# SpaceX

```
['Flight No.', 'Date andtime (UTC)', 'Version,Booster [b]', 'Launch site', 'Payload[c]', 'Payload mass', 'Orbit', 'Customer',
'Launchoutcome', 'Boosterlanding', '1', '2', '3', '4', '5', '6', '7']
```

Starting from the third table is our target table contains the actual launch records.

In [9]:
```
# Let's print the third table and check its content
##first_launch_table = column_names[2]
##print(first_launch_table)
html_tables = soup.find_all('table')

# Print the third table and its content
print(html_tables[2])
```

```
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<
br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
<th scope="col">Launch<br/>outcome
</th>
<th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9 first-stage landing tests">Booster<br/>lan
ding</a>
</th></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">1
```

# Getting all the data and adding it to a dataframe

```
            booster_landing = landing_status(row[8])
            #print(booster_landing)
```

```
F9 B5 △
F9 B5
F9 B5B1051.8
F9 B5B1058.5
F9 B5 △
F9 B5 △
F9 B5 △
F9 B5 △
F9 B5 △
F9 B5B1060.6
F9 B5 △
F9 B5B1061.2
F9 B5B1060.7
F9 B5B1049.9
F9 B5B1051.10
F9 B5B1058.8
F9 B5B1063.2
F9 B5B1067.1
F9 B5
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

In [14]:
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df
```

Out[14]:

| Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome | None | Version Booster | Booster landing | Date | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|

# Data wrangling

Identify and calculate the percentage of the missing values in each attribute

```
In [4]: df.isnull().sum()/len(df)*100
```

```
Out[4]: FlightNumber       0.000000
        Date               0.000000
        BoosterVersion     0.000000
        PayloadMass        0.000000
        Orbit              0.000000
        LaunchSite         0.000000
        Outcome            0.000000
        Flights            0.000000
        GridFins           0.000000
        Reused             0.000000
        Legs               0.000000
        LandingPad        28.888889
        Block              0.000000
        ReusedCount        0.000000
        Serial             0.000000
        Longitude          0.000000
        Latitude           0.000000
        dtype: float64
```

Identify which columns are numerical and categorical:

# Checking the data types.

```
In [5]: df.dtypes
```

```
Out[5]: FlightNumber        int64
        Date               object
        BoosterVersion     object
        PayloadMass       float64
        Orbit              object
        LaunchSite         object
        Outcome            object
        Flights             int64
        GridFins             bool
        Reused               bool
        Legs                 bool
        LandingPad         object
        Block             float64
        ReusedCount         int64
        Serial             object
        Longitude         float64
        Latitude          float64
        dtype: object
```

# Checking the data in the colomns

In [8]:
```
boost=df["Orbit"]
boost
f9=boost.value_counts('None')
print(f9)
```

```
GTO      0.300000
ISS      0.233333
VLEO     0.155556
PO       0.100000
LEO      0.077778
SSO      0.055556
MEO      0.033333
ES-L1    0.011111
HEO      0.011111
SO       0.011111
GEO      0.011111
Name: Orbit, dtype: float64
```

In [9]:
```
# Apply value_counts() on column LaunchSite
count=df['LaunchSite'].value_counts()
count
```

Out[9]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

# Using SQL on the data to retrieve results

- Establishing the connection.

In [3]:
```
%load_ext sql
```

In [4]:
```
import csv, sqlite3

con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

In [5]:
```
!pip install -q pandas==1.1.5
```

In [6]:
```
%sql sqlite:///my_data1.db
```

Out[6]:
```
'Connected: @my_data1.db'
```

In [7]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/dat
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:2882: UserWarning: The spaces in these co
lumn names will not be changed. In pandas versions < 0.14, spaces were converted to underscores.
  both result in 0.1234 being formatted as 0.12.
```

**Note:This below code is added to remove blank rows from table**

In [8]:
```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

```
 * sqlite:///my_data1.db
Done.
```

# Data retrieval with SQL.

```
In [12]: %sql select * from SPACEXTBL WHERE Launch_Site like 'CCA%' limit 5
```

```
 * sqlite:///my_data1.db
Done.
```

Out[12]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

**Task 3**

***Display the total payload mass carried by boosters launched by NASA (CRS)***

```
In [16]: %sql select SUM(PAYLOAD_MASS__KG_) AS SUM FROM SPACEXTBL
```

```
 * sqlite:///my_data1.db
Done.
```

Out[16]:

| SUM |
|---|
| 619967 |

**Task 4**

In [16]:
```
%sql select SUM(PAYLOAD_MASS__KG_) AS SUM FROM SPACEXTBL
```

```
 * sqlite:///my_data1.db
Done.
```

Out[16]:

| SUM |
|---|
| 619967 |

## Task 4

***Display average payload mass carried by booster version F9 v1.1***

In [18]:
```
%sql select AVG(PAYLOAD_MASS__KG_) AS AVG FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[18]:

| AVG |
|---|
| 2928.4 |

In [23]:
```
%sql select min(Date) from SPACEXTBL where Mission_Outcome = 'Success'
```

```
 * sqlite:///my_data1.db
Done.
```

Out[23]:

| min(Date) |
|---|
| 2010-06-04 |

***Task 6***

***List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000***

In [25]:
```
%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ > 4000 AND  PAYLOAD_MASS__KG_ < 6000
```

```
 * sqlite:///my_data1.db
Done.
```

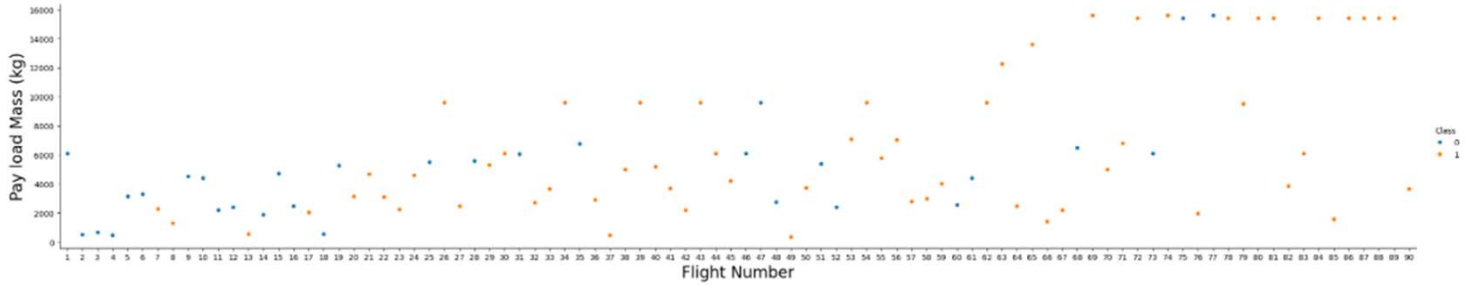
Out[25]:

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| F9 v1.1 | 4535 |
| F9 v1.1 B1011 | 4428 |
| F9 v1.1 B1014 | 4159 |
| F9 v1.1 B1016 | 4707 |
| F9 FT B1020 | 5271 |
| F9 FT B1022 | 4696 |
| F9 FT B1026 | 4600 |
| F9 FT B1030 | 5600 |
| F9 FT B1021.2 | 5300 |
| F9 FT B1032.1 | 5300 |
| F9 B4 B1040.1 | 4990 |

# Exploratory data analysis.

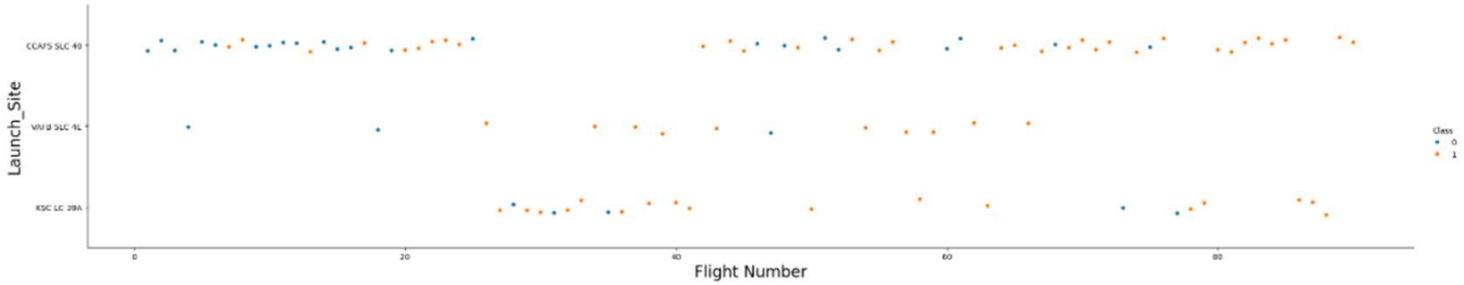
We can plot out the `FlightNumber` vs. `PayloadMass` and overlay the outcome of the launch. We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

In [7]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

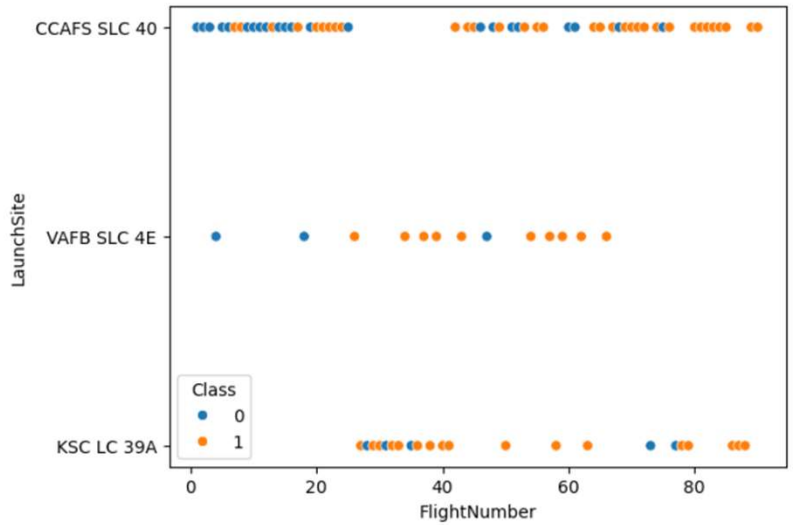


Next, let's drill down to each site visualize its detailed launch records.

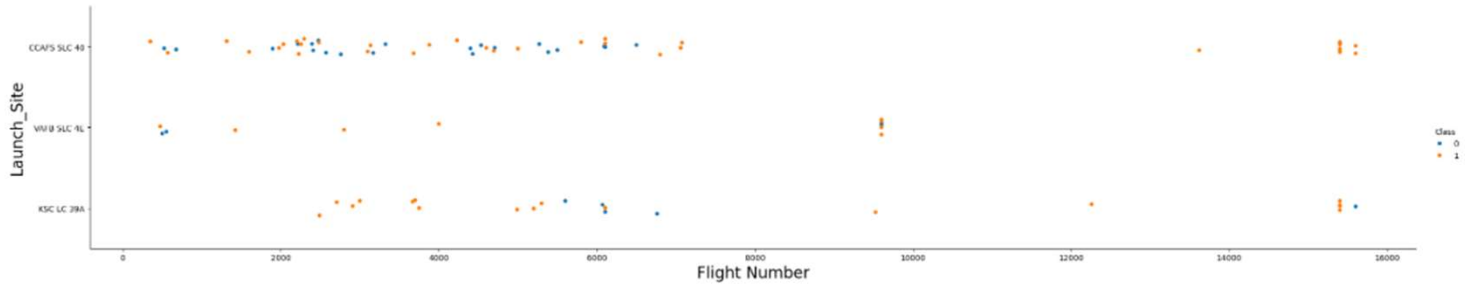
In [8]:
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch_Site",fontsize=20)
plt.show()
```



```
# Assuming you have a DataFrame named df containing the required data

# Plot scatter point chart
sns.scatterplot(data=df, x='FlightNumber', y='LaunchSite', hue='Class')

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

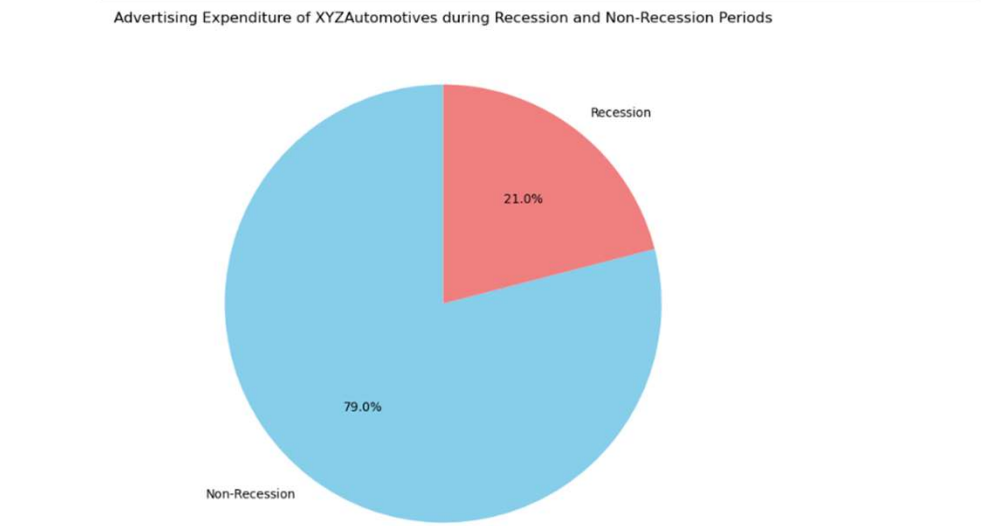


In [12]:
```
### TASK 2: Visualize the relationship between Payload and Launch Site

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch_Site",fontsize=20)
plt.show()
```

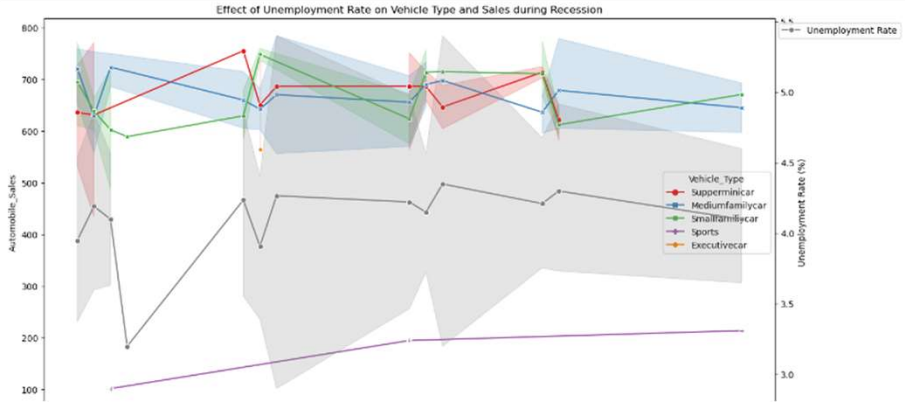


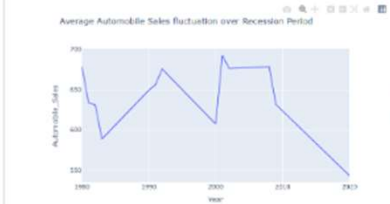
# Folium Results

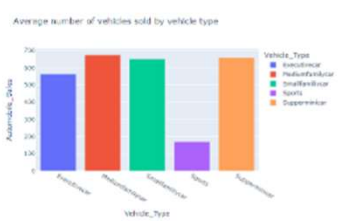
```
# Create a pie chart
labels = ['Non-Recession', 'Recession']
colors = ['skyblue', 'lightcoral']

plt.figure(figsize=(8, 8))
plt.pie(total_expenditure, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title('Advertising Expenditure of XYZAutomotives during Recession and Non-Recession Periods')

# Display the pie chart
plt.show()
```



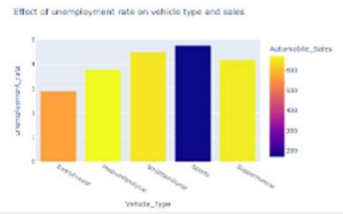
In [58]:
```
recession_data = df[df['Recession'] == 1]

# Using Seaborn to create a line plot
plt.figure(figsize=(15, 8))
sns.lineplot(x='Year', y='Automobile_Sales', hue='Vehicle_Type', style='Vehicle_Type', markers=True, dashes=False, data=recessior

# Add a second y-axis for the unemployment rate
ax2 = plt.gca().twinx()
sns.lineplot(x='Year', y='unemployment_rate', data=recession_data, color='gray', ax=ax2, marker='o', label='Unemployment Rate')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Automobile Sales')
plt.title('Effect of Unemployment Rate on Vehicle Type and Sales during Recession')
ax2.set_ylabel('Unemployment Rate (%)')

# Display the plot
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```

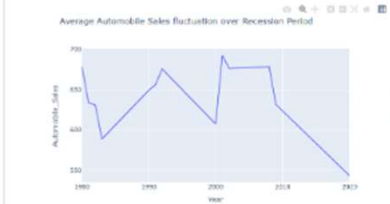


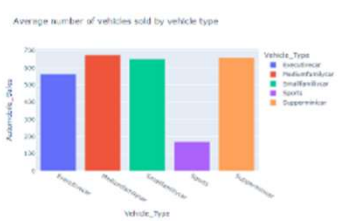

Average Automobile Sales fluctuation over Recession Period
Automobile_Sales
Year


Average number of vehicles sold by vehicle type
Automobile_Sales
Vehicle_Type
Vehicle_Type
Sports

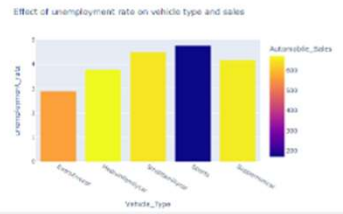

Total expenditure share by vehicle type during recessions
MidsizeFamilycar
SmallFamilycar
Supermidicar
Sports
Executivecar
38.4%
24.2%
24.7%
0.914%


Effect of unemployment rate on vehicle type and sales
unemployment_rate
Vehicle_Type
Automobile_Sales


Average Automobile Sales fluctuation over Recession Period
Automobile_Sales
Year


Average number of vehicles sold by vehicle type
Automobile_Sales
Vehicle_Type
Vehicle_Type
Sports


Total expenditure share by vehicle type during recessions
MidsizeFamilycar
SmallFamilycar
Supermidicar
Sports
Executivecar
38.4%
24.2%
24.7%
0.914%


Effect of unemployment rate on vehicle type and sales
unemployment_rate
Vehicle_Type
Automobile_Sales