IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- **Summary of methodologies**
- Data colletion with API
- Web Scrapping;
- SQl;
- Machine Learning;
- Data Visualization;
- Folium
- **Summary of all results**
- Data analysis result
- Predictive analysis result

Introduction

- Project background and context
- **Problems you want to find answers**
- How many rockets were successful at launch?
- What factors influence the launch of rockets?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - using web scrapping with data from wikipédia
- Perform data wrangling

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

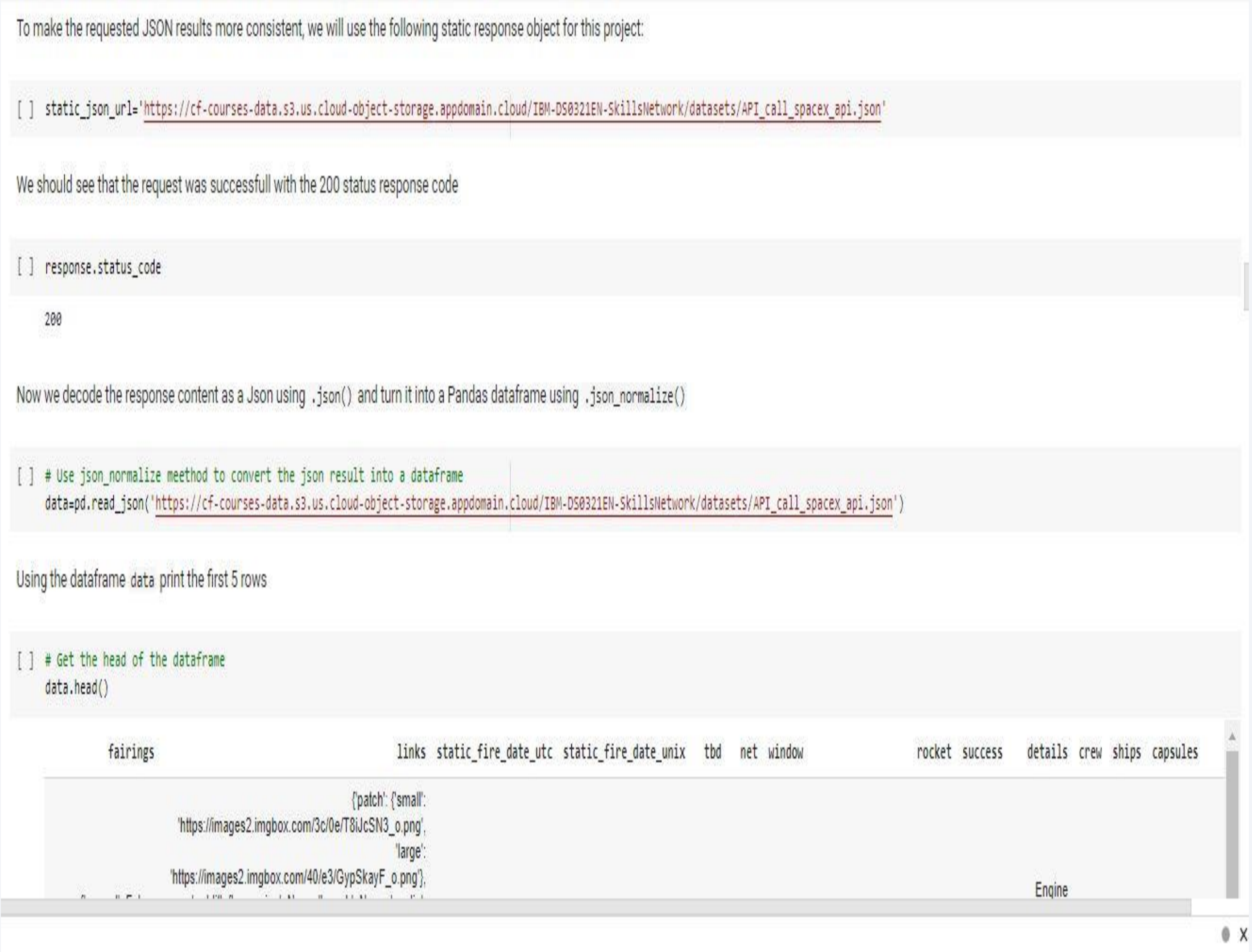
- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts
- data was collected via web scrapping, cleaning the data, the Beautifull Soup library was used to read the data, in which from the cleaning and organization of the data it was possible to convert the data from HTML to a DataFrame and then carry out the appropriate analyses.

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

- https://github.com/jeferson3587/coursera/blob/2bf50629ff63c2749c5bbd697330234d025a2675/jupyter-labs-spacex-data-collection-api.ipynb



To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[ ] static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
[ ] response.status_code

    200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[ ] # Use json_normalize meethod to convert the json result into a dataframe
    data=pd.read_json('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json')
```

Using the dataframe `data` print the first 5 rows

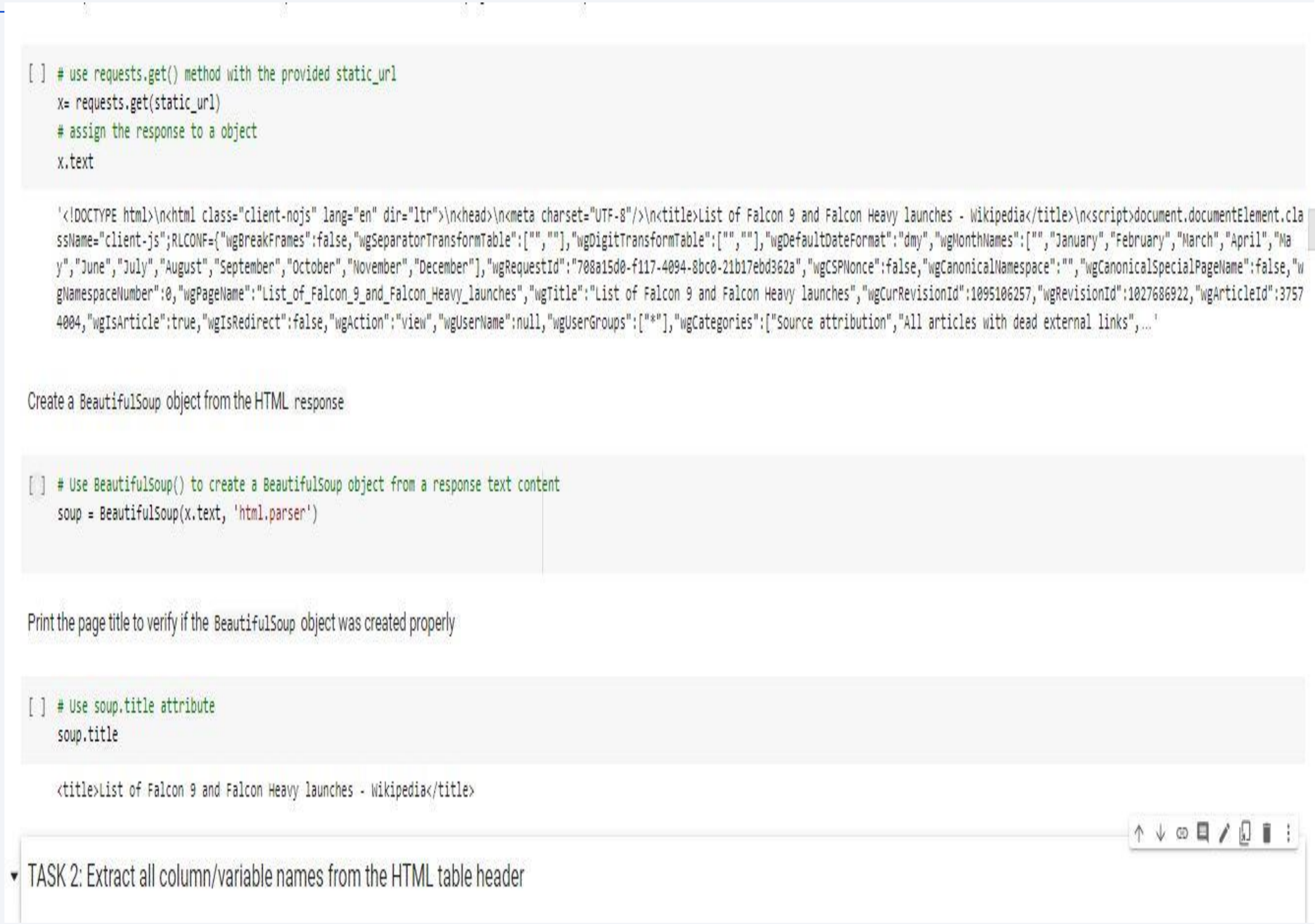
```
[ ] # Get the head of the dataframe
    data.head()
```

```
        fairings                                          links  static_fire_date_utc  static_fire_date_unix  tbd  net  window   rocket  success  details  crew  ships  capsules

                                          {'patch': {'small':
              'https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png',
                                                      'large':
               'https://images2.imgbox.com/40/e3/GypSkayF_o.png'},
                                                                                                                                  Engine
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose
- https://github.com/jeferson3587/coursera/blob/2bf50629ff63c2749c5bbd697330234d025a2675/jupyter_labs_webscraping.ipynb



```
[ ] # use requests.get() method with the provided static_url
    x= requests.get(static_url)
    # assign the response to a object
    x.text
```

```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>\n<script>document.documentElement.cla
ssName="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","Ma
y","June","July","August","September","October","November","December"],"wgRequestId":"708a15d0-f117-4094-8bc0-21b17ebd362a","wgCSPNonce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"w
gNamespaceNumber":0,"wgPageName":"List_of_Falcon_9_and_Falcon_Heavy_launches","wgTitle":"List of Falcon 9 and Falcon Heavy launches","wgCurRevisionId":1095106257,"wgRevisionId":1027686922,"wgArticleId":3757
4004,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Source attribution","All articles with dead external links",…'
```

Create a BeautifulSoup object from the HTML response

```
[ ] # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
    soup = BeautifulSoup(x.text, 'html.parser')
```

Print the page title to verify if the BeautifulSoup object was created properly

```
[ ] # Use soup.title attribute
    soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Data Wrangling

- Describe how data were processed
- You need to present your data wrangling process using key phrases and flowcharts
- After the correct import of the data, some questions regarding the data were answered, such as success rate, which rocket was more successful, main orbits launched and later saved the csv externally.
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- scatter plots were performed between the variables,
- FlightNumber vs. PayloadMass
- Payload and Orbit type
- success rate of each orbit type
- among others, in order to obtain how the variables behave among themselves.
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- import of the csv file in the IBM cloud, then using the loaded data, analyzes such as payload mass carried by booster version F9 v1.1 were performed
- number of successful and failure mission outcomes
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose
- https://github.com/jeferson3587/coursera/blob/2bf50629ff63c2749c5bbd697330234d025a2675/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Explain why you added those objects
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- loading the data, panda and numpy were used, the data was treated in order to clean and organize them, the skit-learn library was used to create the models, train and test the models, executing the accuracy them in order to verify if such a model is really good.
- https://github.com/jeferson3587/coursera/blob/2bf50629ff63c2749c5bbd697330234d025a2675/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

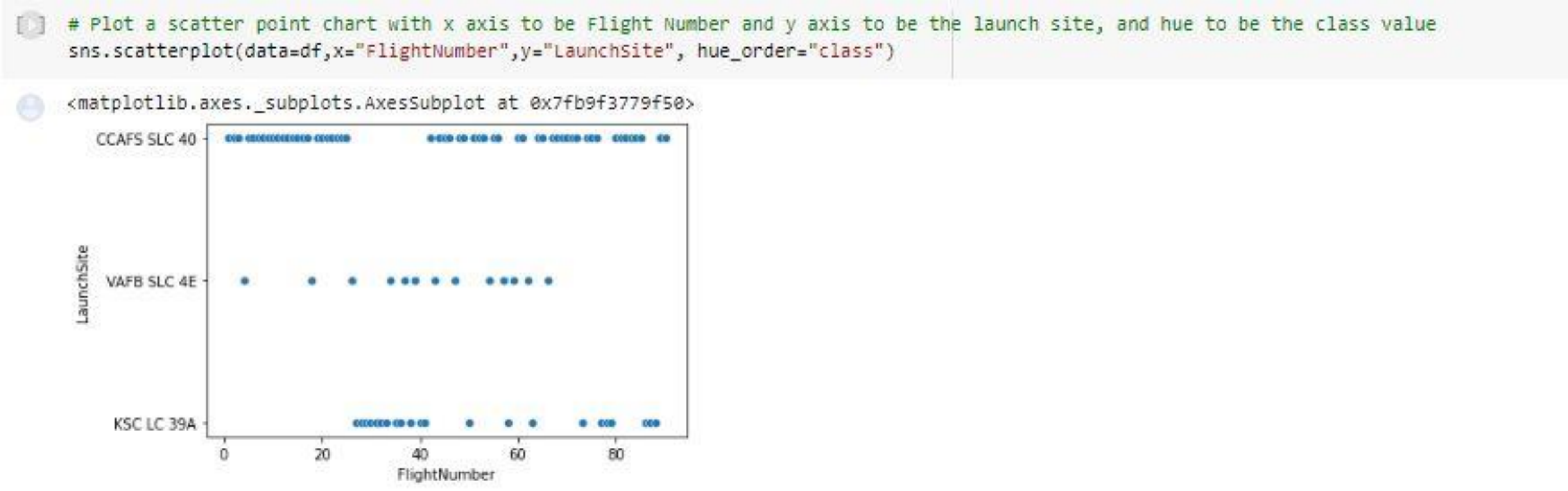
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

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

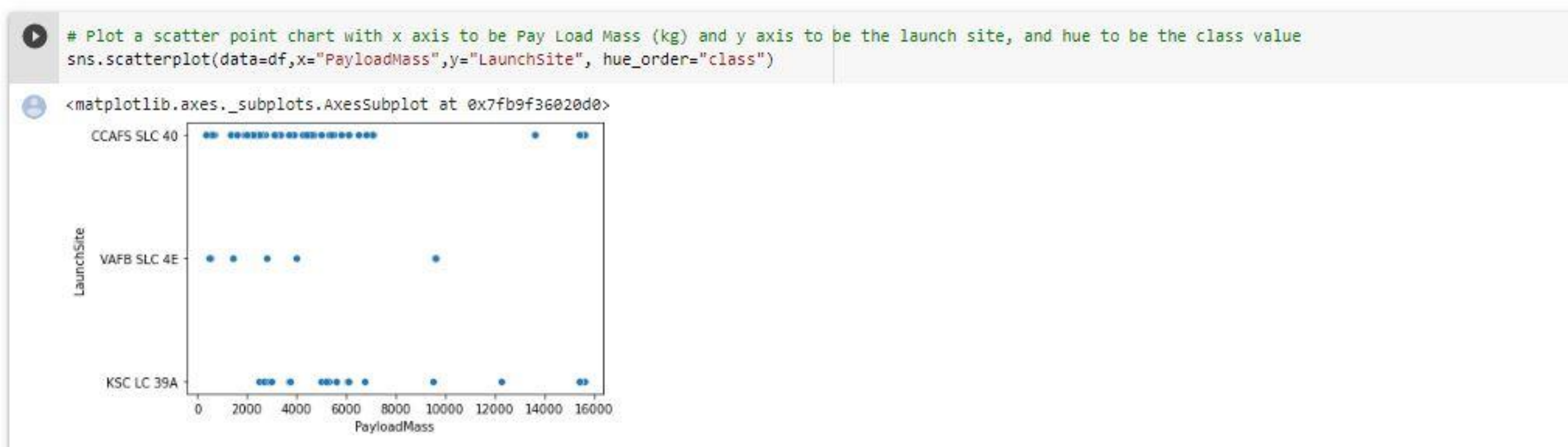


```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.scatterplot(data=df,x="FlightNumber",y="LaunchSite", hue_order="class")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9f3779f50>
```

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations



```
# 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 the class value
sns.scatterplot(data=df,x="PayloadMass",y="LaunchSite", hue_order="class")
```

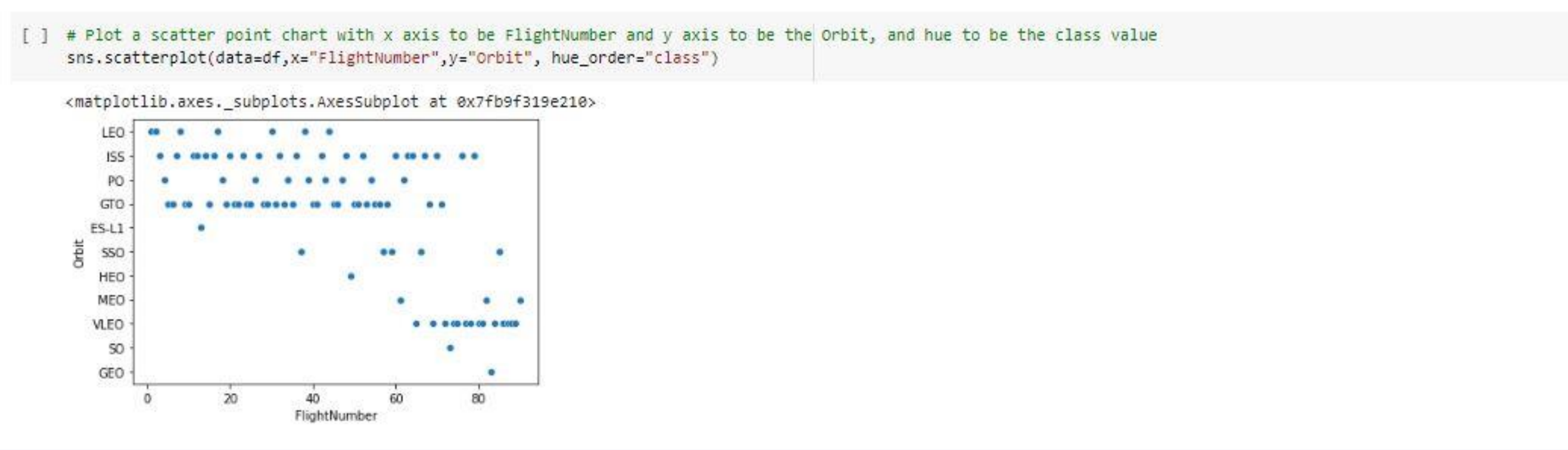
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9f36020d0>
```

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

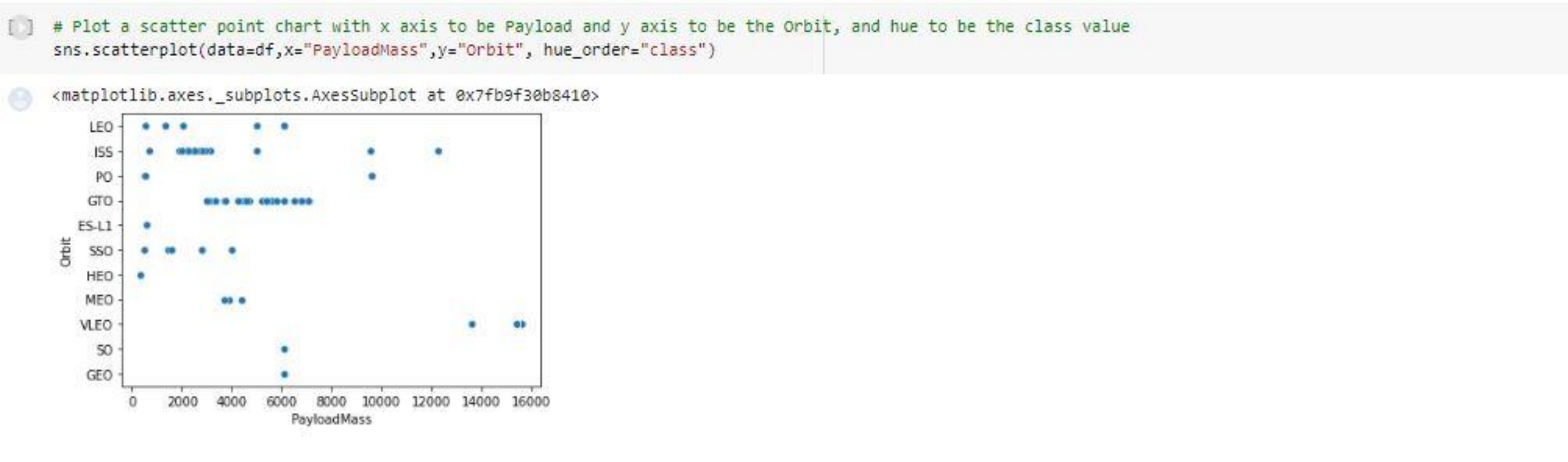


```
[ ]  # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
     sns.scatterplot(data=df,x="FlightNumber",y="Orbit", hue_order="class")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9f319e210>
```

Payload vs. Orbit Type

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



```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.scatterplot(data=df,x="PayloadMass",y="Orbit", hue_order="class")
```

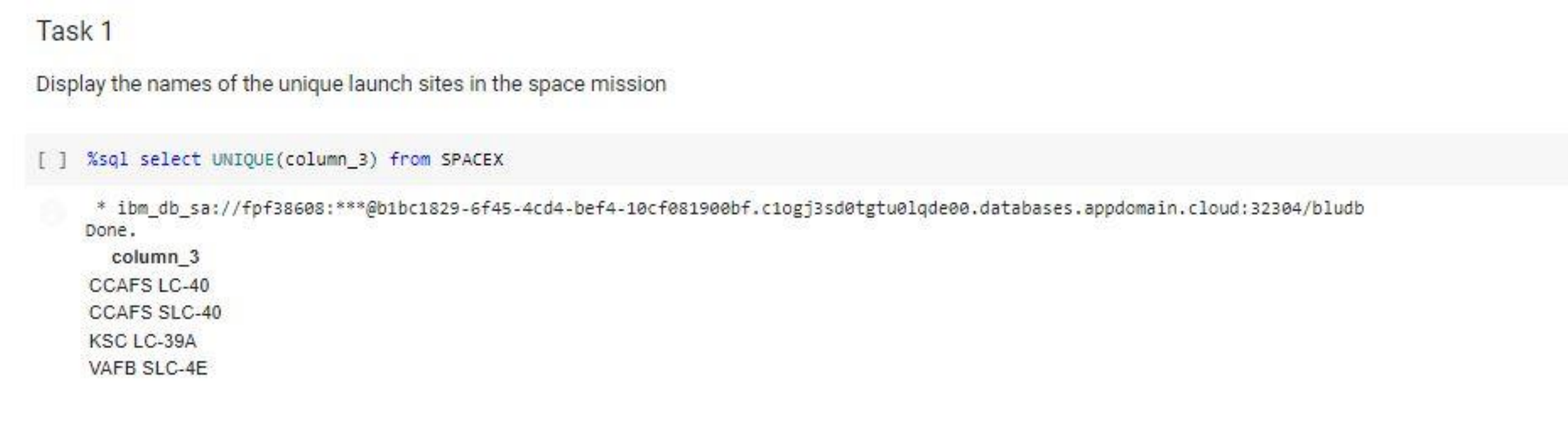
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9f30b8410>
```

Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

All Launch Site Names

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



Task 1

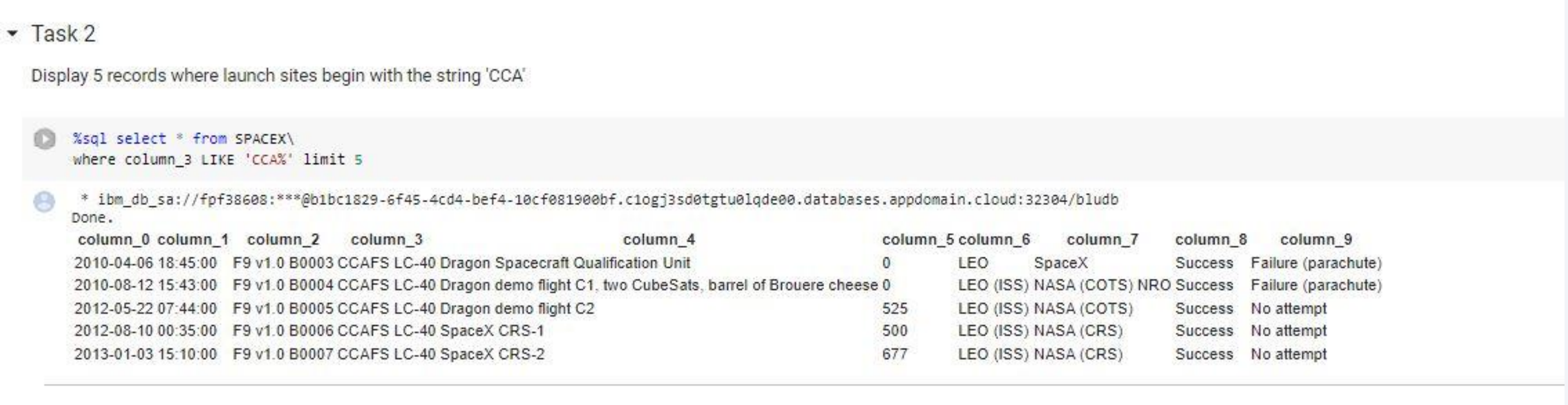
Display the names of the unique launch sites in the space mission

```
[ ] %sql select UNIQUE(column_3) from SPACEX
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
   column_3
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E
```

Launch Site Names Begin with 'CCA'

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



Task 2

Display 5 records where launch sites begin with the string 'CCA'

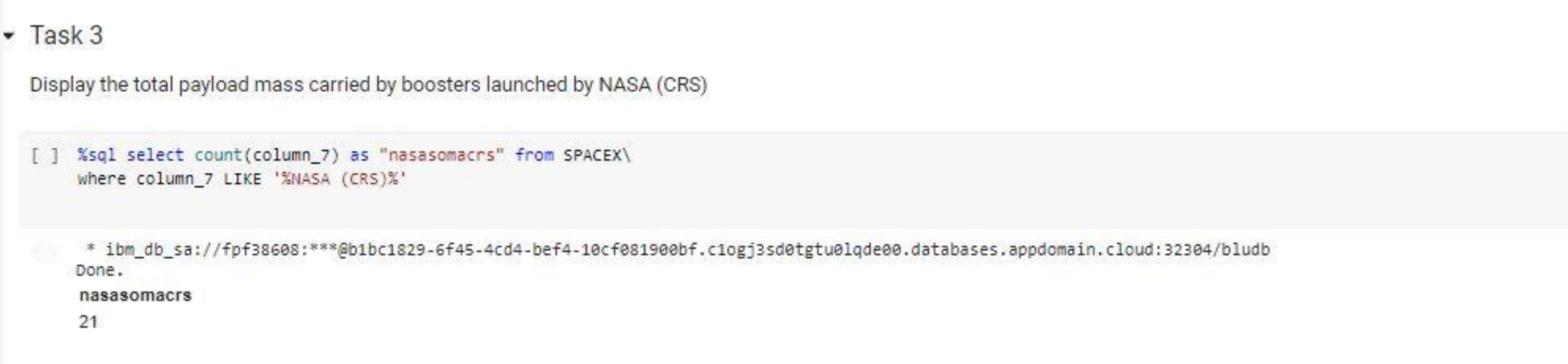
```
%sql select * from SPACEX\
where column_3 LIKE 'CCA%' limit 5
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
```

column_0	column_1	column_2	column_3	column_4	column_5	column_6	column_7	column_8	column_9
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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



Task 3

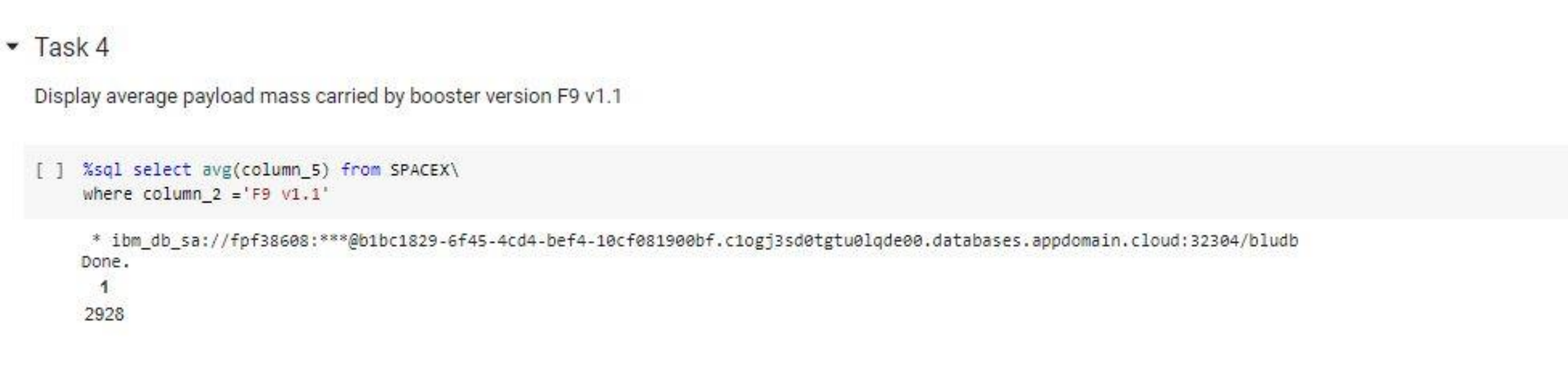
Display the total payload mass carried by boosters launched by NASA (CRS)

```
[ ] %sql select count(column_7) as "nasasomacrs" from SPACEX\
    where column_7 LIKE '%NASA (CRS)%'
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
nasasomacrs
21
```

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

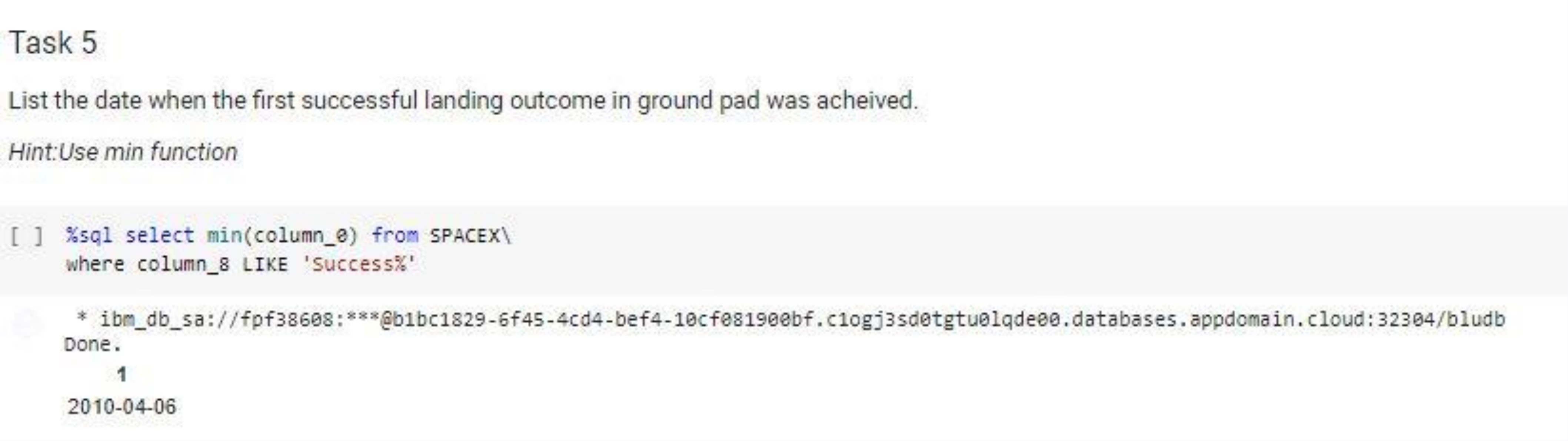
Display average payload mass carried by booster version F9 v1.1

```
[ ] %sql select avg(column_5) from SPACEX\
    where column_2 ='F9 v1.1'
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
  1
2928
```

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 successful landing outcome in ground pad was acheived.

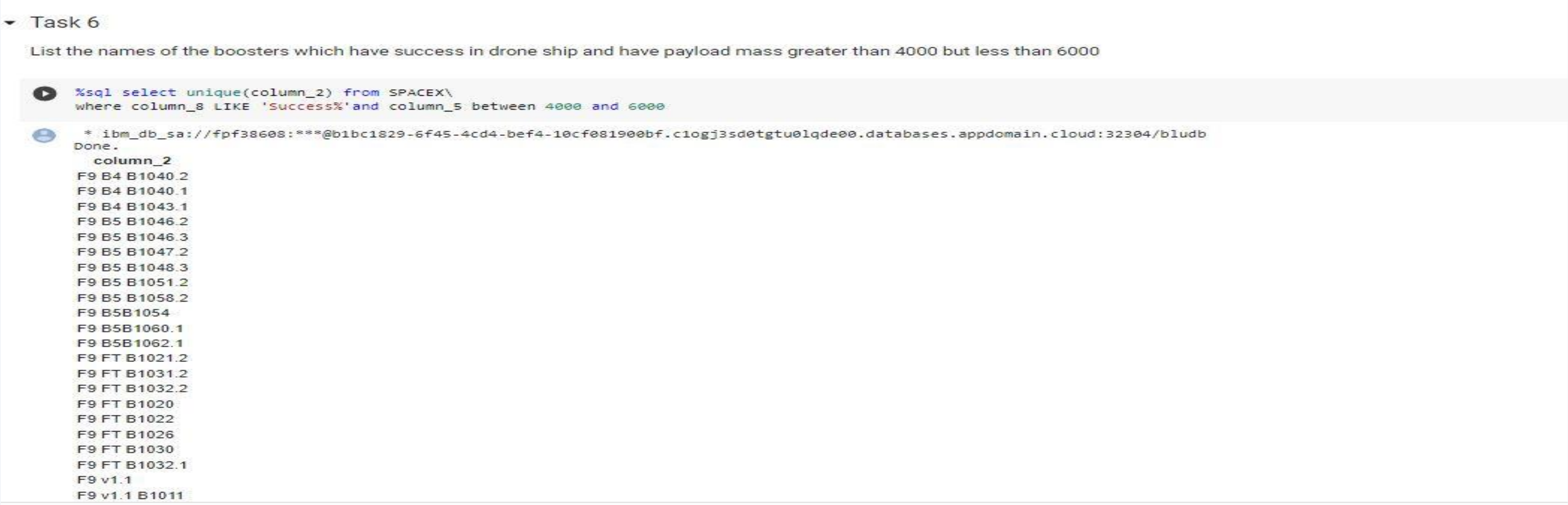
Hint:Use min function

```
[ ] %sql select min(column_0) from SPACEX\
    where column_8 LIKE 'Success%'
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
    1
2010-04-06
```

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

```
%sql select unique(column_2) from SPACEX\
where column_8 LIKE 'Success%'and column_5 between 4000 and 6000
```

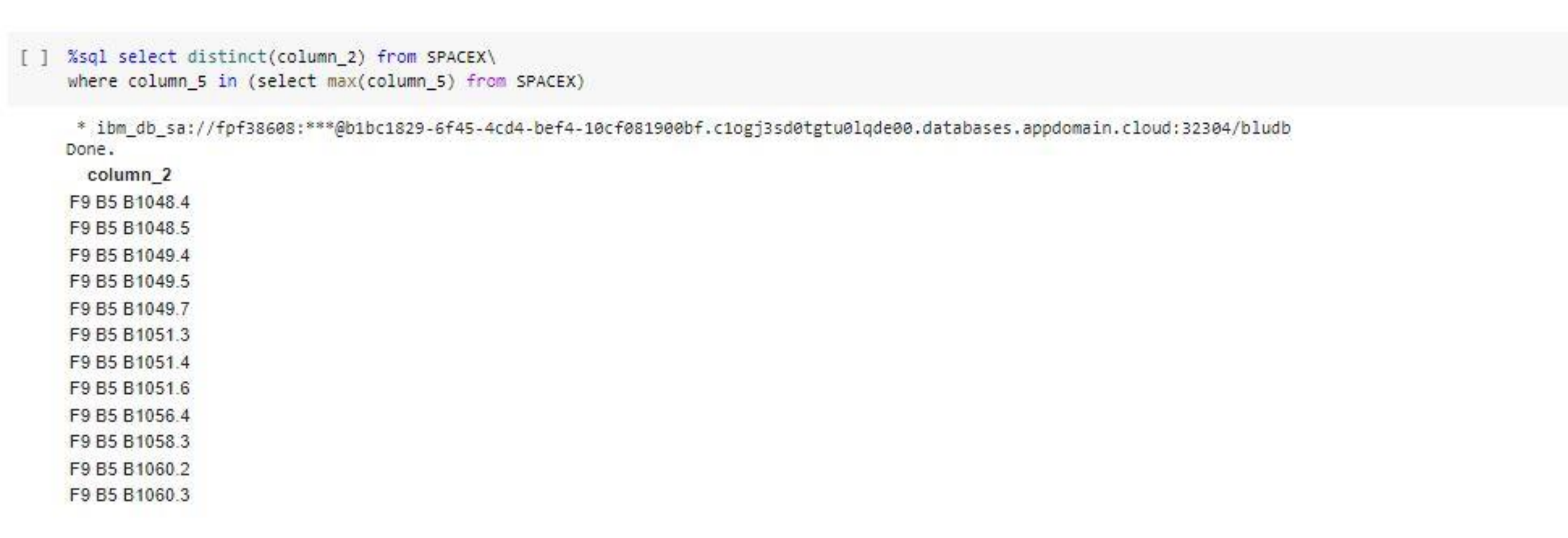
```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
  column_2
F9 B4 B1040.2
F9 B4 B1040.1
F9 B4 B1043.1
F9 B5 B1046.2
F9 B5 B1046.3
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5 B1058.2
F9 B5B1054
F9 B5B1060.1
F9 B5B1062.1
F9 FT B1021.2
F9 FT B1031.2
F9 FT B1032.2
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1032.1
F9 v1.1
F9 v1.1 B1011
```

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

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



```
[ ]  %sql select distinct(column_2) from SPACEX\
     where column_5 in (select max(column_5) from SPACEX)
```

```
 * ibm_db_sa://fpf38608:***@b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32304/bludb
Done.
```

column_2
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

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

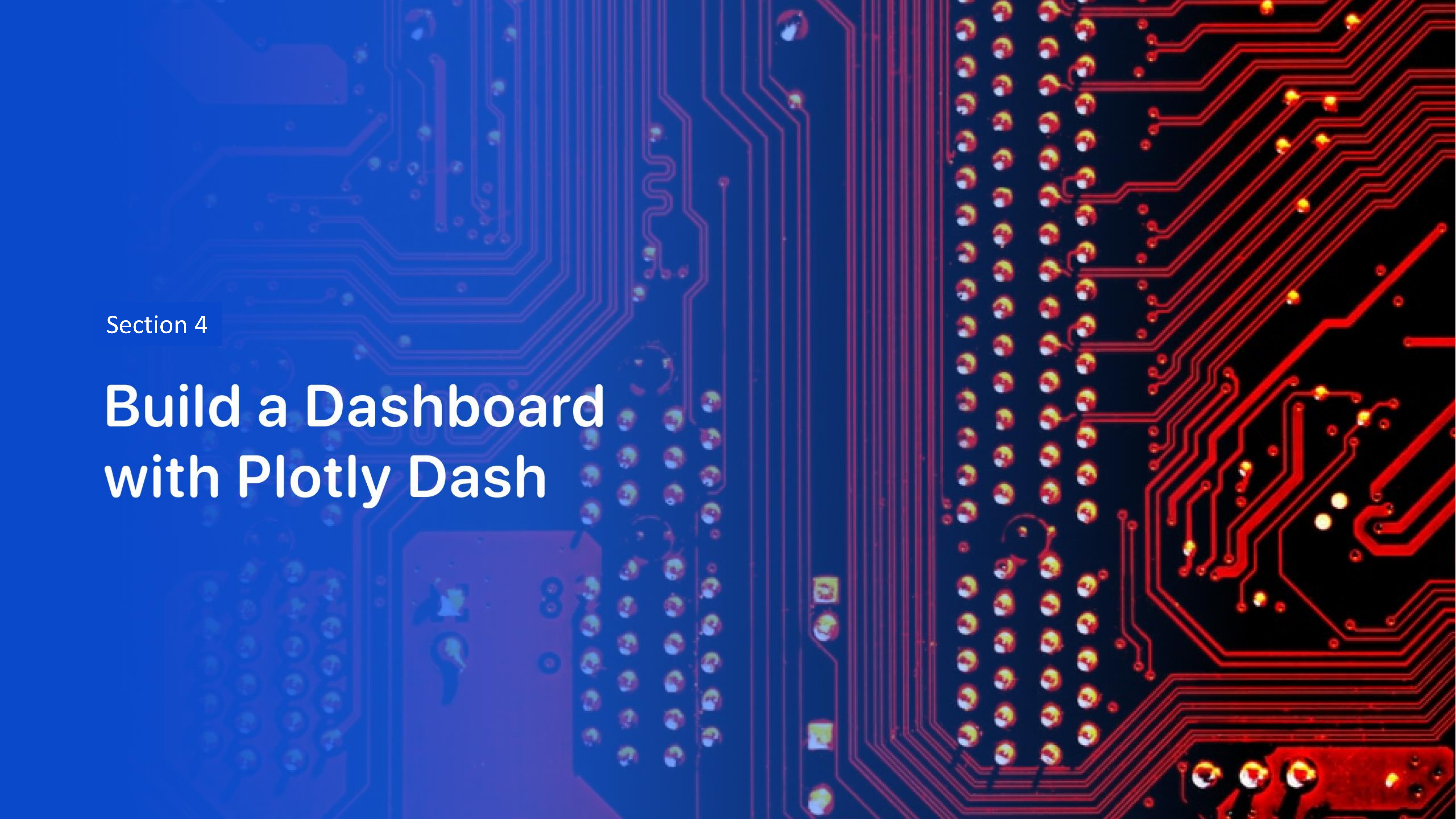
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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 query result with a short explanation here

Section 3

Launch Sites Proximities Analysis



Section 4

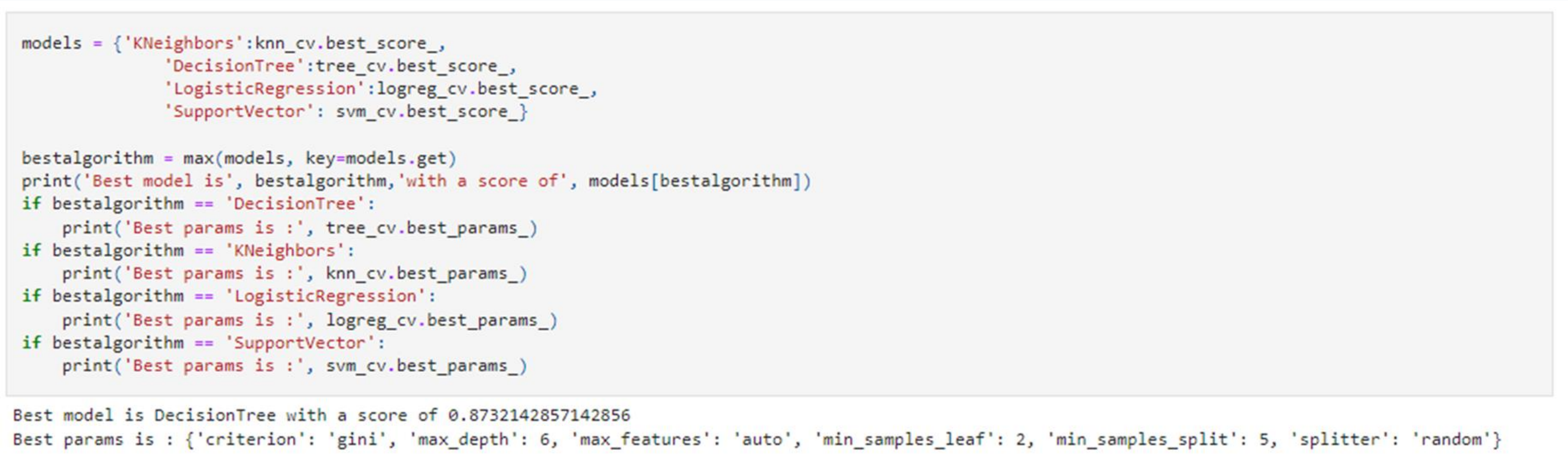
Build a Dashboard with Plotly Dash

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart
- Find which model has the highest classification accuracy



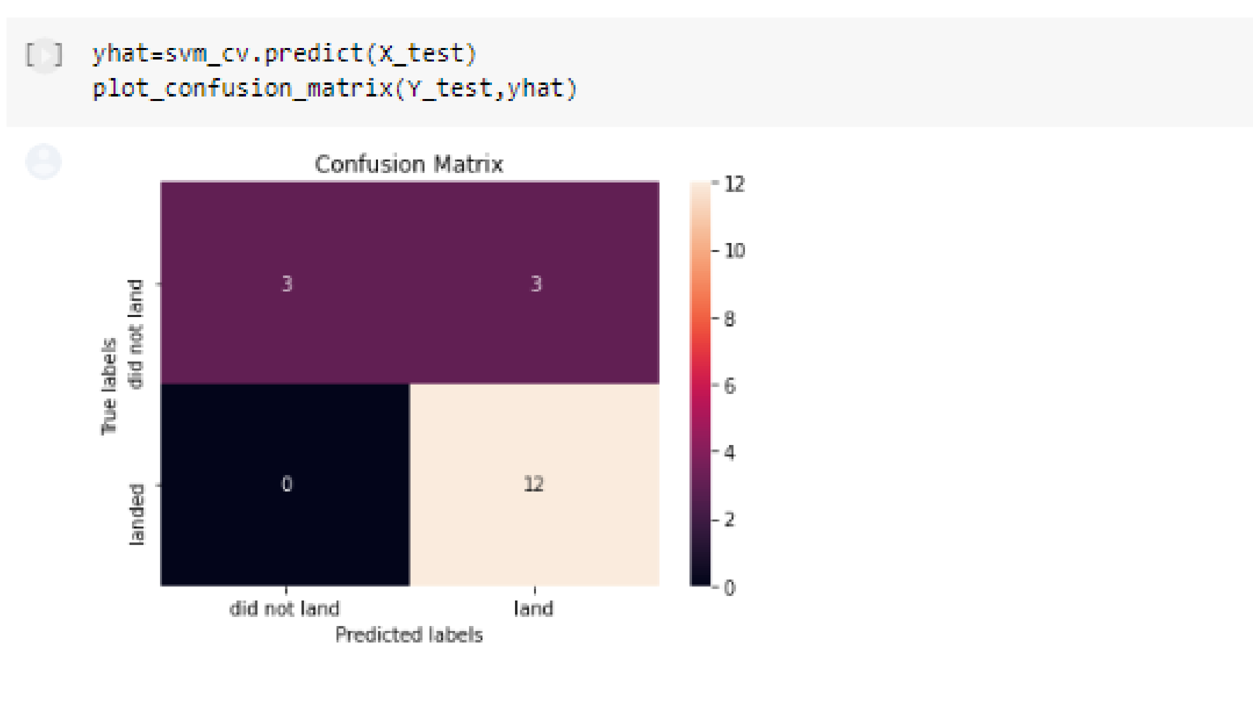
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation



```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Thank you!