In [1]:
```
import requests
import pandas as pd
import numpy as np
import datetime
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
print("All libraries have been imported.")
```

```
All libraries have been imported.
```

In [3]:
```
# NOTE: This code was provided.
# Takes the dataset and uses the rocket column to call the API and append the booster version to the DataFrame
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

In [4]:
```
# NOTE: This code was provided.
# Takes the dataset and uses the launchpad column to call the API and append the latitude and longitude to the DataFr
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

In [5]:
```
# NOTE: This code was provided.
# Takes the dataset and uses the payloads column to call the API and append the payload mass to the DataFrame
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

In [6]:
```
# NOTE: This code was provided.
# Takes the dataset and uses the cores column to call the API and append the data about the cores to the DataFrame
def getCoreData(data):
```

```
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Task 1: Request and parse the SpaceX launch data using the GET request

In [8]:
```
# Convert JSON file into DataFrame
static_json_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/data
response = requests.get(static_json_url)
response_json = response.json()
data_initial = pd.json_normalize(response_json)
data_initial.head(1)
```

Out[8]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	cap
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	

In [9]:
```
data_initial.shape
```

Out[9]: (107, 42)

In [10]:
```
# View column names
pd.DataFrame(data_initial.columns)
```

Out[10]:

	0
0	static_fire_date_utc
1	static_fire_date_unix
2	tbd
3	net
4	window
5	rocket
6	success
7	details
8	crew
9	ships
10	capsules
11	payloads
12	launchpad
13	auto_update
14	failures
15	flight_number
16	name
17	date_utc
18	date_unix
19	date_local
20	date_precision
21	upcoming

	0
22	cores
23	id
24	fairings.reused
25	fairings.recovery_attempt
26	fairings.recovered
27	fairings.ships
28	links.patch.small
29	links.patch.large
30	links.reddit.campaign
31	links.reddit.launch
32	links.reddit.media
33	links.reddit.recovery
34	links.flickr.small
35	links.flickr.original
36	links.presskit
37	links.webcast
38	links.youtube_id
39	links.article
40	links.wikipedia
41	fairings

DataFrame of Launch Data - Selected Information

In [11]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data_initial[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feat
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

In [12]:
```
# Set global variables to be empty lists
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

In [13]:
```
# Confirm list to be empty
BoosterVersion
```

Out[13]:
```
[]
```

In [14]:
```
# Call getBoosterVersion
getBoosterVersion(data)
```

In [15]:
```
# Call getLaunchSite
getLaunchSite(data)
```

In [16]:
```
# Call getPayloadData
getPayloadData(data)
```

In [17]:
```
# Call getCoreData
getCoreData(data)
```

In [18]:
```
# The lists has now been updated
BoosterVersion[0:5]
```

Out[18]:
```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

In [19]:
```
# Combine the columns into a dictionary
launch_dict = {'FlightNumber': list(data['flight_number']),
               'Date': list(data['date']),
               'BoosterVersion':BoosterVersion,
               'PayloadMass':PayloadMass,
               'Orbit':Orbit,
               'LaunchSite':LaunchSite,
               'Outcome':Outcome,
               'Flights':Flights,
               'GridFins':GridFins,
               'Reused':Reused,
               'Legs':Legs,
               'LandingPad':LandingPad,
               'Block':Block,
               'ReusedCount':ReusedCount,
               'Serial':Serial,
               'Longitude': Longitude,
               'Latitude': Latitude}
```

In [20]:
```
# Create a DataFrame from launch_dict
launch_df = pd.DataFrame(launch_dict)
launch_df.head(3)
```

Out[20]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPa
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	Nor
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	Nor
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	Nor

In [21]:
```
launch_df.shape
```

Out[21]:
```
(94, 17)
```

TASK 2

In [22]:
```
# Quantify types of booster versions.
launch_df['BoosterVersion'].value_counts()
```

Out[22]:
```
Falcon 9    90
Falcon 1     4
Name: BoosterVersion, dtype: int64
```

In [23]:
```
# Exclude all launches except those with the Falcon 9 booster.
data_falcon_9 = launch_df.loc[launch_df['BoosterVersion'].isin(['Falcon 9'])]
data_falcon_9.head(2)
```

Out[23]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPa
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	Nor
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	Nor

In [24]:
```
# Confirm that only the Falcon 9 booster is included.
data_falcon_9['BoosterVersion'].value_counts()
```

Out[24]:
```
Falcon 9    90
Name: BoosterVersion, dtype: int64
```

In [25]:
```
# Reset the FlightNumber column
data_falcon_9.loc[:,'FlightNumber'] = list(range(1, data_falcon_9.shape[0]+1))
data_falcon_9.head(2)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

Out[25]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPa
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	Nor
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	Nor

In [26]:
```
data_falcon_9.shape
```

Out[26]:
```
(90, 17)
```

In [27]:
```
data_falcon_9.describe()
```

Out[27]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Longitude	Latitude
count	90.000000	85.000000	90.000000	90.000000	90.000000	90.000000	90.000000
mean	45.500000	6123.547647	1.788889	3.500000	3.188889	-86.366477	29.449963
std	26.124701	4870.916417	1.213172	1.595288	4.194417	14.149518	2.141306
min	1.000000	350.000000	1.000000	1.000000	0.000000	-120.610829	28.561857
25%	23.250000	2482.000000	1.000000	2.000000	0.000000	-80.603956	28.561857
50%	45.500000	4535.000000	1.000000	4.000000	1.000000	-80.577366	28.561857
75%	67.750000	9600.000000	2.000000	5.000000	4.000000	-80.577366	28.608058
max	90.000000	15600.000000	6.000000	5.000000	13.000000	-80.577366	34.632093

Data Wrangeling

In [28]:
```
# There are some missing values in the dataset
data_falcon_9.isnull().sum()
```

Out[28]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Task 3: Dealing with Missing Values

In [32]:
```
# Calculate the mean value of the values in the PayloadMass column and replace the np.nan values with this mean value
mean = data_falcon_9['PayloadMass'].mean()
data_falcon_9['PayloadMass'].replace(np.nan, mean, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

In [34]:
```
# There are now no missing values for 'PayLoadMass'. We keep the 'None' values in the 'LandingPad' column to represer
data_falcon_9.isnull().sum()
```

Out[34]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Export DataFrame to .CSV

In [35]:
```
# Export DataFrame as .csv
data_falcon_9.to_csv('dataset_part_1.csv', index=False)
```

In [ ]:
```
```