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

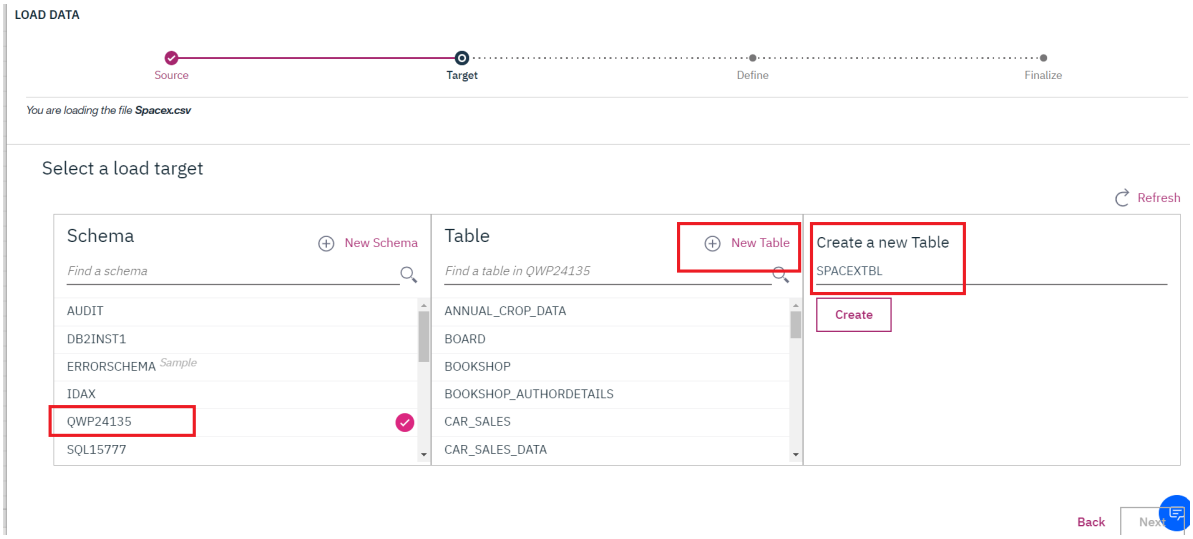
This assignment requires you to load the spacex dataset.

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

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:

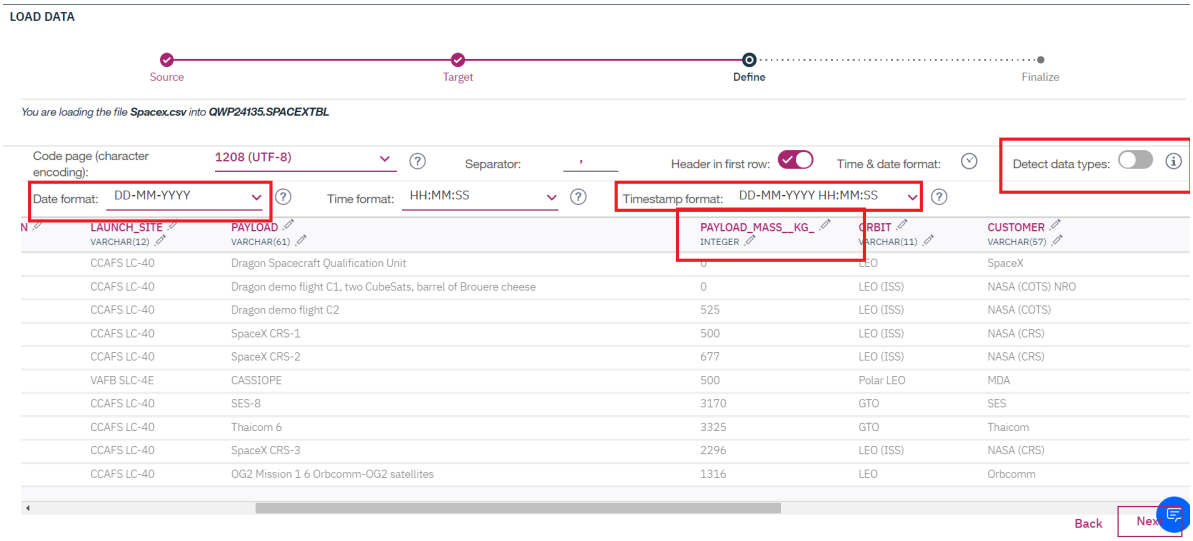
SPACEXDATASET

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).

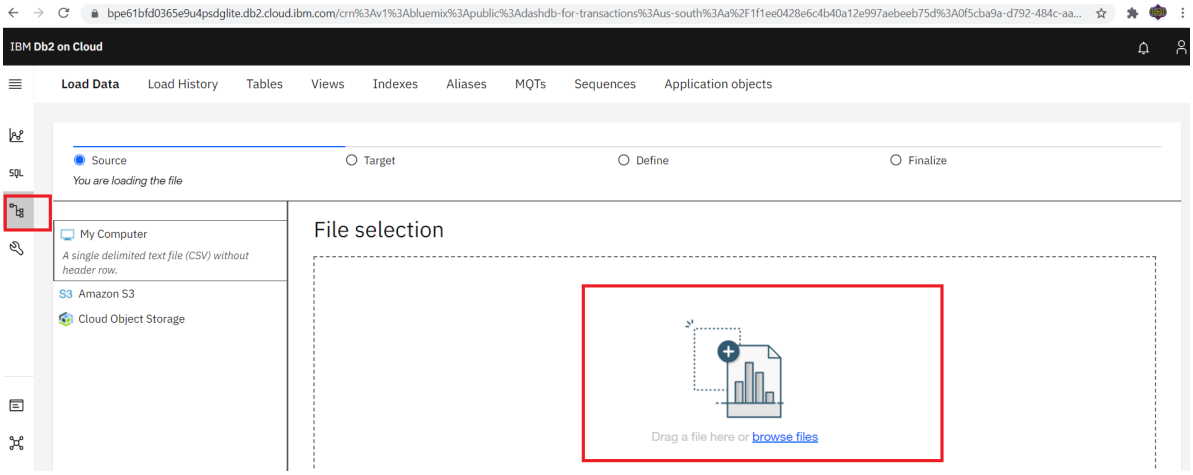
1. Change the Date Format by manually typing DD-MM-YYYY and timestamp format as DD-MM-YYYY HH:MM:SS

2. Change the PAYLOAD_MASS__KG_ datatype to INTEGER.

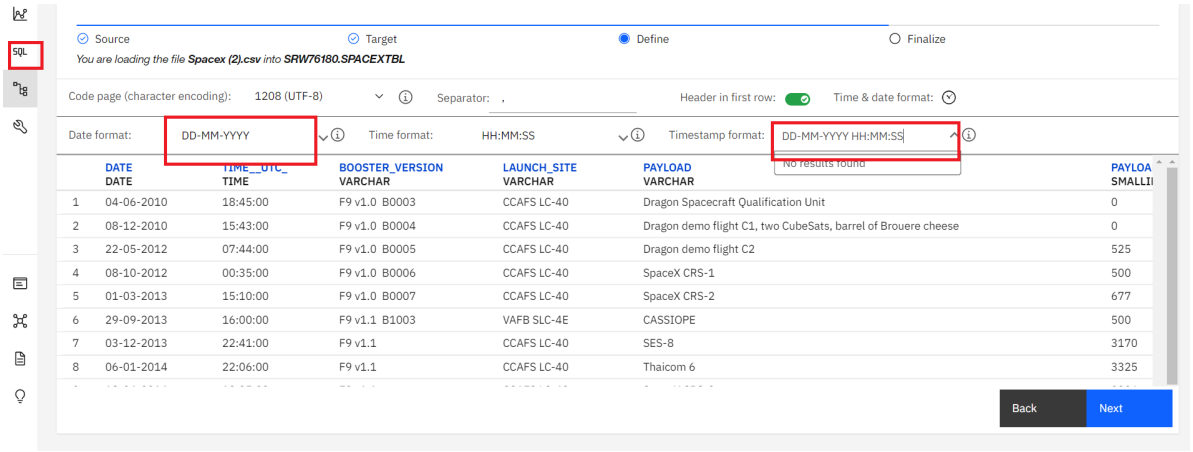


Changes to be considered when having DB2 instance with the new UI having Go to UI screen

- 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.



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 58.8 MB/s eta 0:00:0000:010
0:01
  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=1159121 sha256=c88607bb16645fc1a473f43031af72adab7f699fa124ea28489406f10313b2e5
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/03/71/13/010faf12246f72dc76
b4150e6e599d13a85b4435e06fb9e51f
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 [26]:
```
%load_ext sql
```

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

In [27]:
```
import csv, sqlite3

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

In [28]:
```
!pip install -q pandas==1.1.5
```

In [29]:
```
%sql sqlite:///my_data1.db
```

Out[29]:
```
'Connected: @my_data1.db'
```

In [30]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

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 [31]:
```
%sql SELECT Distinct LAUNCH_SITE from SPACEXTBL
```

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

Out[31]:

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 [32]:
```
%sql SELECT * FROM SPACEXTBL where LAUNCH_SITE like 'CCA%' LIMIT 5
```

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

Out[32]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	M
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	
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	
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	

Task 3

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

In [33]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Customer like 'NASA (CRS)%'
```

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

Out[33]:

SUM(PAYLOAD_MASS__KG_)
48213

Task 4

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

In [34]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%
```

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

Out[34]:

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

Task 5

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

Hint:Use min function

In [48]:
```
%sql select min(Date) from SPACEXTBL where LANDING__OUTCOME = 'Success (ground pad)'
#%sql SELECT min(date) from SPACEXTBL where landing__outcome = 'Success (ground pad)'
#cur.execute('''SELECT MIN(Date), "Landing _Outcome" FROM SPACEXTBL WHERE "Landing _Ou
#cur.fetchall()
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: select min(Date) from SPACEXTBL where LANDING__OUTCOME = 'Success (ground pa
d)']
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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 [37]:
```
%sql SELECT DISTINCT Booster_version FROM SPACEXTBL WHERE Mission_Outcome = 'Success'
```

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

Out[37]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Task 7

List the total number of successful and failure mission outcomes

In [50]:
```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) as Total FROM SPACEXTBL GROUP BY M
```

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

Out[50]:

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
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 [55]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLO
```

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

Out[55]:

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 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.

In [57]:
```
%sql SELECT BOOSTER_VERSION, DATE, LAUNCH_SITE, LANDING__OUTCOME FROM SPACEXTBL WHERE
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT BOOSTER_VERSION, DATE, LAUNCH_SITE, LANDING__OUTCOME FROM SPACEXTBL WHER
E LANDING_OUTCOME = "Failure (drone ship)" AND year = '2015']
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Task 10

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

In [66]:
```
%sql select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between
#%sql SLECT Landing__Outcome, count(Landing__Outcome) From SPACEXTBL Where Date Betwee
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing__Outcome
[SQL: select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE betwee
n '2010-06-04' and '2017-03-20') order by date desc]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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 |