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 [6]:
```
!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 76.9 MB/s eta 0:00:00:0
0:0100:01
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: sqlalchemy
  Building wheel for sqlalchemy (setup.py) ... done
  Created wheel for sqlalchemy: filename=SQLAlchemy-1.3.9-cp37-cp37m-linux_x86_6
4.whl size=1159121 sha256=cd13eeaad2cc8b88ee9ab83a137b0fc4519a722a023398d2097b495
27488a593
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/03/71/13/010faf12246f72
dc76b4150e6e599d13a85b4435e06fb9e51f
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 1.3.24
    Uninstalling SQLAlchemy-1.3.24:
      Successfully uninstalled SQLAlchemy-1.3.24
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 [ ]:
```
#Please uncomment and execute the code below if you are working locally.

#!pip install ipython-sql
```

In [7]:
```
%load_ext sql
```

In [8]:
```
import csv, sqlite3

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

In [9]:
```
!pip install -q pandas==1.1.5
```

In [10]:
```
%sql sqlite:///my_data1.db
```

Out[10]:
```
'Connected: @my_data1.db'
```

In [20]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.c
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

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

In [12]:
```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

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

Out[12]:
```
[]
```

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 [41]:
```
# First, establish a connection to your SQLite database (if not already done)
%reload_ext sql
%sql sqlite:///my_data1.db

# Now execute your SQL query to select distinct launch sites
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

Out[41]:

Launch_Site
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 [46]:
```
# First, ensure you have loaded the SQL extension and connected to your database
%reload_ext sql
%sql sqlite:///my_data1.db

# Then, execute your SQL query to fetch records
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[46]:

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

Task 4

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

In [65]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to calculate average payload mass for booster version 'Falco
%sql select PAYLOAD_MASS__KG_  from SPACEXTABLE where Booster_Version LIKE 'Falc
```

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

Out[65]:

PAYLOAD_MASS__KG_

Task 5

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

Hint:Use min function

In [67]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to find the date of the first successful landing on a ground
%sql SELECT MIN(Date) AS FirstSuccessfulGroundPadLandingDate FROM SPACEXTABLE WH
```

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

Out[67]:

FirstSuccessfulGroundPadLandingDate
2018-07-22

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 [76]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to list boosters meeting the criteria
%sql select  Booster_Version from SPACEXTABLE where  Payload > 4000 AND Payload
```

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

Out[76]:

Booster_Version

Task 7

List the total number of successful and failure mission outcomes

In [78]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to count successful and failure mission outcomes
%sql SELECT Mission_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Mission_Ou
```

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

Out[78]:

Mission_Outcome	Count
Success	98

Task 8

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

In [79]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query using subquery to find booster_versions with maximum payload
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Payload = (SELECT MAX(Payload
```

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

Out[79]:

Booster_Version
F9 B4 B1043.1

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 [92]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to list records meeting the criteria
%sql SELECT Time('%m') as  Landing_Outcome, Booster_Version, Launch_Site FROM SP
```

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

Out[92]:

Landing_Outcome	Booster_Version	Launch_Site

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 [88]:
```
# Load SQL extension and connect to SQLite database
%reload_ext sql
%sql sqlite:///my_data1.db

# Execute SQL query to rank landing outcomes between specified dates
%sql SELECT Landing_Outcome, COUNT(*) AS OutcomeCount FROM SPACEXTABLE WHERE Dat
```

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

Out[88]:

Landing_Outcome	OutcomeCount
Failure (drone ship)	5
Success (ground pad)	3

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

Change log

Date	Version	Changed by	Change Description
2021-07-09	0.2	Lakshmi Holla	Changes made in magic sql
2021-05-20	0.1	Lakshmi Holla	Created Initial Version