jupyter-labs-eda-sql-coursera_sqllite

December 16, 2022

Assignment: SQL Notebook for Peer Assignment

Estimated time needed: **60** minutes.

0.1 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

0.2 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.

0.2.1 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

0.2.2 Store the dataset in database table

it is highly recommended to manually load the table using the database console LOAD tool in DB2.

Now open the Db2 console, open the LOAD tool, Select / Drag the .CSV file for the dataset, Next create a New Table, and then follow the steps on-screen instructions to load the data. Name the new table as follows:

SPACEXDATASET

Follow these steps while using old DB2 UI which is having Open Console Screen

Note:While loading Spacex dataset, ensure that detect datatypes is disabled. Later click on the pencil icon(edit option).

1. Change the Date Format by manually typing DD-MM-YYYY and timestamp format as DD-MM-YYYY HH:MM:SS
2. Change the PAYLOAD_MASS__KG_ datatype to INTEGER.

Changes to be considered when having DB2 instance with the new UI having Go to UI screen

- Refer to this insruction in this link for viewing the new Go to UI screen.
- Later click on **Data link(below SQL)** in the Go to UI screen and click on **Load Data** tab.
- Later browse for the downloaded spacex file.
- Once done select the schema andload the file.

[2]:
```
!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
75.4 MB/s eta 0:00:00:00:0100:01
  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_64.whl size=1159122
sha256=0428e713ab15206688391c56e57c52e5c05b01b03f60b014f1ff589ecfbae7c9
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/ef/95/ac/c232f83b41590
0c26553c64266e1a2b2863bc63e7a5d606c7e
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
```

0.2.3 Connect to the database

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

[3]:
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

[4]:
```
import csv, sqlite3

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

[5]:
```
!pip install -q pandas==1.1.5
```

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

[6]:
```
'Connected: @my_data1.db'
```

[11]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
 ↪cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv")
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-
packages/pandas/core/generic.py:2882: UserWarning: The spaces in these column
names will not be changed. In pandas versions < 0.14, spaces were converted to
underscores.
  both result in 0.1234 being formatted as 0.12.
```

0.3 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"

0.3.1 Task 1

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

[27]:
```
%sql Select Distinct Launch_Site as "Launch_Sites" from SPACEXTBL
```

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

[27]:
```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

0.3.2 Task 2

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

[31]:
```
%sql Select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5;
```

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

[31]:
```
[('04-06-2010', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft
Qualification Unit', 0, 'LEO', 'SpaceX', 'Success', 'Failure (parachute)'),
 ('08-12-2010', '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)'),
 ('22-05-2012', '07:44:00', 'F9 v1.0  B0005', 'CCAFS LC-40', 'Dragon demo flight
C2', 525, 'LEO (ISS)', 'NASA (COTS)', 'Success', 'No attempt'),
 ('08-10-2012', '00:35:00', 'F9 v1.0  B0006', 'CCAFS LC-40', 'SpaceX CRS-1',
500, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt'),
 ('01-03-2013', '15:10:00', 'F9 v1.0  B0007', 'CCAFS LC-40', 'SpaceX CRS-2',
677, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt')]
```

0.3.3 Task 3

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

[32]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM␣
↪SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

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

[32]:
```
[(45596,)]
```

0.3.4 Task 4

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

[41]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Mass from F9 (v1.1)" FROM␣
↪SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[41]:
```
[(2928.4,)]
```

0.3.5 Task 5

List the date when the first succesful landing outcome in ground pad was acheived.
Hint:Use min function

[65]:
```
%sql SELECT MIN(DATE) AS "FIRST SUCCESFUL LANDING" FROM SPACEXTBL WHERE␣
↪"Landing _Outcome" = 'Success (ground pad)';
```

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

[65]: [('01-05-2017',)]

0.3.6 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

[68]:
```
%sql SELECT Booster_Version AS "Success on Dron Ship 4000 < x < 6000 KG" FROM␣
↪SPACEXTBL WHERE "Landing _Outcome" = 'Success (drone ship)' AND␣
↪PAYLOAD_MASS__KG_ > "4000" AND PAYLOAD_MASS__KG_ < "6000"
```

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

[68]: [('F9 FT B1022',), ('F9 FT B1026',), ('F9 FT B1021.2',), ('F9 FT B1031.2',)]

0.3.7 Task 7

List the total number of successful and failure mission outcomes

[82]:
```
%sql SELECT COUNT(Mission_Outcome) AS "Mission Outcomes" FROM SPACEXTBL WHERE␣
↪Mission_Outcome LIKE 'Success%'
```

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

[82]: [(100,)]

[83]:
```
%sql SELECT COUNT(Mission_Outcome) AS "Mission Outcomes" FROM SPACEXTBL WHERE␣
↪Mission_Outcome LIKE 'Failure%'
```

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

[83]: [(1,)]

[87]:
```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end)␣
↪AS "Successful Mission", sum(case when MISSION_OUTCOME LIKE '%Failure%' then␣
↪1 else 0 end) AS "Failure Mission" FROM SPACEXTBL;
```

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

[87]: [(100, 1)]

0.3.8 Task 8

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

[89]:
```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Boosters With Maximum Payload Mass"␣
↪FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM␣
↪SPACEXTBL);
```

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

[89]:
```
[('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 ',)]
```

0.3.9 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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

[91]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE substr(DATE,7,4) =␣
↪'2015' AND LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such table: SPACEX
[SQL: SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE substr(DATE,7,4) =
'2015' AND LANDING__OUTCOME = 'Failure (drone ship)';]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

0.3.10 Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[95]:
```
%sql SELECT COUNT("Landing _Outcome") AS "Rank success count" FROM SPACEXTBL␣
↪WHERE "Landing _Outcome" LIKE '%Success%' AND DATE > '2010-06-04' AND DATE <␣
↪'2017-03-20' ;
```

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

[95]:
```
[(0,)]
```

0.3.11 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

0.4 Author(s)

Lakshmi Holla

0.5 Other Contributors

Rav Ahuja

0.6 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

##