Assignment: SQL Notebook for Peer Assignment

Estimated time needed: **60** minutes.

Introduction

Using this Python notebook you will:

1. Understand the Spacex DataSet
2. Load the dataset into the corresponding table in a Db2 database
3. Execute SQL queries to answer assignment questions

Overview of the DataSet

SpaceX has gained worldwide attention for a series of historic milestones.

It is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars wheras other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage.

Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

This dataset includes a record for each payload carried during a SpaceX mission into outer space.

Download the datasets

This assignment requires you to load the spacex dataset.

In many cases the dataset to be analyzed is available as a .CSV (comma separated values) file, perhaps on the internet. Click on the link below to download and save the dataset (.CSV file):

Spacex DataSet

In [1]:

```
!pip install sqlalchemy==1.3.9
```

```
Collecting sqlalchemy==1.3.9
  Downloading SQLAlchemy-1.3.9.tar.gz (6.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.0/6.0 MB 96.8 MB/s eta 0:0
0:00
  Preparing metadata (setup.py) ... one
Building wheels for collected packages: sqlalchemy
  Building wheel for sqlalchemy (setup.py) ..done
  Created wheel for sqlalchemy: filename=SQLAlchemy-1.3.9-cp312-cp312-linu
x_x86_64.whl size=1160111 sha256=42f4ce89ab7332eb8fe99cfbed0d59ca78d887a3a
04babc58232c9a051c252a8
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/b3/1c/42/0e26b8d
512adc6bce10ff71a05229366b4ccec641cd3b42111
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 2.0.37
    Uninstalling SQLAlchemy-2.0.37:
      Successfully uninstalled SQLAlchemy-2.0.37
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the follo
wing dependency conflicts.
jupyterhub 5.2.1 requires SQLAlchemy>=1.4.1, but you have sqlalchemy 1.3.9
which is incompatible.
Successfully installed sqlalchemy-1.3.9
```

Connect to the database

Let us first load the SQL extension and establish a connection with the database

In [2]:

```
!pip install ipython-sql
!pip install ipython-sql prettytable
```

```
Collecting ipython-sql
  Downloading ipython_sql-0.5.0-py3-none-any.whl.metadata (17 kB)
Collecting prettytable (from ipython-sql)
  Downloading prettytable-3.16.0-py3-none-any.whl.metadata (33 kB)
Requirement already satisfied: ipython in /opt/conda/lib/python3.12/site-p
ackages (from ipython-sql) (8.31.0)
Collecting sqlalchemy>=2.0 (from ipython-sql)
  Downloading sqlalchemy-2.0.41-cp312-cp312-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl.metadata (9.6 kB)
Collecting sqlparse (from ipython-sql)
  Downloading sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: six in /opt/conda/lib/python3.12/site-packa
ges (from ipython-sql) (1.17.0)
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python3.
12/site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: greenlet>=1 in /opt/conda/lib/python3.12/si
te-packages (from sqlalchemy>=2.0->ipython-sql) (3.1.1)
Requirement already satisfied: typing-extensions>=4.6.0 in /opt/conda/lib/
python3.12/site-packages (from sqlalchemy>=2.0->ipython-sql) (4.12.2)
Requirement already satisfied: decorator in /opt/conda/lib/python3.12/site
-packages (from ipython->ipython-sql) (5.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/conda/lib/python3.12/sit
e-packages (from ipython->ipython-sql) (0.19.2)
Requirement already satisfied: matplotlib-inline in /opt/conda/lib/python
3.12/site-packages (from ipython->ipython-sql) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.12/si
te-packages (from ipython->ipython-sql) (4.9.0)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in /opt/cond
a/lib/python3.12/site-packages (from ipython->ipython-sql) (3.0.50)
Requirement already satisfied: pygments>=2.4.0 in /opt/conda/lib/python3.1
2/site-packages (from ipython->ipython-sql) (2.19.1)
Requirement already satisfied: stack_data in /opt/conda/lib/python3.12/sit
e-packages (from ipython->ipython-sql) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in /opt/conda/lib/python
3.12/site-packages (from ipython->ipython-sql) (5.14.3)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.12/site-p
ackages (from prettytable->ipython-sql) (0.2.13)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /opt/conda/lib/pytho
n3.12/site-packages (from jedi>=0.16->ipython->ipython-sql) (0.8.4)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.1
2/site-packages (from pexpect>4.3->ipython->ipython-sql) (0.7.0)
Requirement already satisfied: executing>=1.2.0 in /opt/conda/lib/python3.
12/site-packages (from stack_data->ipython->ipython-sql) (2.1.0)
Requirement already satisfied: asttokens>=2.1.0 in /opt/conda/lib/python3.
12/site-packages (from stack_data->ipython->ipython-sql) (3.0.0)
Requirement already satisfied: pure_eval in /opt/conda/lib/python3.12/site
-packages (from stack_data->ipython->ipython-sql) (0.2.3)
Downloading ipython_sql-0.5.0-py3-none-any.whl (20 kB)
Downloading sqlalchemy-2.0.41-cp312-cp312-manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (3.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 95.2 MB/s eta 0:00:
00
Downloading prettytable-3.16.0-py3-none-any.whl (33 kB)
Downloading sqlparse-0.5.3-py3-none-any.whl (44 kB)
Installing collected packages: sqlparse, sqlalchemy, prettytable, ipython-
sql
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 1.3.9
    Uninstalling SQLAlchemy-1.3.9:
      Successfully uninstalled SQLAlchemy-1.3.9
Successfully installed ipython-sql-0.5.0 prettytable-3.16.0 sqlalchemy-2.
0.41 sqlparse-0.5.3
Requirement already satisfied: ipython-sql in /opt/conda/lib/python3.12/si
te-packages (0.5.0)
Requirement already satisfied: prettytable in /opt/conda/lib/python3.12/si
te-packages (3.16.0)
Requirement already satisfied: ipython in /opt/conda/lib/python3.12/site-p
ackages (from ipython-sql) (8.31.0)
Requirement already satisfied: sqlalchemy>=2.0 in /opt/conda/lib/python3.1
2/site-packages (from ipython-sql) (2.0.41)
Requirement already satisfied: sqlparse in /opt/conda/lib/python3.12/site-
packages (from ipython-sql) (0.5.3)
Requirement already satisfied: six in /opt/conda/lib/python3.12/site-packa
ges (from ipython-sql) (1.17.0)
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python3.
12/site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.12/site-p
ackages (from prettytable) (0.2.13)
Requirement already satisfied: greenlet>=1 in /opt/conda/lib/python3.12/si
te-packages (from sqlalchemy>=2.0->ipython-sql) (3.1.1)
Requirement already satisfied: typing-extensions>=4.6.0 in /opt/conda/lib/
python3.12/site-packages (from sqlalchemy>=2.0->ipython-sql) (4.12.2)
Requirement already satisfied: decorator in /opt/conda/lib/python3.12/site
-packages (from ipython->ipython-sql) (5.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/conda/lib/python3.12/sit
e-packages (from ipython->ipython-sql) (0.19.2)
Requirement already satisfied: matplotlib-inline in /opt/conda/lib/python
3.12/site-packages (from ipython->ipython-sql) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.12/si
te-packages (from ipython->ipython-sql) (4.9.0)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in /opt/cond
a/lib/python3.12/site-packages (from ipython->ipython-sql) (3.0.50)
Requirement already satisfied: pygments>=2.4.0 in /opt/conda/lib/python3.1
2/site-packages (from ipython->ipython-sql) (2.19.1)
Requirement already satisfied: stack_data in /opt/conda/lib/python3.12/sit
e-packages (from ipython->ipython-sql) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in /opt/conda/lib/python
3.12/site-packages (from ipython->ipython-sql) (5.14.3)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /opt/conda/lib/pytho
n3.12/site-packages (from jedi>=0.16->ipython->ipython-sql) (0.8.4)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.1
2/site-packages (from pexpect>4.3->ipython->ipython-sql) (0.7.0)
Requirement already satisfied: executing>=1.2.0 in /opt/conda/lib/python3.
12/site-packages (from stack_data->ipython->ipython-sql) (2.1.0)
Requirement already satisfied: asttokens>=2.1.0 in /opt/conda/lib/python3.
12/site-packages (from stack_data->ipython->ipython-sql) (3.0.0)
Requirement already satisfied: pure_eval in /opt/conda/lib/python3.12/site
-packages (from stack_data->ipython->ipython-sql) (0.2.3)
```

In [3]:

```
%load_ext sql
```

In [4]:

```
import csv, sqlite3
import prettytable
prettytable.DEFAULT = 'DEFAULT'

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

In [5]:

```
!pip install -q pandas
```

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

```
In [7]: import pandas as pd
        df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appd
        df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="mult
```

```
Out[7]: 101
```

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

```
In [8]: #DROP THE TABLE IF EXISTS

        %sql DROP TABLE IF EXISTS SPACEXTABLE;
```

```
 * sqlite:///my_data1.db
Done.
Out[8]: []
```

```
In [9]: %sql create table SPACEXTABLE as select * from SPACEXTBL where Date is no
```

```
 * sqlite:///my_data1.db
Done.
Out[9]: []
```

```
In [12]: columns = df.columns.tolist()
         print("Columns:", columns)
```

```
Columns: ['Date', 'Time (UTC)', 'Booster_Version', 'Launch_Site', 'Payloa
d', 'PAYLOAD_MASS__KG_', 'Orbit', 'Customer', 'Mission_Outcome', 'Landing_
Outcome']
```

Tasks

Now write and execute SQL queries to solve the assignment tasks.

Note: If the column names are in mixed case enclose it in double quotes For Example "Landing_Outcome"

Task 1

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

```
In [13]: unique_launch_sites = df['Launch_Site'].unique()

         print("Unique Launch Sites:")
         for site in unique_launch_sites:
             print("-", site)
```

```
Unique Launch Sites:
- CCAFS LC-40
- VAFB SLC-4E
- KSC LC-39A
- CCAFS SLC-40
```

Task 2

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

In [34]:

```
print("Top 5 launches from CCAFS:")
cca_launches = df[df['Launch_Site'].str.startswith('CCA')]
print(cca_launches.head(5)[['Launch_Site', 'PAYLOAD_MASS__KG_', 'Booster_
```

```
Top 5 launches from CCAFS:
   Launch_Site  PAYLOAD_MASS__KG_ Booster_Version
0  CCAFS LC-40                  0  F9 v1.0  B0003
1  CCAFS LC-40                  0  F9 v1.0  B0004
2  CCAFS LC-40                525  F9 v1.0  B0005
3  CCAFS LC-40                500  F9 v1.0  B0006
4  CCAFS LC-40                677  F9 v1.0  B0007
```

Task 3

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

In [36]:

```
print("\nTotal Payload Mass for NASA (CRS):")
total_payload_nasa = df[df['Customer'].str.contains('NASA', case=False, n
print(f"{total_payload_nasa} kg")
```

```
Total Payload Mass for NASA (CRS):
107010 kg
```

Task 4

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

In [37]:

```
print("\nAvg Payload Mass for F9 v1.1:")
avg_payload_f9v11 = df[df['Booster_Version'] == 'F9 v1.1']['PAYLOAD_MASS_
print(f"Average Payload Mass: {avg_payload_f9v11:.2f} kg")
```

```
Avg Payload Mass for F9 v1.1:
Average Payload Mass: 2928.40 kg
```

Task 5

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

Hint:Use min function

In [38]:

```
print("\nFirst successful ground pad landing:")
first_ground_pad_success = df[df['Landing_Outcome'] == 'Success (ground p
print(f"First success on ground pad: {first_ground_pad_success.date()}")
```

```
First successful ground pad landing:
First success on ground pad: 2015-12-22
```

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 [35]:

```
print("\nBoosters with successful drone ship landing and PayloadMass 4000

filtered = df[
    (df['Landing_Outcome'] == 'Success (drone ship)') &
    (df['PAYLOAD_MASS__KG_'] > 4000) &
    (df['PAYLOAD_MASS__KG_'] < 6000)
]

print(filtered[['Booster_Version', 'PAYLOAD_MASS__KG_']])
```

```
Boosters with successful drone ship landing and PayloadMass 4000–6000 kg:
   Booster_Version  PAYLOAD_MASS__KG_
23      F9 FT B1022               4696
27      F9 FT B1026               4600
31   F9 FT  B1021.2               5300
42   F9 FT  B1031.2               5200
```

In [31]:

```
%%sql
SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE
);
```

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

Out[31]:

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

Task 7

List the total number of successful and failure mission outcomes

In [24]:

```
# Count successful and failed missions
success_count = df['Landing_Outcome'].str.contains('Success').sum()
failure_count = df['Landing_Outcome'].str.contains('Failure').sum()

print("\nTotal Mission Outcomes:")
print(f"Success: {success_count}")
print(f"Failure: {failure_count}")
```

```
Total Mission Outcomes:
Success: 61
Failure: 10
```

In [30]:

```
%%sql
SELECT
    SUM(CASE WHEN Landing_Outcome LIKE 'Success%' THEN 1 ELSE 0 END) AS s
    SUM(CASE WHEN Landing_Outcome LIKE 'Failure%' THEN 1 ELSE 0 END) AS f
FROM SPACEXTABLE;
```

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

Out[30]:

success_count	failure_count
61	10

In [28]:

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS total
FROM SPACEXTABLE
GROUP BY Landing_Outcome;
```

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

Out[28]:

Landing_Outcome	total
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

In [ ]:

```
SELECT booster_version
FROM launches
WHERE payload_mass = (
    SELECT MAX(payload_mass) FROM launches
);
```

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 [21]:
```
from datetime import datetime
# Convert Date to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Extract year and month for filtering and display
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.strftime('%B')  # Full month name (e.g., Janu

# Filter for 2015
df_2015 = df[df['Date'].dt.year == 2015].copy()

# Filter for drone ship failures
df_2015['LandingOutcome'] = df_2015['Landing_Outcome'].str.strip()
df_drone_failures = df_2015[df_2015['Landing_Outcome'] == 'Failure (drone

# Count by month
monthly_drone_fails = df_drone_failures.groupby('Month').size().reset_ind
monthly_drone_fails = monthly_drone_fails.sort_values('Month')

print("\nMonthly Drone Ship Failures in 2015")
print(monthly_drone_fails)
```

```
Monthly Drone Ship Failures in 2015
     Month  Failures
0    April         1
1  January         1
```

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 [23]:
```
start_date = pd.to_datetime('2010-06-04')
end_date = pd.to_datetime('2017-03-20')

# Filter data
df_filtered = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]

# Count landing outcomes
landing_rank = df_filtered['Landing_Outcome'].value_counts().reset_index(
landing_rank.columns = ['Landing_Outcome', 'Count']

print("\nLanding Outcome Counts Between 2010-06-04 and 2017-03-20")
print(landing_rank.sort_values(by='Count', ascending=False))
```

```
Landing Outcome Counts Between 2010-06-04 and 2017-03-20
          Landing_Outcome  Count
0              No attempt     10
1     Failure (drone ship)     5
2     Success (drone ship)     5
3       Controlled (ocean)     3
4     Success (ground pad)     3
5      Failure (parachute)     2
6     Uncontrolled (ocean)     2
7   Precluded (drone ship)     1
```

Reference Links

- Hands-on Lab : String Patterns, Sorting and Grouping
- Hands-on Lab: Built-in functions
- Hands-on Lab : Sub-queries and Nested SELECT Statements
- Hands-on Tutorial: Accessing Databases with SQL magic
- Hands-on Lab: Analyzing a real World Data Set

Author(s)

Lakshmi Holla

Other Contributors

Rav Ahuja