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 [ ]:
```
!pip install sqlalchemy==1.3.9
```

Connect to the database

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

In [ ]:
```
!pip install ipython-sql
!pip install ipython-sql prettytable
```

In [1]:
```
%load_ext sql
```

In [2]:
```
import csv, sqlite3
import prettytable
prettytable.DEFAULT = 'DEFAULT'

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

In [ ]:
```
!pip install -q pandas
```

In [3]:
```
%sql sqlite:///my_data1.db
```

In [4]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storag
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method
```

Out[4]:
```
101
```

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

In [5]:
```
#DROP THE TABLE IF EXISTS

%sql DROP TABLE IF EXISTS SPACEXTABLE;
```

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

Out[5]:
```
[]
```

In [6]:
```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date
```

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

Out[6]:
```
[]
```

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 [9]:
```
query = 'SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;'

# Execute the query and fetch the results
unique_launch_sites = pd.read_sql_query(query, con)

# Display the results
print(unique_launch_sites)
```

```
   Launch_Site
0   CCAFS LC-40
1   VAFB SLC-4E
2    KSC LC-39A
3  CCAFS SLC-40
```

Task 2

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

In [10]:
```
query = 'SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE "CCA%"

# Execute the query and fetch the results
cca_launch_sites = pd.read_sql_query(query, con)

# Display the results
print(cca_launch_sites)
```

```
         Date Time (UTC) Booster_Version  Launch_Site  \
0  2010-06-04   18:45:00  F9 v1.0  B0003  CCAFS LC-40
1  2010-12-08   15:43:00  F9 v1.0  B0004  CCAFS LC-40
2  2012-05-22    7:44:00  F9 v1.0  B0005  CCAFS LC-40
3  2012-10-08    0:35:00  F9 v1.0  B0006  CCAFS LC-40
4  2013-03-01   15:10:00  F9 v1.0  B0007  CCAFS LC-40

                                                Payload  PAYLOAD_MASS__
KG_  \
0                Dragon Spacecraft Qualification Unit
0
1  Dragon demo flight C1, two CubeSats, barrel of...
0
2                               Dragon demo flight C2
525
3                                         SpaceX CRS-1
500
4                                         SpaceX CRS-2
677

       Orbit         Customer Mission_Outcome      Landing_Outcome
0        LEO           SpaceX         Success  Failure (parachute)
1  LEO (ISS)  NASA (COTS) NRO         Success  Failure (parachute)
2  LEO (ISS)      NASA (COTS)         Success           No attempt
3  LEO (ISS)       NASA (CRS)         Success           No attempt
4  LEO (ISS)       NASA (CRS)         Success           No attempt
```

Task 3

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

In [27]:
```
query = '''
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" LIKE "%NASA%" AND "Payload" LIKE "%CRS%";
'''

# Execute the query and fetch results
total_payload_mass = pd.read_sql_query(query, con)
print(total_payload_mass)
```

```
   Total_Payload_Mass
0               60268
```

Task 4

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

In [28]:
```
query = '''
SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" LIKE "F9 v1.1";
'''

# Execute the query and fetch the results
```

```
average_payload_mass = pd.read_sql_query(query, con)
print(average_payload_mass)
```

```
   Average_Payload_Mass
0                2928.4
```

Task 5

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

Hint:Use min function

In [29]:
```
query = '''
SELECT MIN("Date") AS First_Successful_Landing_Date
FROM SPACEXTABLE
WHERE "Landing_Outcome" = "Success (ground pad)";
'''

# Execute the query and fetch the result
first_successful_landing_date = pd.read_sql_query(query, con)
print(first_successful_landing_date)
```

```
  First_Successful_Landing_Date
0                    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 [30]:
```
query = '''
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = "Success (drone ship)"
  AND "PAYLOAD_MASS__KG_" > 4000
  AND "PAYLOAD_MASS__KG_" < 6000;
'''

# Execute the query and fetch the results
boosters = pd.read_sql_query(query, con)
print(boosters)
```

```
  Booster_Version
0     F9 FT B1022
1     F9 FT B1026
2  F9 FT  B1021.2
3  F9 FT  B1031.2
```

Task 7

List the total number of successful and failure mission outcomes

In [31]:
```
query = '''
SELECT "Mission_Outcome", COUNT(*) AS Total_Missions
FROM SPACEXTABLE
```

```
GROUP BY "Mission_Outcome";
'''

# Execute the query and fetch the results
mission_outcomes = pd.read_sql_query(query, con)
print(mission_outcomes)
```

```
                  Mission_Outcome  Total_Missions
0              Failure (in flight)               1
1                          Success              98
2                         Success               1
3  Success (payload status unclear)              1
```

Task 8

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

In [32]:
```
query = '''
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
);
'''

# Execute the query and fetch the results
max_payload_boosters = pd.read_sql_query(query, con)
print(max_payload_boosters)
```

```
   Booster_Version
0    F9 B5 B1048.4
1    F9 B5 B1049.4
2    F9 B5 B1051.3
3    F9 B5 B1056.4
4    F9 B5 B1048.5
5    F9 B5 B1051.4
6    F9 B5 B1049.5
7   F9 B5 B1060.2
8   F9 B5 B1058.3
9    F9 B5 B1051.6
10   F9 B5 B1060.3
11  F9 B5 B1049.7
```

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 [33]:
```
query = '''
```

```
SELECT
    CASE
        WHEN SUBSTR("Date", 6, 2) = '01' THEN 'January'
        WHEN SUBSTR("Date", 6, 2) = '02' THEN 'February'
        WHEN SUBSTR("Date", 6, 2) = '03' THEN 'March'
        WHEN SUBSTR("Date", 6, 2) = '04' THEN 'April'
        WHEN SUBSTR("Date", 6, 2) = '05' THEN 'May'
        WHEN SUBSTR("Date", 6, 2) = '06' THEN 'June'
        WHEN SUBSTR("Date", 6, 2) = '07' THEN 'July'
        WHEN SUBSTR("Date", 6, 2) = '08' THEN 'August'
        WHEN SUBSTR("Date", 6, 2) = '09' THEN 'September'
        WHEN SUBSTR("Date", 6, 2) = '10' THEN 'October'
        WHEN SUBSTR("Date", 6, 2) = '11' THEN 'November'
        WHEN SUBSTR("Date", 6, 2) = '12' THEN 'December'
    END AS "Month_Name",
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = "Failure (drone ship)"
  AND SUBSTR("Date", 0, 5) = '2015';
'''

# Execute the query and fetch the results
failure_landing_2015 = pd.read_sql_query(query, con)
print(failure_landing_2015)
```

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

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 [34]:

```
query = '''
SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
'''

# Execute the query and fetch the results
landing_outcome_ranking = pd.read_sql_query(query, con)
print(landing_outcome_ranking)
```

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