Skills Network

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 [2]:
```
!pip install ipython-sql prettytable
```
```
Requirement already satisfied: ipython-sql in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (0.5.0)
Requirement already satisfied: prettytable in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (3.12.0)
Requirement already satisfied: ipython in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython-sql) (8.30.0)
Requirement already satisfied: sqlalchemy>=2.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython-sql) (2.0.36)
Requirement already satisfied: sqlparse in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython-sql) (0.5.2)
Requirement already satisfied: six in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython-sql) (1.16.0)
Requirement already satisfied: ipython-genutils in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: wcwidth in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from prettytable) (0.2.13)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (4.12.2)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from sqlalchemy>=2.0->ipython-sql) (3.1.1)
Requirement already satisfied: colorama in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (0.4.6)
Requirement already satisfied: decorator in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (0.19.2)
Requirement already satisfied: matplotlib-inline in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (0.1.7)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (3.0.48)
Requirement already satisfied: pygments>=2.4.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (2.18.0)
Requirement already satisfied: stack_data in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from ipython->ipython-sql) (5.14.3)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from jedi>=0.16->ipython->ipython-sql) (0.8.4)
Requirement already satisfied: executing>=1.2.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from stack_data->ipython->ipython-sql) (2.1.0)
Requirement already satisfied: asttokens>=2.1.0 in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from stack_data->ipython->ipython-sql) (3.0.0)
Requirement already satisfied: pure-eval in c:\users\kusum\appdata\local\programs\python\python312\lib\site-packages (from stack_data->ipython->ipython-sql) (0.2.3)
```

In [3]:
```
# Import necessary libraries

import sqlalchemy
import csv
import sqlite3
import pandas as pd
import prettytable
```

In [4]:
```
prettytable.DEFAULT = 'DEFAULT'
```

Connect to the database

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

In [5]:
```
# Load the SQL extension
%load_ext sql

# Establish a connection with your SQLite database
%sql sqlite:///my_data1.db
```

In [6]:
```
# Connect to the SQLite database
con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

In [7]:
```
# Load the dataset into a DataFrame
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")
```

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 not null
```
```
 * sqlite:///my_data1.db
Done.
```
Out[9]: []

In [12]:
```
%%sql
SELECT *
FROM SPACEXTBL
LIMIT 10;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[12]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-09-29	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500	Polar LEO	MDA	Success	Uncontrolled (ocean)
2013-12-03	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt
2014-01-06	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	3325	GTO	Thaicom	Success	No attempt
2014-04-18	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
2014-07-14	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316	LEO	Orbcomm	Success	Controlled (ocean)

In [24]:
```
print(df.shape)
```
```
(101, 10)
```

In [18]:
```
print(len(df))
```
```
101
```

In [15]:
```
%%sql
SELECT COUNT(*) AS row_count
FROM SPACEXTBL;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[15]:

row_count
101

In [16]:
```
df.dtypes
```
Out[16]:
```
Date                 object
Time (UTC)           object
Booster_Version      object
Launch_Site          object
Payload              object
PAYLOAD_MASS__KG_     int64
Orbit                object
Customer             object
Mission_Outcome      object
Landing_Outcome      object
dtype: object
```

In [17]:
```
print(len(df.columns))
```
```
10
```

In [14]:
```
%sql PRAGMA table_info(SPACEXTBL);
```
```
 * sqlite:///my_data1.db
Done.
```
Out[14]:

cid	name	type	notnull	dflt_value	pk
0	Date	TEXT	0	None	0
1	Time (UTC)	TEXT	0	None	0
2	Booster_Version	TEXT	0	None	0
3	Launch_Site	TEXT	0	None	0
4	Payload	TEXT	0	None	0
5	PAYLOAD_MASS__KG_	INTEGER	0	None	0
6	Orbit	TEXT	0	None	0
7	Customer	TEXT	0	None	0
8	Mission_Outcome	TEXT	0	None	0
9	Landing_Outcome	TEXT	0	None	0

The following columnsare :

cid: The column ID (an integer).

name: The name of the column.

type: The data type of the column.

notnull: Indicates whether the column can be NULL (0 means it can be NULL, 1 means it cannot be NULL).

dflt_value: The default value for the column, if any.

pk: Indicates whether the column is part of the primary key (0 means it is not part of the primary key, 1 means it is part of the primary key).

In [19]:
```
# Combine the count of NULL and non-NULL values in one DataFrame
null_counts = df.isnull().sum()
not_null_counts = df.notnull().sum()

combined_counts = pd.DataFrame({
    'null_count': null_counts,
    'not_null_count': not_null_counts
})

print(combined_counts)
```
```
                   null_count  not_null_count
Date                        0             101
Time (UTC)                  0             101
Booster_Version             0             101
Launch_Site                 0             101
Payload                     0             101
PAYLOAD_MASS__KG_           0             101
Orbit                       0             101
Customer                    0             101
Mission_Outcome             0             101
Landing_Outcome             0             101
```

In [21]:
```
print(df.nunique())
```
```
Date                 101
Time (UTC)            97
Booster_Version       97
Launch_Site            4
Payload              101
PAYLOAD_MASS__KG_     78
Orbit                  8
Customer              53
Mission_Outcome        4
Landing_Outcome       11
dtype: int64
```

In [20]:
```
print(df.describe())
```
```
       PAYLOAD_MASS__KG_
count         101.000000
mean         6138.287129
std          4900.998607
min             0.000000
25%          2500.000000
50%          4535.000000
75%          9600.000000
max         15600.000000
```

In [22]:
```
print(df.duplicated().sum())
```
```
0
```

In [25]:
```
print(df.isnull().any())
```
```
Date                 False
Time (UTC)           False
Booster_Version      False
Launch_Site          False
Payload              False
PAYLOAD_MASS__KG_    False
Orbit                False
Customer             False
Mission_Outcome      False
Landing_Outcome      False
dtype: bool
```

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 [28]:
```
%%sql
SELECT DISTINCT("Launch_Site")
FROM SPACEXTBL
```
```
 * sqlite:///my_data1.db
Done.
```
Out[28]:

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 [33]:
```
%%sql
SELECT "Launch_Site"
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[33]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Task 3

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

In [43]:
```
%%sql
SELECT "Customer", "Booster_Version", SUM("PAYLOAD_MASS__KG_") AS "Total_Palyoad_Mass_by_Booster"
FROM SPACEXTBL
WHERE "Customer" LIKE 'NASA (CRS)'
GROUP BY "Customer","Booster_Version"
ORDER BY "Total_Palyoad_Mass_by_Booster" DESC;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[43]:

Customer	Booster_Version	Total_Palyoad_Mass_by_Booster
NASA (CRS)	F9 B4 B1039.1	3310
NASA (CRS)	F9 FT B1021.1	3136
NASA (CRS)	F9 B5 B1058.4	2972
NASA (CRS)	F9 FT B1035.1	2708
NASA (CRS)	F9 B4 B1045.2	2697
NASA (CRS)	F9 B4 B1039.2	2647
NASA (CRS)	F9 B5B1050	2500
NASA (CRS)	F9 B5B1056.1	2495
NASA (CRS)	F9 FT B1031.1	2490
NASA (CRS)	F9 v1.1 B1012	2395
NASA (CRS)	F9 v1.1	2296
NASA (CRS)	F9 B5 B1056.2	2268
NASA (CRS)	F9 FT B1025.1	2257
NASA (CRS)	F9 v1.1 B1010	2216
NASA (CRS)	F9 FT B1035.2	2205
NASA (CRS)	F9 B5 B1059.2	1977
NASA (CRS)	F9 v1.1 B1018	1952
NASA (CRS)	F9 v1.1 B1015	1898
NASA (CRS)	F9 v1.0 B0007	677
NASA (CRS)	F9 v1.0 B0006	500

Task 4

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

%%sql SELECT COUNT("Booster_Version") FROM SPACEXTBL WHERE "Booster_Version" LIKE 'F9 v1.1';%%sql SELECT "Booster_Version", "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "Booster_Version" LIKE 'F9 v1.1' ORDER BY "PAYLOAD_MASS__KG_" DESC;

In [70]:
```
%%sql
SELECT "Booster_Version", AVG("PAYLOAD_MASS__KG_") AS Avergae_Payload_Mass
FROM SPACEXTBL
WHERE "Booster_Version" LIKE 'F9 v1.1';
```
```
 * sqlite:///my_data1.db
Done.
```
Out[70]:

Booster_Version	Avergae_Payload_Mass
F9 v1.1	2928.4

In [75]:
```
%%sql
SELECT "Booster_Version", ROUND(AVG("PAYLOAD_MASS__KG_"), 2) AS Avergae_Payload_Mass
FROM SPACEXTBL
WHERE "Booster_Version" LIKE 'F9 v1.1%';
```
```
 * sqlite:///my_data1.db
Done.
```
Out[75]:

Booster_Version	Avergae_Payload_Mass
F9 v1.1 B1003	2534.67

Task 5

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

Hint:Use min function

%%sql SELECT DISTINCT "Landing_Outcome" FROM SPACEXTBL;

In [85]:
```
%%sql
SELECT MIN("Date"), "Landing_Outcome", "Mission_Outcome"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)';
```
```
 * sqlite:///my_data1.db
Done.
```
Out[85]:

MIN("Date")	Landing_Outcome	Mission_Outcome
2015-12-22	Success (ground pad)	Success

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 [90]:
```
%%sql
SELECT "Booster_Version","Landing_Outcome", "Mission_Outcome", "PAYLOAD_MASS__KG_"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000
ORDER BY "PAYLOAD_MASS__KG_";
```
```
 * sqlite:///my_data1.db
Done.
```
Out[90]:

Booster_Version	Landing_Outcome	Mission_Outcome	PAYLOAD_MASS__KG_
F9 FT B1026	Success (drone ship)	Success	4600
F9 FT B1022	Success (drone ship)	Success	4696
F9 FT B1031.2	Success (drone ship)	Success	5200
F9 FT B1021.2	Success (drone ship)	Success	5300

Task 7

List the total number of successful and failure mission outcomes

In [111…
```
combined_counts = pd.DataFrame({
    "Mission_Outcome_Count": [df["Mission_Outcome"].count()],
    "Landing_Outcome_Count": [df["Landing_Outcome"].count()]
})

print(combined_counts)
```
```
   Mission_Outcome_Count  Landing_Outcome_Count
0                    101                    101
```

In [102…
```
%%sql
SELECT DISTINCT("Mission_Outcome") , "Landing_Outcome"
FROM SPACEXTBL
ORDER BY "PAYLOAD_MASS__KG_";
```
```
 * sqlite:///my_data1.db
Done.
```
Out[102…

Mission_Outcome	Landing_Outcome
Success	Failure (parachute)
Success	Uncontrolled (ocean)
Success	No attempt
Failure (in flight)	Precluded (drone ship)
Success	Success (ground pad)
Success	Controlled (ocean)
Success	Failure (drone ship)
Success	Failure
Success	Success (drone ship)
Success	No attempt
Success (payload status unclear)	Success (ground pad)
Success	No attempt
Success	Success

In [113…
```
%%sql
SELECT
    CASE
        WHEN "Mission_Outcome" LIKE 'Success%' THEN 'Success'
        ELSE 'Failure'
    END AS "Outcome",
    COUNT(*) AS "Count"
FROM SPACEXTBL
GROUP BY "Outcome"
ORDER BY "Count" DESC;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[113…

Outcome	Count
Success	100
Failure	1

In [107…
```
%%sql
SELECT
    CASE
        WHEN "Landing_Outcome" LIKE 'Success%' THEN 'Success'
        ELSE 'Failure'
    END AS "Outcome",
    COUNT(*) AS "Count"
FROM SPACEXTBL
GROUP BY "Outcome"
ORDER BY "Count" DESC;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[107…

Outcome	Count
Success	61
Failure	40

Task 8

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

In [122…
```
%%sql
SELECT "Booster_Version", "PAYLOAD_MASS__KG_"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" =
        (SELECT MAX("PAYLOAD_MASS__KG_")
        FROM SPACEXTBL)
ORDER BY "PAYLOAD_MASS__KG_";
```
```
 * sqlite:///my_data1.db
Done.
```
Out[122…

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

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 [131…
```
%%sql
SELECT SUBSTR(Date,0,5) AS "Year", SUBSTR(Date,6,2) as "Month", "Booster_Version", "Launch_Site","Landing_Outcome"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Failure (drone ship)' AND SUBSTR(Date,0,5) = '2015'
ORDER BY "Month";
```
```
 * sqlite:///my_data1.db
Done.
```
Out[131…

Year	Month	Booster_Version	Launch_Site	Landing_Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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 [142…
```
%%sql
SELECT SUBSTR(Date,0,5) AS "Year", SUBSTR(Date,6,2) as "Month", "Landing_Outcome", COUNT(*) AS "Success / Failure Count"
FROM SPACEXTBL
WHERE SUBSTR(Date,0,5) >= '2010' AND SUBSTR(Date,0,5) <= '2017'
GROUP BY "Landing_Outcome"
ORDER BY "Success / Failure Count" DESC;
```
```
 * sqlite:///my_data1.db
Done.
```
Out[142…

Year	Month	Landing_Outcome	Success / Failure Count
2016	04	Success (drone ship)	12
2012	05	No attempt	12
2015	12	Success (ground pad)	8
2015	01	Failure (drone ship)	5
2014	04	Controlled (ocean)	4
2013	09	Uncontrolled (ocean)	2
2010	06	Failure (parachute)	2
2015	06	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