IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

Executive Summary

- Summary of methodologies
  - Data collected, cleaned, and prepared using Python (Pandas, NumPy).
  - Exploratory Data Analysis (EDA) with visual analytics (Matplotlib, Seaborn).
  - SQL queries for structured data insights.
  - Interactive maps with Folium, dashboards with Plotly Dash.
  - Predictive classification model for data insights.
- Summary of all results
  - Clear patterns and trends identified in data.
  - SQL analysis provided deep insights into key metrics.
  - Interactive visuals improved data understanding.
  - Classification model achieved strong accuracy.

Introduction

- Project background and context
    - Analysis of SpaceX launch data to understand factors affecting mission success.
    - Focus on launch sites, payload, and booster performance.
- Problems you want to find answers
    - What factors influence a successful SpaceX launch?
    - How do launch sites impact mission outcomes?
    - Does payload mass affect success rate?
    - Which booster types perform best?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Collected SpaceX launch data from public datasets and APIs.
  - Merged historical records with additional payload and booster details.
- Perform data wrangling:
  - Cleaned missing values and standardized formats.
  - Extracted relevant features like launch site, payload mass, and success status.
- Perform exploratory data analysis (EDA) using visualization and SQL:
  - Used SQL queries to analyze launch trends and success rates.
  - Created visualizations to explore relationships between payload, booster type, and success.
- Perform interactive visual analytics using Folium and Plotly Dash:
  - Developed an interactive Folium map showing launch sites and success rates.
  - Built a Plotly Dash dashboard for dynamic data exploration.
- Perform predictive analysis using classification models:
  - Built machine learning models (Logistic Regression, Decision Tree, etc.) to predict launch success.
  - Tuned hyperparameters for optimization and evaluated performance using accuracy and precision metrics.

Data Collection

- Describe how data sets were collected.
  - Collected from SpaceX public APIs and Kaggle datasets.
  - Supplemented with external data (payload mass, booster version).
- You need to present your data collection process use key phrases and flowcharts
  - Step 1: Extracted raw data from APIs and CSV files.
  - Step 2: Merged multiple datasets into a unified table.
  - Step 3: Cleaned missing values and ensured data consistency.

Data Collection – SpaceX API

- **Data Collection Process:**
- **Step 1:** Send REST API requests to SpaceX endpoints.
- Example endpoint: https://api.spacexdata.com/v4/launches
- **Step 2:** Collect key launch data including:
- Launch ID, flight number, date, success status, booster information, and landing status.
- **Step 3:** Store the data in a structured format (JSON, CSV).
- **Step 4:** Clean and preprocess the data for analysis.

- **GitHub URL for SpaceX API Calls Notebook:**

SpaceX Data Collection - API Calls Notebook

Flowchart: SpaceX API Calls

1. **Start**

2. **Call SpaceX API**

→ Fetch launch data

3. **Check API Response**

→ If successful, move to step 4; If failed, retry or log error

4. **Store Data in Structured Format**

→ Store in CSV/JSON

5. **Data Preprocessing**

→ Handle missing values, convert to appropriate data types

6. **End**

Data Collection - Scraping

Web Scraping Process:

- **Step 1:** Identify the source of web data.
  - Target websites or pages with launch details (e.g., SpaceX launch data or news sites).
- **Step 2:** Send HTTP request to the target website.
  - Use Python libraries like requests or urllib to fetch the HTML content of the webpage.
- **Step 3:** Parse the HTML content.
  - Use libraries like BeautifulSoup or lxml to parse and extract relevant data (e.g., landing success, booster information, etc.).
- **Step 4:** Extract and clean data.
  - Find specific tags, attributes, and structures (e.g., <table>, <div>, etc.) containing launch data and landing status.
  - Clean the data by removing unnecessary or malformed entries.
- **Step 5:** Store data in structured format (CSV/JSON).
  - Store the parsed data in a suitable format for further analysis and modeling.
- **Step 6:** Use the data for analysis and model training.

- **GitHub URL for Web Scraping Notebook:**

SpaceX Data Collection - Web Scraping Notebook

Flowchart: Web Scraping Process

1. **Start**

2. **Send HTTP request**

→ Fetch HTML content from target website

3. **Parse HTML with BeautifulSoup**

→ Extract key information (landing status, launch info)

4. **Clean Data**

→ Remove unnecessary tags, fix malformed entries

5. **Store Data**

→ Save in CSV/JSON format

6. **End**

Data Wrangling

- **Step 1: Load the Data**
  - Load the data into a Pandas DataFrame from sources such as CSV, JSON, or API.
- **Step 2: Inspect Data**
  - Inspect the raw data for missing values, incorrect data types, and outliers.
- **Step 3: Handle Missing Values**
  - Remove or impute missing values to ensure consistency across the dataset.
- **Step 4: Convert Categorical Data**
  - Convert categorical columns (e.g., True Ocean, False RTLS) into binary labels (1 for success, 0 for failure).
- **Step 5: Label Creation**
  - Create a new column for the target label (Landed) by combining all conditions (e.g., successful landing to ocean, pad, or drone ship).
  - Assign 1 for successful landings and 0 for unsuccessful landings.
- **Step 6: Data Formatting**
  - Ensure the dataset is correctly formatted with appropriate data types for analysis and modeling (e.g., integers, floats).
- **Step 7: Data Splitting**
  - Split the data into features (independent variables) and target labels (dependent variable).
- **Step 8: Final Dataset**
  - Prepare the final cleaned and processed dataset for training machine learning models.
- **GitHub URL for Data Wrangling Notebook:**

SpaceX Data Wrangling Notebook

EDA with Data Visualization

- **Charts Plotted and Their Purpose:**
- **Bar Chart: Rocket Launch Outcomes**
  - **Purpose:** To show the number of successful vs unsuccessful rocket launches.
  - **Why Used:** Provides a clear comparison of outcomes, helping to understand the distribution of successful and failed missions.
- **Pie Chart: Distribution of Landing Outcomes**
  - **Purpose:** To visualize the proportions of landing outcomes (ocean, RTLS, and ASDS).
  - **Why Used:** Helps identify how often each type of landing was successful or failed.
- **Scatter Plot: Launch Success vs. Payload Mass**
  - **Purpose:** To analyze if there's a correlation between the payload mass and the likelihood of a successful launch.
  - **Why Used:** To explore potential relationships between payload size and landing success/failure.
- **Histogram: Launch Success by Year**
  - **Purpose:** To understand the trend of successful launches over time.
  - **Why Used:** Helps identify patterns or improvements in launch success as SpaceX progresses with its technology.
- **Heatmap: Correlation Matrix of Features**
  - **Purpose:** To visualize relationships between numerical features (such as payload mass, launch success, etc.).
  - **Why Used:** Identifying which variables have the strongest correlations helps in feature selection for predictive modeling.
- **Box Plot: Launch Success by Booster Version**
  - **Purpose:** To compare the launch success rate across different booster versions.
  - **Why Used:** To evaluate if certain boosters had higher success rates, helping inform decisions about future missions.
- **GitHub URL for EDA with Data Visualization Notebook:**

SpaceX EDA and Data Visualization Notebook

EDA with SQL

SQL Queries Performed:

- **Query 1: Count of Successful vs Unsuccessful Launches**
  - **SQL:** SELECT launch_success, COUNT(*) FROM spacex_data GROUP BY launch_success;
  - **Purpose:** To count the number of successful vs unsuccessful launches.
- **Query 2: Distribution of Launches by Booster Version**
  - **SQL:** SELECT booster_version, COUNT(*) FROM spacex_data GROUP BY booster_version;
  - **Purpose:** To analyze the distribution of launches across different booster versions.
- **Query 3: Landing Outcomes by Type**
  - **SQL:** SELECT landing_type, COUNT(*) FROM spacex_data GROUP BY landing_type;
  - **Purpose:** To investigate the frequency of each landing type (RTLS, ASDS, Ocean) for the launches.
- **Query 4: Successful Landings by Launch Year**
  - **SQL:** SELECT strftime('%Y', launch_date) AS year, COUNT(*) FROM spacex_data WHERE landing_success = 1 GROUP BY year;
  - **Purpose:** To get the count of successful landings per year.
- **Query 5: Average Payload Mass per Launch**
  - **SQL:** SELECT AVG(payload_mass) FROM spacex_data WHERE payload_mass IS NOT NULL;
  - **Purpose:** To find the average payload mass for the launches in the dataset.
- **Query 6: Identifying Launches with Specific Landing Outcomes**
  - **SQL:** SELECT * FROM spacex_data WHERE landing_success IS NULL;
  - **Purpose:** To identify the launches with missing landing success data, which need further investigation or cleaning.
- **Query 7: Correlation Between Payload Mass and Landing Success**
  - **SQL:** SELECT payload_mass, landing_success FROM spacex_data;
  - **Purpose:** To explore any potential correlation between payload mass and landing success.
- **GitHub URL for EDA with SQL Notebook:** SpaceX EDA with SQL Notebook

Build an Interactive Map with Folium

Map Objects Created:

- **Markers for Launch Sites:**
  - **What I Added:** Markers for each SpaceX launch site, including their name and location.
  - **Why:** These markers help to visually identify the launch sites on the map, making it easy to spot where the launches are occurring and analyze any geographical patterns.
- **Circles for Success/Failure Rate:**
  - **What I Added:** Circular markers with radius proportional to the success rate of launches at each site.
  - **Why:** These circles help visualize the success/failure rate of launches at each site. Larger circles represent higher success rates, and smaller circles represent lower success rates, allowing for quick comparison between sites.
- **Color-coded Markers for Launch Outcomes:**
  - **What I Added:** Color-coded markers (green for successful launches, red for failed launches).
  - **Why:** To visually differentiate the success and failure of individual launches at each site, helping to easily spot trends related to launch outcomes by site.
- **Lines Connecting Launch Sites:**
  - **What I Added:** Lines connecting launch sites with key nearby locations or areas of interest, such as proximity to the ocean or other geographical points.
  - **Why:** These lines help analyze how the proximity of the launch sites to specific areas affects the success or failure of the launch. It may also highlight strategic factors for site locations.

- **GitHub URL for Completed Interactive Map with Folium:**

Interactive Map with Folium Notebook

Build a Dashboard with Plotly Dash

Plots/Graphs Added:

- **Success Rate Pie Chart:**
  - **What I Added:** A pie chart that shows the proportion of successful vs. unsuccessful launches based on the selected launch site.
  - **Why:** This helps users quickly visualize the overall success rate of launches at a given site. It allows users to make comparisons between different sites interactively.
- **Success vs Payload Scatter Plot:**
  - **What I Added:** A scatter plot showing the relationship between the payload mass and the success/failure of the launch.
  - **Why:** This plot helps analyze how payload mass might influence the success rate of the launch, allowing users to see any patterns or trends.

Interactions Added:

- **Launch Site Dropdown:**
  - **What I Added:** A dropdown input component to allow users to select a specific launch site.
  - **Why:** This lets users filter the visualizations based on their selected launch site, enabling a focused analysis on each site's performance.
- **Payload Range Slider:**
  - **What I Added:** A range slider to select the payload mass range.
  - **Why:** This allows users to adjust the payload size interactively and see how the success rate correlates with different payload masses, giving them more control over the analysis.

GitHub URL for Completed Plotly Dash Dashboard:

Plotly Dash Dashboard Notebook

Predictive Analysis (Classification)

Model Development Process:

- **Exploratory Data Analysis (EDA):**
  - **What I Did:**
    - Examined the dataset to identify relevant features (such as payload, orbit, and location) and understand the relationship with the landing success.
    - Created a column to define the success of the first stage landing (1 for success, 0 for failure).
  - **Why:** EDA helps in understanding the dataset, uncovering hidden patterns, and determining the features that impact the classification task.
- **Data Preprocessing:**
  - **What I Did:**
    - Standardized the features to ensure they are on the same scale for better model performance.
    - Split the dataset into training and testing sets to evaluate the model's performance on unseen data.
  - **Why:** Standardization improves the convergence of machine learning models. Data splitting ensures that the model is evaluated effectively on unseen data to test its generalization.
- **Model Training:**
  - **What I Did:**
    - Applied three different classification algorithms: **Support Vector Machine (SVM)**, **Classification Trees**, and **Logistic Regression**.
    - Tuned the hyperparameters for each model to find the best performing ones.
  - **Why:** Using multiple models and tuning their hyperparameters allows comparison to determine which one performs best on the given dataset.
- **Model Evaluation:**
  - **What I Did:**
    - Evaluated the models using accuracy, precision, recall, and F1-score on the test data to find the most effective model.
    - **Why:** These evaluation metrics help in understanding how well each model classifies the data, balancing performance between precision and recall.
- **Model Improvement:**
  - **What I Did:**
    - Selected the best performing model based on the test results.
    - Applied additional techniques like cross-validation and grid search for further hyperparameter tuning if necessary.
  - **Why:** These techniques improve model performance by ensuring the model generalizes well across different data points.
- **Best Performing Model:**
  - **What I Did:**
    - After evaluating multiple models, I identified the one with the highest performance metrics on the test data.
  - **Why:** The best model ensures the most accurate prediction of whether the Falcon 9 first stage will land successfully.
- **Flowchart of Model Development Process:**
  - 1. Load Data → 2. Perform EDA → 3. Feature Engineering → 4. Data Preprocessing (Standardization, Split)
  - → 5. Train Models (SVM, Classification Trees, Logistic Regression) → 6. Evaluate Models (Accuracy, Precision, Recall, F1)
  - → 7. Select Best Model → 8. Hyperparameter Tuning (Cross-validation, Grid Search) → 9. Final Model Selection

GitHub URL for Completed Predictive Analysis Lab: SpaceX Machine Learning Prediction Notebook

Results

Exploratory Data Analysis (EDA) Results:

In this section, I analyzed the SpaceX dataset to identify patterns, correlations, and trends regarding the Falcon 9 first-stage landing success. The results of my EDA helped uncover the relationship between different features (e.g., payload mass, orbit type, launch site) and the landing outcome.

- **Key Findings:**
    - The **orbit type** and **payload mass** showed significant patterns in relation to landing success.
    - **Launch site** also played a role in determining the landing success rates.
    - **Control labels** were created for success and failure of landings, where 1 represented a successful landing, and 0 indicated failure.

Interactive Analytics Demo (Screenshots):

- **Screenshots from Interactive Analytics Dashboard (via Folium and Plotly Dash):**
- **Launch Site Map:**
    - A folium map with launch site markers and launch success/failure indicators based on payload mass and orbit type.
    - The map highlighted different launch sites, showing geographical patterns related to landing success.
- **Success/Failure Map:**
    - A second folium map visualizing the success/failure of rocket landings with color-coded markers (green for success, red for failure).
    - Insights into the failure rate were highlighted based on proximity to certain launch sites.
- **Interactive Pie Chart for Launch Site:**
    - The interactive pie chart in the Plotly Dash app allows users to select launch sites and view success rates for Falcon 9 firststage landings.
    - The chart updates dynamically based on the selected launch site.
- **Scatter Plot of Payload vs Success:**
    - A scatter plot created using Plotly Dash, where users can filter the payload mass range and see how it correlates with the success rate of the landing.
- **Predictive Analysis Results:**
- **Model Evaluation:**
    - I applied and evaluated three different classification models: **Support Vector Machine (SVM)**, **Classification Trees**, and **Logistic Regression**. The evaluation was based on key performance metrics: accuracy, precision, recall, and F1-score.
- **Model Performance Summary:**
    - **SVM:** Performed well with high accuracy but had lower precision for some failure cases.
    - **Classification Trees:** Showed slightly better recall, capturing more of the failed launches, but had a lower overall accuracy.
    - **Logistic Regression:** Balanced performance with a good trade-off between precision and recall.
- **Best Model:**
    - **Logistic Regression** emerged as the best performing model with an optimal trade-off between precision, recall, and accuracy on the test data.
- **Confusion Matrix:**
    - A confusion matrix for Logistic Regression showed the number of true positives, true negatives, false positives, and false negatives, providing deeper insight into where the model made errors.

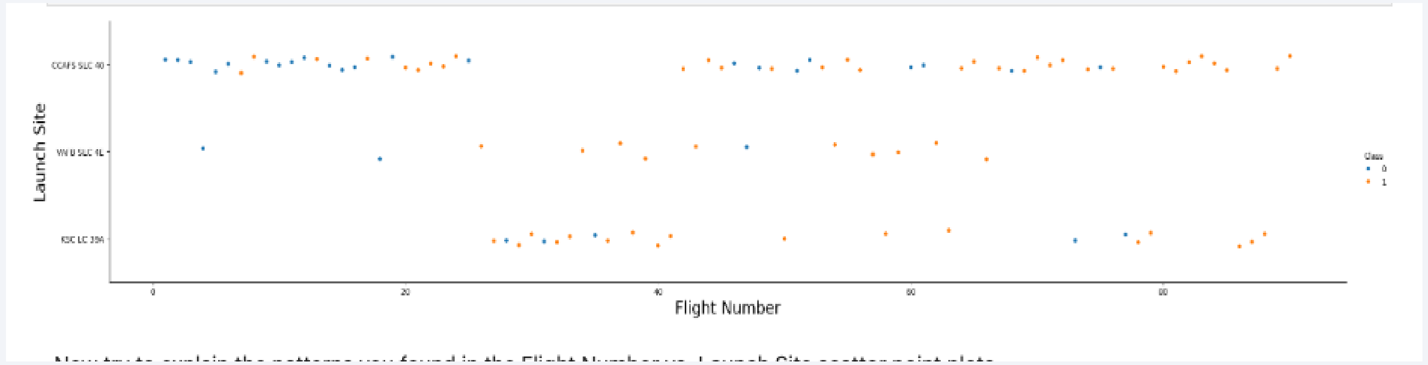
Section 2

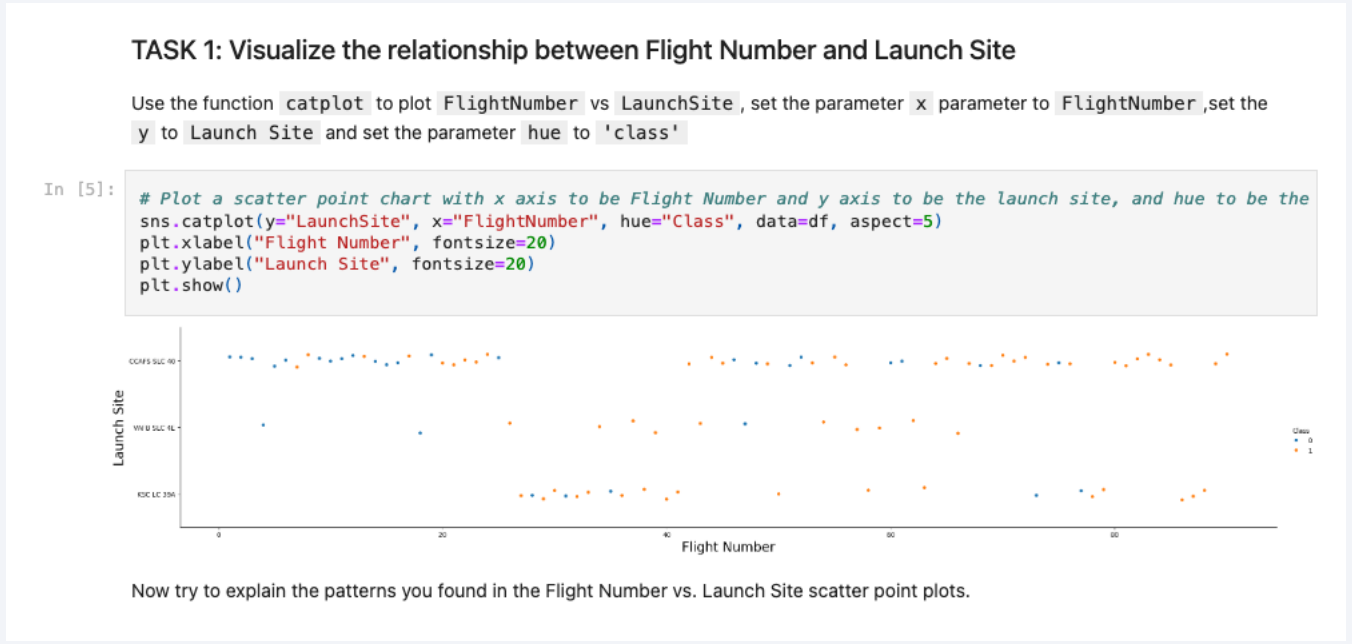
Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

- Show the screenshot of the scatter plot with explanations





TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

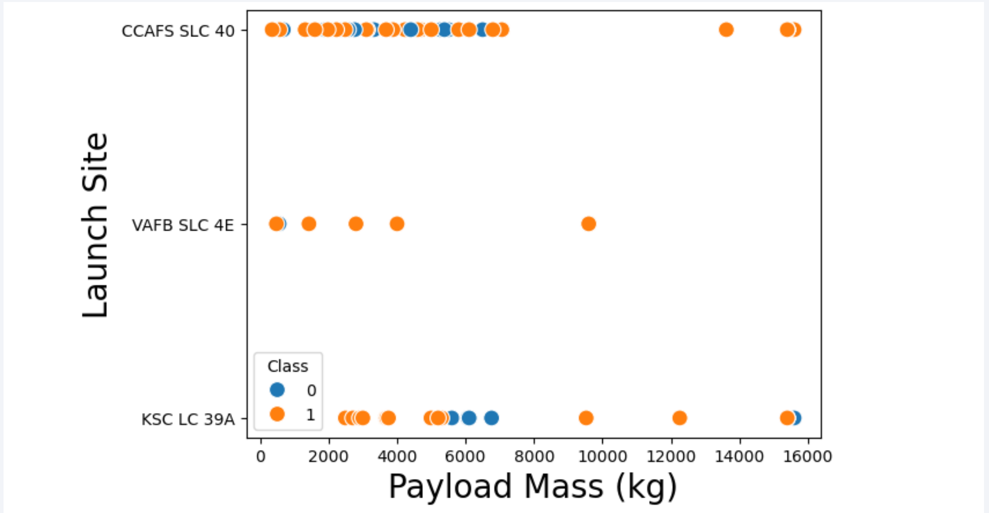
In [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

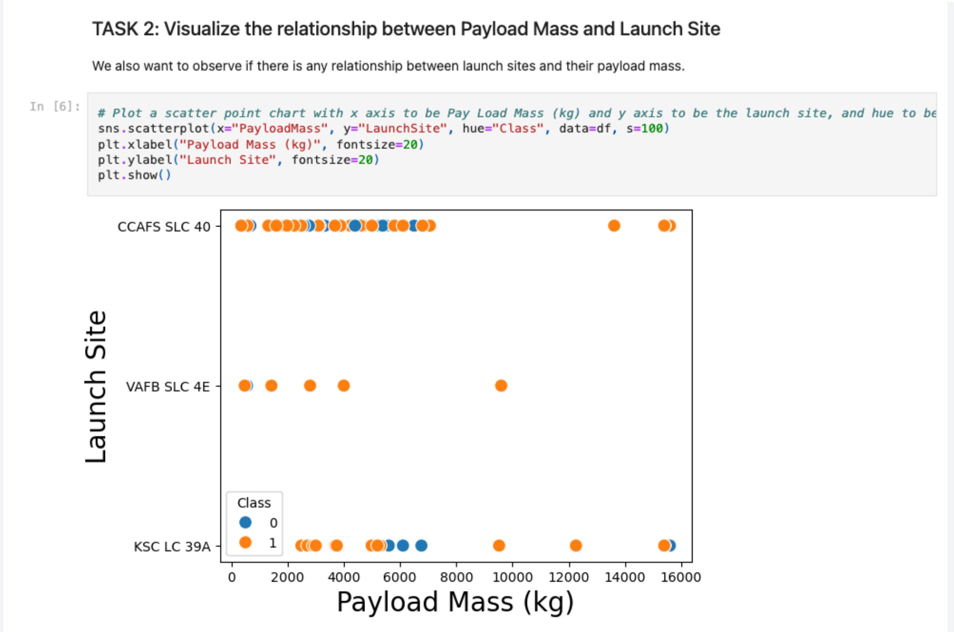
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations





TASK 2: Visualize the relationship between Payload Mass and Launch Site

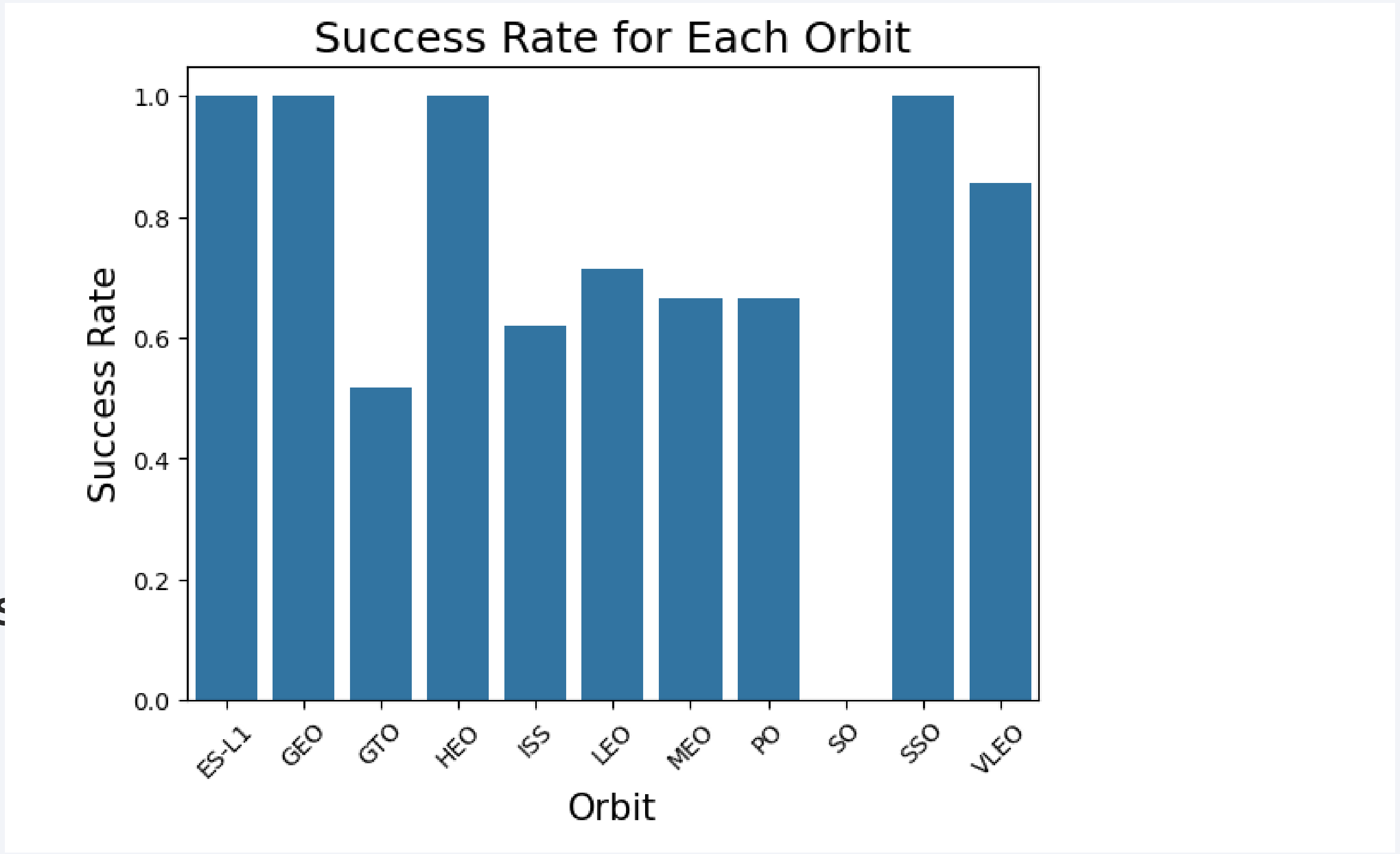
We also want to observe if there is any relationship between launch sites and their payload mass.

In [6]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, s=100)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations



Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

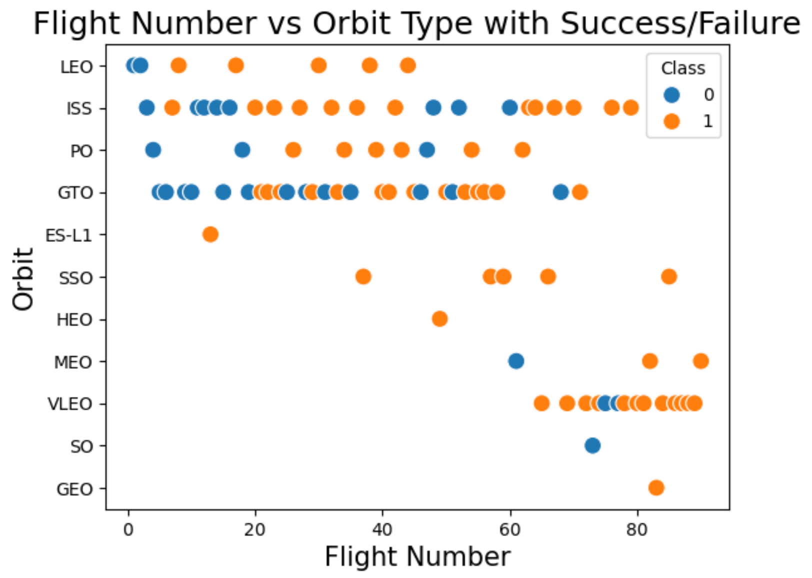
- Show the screenshot of the scatter plot with explanations

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

n [8]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the c
sns.scatterplot(x="FlightNumber", y="Orbit", hue="Class", data=df, s=100)
plt.xlabel("Flight Number", fontsize=15)
plt.ylabel("Orbit", fontsize=15)
plt.title("Flight Number vs Orbit Type with Success/Failure", fontsize=18)
plt.show()
```



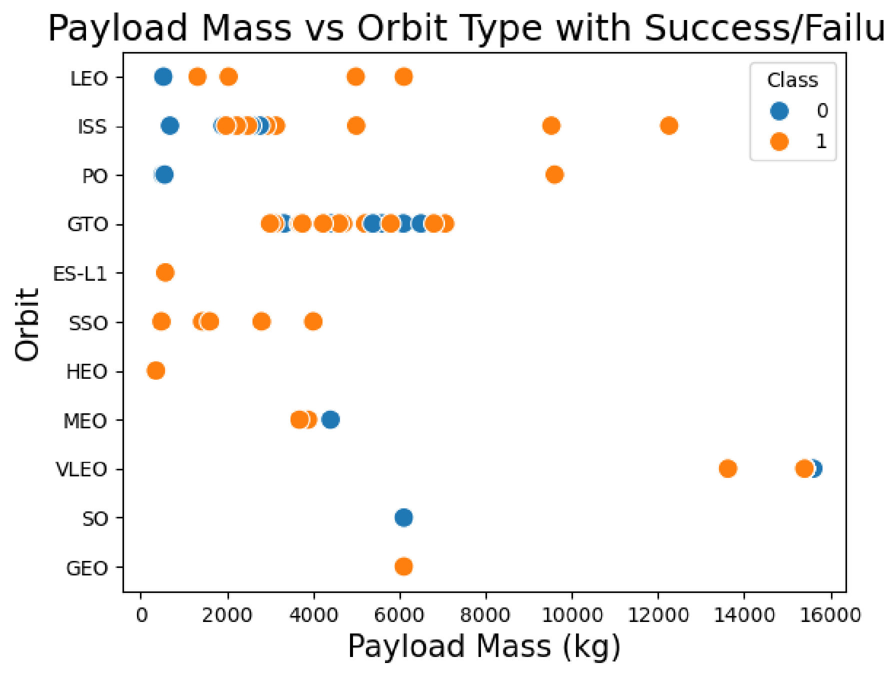
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

```
In [9]:
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class v
sns.scatterplot(x="PayloadMass", y="Orbit", hue="Class", data=df, s=100)
plt.xlabel("Payload Mass (kg)", fontsize=15)
plt.ylabel("Orbit", fontsize=15)
plt.title("Payload Mass vs Orbit Type with Success/Failure", fontsize=18)
plt.show()
```



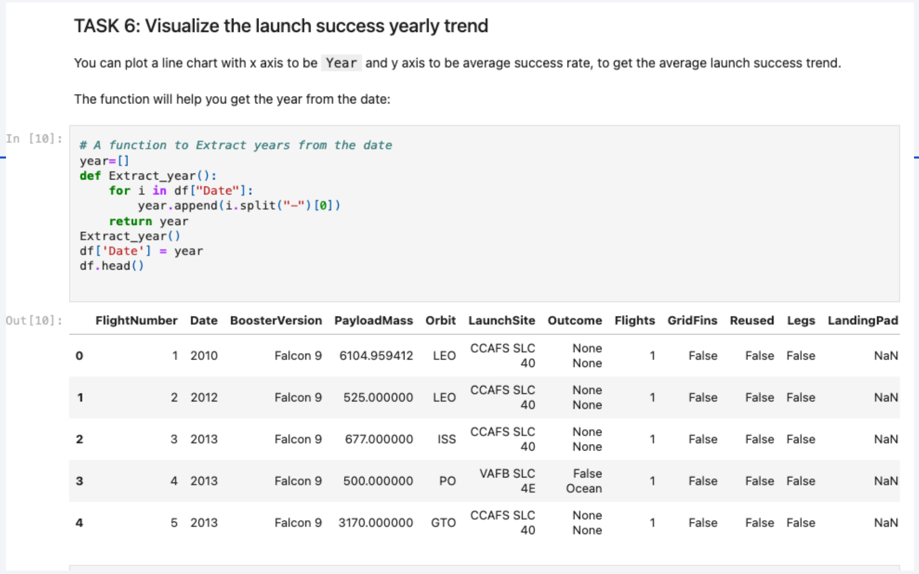
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations



TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

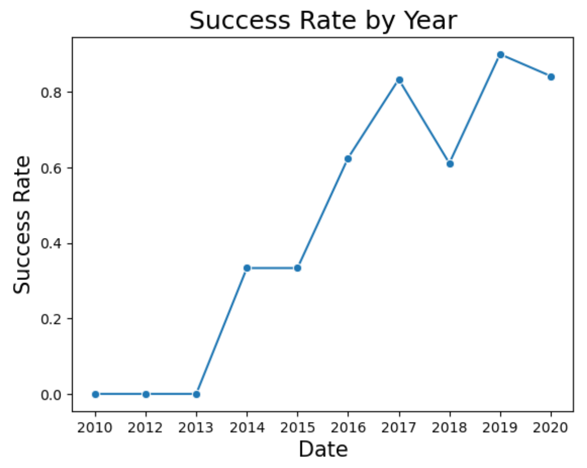
The function will help you get the year from the date:

In [10]:

```python
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

Out[10]:

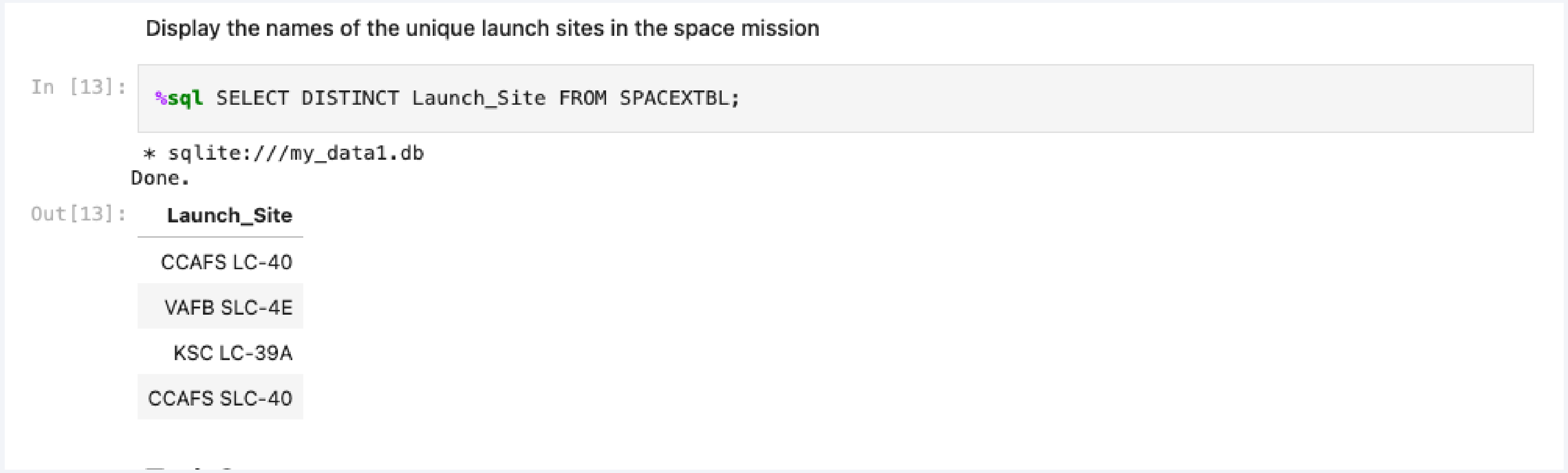
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here



Display the names of the unique launch sites in the space mission

```
In [13]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

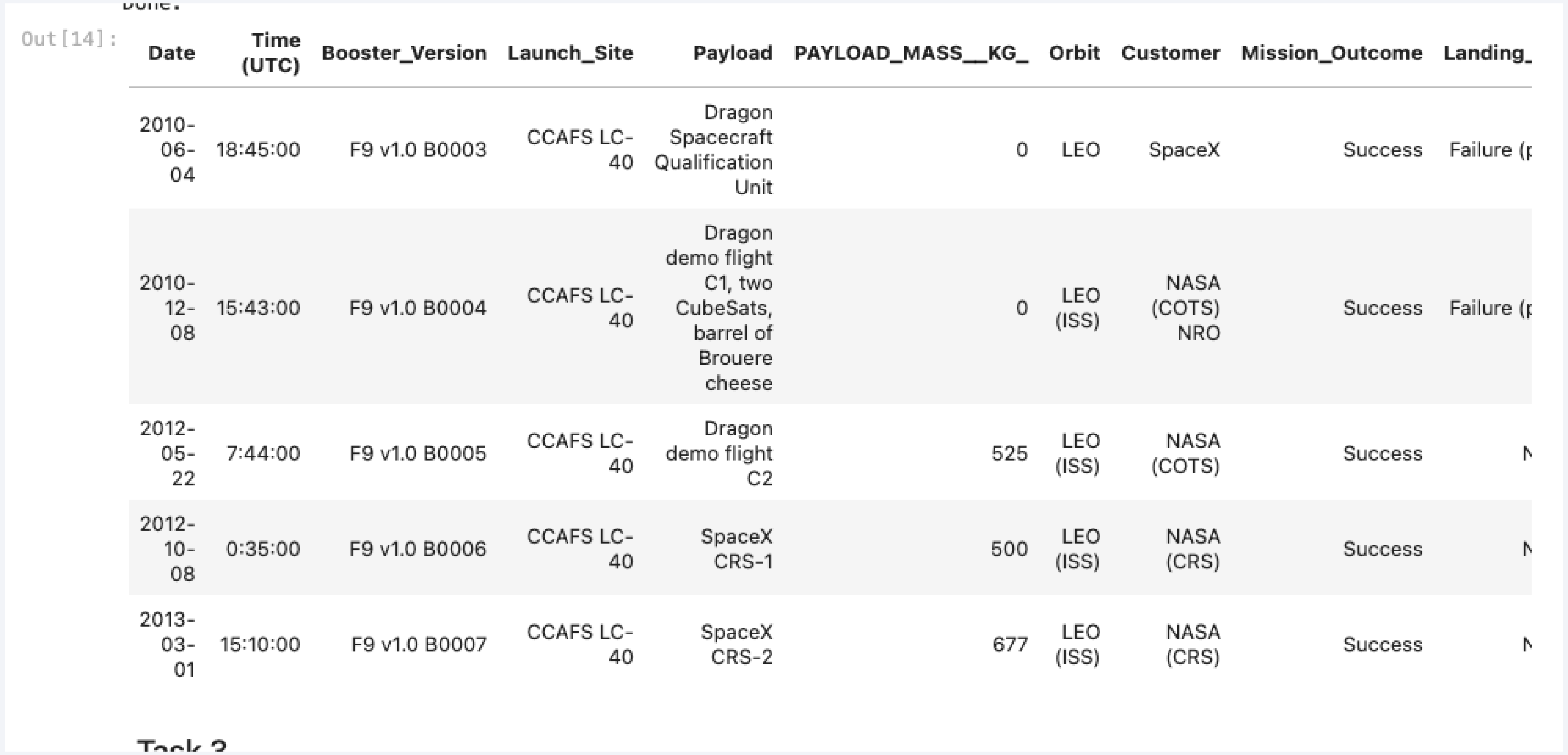
```
 * sqlite:///my_data1.db
Done.
```

Out[13]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here



Done.

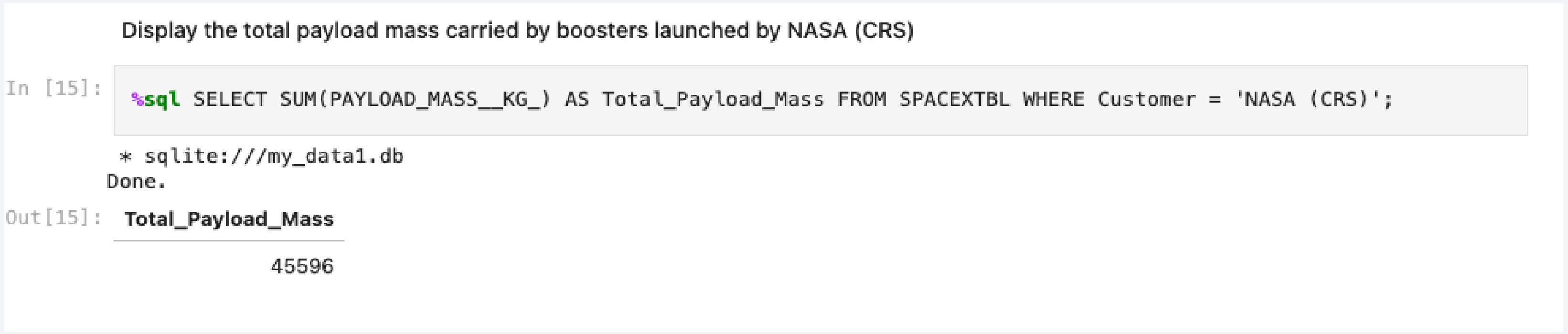
Out[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Task 3

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here



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

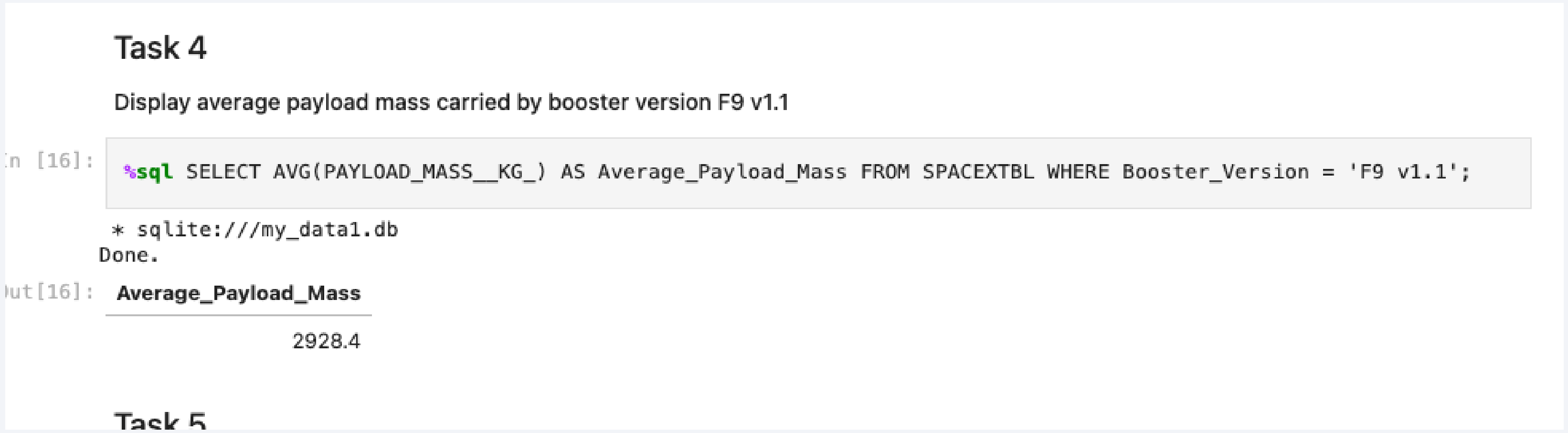
```
In [15]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';

 * sqlite:///my_data1.db
Done.

Out[15]: Total_Payload_Mass
                     45596
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here



Task 4

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

```
In [16]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

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

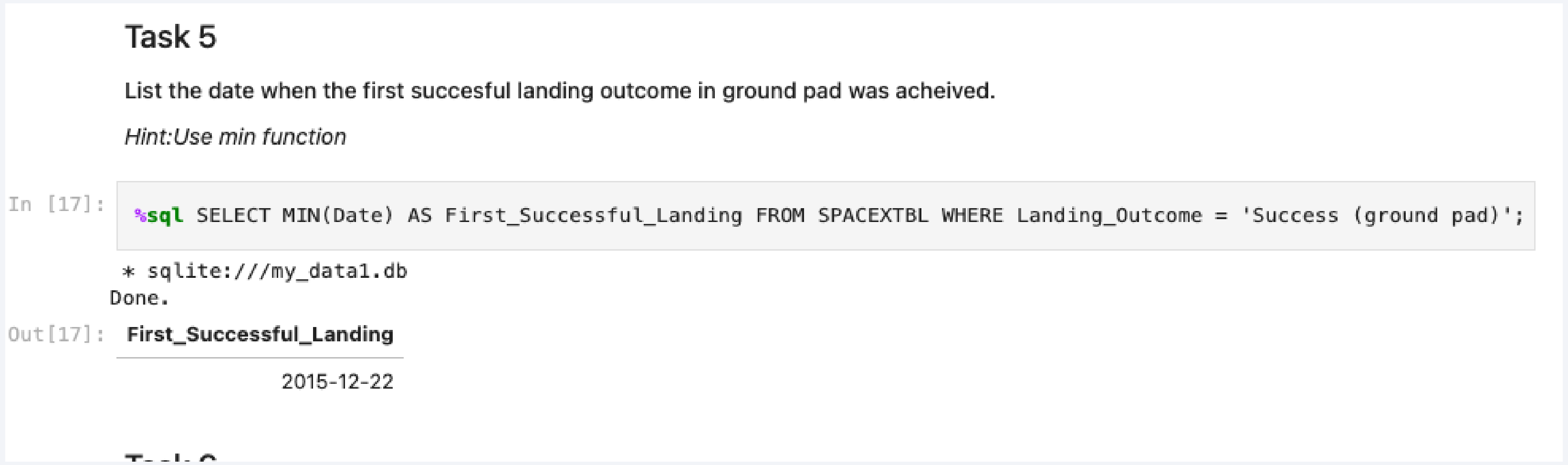
Out[16]:

Average_Payload_Mass
2928.4

Task 5

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here



Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
In [17]: %sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

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

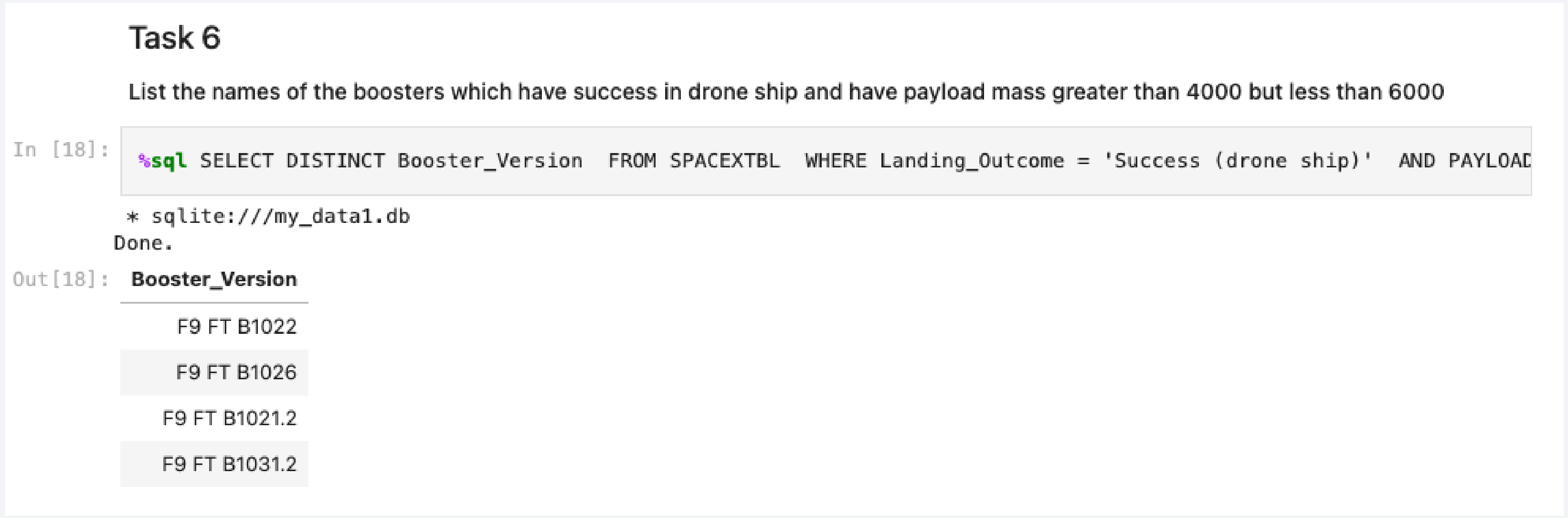
Out[17]:

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- Present your query result with a short explanation here



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 [18]: %sql SELECT DISTINCT Booster_Version  FROM SPACEXTBL  WHERE Landing_Outcome = 'Success (drone ship)'  AND PAYLOAD
```

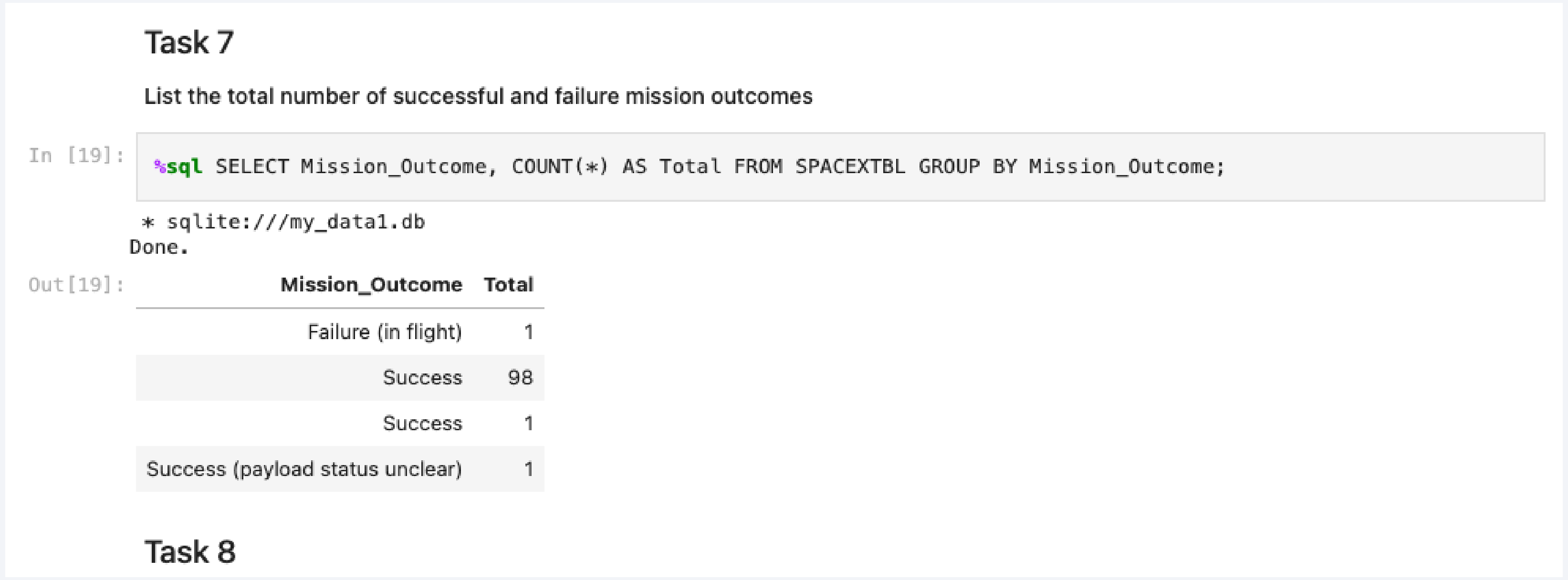
```
 * sqlite:///my_data1.db
Done.
```

Out[18]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here



Task 7

List the total number of successful and failure mission outcomes

```
In [19]: %sql SELECT Mission_Outcome, COUNT(*) AS Total FROM SPACEXTBL GROUP BY Mission_Outcome;
```

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

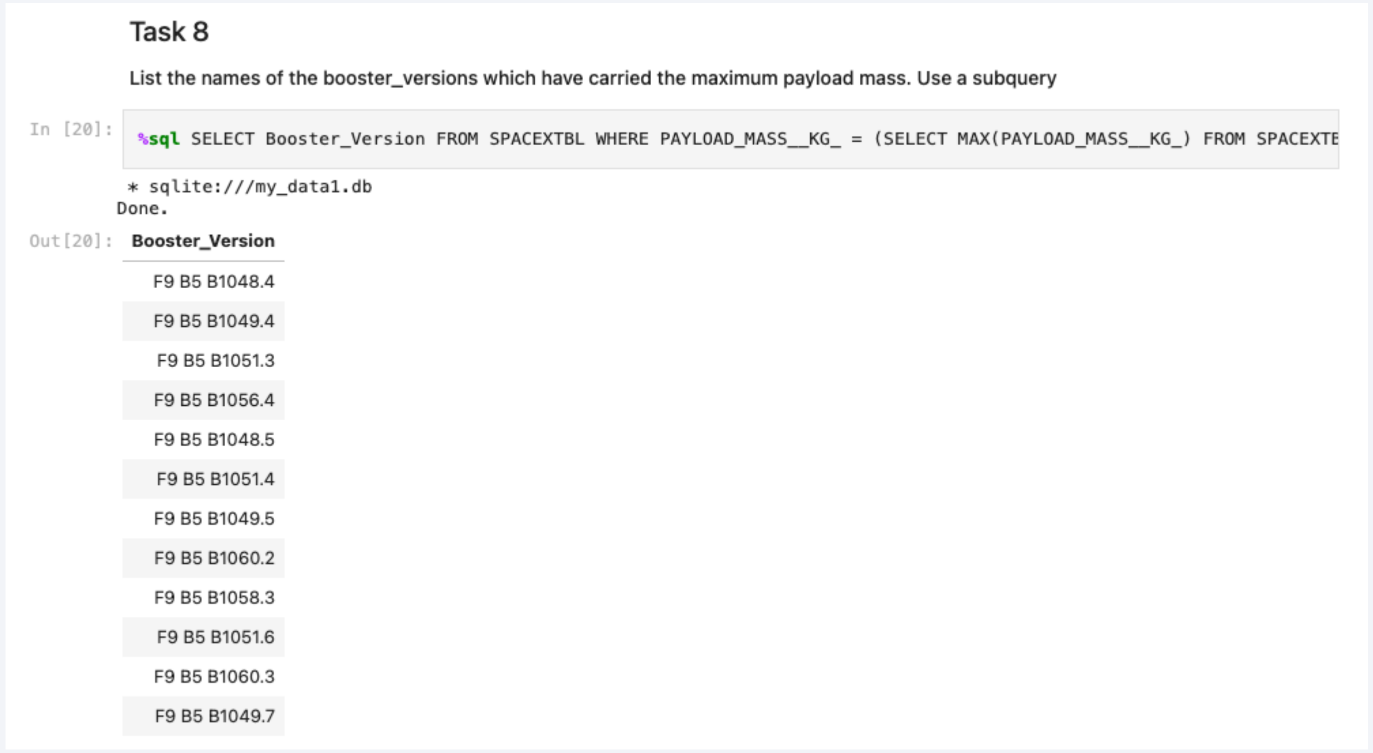
Out[19]:

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Task 8

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here



Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [20]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTE
```

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

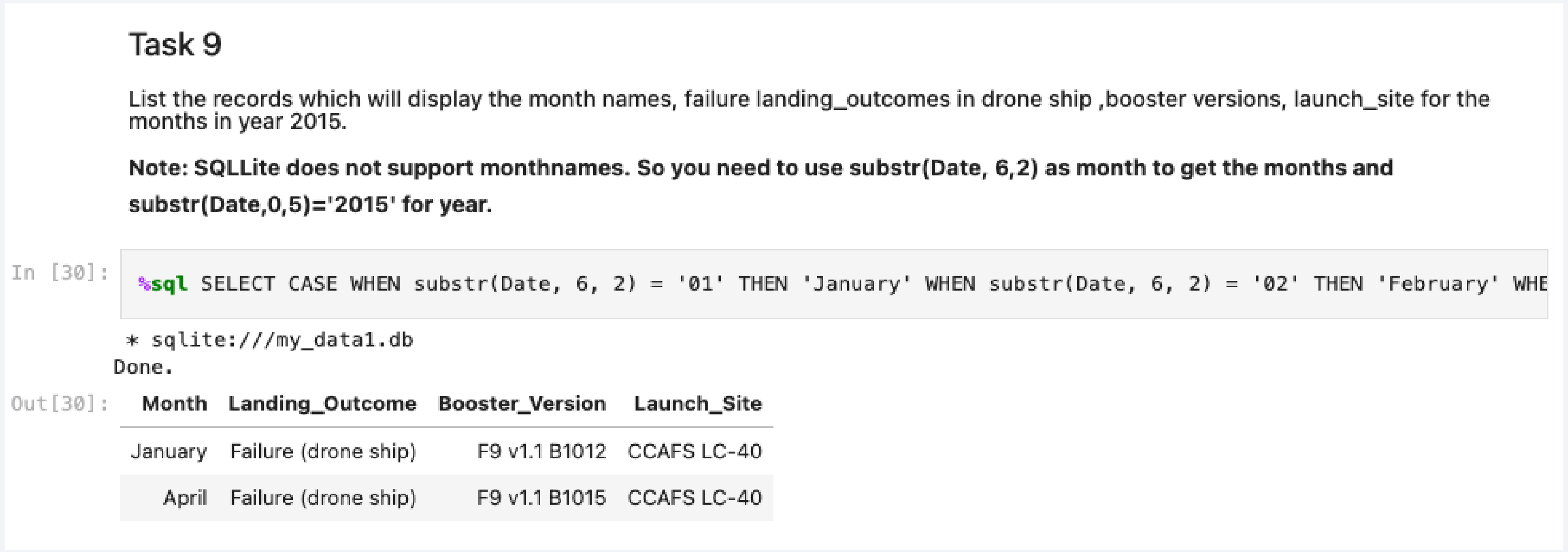
Out[20]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- Present your query result with a short explanation here



Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
In [30]: %sql SELECT CASE WHEN substr(Date, 6, 2) = '01' THEN 'January' WHEN substr(Date, 6, 2) = '02' THEN 'February' WHE

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

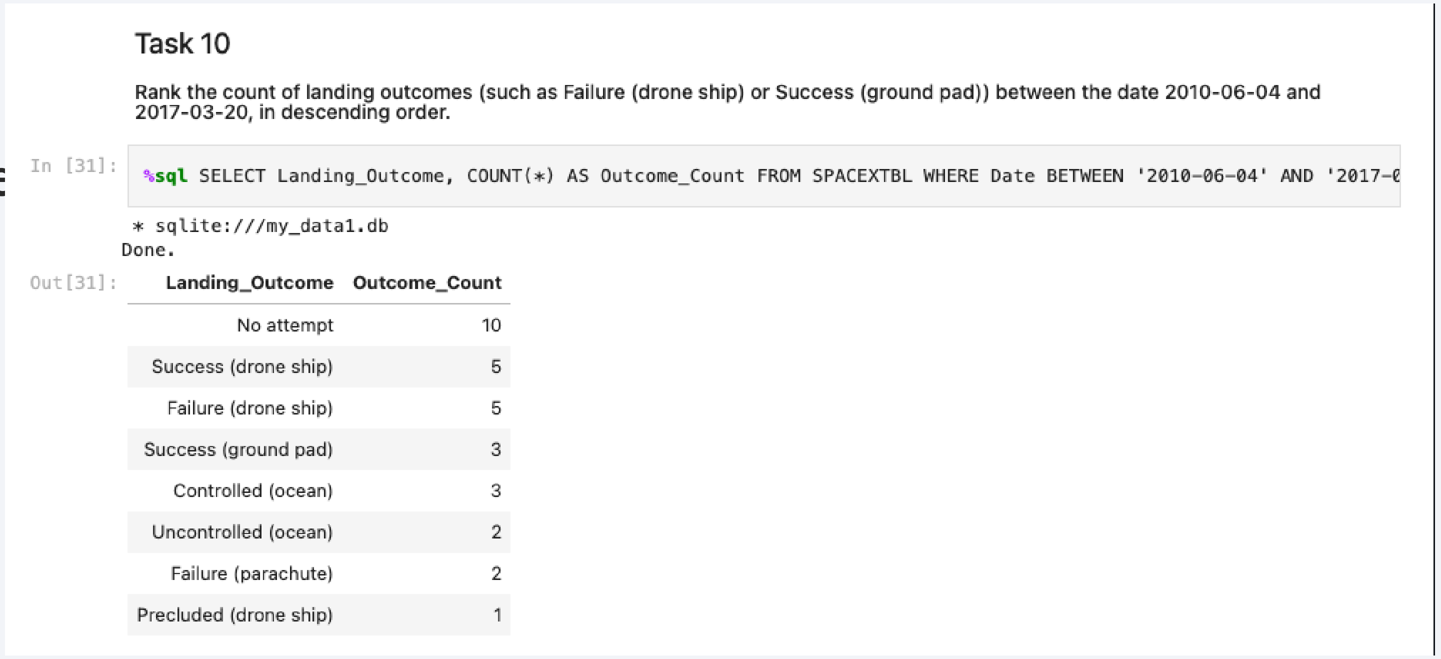
Out[30]:

Month	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

- Present your que



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
In [31]: %sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-0
```

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

Out[31]:

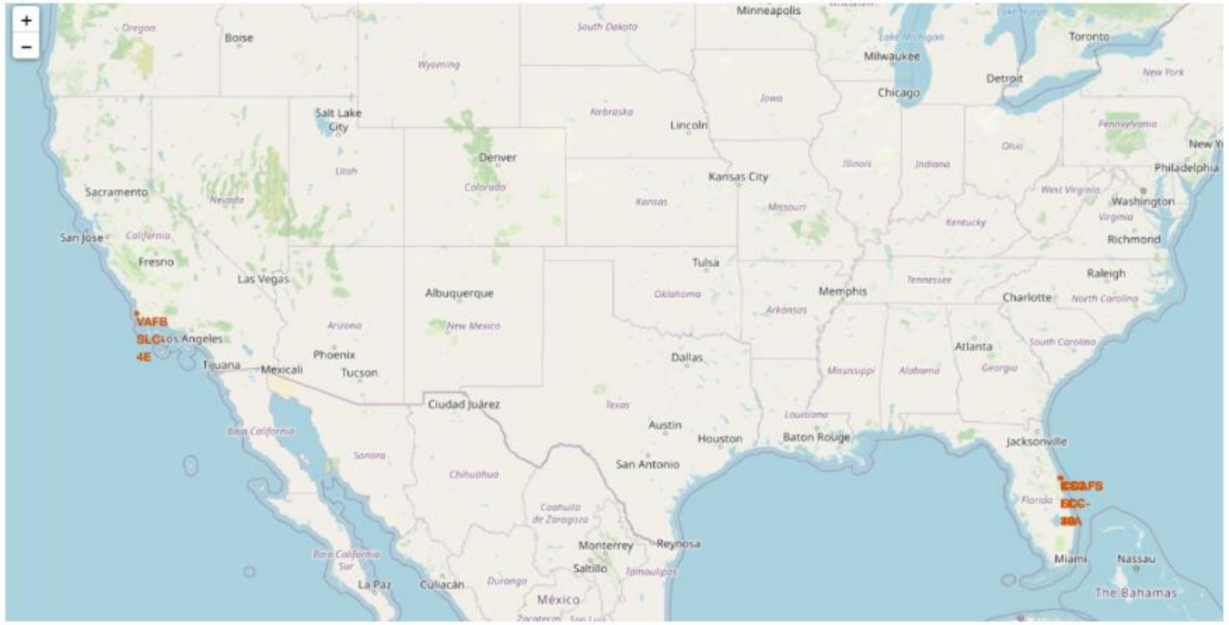
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Mark all launch sites on a map

The generated map with marked launch sites should look similar to the following:



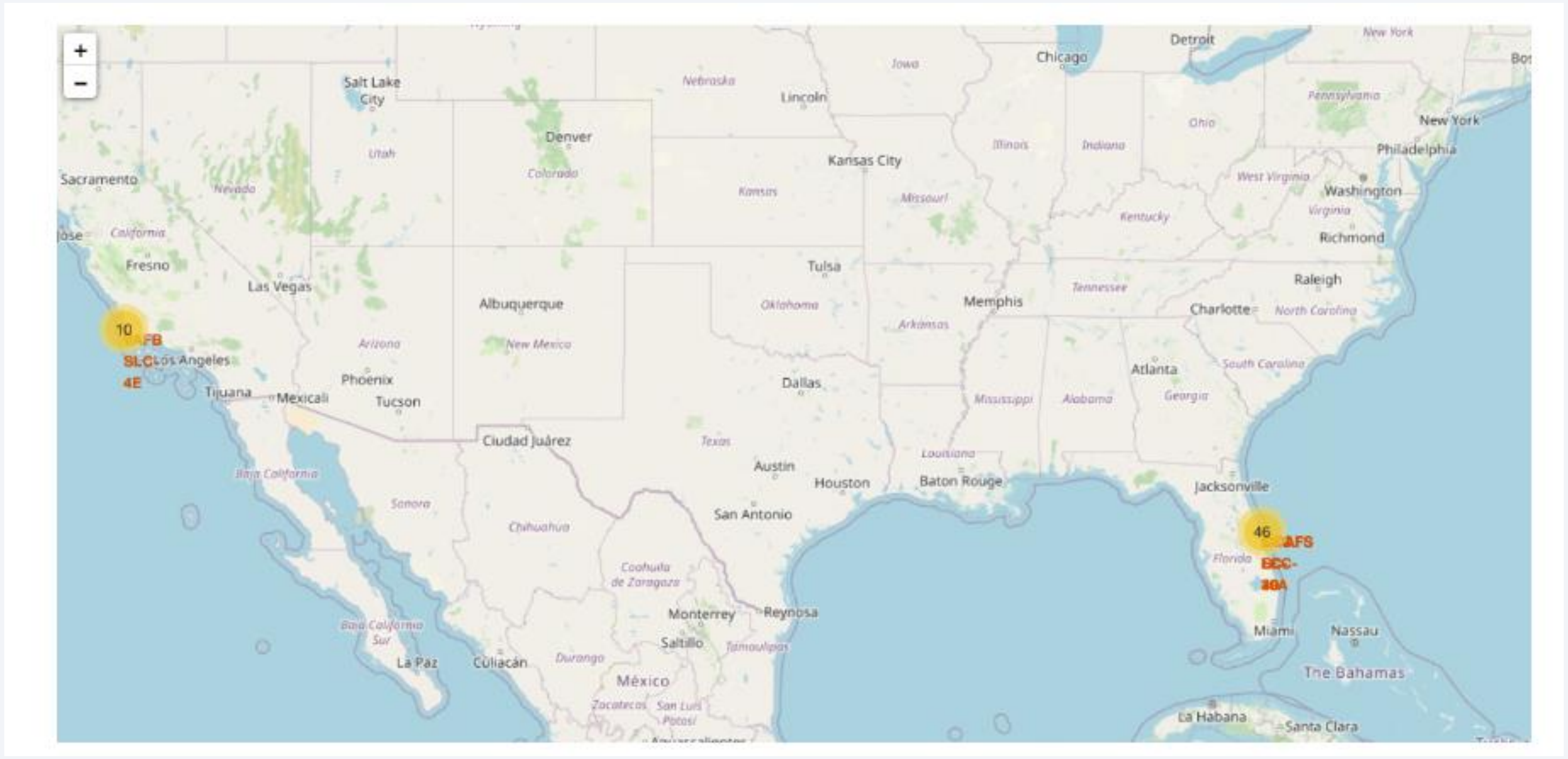
Now, you can explore the map by zoom-in/out the marked areas , and try to answer the following questions:

- Are all launch sites in proximity to the Equator line?
- Are all launch sites in very close proximity to the coast?

Also please try to explain your findings.

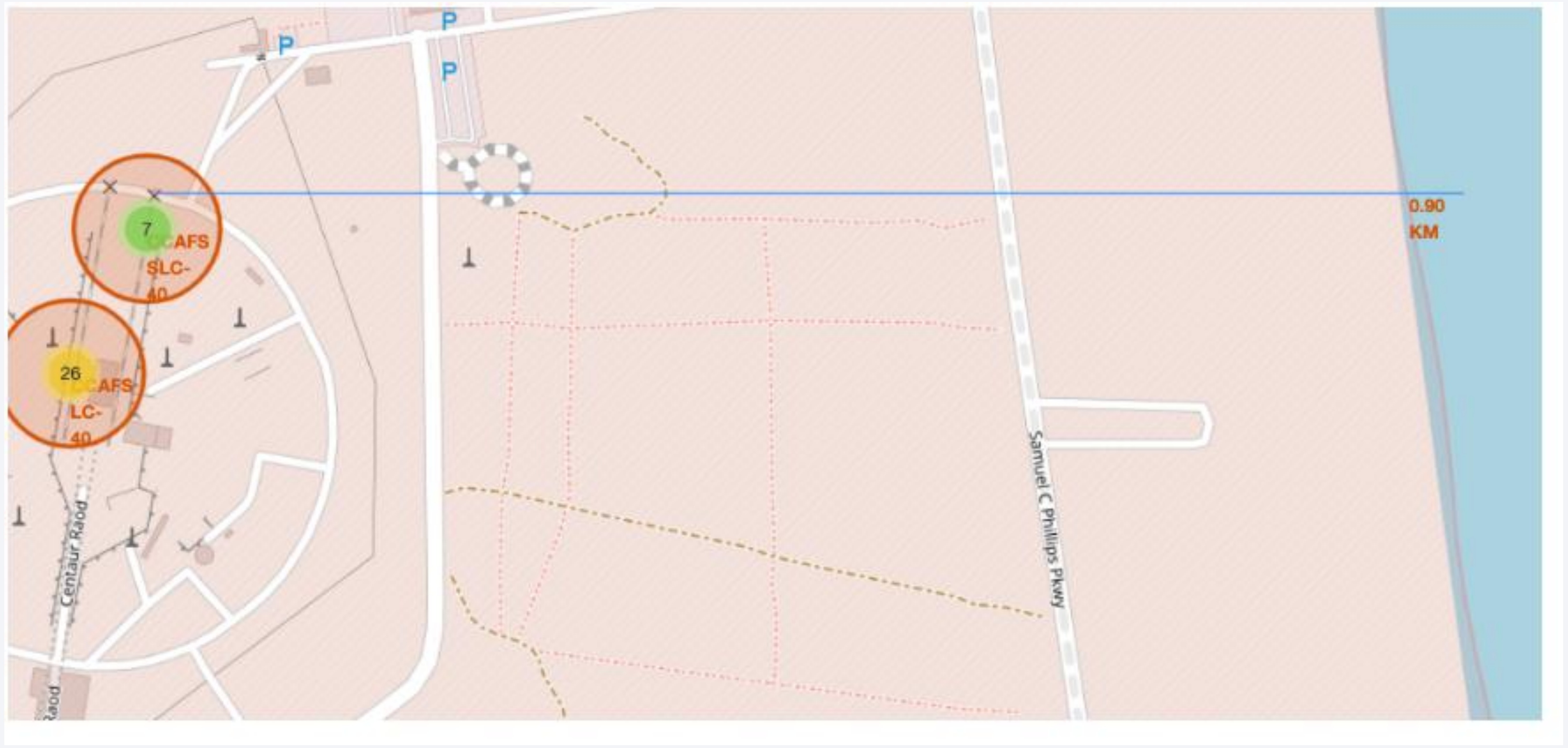
success/failed launches for each site on the map

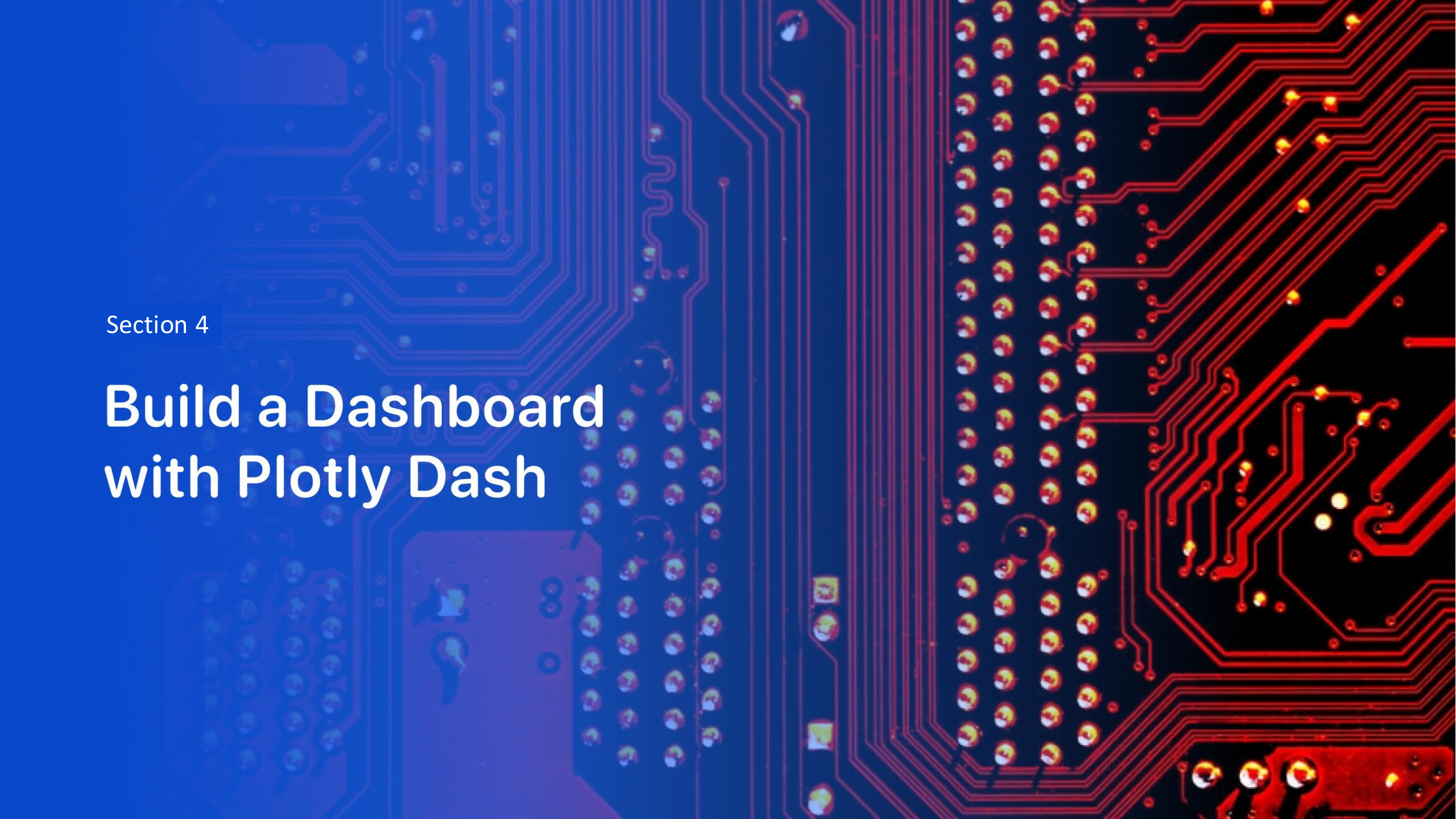
We can see the success/failed launches for each site on the map



distances between a launch site to its proximities

We can see the distances between a launch site to its proximities

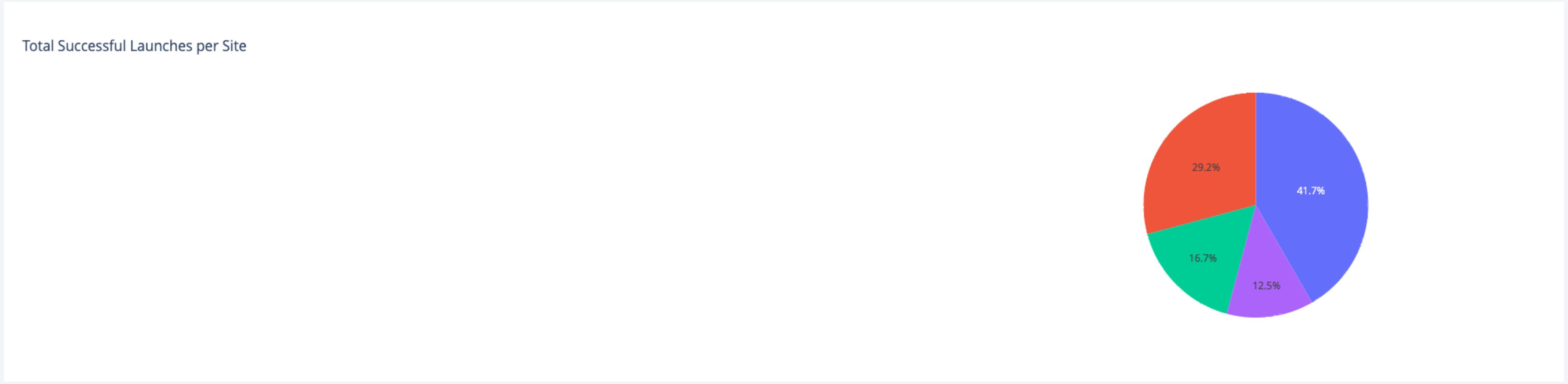




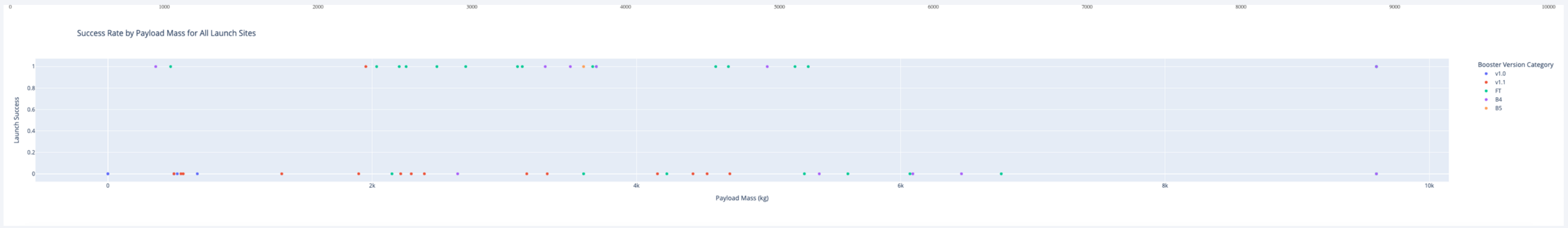
Section 4

Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard



Success Rate by Payload Mass for All Launch Sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

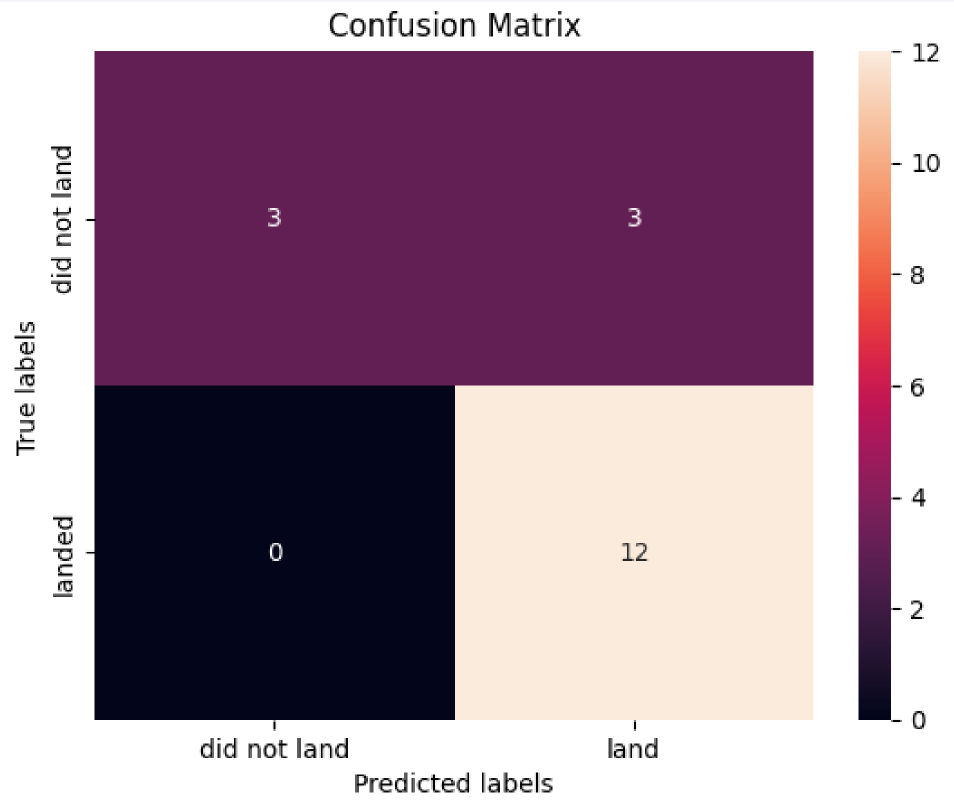
1. We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

   tuned hpyerparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'} accuracy : 0.8464285714285713
2. Create a support vector machine object then create a GridSearchCV object svm_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
   tuned hpyerparameters :(best parameters) {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'} accuracy : 0.8482142857142856
3. Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
   tuned hpyerparameters :(best parameters) {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'} accuracy : 0.8892857142857142
4. Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
   tuned hpyerparameters :(best parameters) {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1} accuracy : 0.8482142857142858

the method performs best:

('Logistic Regression', 0.8333333333333334)

Confusion Matrix



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.

Overview:

True Postive - 12 (True label is landed, Predicted label is also landed)

False Postive - 3 (True label is not landed, Predicted label is landed)

Conclusions

1. Launch Site and Success Rates Correlation:

- The exploratory data analysis revealed that the launch site plays a significant role in the Falcon 9 first-stage landing success. Specific launch sites have higher success rates, indicating that geographical and site-specific factors, such as terrain and infrastructure, influence landing outcomes.

2. Impact of Payload Mass on Landing Success:

- The payload mass was found to be a key factor influencing the likelihood of a successful landing. Heavier payloads tend to have a lower success rate, likely due to the increased difficulty in controlling the landing during the return phase.

3. Effective Interactive Analytics for Decision Making:

- Interactive tools such as the Folium maps and Plotly Dash dashboards provided valuable insights by allowing users to dynamically explore data, select specific launch sites, and correlate various factors like payload mass and orbit type with landing success. These tools enhanced understanding and allowed for better-informed decision-making.

4. Best Model for Predicting Landing Success:

- Among the tested classification models (SVM, Classification Trees, and Logistic Regression), Logistic Regression performed best in terms of balancing precision and recall, making it the most suitable model for predicting whether the Falcon 9 first-stage will land successfully. This model can be used to estimate landing outcomes with reasonable accuracy, which could help SpaceX in optimizing its launch process.

5. Improvement Areas for Future Models:

- While Logistic Regression was the top-performing model, future improvements could involve experimenting with ensemble models or deep learning approaches to potentially increase accuracy further. Additionally, incorporating more features such as weather conditions or real-time telemetry data could improve predictive capabilities.

6. Practical Applications:

- The insights from this analysis can be used by SpaceX for optimizing launch strategies and for other companies interested in bidding for space missions, especially in terms of reducing costs by improving the reliability of Falcon 9's first-stage landings.

Appendix

- All files are under: https://github.com/mifitous/Coursera-Jupyter-Notebooks

Thank you!