Observations and Insights

In [34]:

```
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as st
import numpy as np
from scipy.stats import linregress

# Study data files
mouse_metadata_path = "Mouse_metadata.csv"
study_results_path = "Study_results.csv"

# Read the mouse data and the study results
mouse_metadata = pd.read_csv(mouse_metadata_path)
study_results = pd.read_csv(study_results_path)

# Combine the data into a single dataset
merge_table = pd.merge(mouse_metadata, study_results, on="Mouse ID", how="left")

# Display the data table for preview
merge_table.head()
```

Out[34]:

	Mouse ID	Drug Regimen	Sex	Age_months	Weight (g)	Timepoint	Tumor Volume (mm3)	Metastatic Sites
0	k403	Ramicane	Male	21	16	0	45.000000	0
1	k403	Ramicane	Male	21	16	5	38.825898	0
2	k403	Ramicane	Male	21	16	10	35.014271	1
3	k403	Ramicane	Male	21	16	15	34.223992	1
4	k403	Ramicane	Male	21	16	20	32.997729	1

In [35]:

```
# Checking the number of mice.
merge_table["Mouse ID"].value_counts().unique
```

Out[35]:

```
<bound method Series.unique of g989    13
a275    10
a685    10
l725    10
e476    10
       ..
v199     1
h428     1
f932     1
t573     1
b447     1
Name: Mouse ID, Length: 249, dtype: int64>
```

In [36]:

```
# Getting the duplicate mice by ID number that shows up for Mouse ID and Timepoint.
duplicateMice = merge_table.loc[merge_table.duplicated(subset=["Mouse ID", "Timepoint"]), "Mouse ID"].unique()
```

In [4]:

```
# Optional: Get all the data for the duplicate mouse ID.
```

In [37]:

```
# Create a clean DataFrame by dropping the duplicate mouse by its ID.
cleanDF = merge_table[merge_table["Mouse ID"].isin(duplicateMice) == False]

cleanDF
```

Out[37]:

	Mouse ID	Drug Regimen	Sex	Age_months	Weight (g)	Timepoint	Tumor Volume (mm3)	Metastatic Sites
0	k403	Ramicane	Male	21	16	0	45.000000	0
1	k403	Ramicane	Male	21	16	5	38.825898	0
2	k403	Ramicane	Male	21	16	10	35.014271	1
3	k403	Ramicane	Male	21	16	15	34.223992	1
4	k403	Ramicane	Male	21	16	20	32.997729	1
...	...	...	...	...	...	...	...	...
1888	z969	Naftisol	Male	9	30	25	63.145652	2
1889	z969	Naftisol	Male	9	30	30	65.841013	3
1890	z969	Naftisol	Male	9	30	35	69.176246	4
1891	z969	Naftisol	Male	9	30	40	70.314904	4
1892	z969	Naftisol	Male	9	30	45	73.867845	4

1880 rows × 8 columns

In [38]:

```
# Checking the number of mice in the clean DataFrame.
cleanDF["Mouse ID"].value_counts().unique
```

Out[38]:

```
<bound method Series.unique of a275    10
a203    10
e476    10
r944    10
a520    10
        ..
u153     1
v199     1
b447     1
f932     1
x336     1
Name: Mouse ID, Length: 248, dtype: int64>
```

Summary Statistics

In [7]:

```
# Generate a summary statistics table of mean, median, variance, standard deviation, and SEM of the tumor volume for each regimen

# This method is the most straighforward, creating multiple series and putting them all together at the end.
```

In [8]:

```
# Generate a summary statistics table of mean, median, variance, standard deviation, and SEM of the tumor volume for each regimen

# This method produces everything in a single groupby function
```

In [39]:

```
newDF = cleanDF.loc[:, ["Mouse ID","Drug Regimen","Tumor Volume (mm3)"]]

mean = newDF.groupby(["Drug Regimen"]).mean()["Tumor Volume (mm3)"]
median = newDF.groupby(["Drug Regimen"]).median()["Tumor Volume (mm3)"]
variance = newDF.groupby(["Drug Regimen"]).var()["Tumor Volume (mm3)"]
sdev = newDF.groupby(["Drug Regimen"]).std()["Tumor Volume (mm3)"]
sem = newDF.groupby(["Drug Regimen"]).sem()["Tumor Volume (mm3)"]

sumStatDF = pd.DataFrame({"mean TV":mean,"median TV":median,"variance":variance,"sd":sdev,"sem":sem})

sumStat = sumStatDF.round(2)

sumStat
```

Out[39]:

Drug Regimen	mean TV	median TV	variance	sd	sem
Capomulin	40.68	41.56	24.95	4.99	0.33
Ceftamin	52.59	51.78	39.29	6.27	0.47
Infubinol	52.88	51.82	43.13	6.57	0.49
Ketapril	55.24	53.70	68.55	8.28	0.60
Naftisol	54.33	52.51	66.17	8.13	0.60
Placebo	54.03	52.29	61.17	7.82	0.58
Propriva	52.32	50.45	43.85	6.62	0.54
Ramicane	40.22	40.67	23.49	4.85	0.32
Stelasyn	54.23	52.43	59.45	7.71	0.57
Zoniferol	53.24	51.82	48.53	6.97	0.52

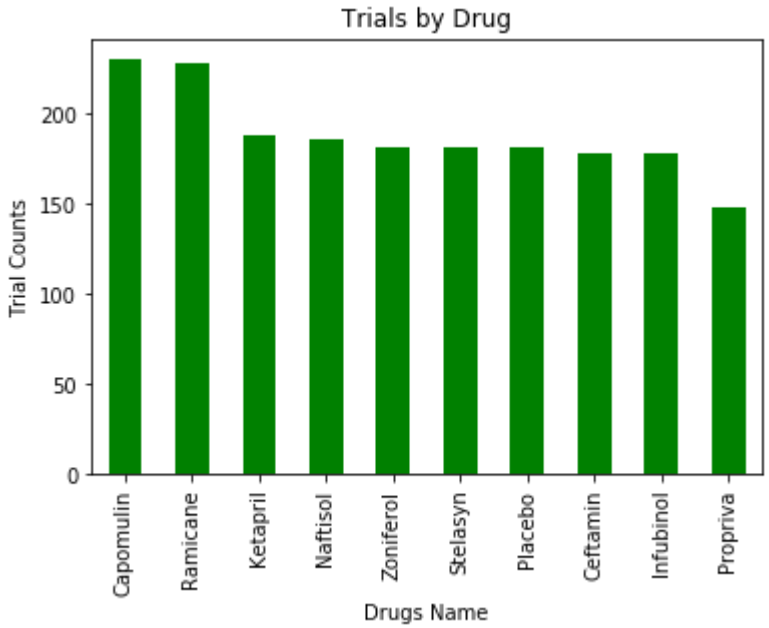
Bar and Pie Charts

In [40]:

```
# Generate a bar plot showing the total number of mice for each treatment throughout the course of the study using panda
s.
drugCounts = cleanDF["Drug Regimen"].value_counts()

y_axis = drugCounts.values
x_axis = drugCounts.index
drugCounts.plot(kind = "bar", facecolor = "green",)

plt.ylabel("Trial Counts")
plt.xlabel("Drugs Name")
plt.title("Trials by Drug")
plt.xticks(rotation = 90)
plt.show()
```



In [41]:

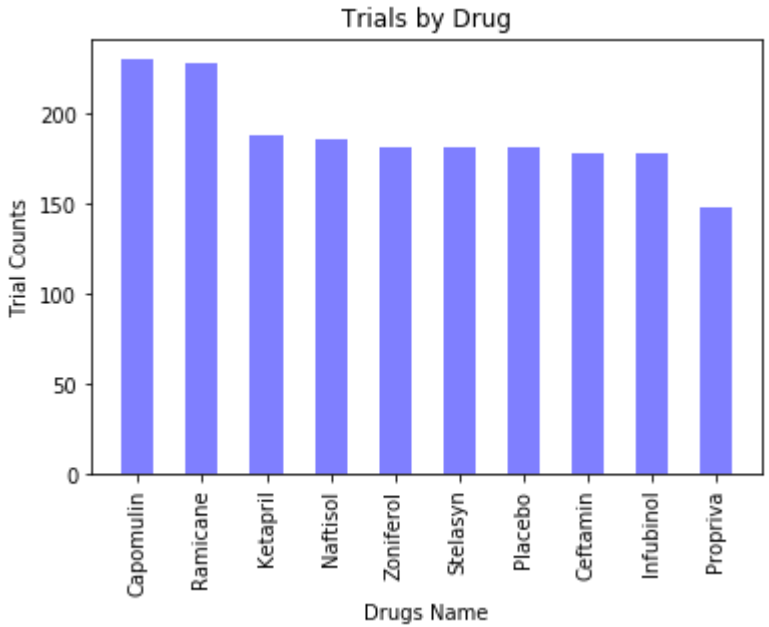
```
# Generate a bar plot showing the total number of mice for each treatment throughout the course of the study using pyplo
t.
drugCounts = cleanDF["Drug Regimen"].value_counts()

y_axis = drugCounts.values
x_axis = np.arange(0, len(y_axis),1)
x_drugs = drugCounts.index

plt.bar(x_drugs, y_axis, color = "b", alpha = 0.5, align = "center", width = .5)

plt.ylabel("Trial Counts")
plt.xlabel("Drugs Name")
plt.title("Trials by Drug")
plt.xticks(rotation = 90)

plt.show()
```

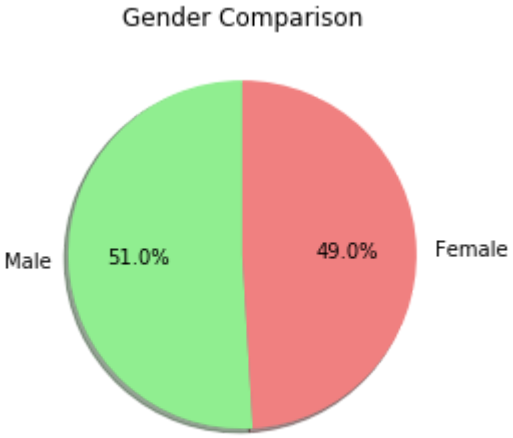


In [42]:

```
# Generate a pie plot showing the distribution of female versus male mice using pandas
maFeCounts = cleanDF["Sex"].value_counts()
labels = maFeCounts.values
sizes = maFeCounts.values
colors = ["lightgreen", "lightcoral"]

maFeCounts.plot(kind = "pie", colors = colors, autopct = "%1.1f%%", shadow = True, startangle = 90)

plt.title("Gender Comparison")
plt.ylabel(" ")
plt.show()
```



In [43]:

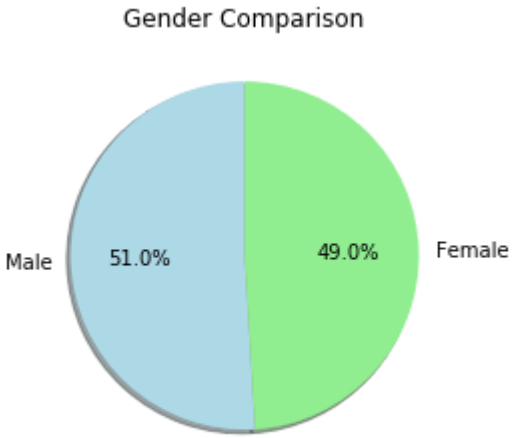
```
# Generate a pie plot showing the distribution of female versus male mice using pyplot
maFeCounts = cleanDF["Sex"].value_counts()
label = maFeCounts.index
size = maFeCounts.values

colors = ["lightblue", "lightgreen"]

plt.pie(sizes, labels = label, colors = colors, autopct = "%1.1f%%", shadow = True, startangle = 90)

plt.title("Gender Comparison")

plt.show()
```



Quartiles, Outliers and Boxplots

In [13]:

```
# Calculate the final tumor volume of each mouse across four of the treatment regimens:
# Capomulin, Ramicane, Infubinol, and Ceftamin
# Start by getting the last (greatest) timepoint for each mouse
# Merge this group df with the original dataframe to get the tumor volume at the last timepoint
```

In [44]:

```
finalTumor = cleanDF.groupby("Mouse ID").max().reset_index()
mergeD = finalTumor[["Mouse ID", "Timepoint"]].merge(cleanDF, on = ["Mouse ID", "Timepoint"], how = "left")
mergeD.head().round(2)
```

Out[44]:

	Mouse ID	Timepoint	Drug Regimen	Sex	Age_months	Weight (g)	Tumor Volume (mm3)	Metastatic Sites
0	a203	45	Infubinol	Female	20	23	67.97	2
1	a251	45	Infubinol	Female	21	25	65.53	1
2	a262	45	Placebo	Female	17	29	70.72	4
3	a275	45	Ceftamin	Female	20	28	63.00	3
4	a366	30	Stelasyn	Female	16	29	63.44	1

In [63]:

```
capomulin = mergeD.loc[mergeD["Drug Regimen"] == "Capomulin"]["Tumor Volume (mm3)"]
ramicane = mergeD.loc[mergeD["Drug Regimen"] == "Ramicane"]["Tumor Volume (mm3)"]
infubinol = mergeD.loc[mergeD["Drug Regimen"] == "Infubinol"]["Tumor Volume (mm3)"]
ceftamin = mergeD.loc[mergeD["Drug Regimen"] == "Ceftamin"]["Tumor Volume (mm3)"]

quartileCap = capomulin.quantile([.25,.5,.75])
lowerCap = quartileCap[.25]
upperCap = quartileCap[.75]
iqr = upperCap-lowerCap

lowerCap = lowerCap-(1.5*iqr)
upperCap = upperCap+(1.5*iqr)

print("outlier: any number below ",format(lowerCap,".2f"))
print("outlier: any number above ",format(upperCap,".2f"))
```

```
outlier: any number below  20.70
outlier: any number above  51.83
```

In [14]:

```
# Put treatments into a list for for loop (and later for plot labels)

# Create empty list to fill with tumor vol data (for plotting)

# Calculate the IQR and quantitatively determine if there are any potential outliers.

    # Locate the rows which contain mice on each drug and get the tumor volumes

    # add subset

    # Determine outliers using upper and lower bounds
```

In [62]:

```
quartileRam = ramicane.quantile([.25,.5,.75])
lowerRam = quartileRam[.25]
upperRam = quartileRam[.75]
iqr = upperRam - lowerRam

lowerRam = lowerRam-(1.5*iqr)
upperRam = upperRam+(1.5*iqr)

print("outlier: any number below ",format(lowerRam,".2f"))
print("outlier: any numbre above ",format(upperRam,".2f"))
```

```
outlier: any number below  17.91
outlier: any numbre above  54.31
```

In [61]:

```
quartileIn = infubinol.quantile([.25,.5,.75])
lowerIn = quartileIn[.25]
upperIn = quartileIn[.75]
iqr = upperIn - lowerIn

lowerIn = lowerIn-(1.5*iqr)
upperIn = upperIn+(1.5*iqr)

print("outlier: any number below ",format(lowerIn,".2f"))
print("outlier: any number above ",format(upperIn,".2f"))
```

```
outlier: any number below  36.83
outlier: any number above  82.74
```

In [59]:

```
quartileCef = ceftamin.quantile([.25,.5,.75])
lowerCef = quartileCef[.25]
upperCef = quartileCef[.75]
iqr = upperCef - lowerCef

lowerCef = lowerCef - (1.5*iqr)
upperCef = upperCef + (1.5*iqr)

print("outlier: any number below ",format(lowerCef,".2f"))
print("outlier: any number above ",format(upperCef,".2f"))
```

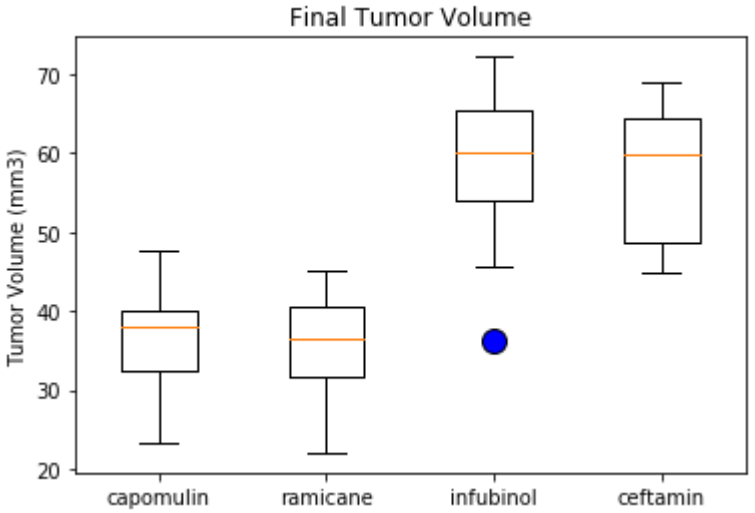
```
outlier: any number below  25.36
outlier: any number above  87.67
```

In [49]:

```
# Generate a box plot of the final tumor volume of each mouse across four regimens of interest
orange_out = dict(markerfacecolor = "blue", markersize = 12)

plt.boxplot([capomulin, ramicane, infubinol, ceftamin], labels = ["capomulin", "ramicane", "infubinol", "ceftamin"], flie
rprops = orange_out)

plt.title("Final Tumor Volume")
plt.ylabel("Tumor Volume (mm3)")
plt.show()
```



Line and Scatter Plots

In [16]:

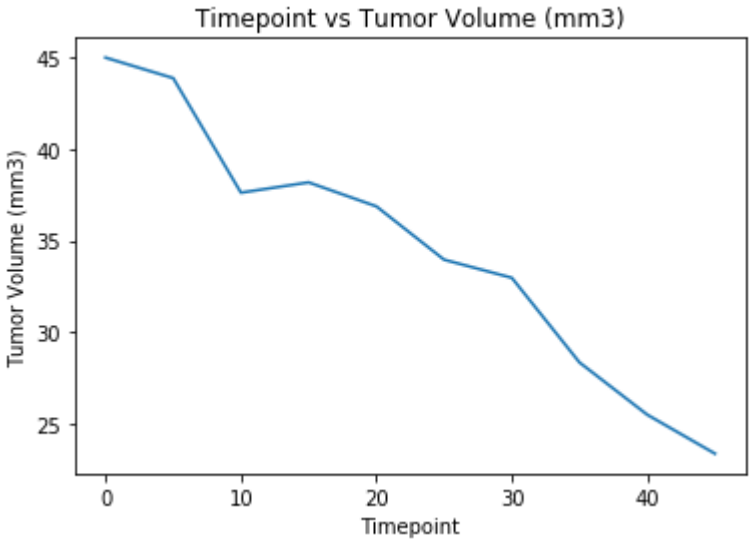
```
# Generate a line plot of time point versus tumor volume for a mouse treated with Capomulin
```

In [50]:

```
capomulinTab = cleanDF.loc[cleanDF["Drug Regimen"] == "Capomulin"]
mouse = cleanDF.loc[cleanDF["Mouse ID"] == "s185"]

plt.plot(mouse["Timepoint"], mouse["Tumor Volume (mm3)"])
plt.xlabel("Timepoint")
plt.ylabel("Tumor Volume (mm3)")
plt.title("Timepoint vs Tumor Volume (mm3)")

plt.show()
```



In [17]:

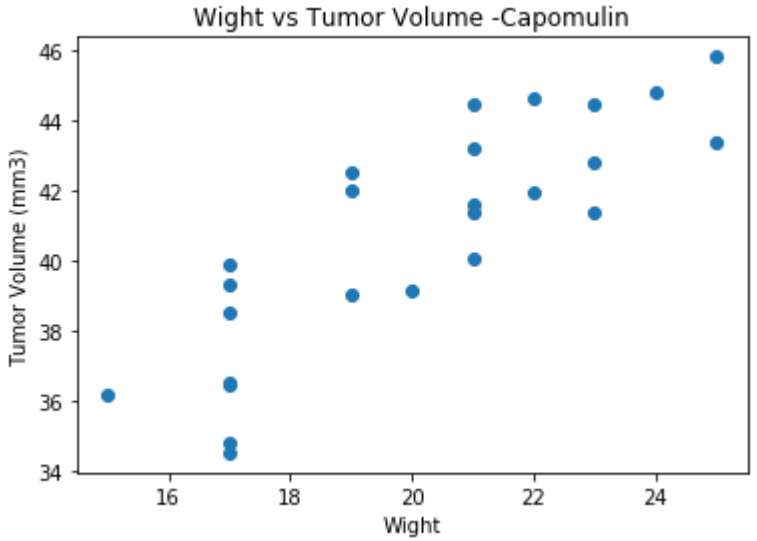
```
# Generate a scatter plot of mouse weight versus average tumor volume for the Capomulin regimen
```

In [51]:

```
capomulinWe = cleanDF.loc[cleanDF["Drug Regimen"] == "Capomulin"]
capAvg = capomulinWe.groupby(["Mouse ID"]).mean()

plt.scatter(capAvg["Weight (g)"], capAvg["Tumor Volume (mm3)"])
plt.xlabel("Wight")
plt.ylabel("Tumor Volume (mm3)")
plt.title("Wight vs Tumor Volume -Capomulin")

plt.show()
```



Correlation and Regression

In [18]:

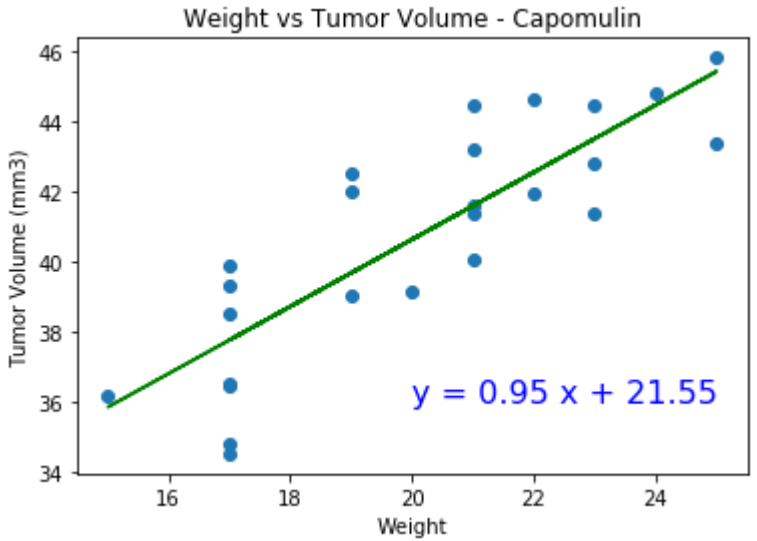
```
# Calculate the correlation coefficient and linear regression model
# for mouse weight and average tumor volume for the Capomulin regimen
```

In [53]:

```
(slope, intercept, rvalue, pvalue, stderr) = linregress(capAvg["Weight (g)"], capAvg["Tumor Volume (mm3)"])
regressValues = capAvg["Weight (g)"]* slope + intercept
lineEq = f"y = {round(slope, 2)} x + {round(intercept, 2)}"

plt.scatter(capAvg["Weight (g)"], capAvg["Tumor Volume (mm3)"])
plt.plot(capAvg["Weight (g)"], regressValues, color = "green")
plt.annotate(lineEq,(20,36), color = "blue", fontsize=16)
plt.xlabel("Weight")
plt.ylabel("Tumor Volume (mm3)")
plt.title("Weight vs Tumor Volume - Capomulin")

plt.show()
```



In [64]:

```
print("Correlation coefficient = ",format(rvalue,".2f"))
```

```
Correlation coefficient =  0.84
```

In [ ]: