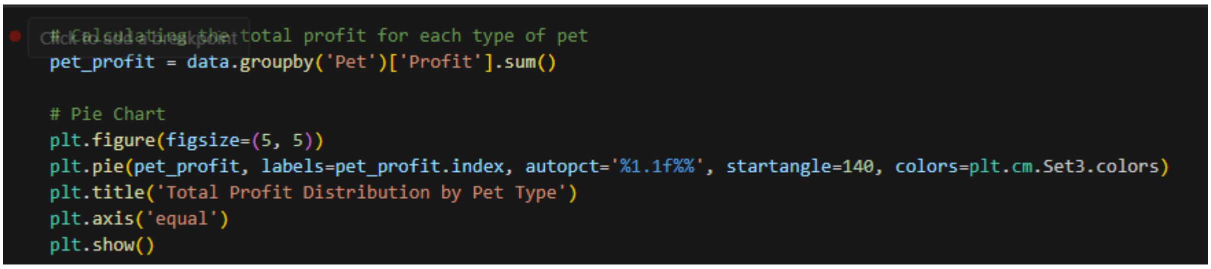


Figure 15- Preview of Lin chart of Monthly Profit

They have reached the maximum profit every month and then dropped the next month, which indicates a lack of stability in the sales strategy. Profits reached the head in June, October, and December, at over **300,000**£.

Investigating the Effect of pet on Profit:



```
# Calculating the total profit for each type of pet
pet_profit = data.groupby('Pet')['Profit'].sum()

# Pie Chart
plt.figure(figsize=(5, 5))
plt.pie(pet_profit, labels=pet_profit.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Set3.colors)
plt.title('Total Profit Distribution by Pet Type')
plt.axis('equal')
plt.show()
```

Figure 16- Python Codes for effect of pet on profit

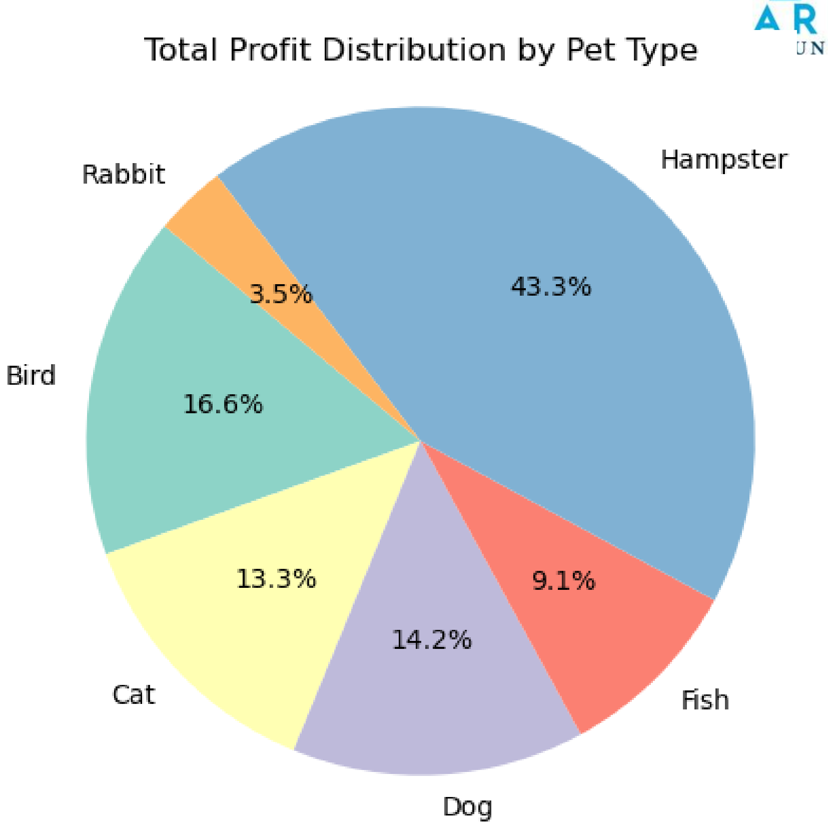
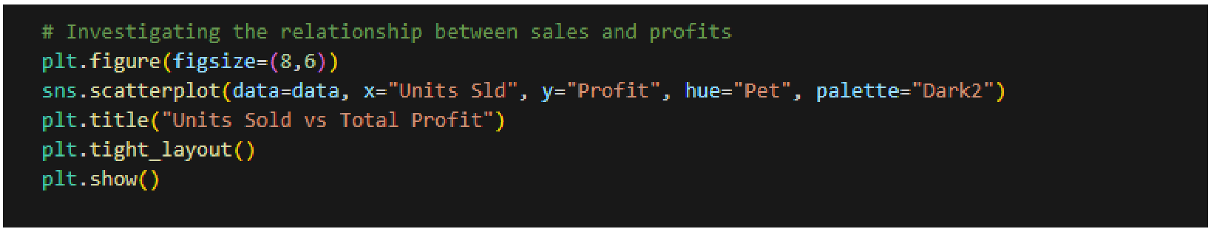


Figure 17- Preview of Pie chart

Hamsters account for the highest profit, at 43.3%, and it appear to be the most in-demand pet during the observed period. On the other hand**, rabbits** are the least demanded, at 3.5%, and consequently the least profitable pet type. The profit percentages for **dogs**, **cats**, and **birds** indicate a relatively stable demand for these three categories.

Survey the relationship sale & profits:



```
# Investigating the relationship between sales and profits
plt.figure(figsize=(8,6))
sns.scatterplot(data=data, x="Units Sld", y="Profit", hue="Pet", palette="Dark2")
plt.title("Units Sold vs Total Profit")
plt.tight_layout()
plt.show()
```

Figure 18- Python Codes for analyzing sale unit and profit

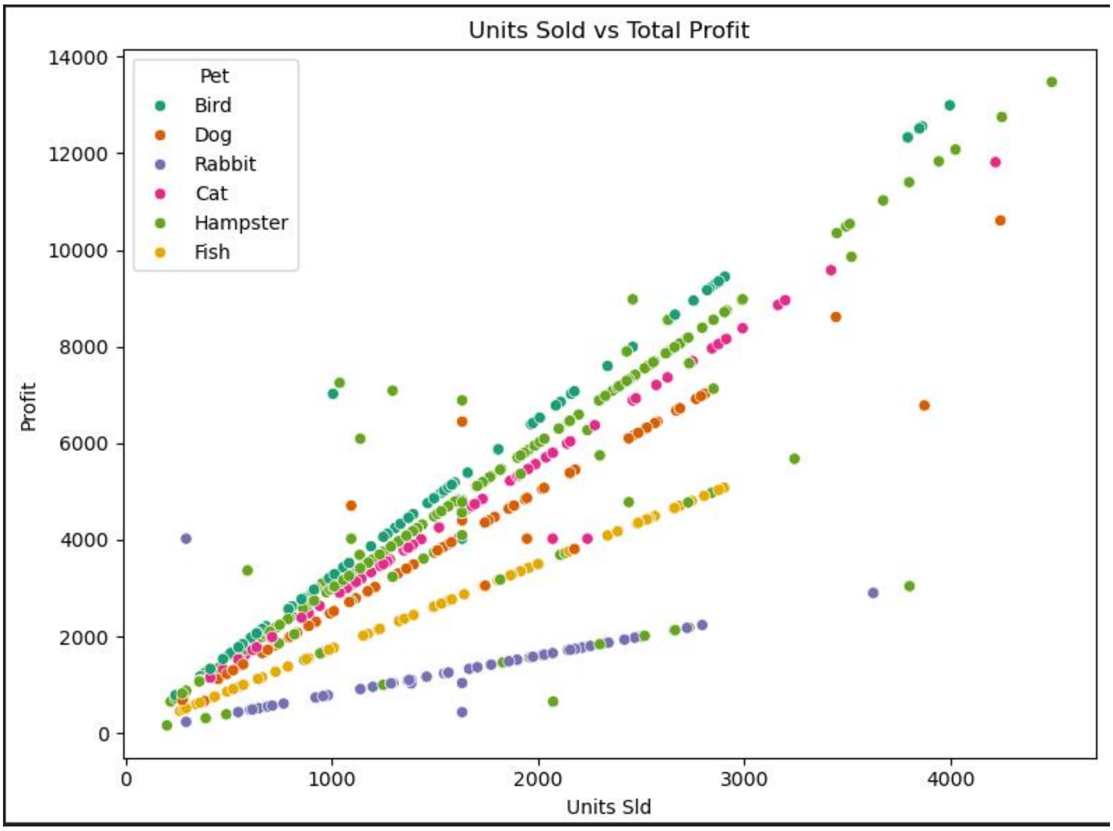
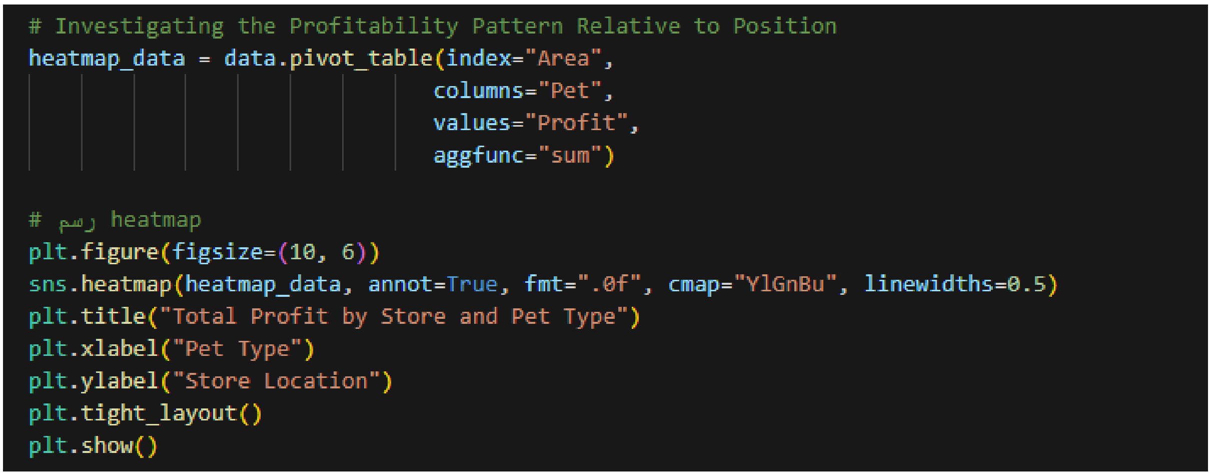


Figure 19- Preview of Scatter Plot

Overall, all pets had an upward trend in Unit Sale and Profit over the Time. There were some Outliers that exceed 13000£.

Area & Pet



```
# Investigating the Profitability Pattern Relative to Position
heatmap_data = data.pivot_table(index="Area",
                                columns="Pet",
                                values="Profit",
                                aggfunc="sum")

# رسم heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlGnBu", linewidths=0.5)
plt.title("Total Profit by Store and Pet Type")
plt.xlabel("Pet Type")
plt.ylabel("Store Location")
plt.tight_layout()
plt.show()
```

Figure 20- Python Codes for analyzing Area on pet

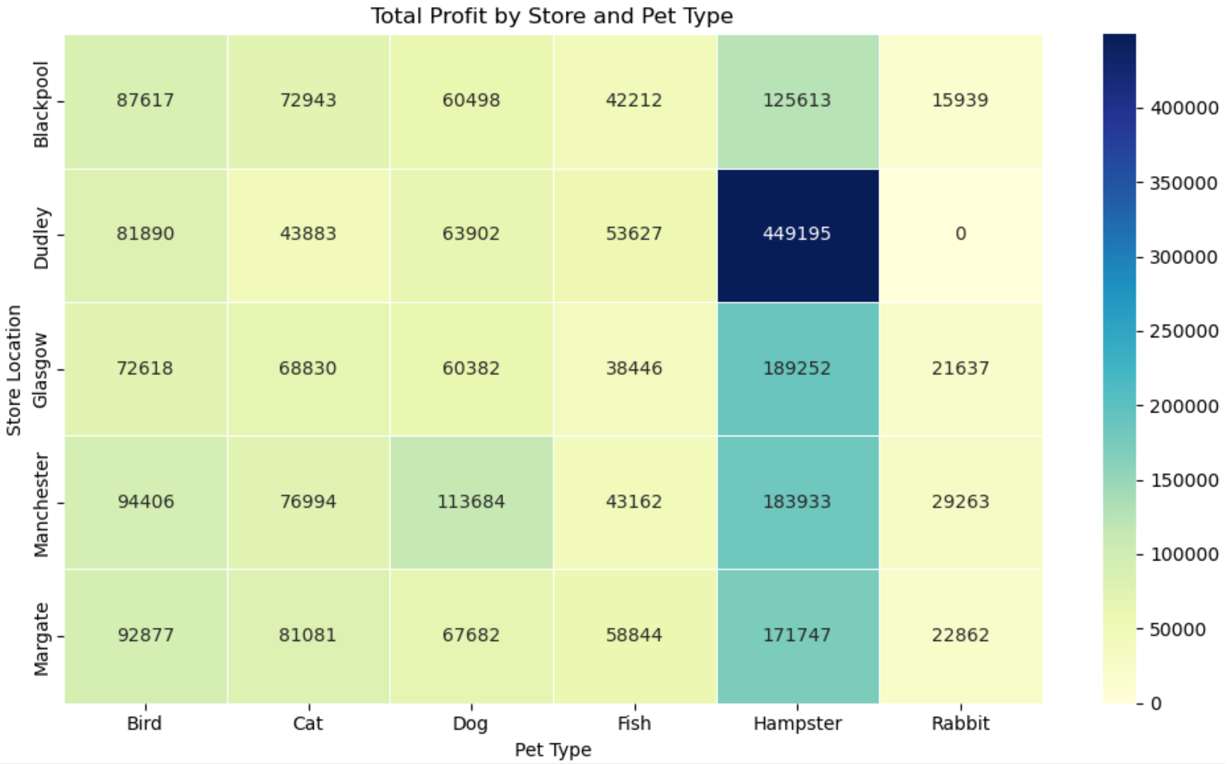
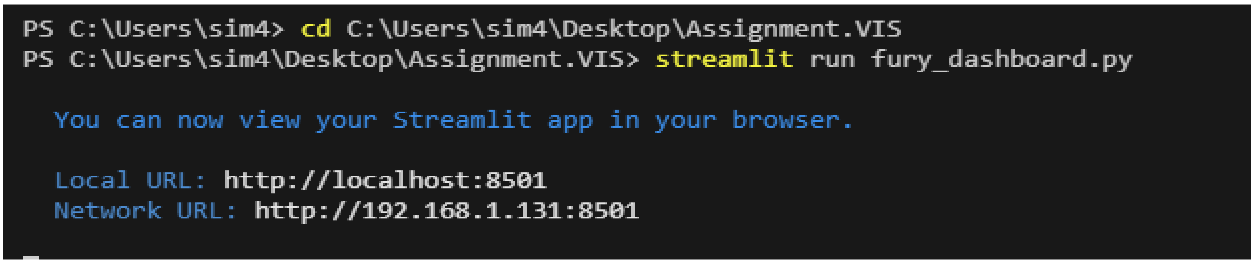


Figure 21- Preview of Heatmap

The focus of all branches is on **Hamster**. Unlike Dudley, which has not made a profit from the sale of rabbits, **Manchester** has managed to strike a profitable balance across all of its products, yet it is still the second most profitable store after Dudley.

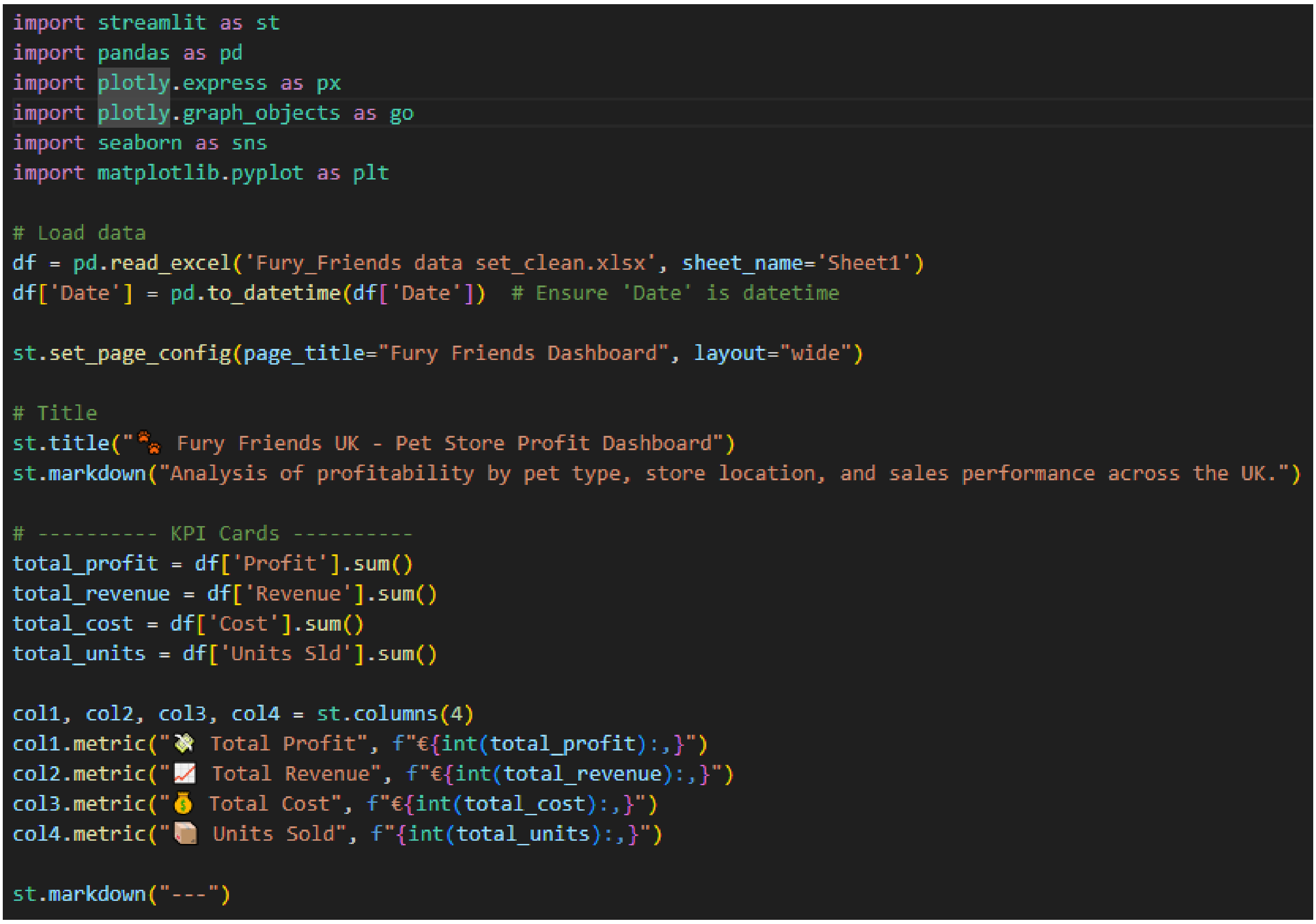
An Overview of Streamlit's Interactive Dashboard

Streamlit Dashboard is a high-level user interface built on Python that allows users to visually explore data through dynamic and interactive controls (Cody, 2020).

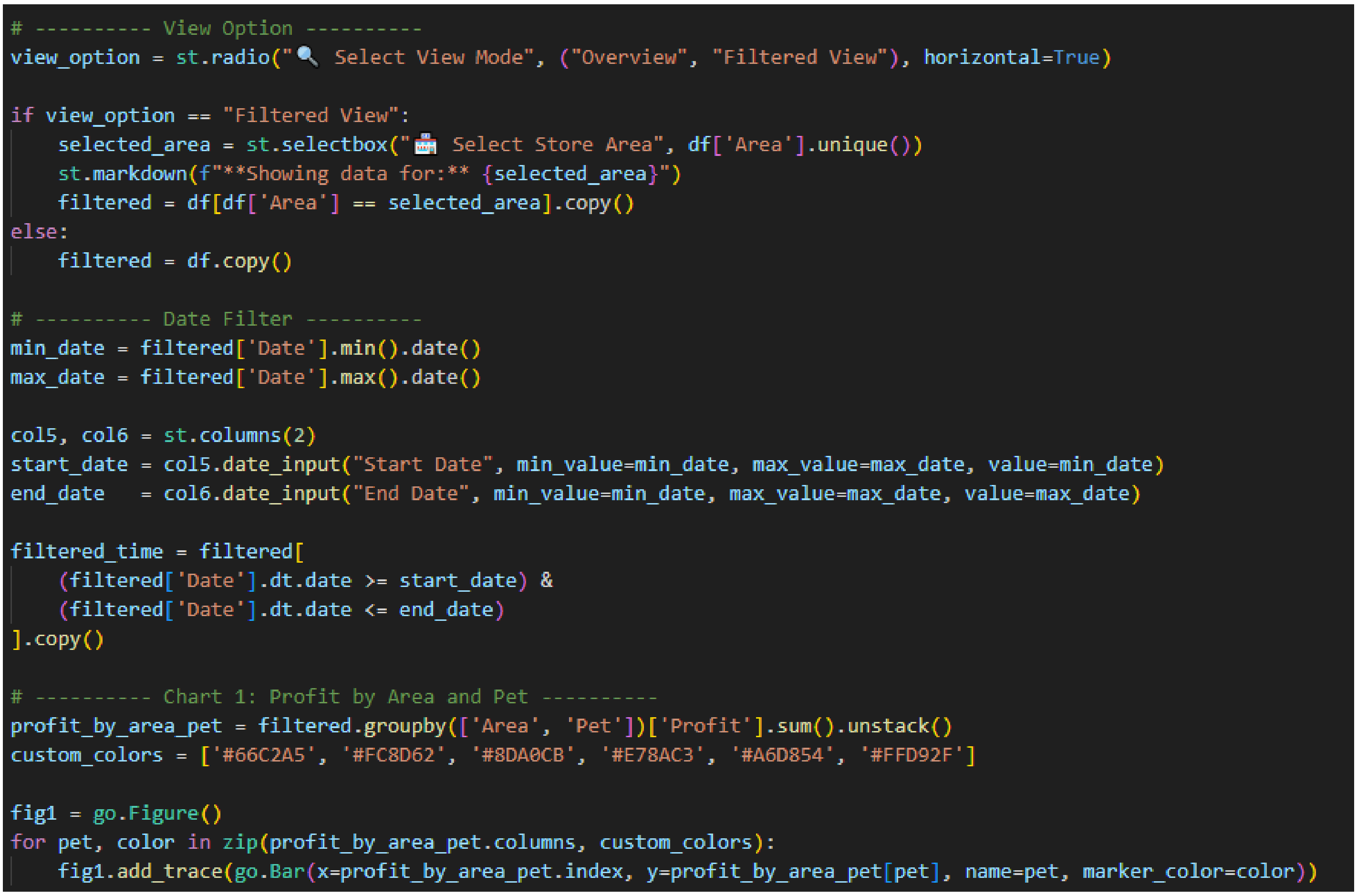


```
PS C:\Users\sim4> cd C:\Users\sim4\Desktop\Assignment.VIS
PS C:\Users\sim4\Desktop\Assignment.VIS> streamlit run fury_dashboard.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.131:8501
```

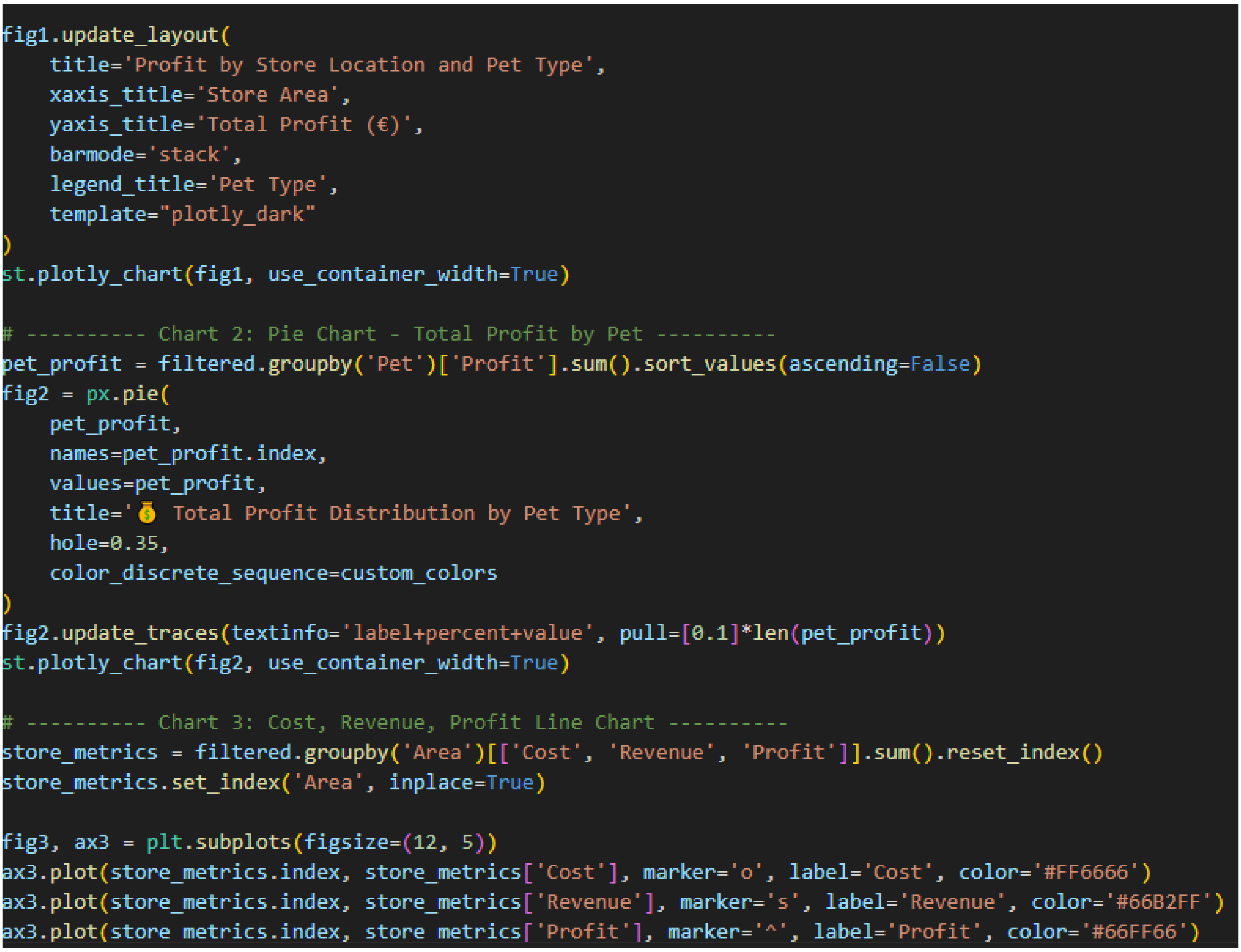
Figure 22- Python Codes for analyzing Area on pet



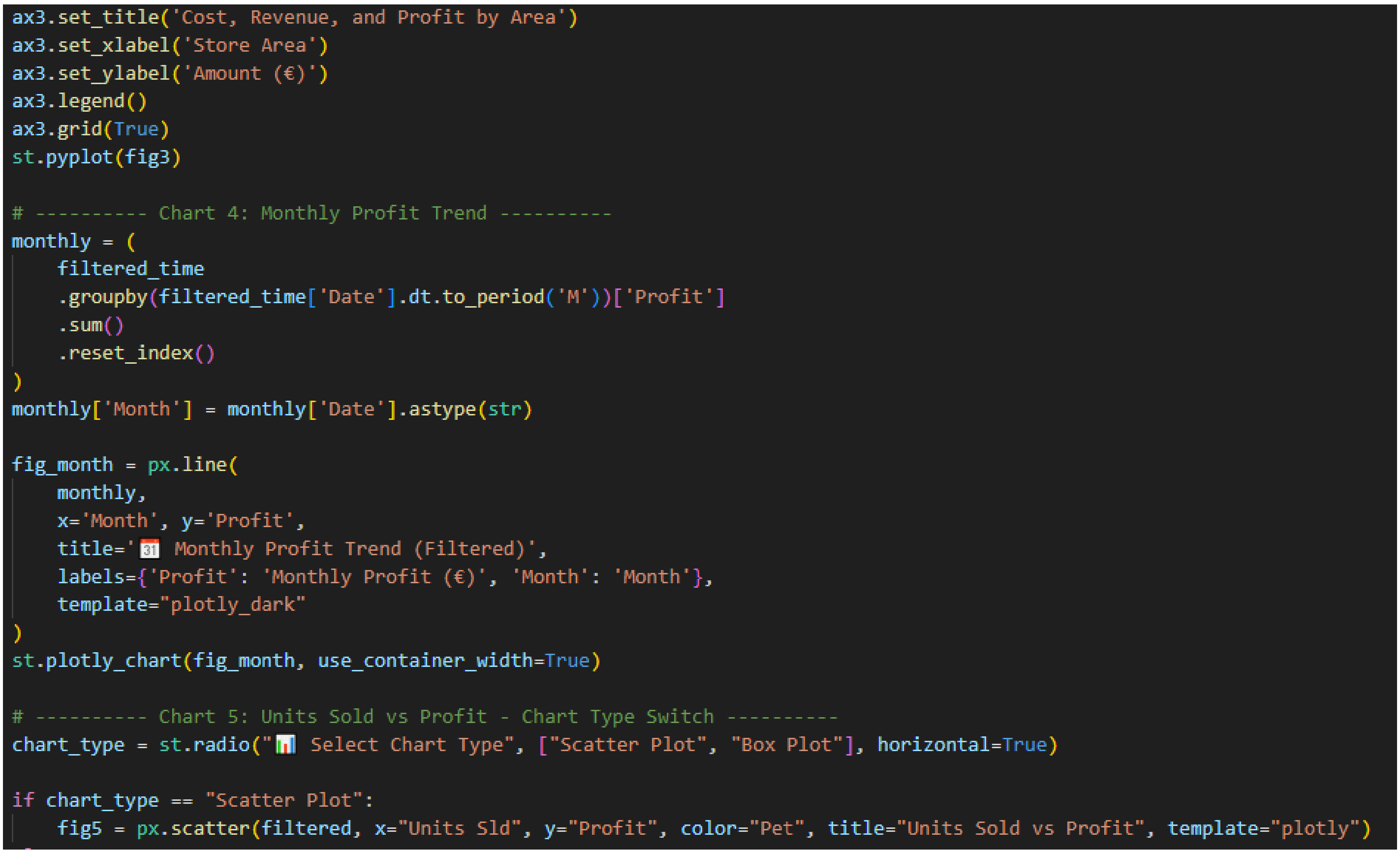
```
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = pd.read_excel('Fury_Friends data set_clean.xlsx', sheet_name='Sheet1')
df['Date'] = pd.to_datetime(df['Date'])  # Ensure 'Date' is datetime

st.set_page_config(page_title="Fury Friends Dashboard", layout="wide")

# Title
st.title("🐾 Fury Friends UK - Pet Store Profit Dashboard")
st.markdown("Analysis of profitability by pet type, store location, and sales performance across the UK.")

# ---------- KPI Cards ----------
total_profit = df['Profit'].sum()
total_revenue = df['Revenue'].sum()
total_cost = df['Cost'].sum()
total_units = df['Units Sld'].sum()

col1, col2, col3, col4 = st.columns(4)
col1.metric("💸 Total Profit", f"€{int(total_profit):,}")
col2.metric("📈 Total Revenue", f"€{int(total_revenue):,}")
col3.metric("💰 Total Cost", f"€{int(total_cost):,}")
col4.metric("📦 Units Sold", f"{int(total_units):,}")

st.markdown("---")
```



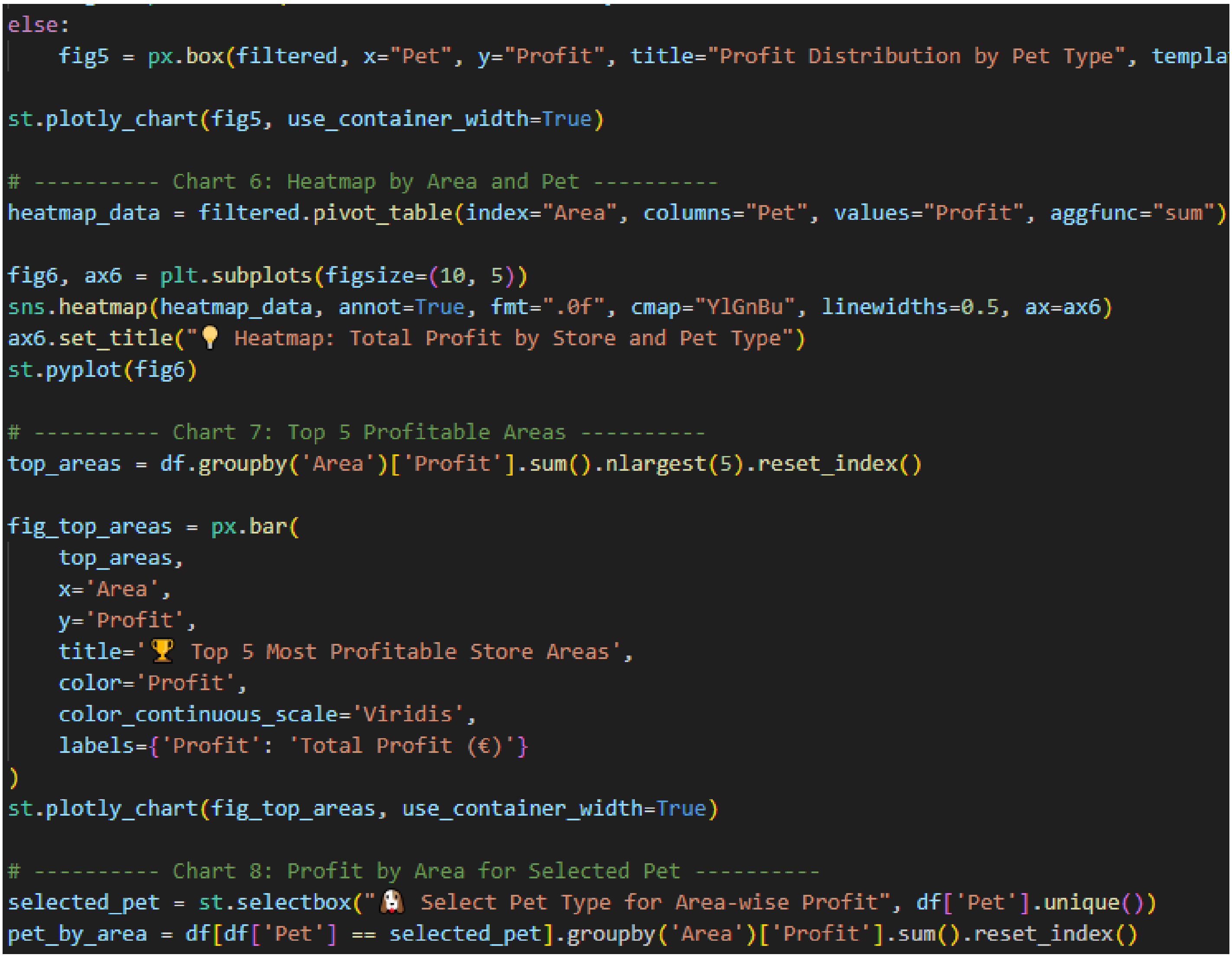
```
# ---------- View Option ----------
view_option = st.radio("🔍 Select View Mode", ("Overview", "Filtered View"), horizontal=True)

if view_option == "Filtered View":
    selected_area = st.selectbox("🏪 Select Store Area", df['Area'].unique())
    st.markdown(f"**Showing data for:** {selected_area}")
    filtered = df[df['Area'] == selected_area].copy()
else:
    filtered = df.copy()

# ---------- Date Filter ----------
min_date = filtered['Date'].min().date()
max_date = filtered['Date'].max().date()

col5, col6 = st.columns(2)
start_date = col5.date_input("Start Date", min_value=min_date, max_value=max_date, value=min_date)
end_date   = col6.date_input("End Date", min_value=min_date, max_value=max_date, value=max_date)

filtered_time = filtered[
    (filtered['Date'].dt.date >= start_date) &
    (filtered['Date'].dt.date <= end_date)
].copy()

# ---------- Chart 1: Profit by Area and Pet ----------
profit_by_area_pet = filtered.groupby(['Area', 'Pet'])['Profit'].sum().unstack()
custom_colors = ['#66C2A5', '#FC8D62', '#8DA0CB', '#E78AC3', '#A6D854', '#FFD92F']

fig1 = go.Figure()
for pet, color in zip(profit_by_area_pet.columns, custom_colors):
    fig1.add_trace(go.Bar(x=profit_by_area_pet.index, y=profit_by_area_pet[pet], name=pet, marker_color=color))
```



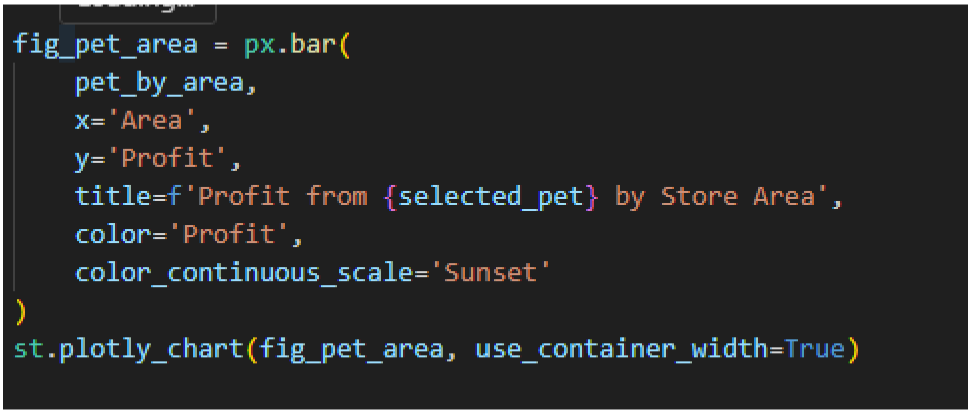
```
fig1.update_layout(
    title='Profit by Store Location and Pet Type',
    xaxis_title='Store Area',
    yaxis_title='Total Profit (€)',
    barmode='stack',
    legend_title='Pet Type',
    template="plotly_dark"
)
st.plotly_chart(fig1, use_container_width=True)

# ---------- Chart 2: Pie Chart - Total Profit by Pet ----------
pet_profit = filtered.groupby('Pet')['Profit'].sum().sort_values(ascending=False)
fig2 = px.pie(
    pet_profit,
    names=pet_profit.index,
    values=pet_profit,
    title='💰 Total Profit Distribution by Pet Type',
    hole=0.35,
    color_discrete_sequence=custom_colors
)
fig2.update_traces(textinfo='label+percent+value', pull=[0.1]*len(pet_profit))
st.plotly_chart(fig2, use_container_width=True)

# ---------- Chart 3: Cost, Revenue, Profit Line Chart ----------
store_metrics = filtered.groupby('Area')[['Cost', 'Revenue', 'Profit']].sum().reset_index()
store_metrics.set_index('Area', inplace=True)

fig3, ax3 = plt.subplots(figsize=(12, 5))
ax3.plot(store_metrics.index, store_metrics['Cost'], marker='o', label='Cost', color='#FF6666')
ax3.plot(store_metrics.index, store_metrics['Revenue'], marker='s', label='Revenue', color='#66B2FF')
ax3.plot(store_metrics.index, store_metrics['Profit'], marker='^', label='Profit', color='#66FF66')
```



```
ax3.set_title('Cost, Revenue, and Profit by Area')
ax3.set_xlabel('Store Area')
ax3.set_ylabel('Amount (€)')
ax3.legend()
ax3.grid(True)
st.pyplot(fig3)

# ---------- Chart 4: Monthly Profit Trend ----------
monthly = (
    filtered_time
    .groupby(filtered_time['Date'].dt.to_period('M'))['Profit']
    .sum()
    .reset_index()
)
monthly['Month'] = monthly['Date'].astype(str)

fig_month = px.line(
    monthly,
    x='Month', y='Profit',
    title='📅 Monthly Profit Trend (Filtered)',
    labels={'Profit': 'Monthly Profit (€)', 'Month': 'Month'},
    template="plotly_dark"
)
st.plotly_chart(fig_month, use_container_width=True)

# ---------- Chart 5: Units Sold vs Profit - Chart Type Switch ----------
chart_type = st.radio("📊 Select Chart Type", ["Scatter Plot", "Box Plot"], horizontal=True)

if chart_type == "Scatter Plot":
    fig5 = px.scatter(filtered, x="Units Sld", y="Profit", color="Pet", title="Units Sold vs Profit", template="plotly")
```

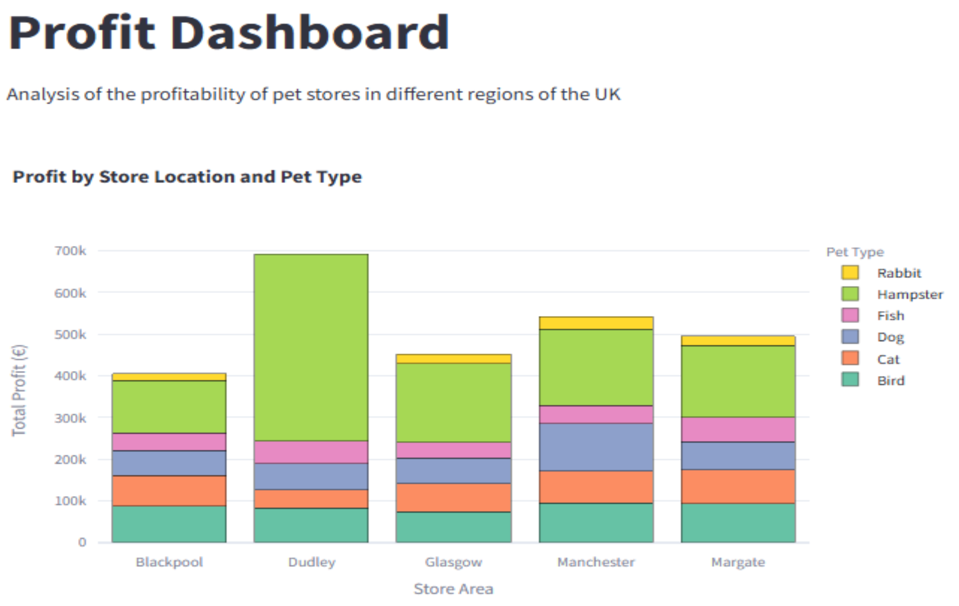


```
else:
    fig5 = px.box(filtered, x="Pet", y="Profit", title="Profit Distribution by Pet Type", templa

st.plotly_chart(fig5, use_container_width=True)

# ---------- Chart 6: Heatmap by Area and Pet ----------
heatmap_data = filtered.pivot_table(index="Area", columns="Pet", values="Profit", aggfunc="sum")

fig6, ax6 = plt.subplots(figsize=(10, 5))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlGnBu", linewidths=0.5, ax=ax6)
ax6.set_title("💡 Heatmap: Total Profit by Store and Pet Type")
st.pyplot(fig6)

# ---------- Chart 7: Top 5 Profitable Areas ----------
top_areas = df.groupby('Area')['Profit'].sum().nlargest(5).reset_index()

fig_top_areas = px.bar(
    top_areas,
    x='Area',
    y='Profit',
    title='🏆 Top 5 Most Profitable Store Areas',
    color='Profit',
    color_continuous_scale='Viridis',
    labels={'Profit': 'Total Profit (€)'}
)
st.plotly_chart(fig_top_areas, use_container_width=True)

# ---------- Chart 8: Profit by Area for Selected Pet ----------
selected_pet = st.selectbox("🐶 Select Pet Type for Area-wise Profit", df['Pet'].unique())
pet_by_area = df[df['Pet'] == selected_pet].groupby('Area')['Profit'].sum().reset_index()
```



```
fig_pet_area = px.bar(
    pet_by_area,
    x='Area',
    y='Profit',
    title=f'Profit from {selected_pet} by Store Area',
    color='Profit',
    color_continuous_scale='Sunset'
)
st.plotly_chart(fig_pet_area, use_container_width=True)
```

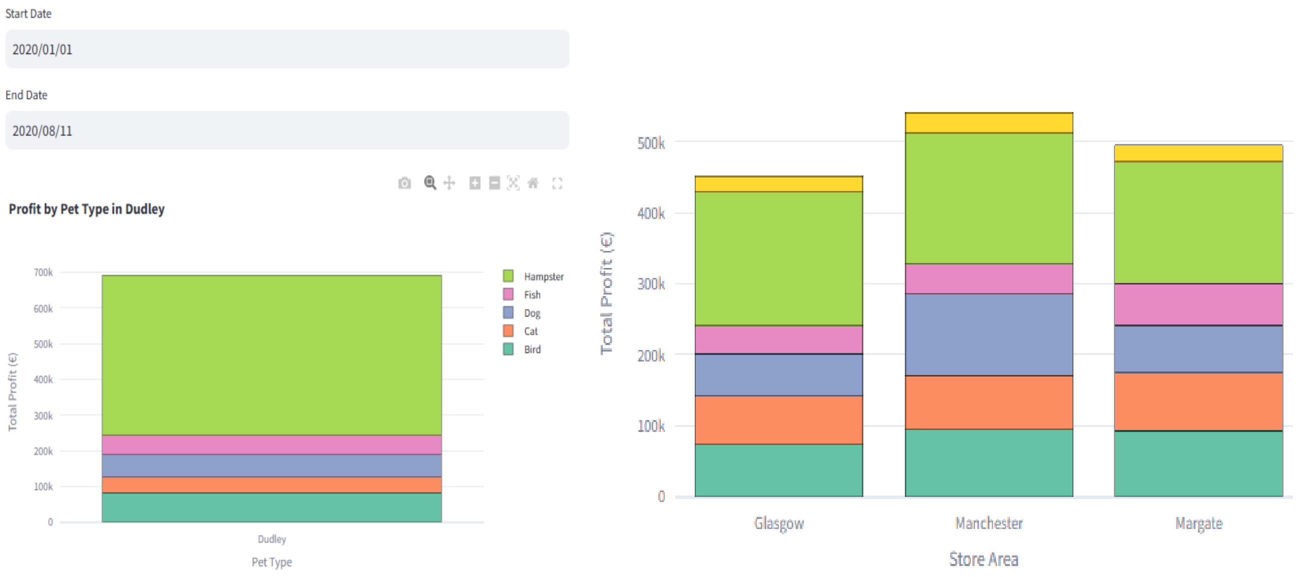
Figure 23- Python Codes for Implementing Dashboard



Profit Dashboard

Analysis of the profitability of pet stores in different regions of the UK

Profit by Store Location and Pet Type



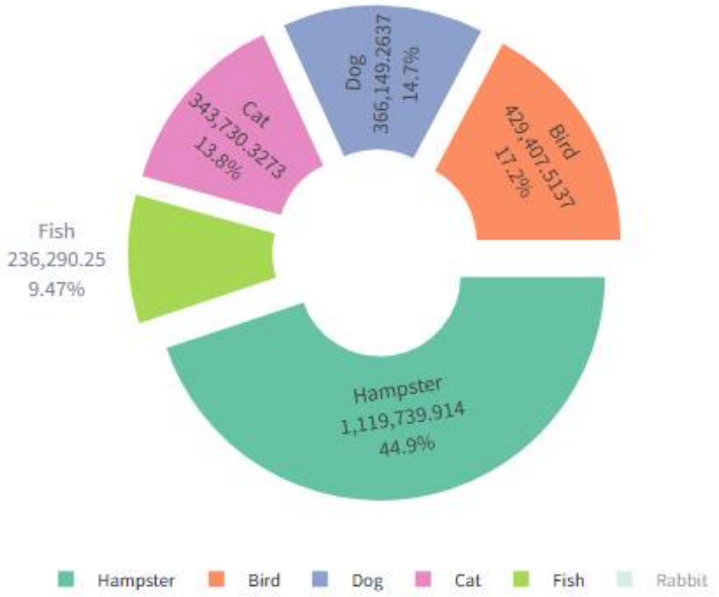
Start Date

2020/01/01

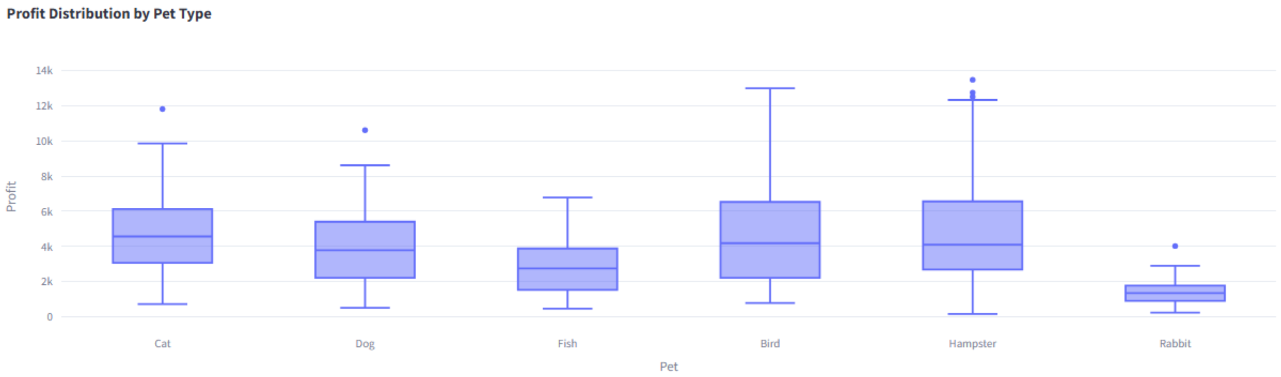
End Date

2020/08/11

Profit by Pet Type in Dudley

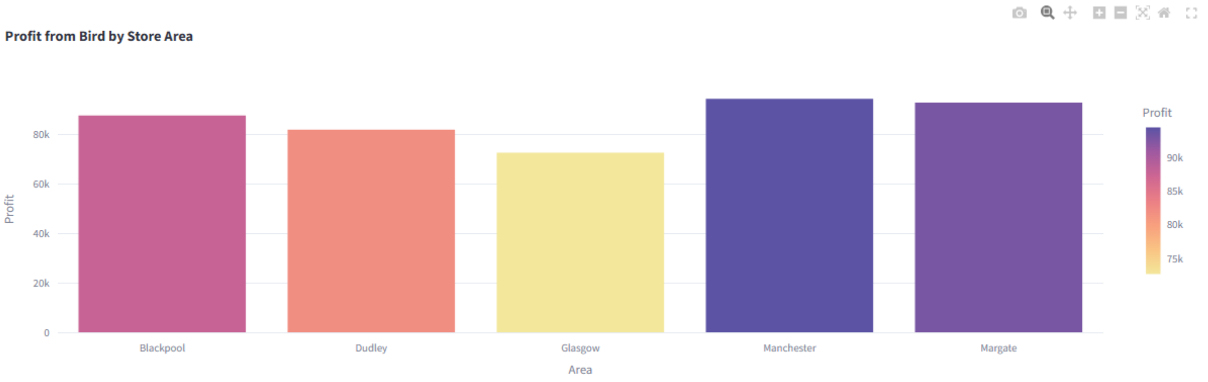


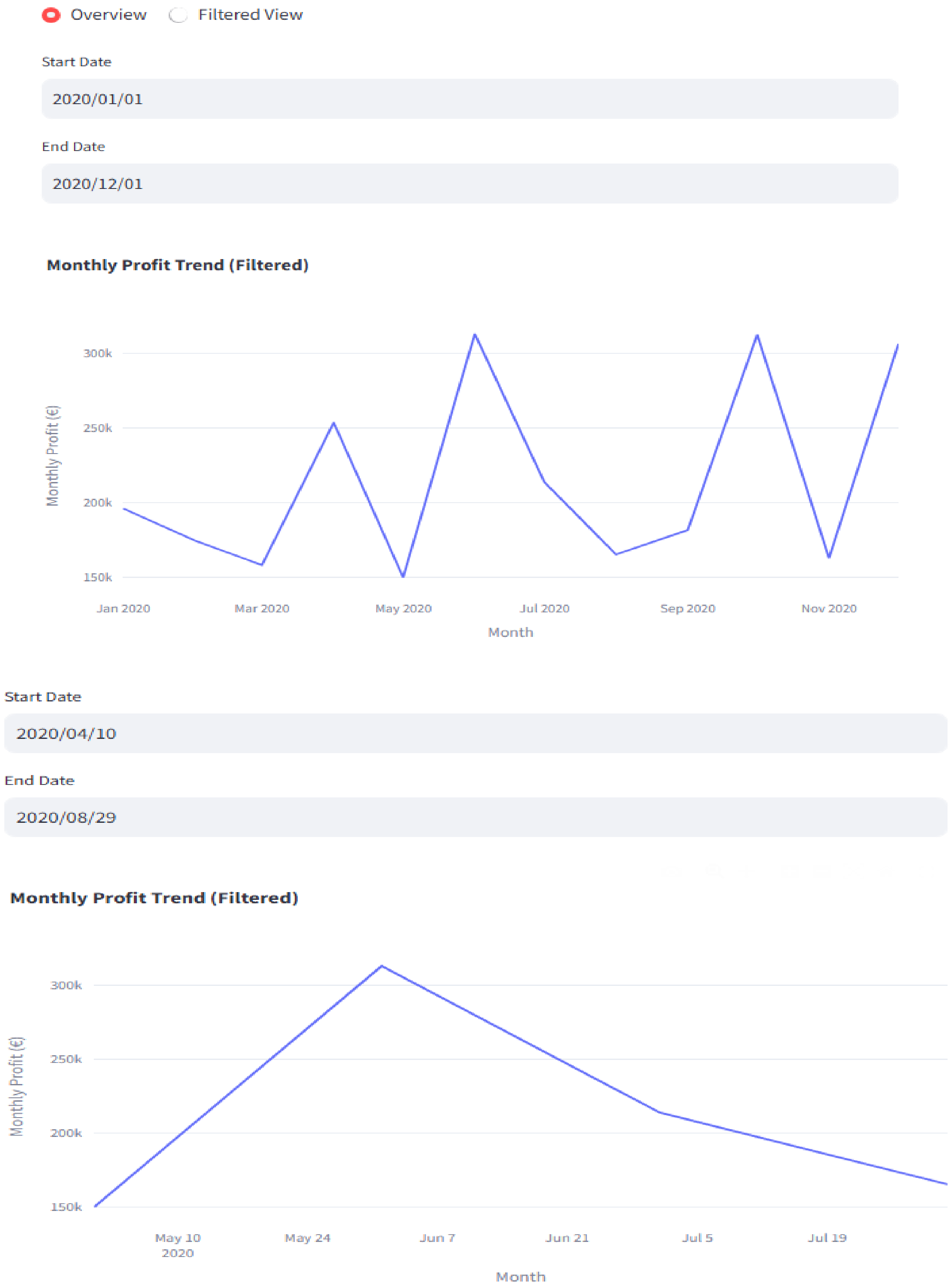
○ Scatter Plot ● Box Plot


Profit Distribution by Pet Type

Select Pet Type for Area-wise Profit

Bird


Profit from Bird by Store Area



● Overview ○ Filtered View

Start Date

2020/01/01

End Date

2020/12/01

Monthly Profit Trend (Filtered)

Start Date

2020/04/10

End Date

2020/08/29

Monthly Profit Trend (Filtered)

Figure 24- Preview of Streamlit Dashboard

TASK 2: Analysing the Performance of implementing and Visualization – Fury Friends Stores in the UK:

During the study of store data, we discovered the linear and reciprocal relationships of data to each other using the Python programming language, and then based on data visualization techniques, we conducted an exploratory and critical evaluation of the dataset. And finally, by using the interactive dashboard of Streamlit, we tried to interact as much as possible with user so that they can access accurate information about a specific date or store by applying filters. Our goal is to try to find the factors that affect profitability and/or losses, as well as try to convey visual insights to the board. The reasons for choosing charts and the structure of the dashboard are explained below, and evidence-based recommendations are provided to support strategic business decisions.

Step 1: Data Preparation and Cleansing

These steps made the data more accurately represent the realities in stores and helped to analyse more accurately and credibly. Operations such as **data cleansing**, **eliminating duplicate records** and **missing values**, replacement of incorrect data, and completion of incomplete values, as well as investigation and classification of variables to improve the quality of the data, were performed.

Step2: Discovery and Analysis of Relationships between Factors (EDA)

In this stage, the factors affecting profit, such as cost, income, store location, and the type of animals are examined and visualize. The following is an analytical report:

- **Linear relationship between sales, profits, costs, and stores**

  There is a linear and positive relationship between profit and income. The highest **Revenue, cost,** and profit (692,497) among all stores belongs to the **Dudley** branch. The store is the main hub of profitability. Aalthough **Blackpool** has a low income and cost rather than **Glasgow**, it is more profitable. With this analysis, **Glasgow** needs to rethink its cost reduction or sales strategy.

- **Total profit by branch and type of animal**

  According to the bar chart, heat map, and pie chart, hamsters were in high demand at 43.3% (£1.5 million), particularly in **Dudley**, a profit greater than the cat and dog profit collection. In terms of profitability, the **Glasgow**, **Manchester**, and **Margate** branches have made almost equal profits from **hamster-related** sales. Other types of pets show significant variation in sales rankings across branches.

The **rabbits** have the lowest market shares at **£140,000**. Given their performance. The purpose of visualizing this analysis with 3 types of graphs is to emphasize the concentration of the importance of these factors. In order to give this insight to the board of directors to become more familiar with the tastes and interests of the buyers of each city, which will undoubtedly be effective in increasing profitability.

- **Relationship between profit, unit sales, and pet:**

There is a positive linear relationship between **profits** and **sales**. **Birds**, **cats,** and **hamsters** are the **top-performing** pet categories in terms of profitability, respectively. On the other hand, **rabbits** generate **low** profit despite relatively high sales, indicating a possible imbalance between cost and pricing. **Fish** have the **lowest** and display the frequency, upward trend, and order of type of animals in terms of profitability.

- **Survey the Monthly Profits:**

Peak profits usually occur in the summer (**June to August**) at over **300,000£**—mainly due to increased demand during the holiday season. After August, there is a slight decline, but again there is a secondary peak in **September** and **October**. These fluctuations after each peak indicate the lack of a profitable and flexible strategy suitable for each season. The line chart was able to depict this fluctuation clearly.

Step 3: Streamlit Interactive Dashboard

An interactive dashboard was designed using Streamlit. This dashboard allows users to filter data based on location and animal type and view dynamic visualizations of revenue and profits.

Features:

Drop-down menus to select the location and type of animal, time frame

this dashboard provides a user-friendly tool for data discovery. Store managers and board members can easily inspect specific areas or types of animals and identify opportunities for improvement.

Step 4: Results and Recommendations

- Focus on high-demand products, such as Hamsters, birds, cats, and dogs, respectively. Increasing their supply and investing in sales can be effective.
- Focus on sales strategy in profitable stores to implement in low-profit stores.
- Strengthen marketing strategies in the summer and increase ad campaigns in the months when profitability is low.
- A weak correlation between income and profit suggests that costs are significantly variable. Optimize costs in stores with high sales and low profits, and Detailed cost analysis to identify areas for cost reduction, such as: negotiating better contracts with suppliers, optimizing the workforce, or Streamlining operations can be effective.
- No store deserves to be closed because they are all profitable with different percentages. It is only by applying a series of reforms that this percentage can be increased. Finally, it is suggested to the board of directors to merge the **Blackpool** and **Glasgow** branches.
- Having more detailed information about the stores, such as the size of the store, the number of staff, helps in a more accurate analysis.

Step 5: Conclusion

Based on the analysis conducted, significant differences in the financial performance and profitability of various branches were identified. For example, Dudley and Manchester emerged as the most profitable stores, while branches such as Glasgow and Blackpool showed relatively lower profitability and were recommended for either consolidation or strategic review. It was also observed that while store location plays a role, the type of animals sold has an even greater impact on profitability. Furthermore, it was suggested that the company adopt best practices from high-performing stores as a benchmark. Therefore, it is recommended that the company strategically reassess its sales policies, product distribution, and the management of underperforming branches in order to optimize overall profitability across the retail network.

References:

- Cody, P. (2020) *Streamlit for Data Science*. Birmingham· Packt Publishing, UK.
- Healy, K. (2018) Data Visualization: A Practical Introduction. Princeton University Press, USA.

Appendix: Python code

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

data = pd.read_excel('Fury_Friends data set_4376.xlsx')

data

print(data.shape)

print(data.info())

print(data.describe())

print("Location:", data['Area'].unique())

print("Type of Pet:", data['Pet'].unique())

print(data.dtypes)

print("Number of Null Values:",data.isnull().sum())

data["Units Sld"].fillna(data["Units Sld"].mean(), inplace=True)

 data["Revenue"].fillna(data["Revenue"].mean(), inplace=True)

data["Cost"].fillna(data["Cost"].mean(), inplace=True)

data["Profit"].fillna(data["Profit"].mean(), inplace=True)

print("Number of Duplicated Values:",data.duplicated().sum())

data = data.drop_duplicates()


data.to_excel("Fury_Friends data set_clean.xlsx", index=False)

data['Pet'] = data['Pet'].astype('category')

data['Area'] = data['Area'].astype('category')

data.to_excel("Fury_Friends data set_clean.xlsx", index=False)

profit_by_area_pet = data.groupby(['Area', 'Pet'])['Profit'].sum().unstack()
```

```
custom_colors = ['#66C2A5',
        '#FC8D62',
        '#8DA0CB',
        '#E78AC3',
        '#A6D854',
        '#FFD92F']

profit_by_area_pet.plot(kind='bar', figsize=(12, 6),
           color=custom_colors[:len(profit_by_area_pet.columns)])

plt.title('Profit by Store Location and Pet Type')
plt.xlabel('Store Area')
plt.ylabel('Total Profit (€)')
plt.legend(title='Pet Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
store_metrics = data.groupby('Area')[['Cost', 'Revenue', 'Profit']].sum().reset_index()
store_metrics.set_index('Area', inplace=True)

plt.figure(figsize=(12, 6))
for column, color, marker in zip(['Cost', 'Revenue', 'Profit'],
                ['#FF6666', '#66B2FF', '#66FF66'],
                ['o', 's', '^']):
  plt.plot(store_metrics.index, store_metrics[column], marker=marker, label=column, color=color)

  for i, value in enumerate(store_metrics[column]):
    plt.text(x=i, y=value + max(store_metrics[column]) * 0.01,
                    s=f'{int(value):,}', ha='center', fontsize=9)
plt.title('Cost, Revenue, and Profit by Area')
plt.xlabel('Store Area')
plt.ylabel('Amount (€)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
df_time = data.dropna(subset=['Date'])
monthly_profit = df_time.groupby(df_time['Date'].dt.to_period('M'))['Profit'].sum().reset_index()
monthly_profit['Date'] = monthly_profit['Date'].astype(str)  # Convert to String for X-Axis
fig = px.line(
    monthly_profit,
    x='Date',
    y='Profit',
    title='Monthly Profit Trend',
    labels={'Profit': 'Monthly Profit (£)', 'Date': 'Month'}
)
fig.show()
pet_profit = data.groupby('Pet')['Profit'].sum()
plt.figure(figsize=(5, 5))
plt.pie(pet_profit, labels=pet_profit.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Set3.colors)
plt.title('Total Profit Distribution by Pet Type')
plt.axis('equal')
plt.show()
plt.figure(figsize=(8,6))
sns.scatterplot(data=data, x="Units Sld", y="Profit", hue="Pet", palette="Dark2")
plt.title("Units Sold vs Total Profit")
plt.tight_layout()
plt.show()
heatmap_data = data.pivot_table(index="Area",
                    columns="Pet",
                    values="Profit",
                    aggfunc="sum")

plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlGnBu", linewidths=0.5)
plt.title("Total Profit by Store and Pet Type")
plt.xlabel("Pet Type")
plt.ylabel("Store Location")
plt.tight_layout()
plt.show()
```

Appendix: Streamlit Dashboard Code

```
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_excel('Fury_Friends data set_clean.xlsx', sheet_name='Sheet1')
df['Date'] = pd.to_datetime(df['Date'])  # Ensure 'Date' is datetime

st.set_page_config(page_title="Fury Friends Dashboard", layout="wide")
st.title("🐾 Fury Friends UK - Pet Store Profit Dashboard")
st.markdown("Analysis of profitability by pet type, store location, and sales performance across the UK.")
```

```
total_profit = df['Profit'].sum()

total_revenue = df['Revenue'].sum()

total_cost = df['Cost'].sum()

total_units = df['Units Sld'].sum()

col1, col2, col3, col4 = st.columns(4)

col1.metric("💸 Total Profit", f"€{int(total_profit):,}")

col2.metric("📈 Total Revenue", f"€{int(total_revenue):,}")

col3.metric("💰 Total Cost", f"€{int(total_cost):,}")

col4.metric("📦 Units Sold", f"{int(total_units):,}")

st.markdown("---")

view_option = st.radio("🔍 Select View Mode", ("Overview", "Filtered View"), horizontal=True)

if view_option == "Filtered View":

    selected_area = st.selectbox("🏬 Select Store Area", df['Area'].unique())

    st.markdown(f"**Showing data for:** {selected_area}")

    filtered = df[df['Area'] == selected_area].copy()

else:

    filtered = df.copy()

min_date = filtered['Date'].min().date()

max_date = filtered['Date'].max().date()




col5, col6 = st.columns(2)

start_date = col5.date_input("Start Date", min_value=min_date, max_value=max_date, value=min_date)

end_date   = col6.date_input("End Date", min_value=min_date, max_value=max_date, value=max_date)

filtered_time = filtered[

    (filtered['Date'].dt.date >= start_date) &
    (filtered['Date'].dt.date <= end_date)
].copy()

profit_by_area_pet = filtered.groupby(['Area', 'Pet'])['Profit'].sum().unstack()
custom_colors = ['#66C2A5', '#FC8D62', '#8DA0CB', '#E78AC3', '#A6D854', '#FFD92F']

fig1 = go.Figure()
for pet, color in zip(profit_by_area_pet.columns, custom_colors):
    fig1.add_trace(go.Bar(x=profit_by_area_pet.index, y=profit_by_area_pet[pet], name=pet, marker_color=color))

fig1.update_layout(
    title='Profit by Store Location and Pet Type',
    xaxis_title='Store Area',
    yaxis_title='Total Profit (€)',
    barmode='stack',
    legend_title='Pet Type',
    template="plotly_dark"
)
st.plotly_chart(fig1, use_container_width=True)

pet_profit = filtered.groupby('Pet')['Profit'].sum().sort_values(ascending=False)
fig2 = px.pie(
    pet_profit,
    names=pet_profit.index,
    values=pet_profit,
    title='💰 Total Profit Distribution by Pet Type',
    hole=0.35,
    color_discrete_sequence=custom_colors
```

```
)
fig2.update_traces(textinfo='label+percent+value', pull=[0.1]*len(pet_profit))
st.plotly_chart(fig2, use_container_width=True)

store_metrics = filtered.groupby('Area')[['Cost', 'Revenue', 'Profit']].sum().reset_index()
store_metrics.set_index('Area', inplace=True)
fig3, ax3 = plt.subplots(figsize=(12, 5))
ax3.plot(store_metrics.index, store_metrics['Cost'], marker='o', label='Cost', color='#FF6666')
ax3.plot(store_metrics.index, store_metrics['Revenue'], marker='s', label='Revenue', color='#66B2FF')
ax3.plot(store_metrics.index, store_metrics['Profit'], marker='^', label='Profit', color='#66FF66')
ax3.set_title('Cost, Revenue, and Profit by Area')
ax3.set_xlabel('Store Area')
ax3.set_ylabel('Amount (€)')
ax3.legend()
ax3.grid(True)
st.pyplot(fig3)
monthly = (
    filtered_time
    .groupby(filtered_time['Date'].dt.to_period('M'))['Profit']
    .sum()
    .reset_index()
)
monthly['Month'] = monthly['Date'].astype(str)

fig_month = px.line(
    monthly,
    x='Month', y='Profit',
    title='📅 Monthly Profit Trend (Filtered)',
    labels={'Profit': 'Monthly Profit (€)', 'Month': 'Month'},
    template="plotly_dark"
)
st.plotly_chart(fig_month, use_container_width=True)
chart_type = st.radio("📊 Select Chart Type", ["Scatter Plot", "Box Plot"], horizontal=True)
if chart_type == "Scatter Plot":
    fig5 = px.scatter(filtered, x="Units Sld", y="Profit", color="Pet", title="Units Sold vs Profit", template="plotly")
else:
    fig5 = px.box(filtered, x="Pet", y="Profit", title="Profit Distribution by Pet Type", template="plotly_dark")

st.plotly_chart(fig5, use_container_width=True)
heatmap_data = filtered.pivot_table(index="Area", columns="Pet", values="Profit", aggfunc="sum")
fig6, ax6 = plt.subplots(figsize=(10, 5))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlGnBu", linewidths=0.5, ax=ax6)
ax6.set_title("💡 Heatmap: Total Profit by Store and Pet Type")
st.pyplot(fig6)
top_areas = df.groupby('Area')['Profit'].sum().nlargest(5).reset_index()
fig_top_areas = px.bar(
    top_areas,
    x='Area',
    y='Profit',
    title='🏆 Top 5 Most Profitable Store Areas',
    color='Profit',
    color_continuous_scale='Viridis',
    labels={'Profit': 'Total Profit (€)'}
)
st.plotly_chart(fig_top_areas, use_container_width=True)
selected_pet = st.selectbox("🐶 Select Pet Type for Area-wise Profit", df['Pet'].unique())
```

```
pet_by_area = df[df['Pet'] == selected_pet].groupby('Area')['Profit'].sum().reset_index()

fig_pet_area = px.bar(

    pet_by_area,

    x='Area',

    y='Profit',

    title=f'Profit from {selected_pet} by Store Area',

    color='Profit',

    color_continuous_scale='Sunset'

)

st.plotly_chart(fig_pet_area, use_container_width=True)
```