

# Assignment: SQL Notebook

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 whereas 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.**

The screenshot shows the 'LOAD DATA' interface in IBM Data Studio. The 'Source' tab is active, indicating the file 'Spacex.csv' is being loaded. The 'Target' tab is also active, showing the schema 'QWP24135' and the table 'SPACEXTBL'. A red box highlights the 'New Table' button and the 'Create a new Table' option, which is set to 'SPACEXTBL'. Other tables listed include ANNUAL\_CROP\_DATA, BOARD, BOOKSHOP, BOOKSHOP\_AUTHORDETAILS, CAR\_SALES, and CAR\_SALES\_DATA. Navigation buttons for 'Back' and 'Next' are visible at the bottom right.

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.

Here you should place the cursor at Date field and manually type as DD-MM-YYYY.

2. Change the PAYLOAD\_MASS\_KG\_ datatype to INTEGER.

The screenshot shows the 'LOAD DATA' interface in IBM Data Studio. The 'Source' tab is active, indicating the file 'Spacex.csv' is being loaded into the table 'SPACEXTBL' in schema 'QWP24135'. The 'Target' tab shows the table structure with columns: LAUNCH\_SITE, PAYLOAD, PAYLOAD\_MASS\_KG\_, ORBIT, and CUSTOMER. A red box highlights the 'Date format' dropdown set to 'DD-MM-YYYY', the 'Timestamp format' dropdown set to 'DD-MM-YYYY HH:MM:SS', and the 'Detect data types' toggle switch which is turned off. The table data includes entries for various flights and their details. Navigation buttons for 'Back' and 'Next' are visible at the bottom right.

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

- Refer to this instruction 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.

The screenshot shows the IBM Db2 on Cloud interface with the 'Load Data' tab selected. On the left, there's a sidebar with icons for SQL, Tables, Views, Indexes, Aliases, MQTs, Sequences, and Application objects. The main area has tabs for Source (selected), Target, Define, and Finalize. Below the tabs, it says 'You are loading the file'. On the left, under 'Source', there are options for 'My Computer' (highlighted with a red box) and 'Amazon S3'. On the right, there's a 'File selection' section with a 'Drag a file here or browse files' button, also highlighted with a red box.

- Once done select the schema andload the file.

The screenshot shows the 'Load Data' configuration screen for the 'Spacex (2).csv' file. It includes fields for 'Code page (character encoding)', 'Separator', 'Header in first row', 'Date format' (set to 'DD-MM-YYYY'), 'Time format' (set to 'HH:MM:SS'), 'Timestamp format' (set to 'DD-MM-YYYY HH:MM:SS'), and a preview table with columns: DATE, TIME\_UTC\_TIME, BOOSTER\_VERSION, LAUNCH\_SITE, PAYLOAD. The preview table shows 8 rows of data from 2010 to 2014. At the bottom, there are 'Back' and 'Next' buttons.

```
In [1]: !pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
```

```
Collecting sqlalchemy==1.3.9
  Downloading SQLAlchemy-1.3.9.tar.gz (6.0 MB)
    |██████████| 6.0 MB 14.5 MB/s eta 0:00:01
Building wheels for collected packages: sqlalchemy
  Building wheel for sqlalchemy (setup.py) ... done
    Created wheel for sqlalchemy: filename=SQLAlchemy-1.3.9-cp38-cp38-linux_x86_64.whl
    size=1209506 sha256=8fcc47c0e919a9f01b0c21e23c63b72c6cb9554d48d4fd991e2a4bd3b799a182
    Stored in directory: /tmp/wsuser/.cache/pip/wheels/cb/43/46/fa638f2422554332b7865d
600275b24568bf60e76104a94bb4
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 1.4.22
    Uninstalling SQLAlchemy-1.4.22:
      Successfully uninstalled SQLAlchemy-1.4.22
Successfully installed sqlalchemy-1.3.9
Requirement already satisfied: ibm_db_sa in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (0.3.7)
Requirement already satisfied: ibm-db>=2.0.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_db_sa) (3.0.4)
Requirement already satisfied: sqlalchemy>=0.7.3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_db_sa) (1.3.9)
Collecting ipython-sql
  Downloading ipython_sql-0.4.0-py3-none-any.whl (19 kB)
Requirement already satisfied: six in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython-sql) (1.15.0)
Requirement already satisfied: sqlalchemy>=0.6.7 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython-sql) (1.3.9)
Requirement already satisfied: ipython-genutils>=0.1.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: ipython>=1.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython-sql) (7.27.0)
Collecting prettytable<1
  Downloading prettytable-0.7.2.zip (28 kB)
Collecting sqlparse
  Downloading sqlparse-0.4.2-py3-none-any.whl (42 kB)
    |██████████| 42 kB 2.6 MB/s eta 0:00:01
Requirement already satisfied: setuptools>=18.5 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (52.0.0.post20211006)
Requirement already satisfied: traitlets>=4.2 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (5.0.5)
Requirement already satisfied: backcall in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: decorator in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (5.0.9)
Requirement already satisfied: pickleshare in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (0.7.5)
Requirement already satisfied: pexpect>4.3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (4.8.0)
Requirement already satisfied: jedi>=0.16 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (0.17.2)
Requirement already satisfied: matplotlib-inline in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (0.1.2)
Requirement already satisfied: pygments in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (2.9.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!>=3.0.1,<3.1.0,>=2.0.0 in /opt/
```

```

conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql) (3.0.20)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from jedi>=0.16->ipython>=1.0->ipython-sql) (0.7.0)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from pexpect>4.3->ipython>=1.0->ipython-sql) (0.7.0)
Requirement already satisfied: wcwidth in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from prompt-toolkit!=3.0.0,!>=3.0.1,<3.1.0,>=2.0.0->ipython>=1.0->ipython-sql) (0.2.5)
Building wheels for collected packages: prettytable
  Building wheel for prettytable (setup.py) ... done
  Created wheel for prettytable: filename=prettytable-0.7.2-py3-none-any.whl size=13700 sha256=fa982d74cf33b1ed21936c1f2271c7b91cf6702b8a806112626d46e8142994d
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/48/6d/77/9517cb933af254f51a446f1a5ec9c2be3e45f17384940bce68
Successfully built prettytable
Installing collected packages: sqlparse, prettytable, ipython-sql
Successfully installed ipython-sql-0.4.0 prettytable-0.7.2 sqlparse-0.4.2

```

## Connect to the database

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

```
In [2]: %load_ext sql
```

### DB2 magic in case of old UI service credentials.

In the next cell enter your db2 connection string. Recall you created Service Credentials for your Db2 instance before. From the **uri** field of your Db2 service credentials copy everything after db2:// (except the double quote at the end) and paste it in the cell below after ibm\_db\_sa://

```

{
  "host": "dashdb-txn-sbox-yp-dal09-03.services.dal.bluemix.net",
  "jdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-dal09-03.services.dal.bluemix.net:50000/BLUDB",
  "uri": "db2://fbv67412:c...@dashdb-txn-sbox-yp-dal09-03.services.dal.bluemix.net:50000/BLUDB",
  "db": "BLUDB",
  "dsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-03.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCP"
}

```

in the following format

```
%sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name
```

### DB2 magic in case of new UI service credentials.

```

    "method": "direct",
    "password": "*****",
    "username": "qdg93144"
},
"certificate": {
    "certificate_base64": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURFakNDQWZxZ0F3SUJBZ0lKQVA1S0R3ZTNCTkxiTUEwR0NTcFFQkN3VUFNQjR4SERBYUjnT1YKQKFNTUUhbwENUUJEYkc5MViPDKvZWFJ0WW1GelpYTxDiagNOTwpBd01qSTVNRFF5TVRBeVdoY05NekF3TwpJMpNRFF5TVNUnd3R2dZRFZRUERQk5KUwsZ1EyhZkV1FnUkdGMF1xSmhjmM1z6Tu1JQk1qQU5C22txCmhraUc5dZBCQVFTRFBT0NBUThBTU1JQkNnSONBUUVBdXUvbItiNU8xSGpealpstK251yje4UKR4ZGwKtzRUL3FoUGMxMTRERY1FUk0p1RXdhG13aGljTGxa0nf2QWFmb1hrbmhqSVF0MG01L0x5YzBY291VNMsGR0QwpDVGrDMrTHM3d1dTakxqVE96N3M3M1zUSU5yYmx3cnIRUlVMIzWTkV6SkNHYw5LSDzMWZVSUtrc1dNm1R0SD15cnFsSGN0Z2pIu1fMrKvTRmlYaHj10dhsQmdoarpcCaTFBeEvadWNobWZ2QVRnEN03EKY210cHNrdBPTnI0YnhJMVRyUwxEemN1hMSFBzWw91UpzdnVzMUzaTeYsSmRN1MzK3labFZPMUznZku3bwPKMjlGOgtIU0NMSkJvTTFSZ3FPZG90Vm500c9E0WZhamNNN01wd2V4aa01sOTNKR1fJREFRQUJyMu13CLVU0WRCZ05WSFE0RUzUVV1Q3JzanFJ0zc1VUpxVmZEMDh:UmN3SHdZRFZSMGpCQmd3Rm9BVwVdc1kkanFJQzc1VUpxVmZEMDh1ZwdqeDZiumN3RldZRFZSMFRBUUgvQkFVdf0f3RUiVekF0QmdrcWhraUc5dzBcqVfZRpB0UkyRTBU0U13M1N3rjJ2MXBqaHV4M01kwW25GFVskRmb0tPd6hSRnfS0hgxZ2drCgVEcfBnMk55Ckx3R08yek85SwZUMmhLaWd1d2orwnJ5SGxxchIxQ0pLOVPekIyWmE251YzQTVscEttMwdjV3VHYzMkk1UrVTFzTdd1Ujd3ZFVuuj0UTVU4aERvNi9sVHMRMRVb2Mnc3V1NPs1FDK013ejgtTFJMdjVHSW5BN1JySwNhKw4ZEttd1pLYThWcnBnMXJ3QzRnY3d1YUhYMUNEWE42K0J1bzvhwG5Wkh6UG91cldYS1BoaGdXX2J5CkNdcUdIK0NwNnQ1eFg3b05Ns3VNSNqFZVZndnNLWnRNVZZbhQ0b13dTf1bGdZRDNjekltbj1lREQKNHB1REFvYTzyMktZZE4xVxuN3F3VG1tBd1TU05RPT0KLS0tLS1FTkQg00SVSE1GSUNBVEutLS0tLQo=",
    "name": "1cbbb1b6-3a1a-4d49-9262-3102a8f7a7c8"
},
"composed": [
    "*****"
],
"replicaSet": "*****"
3/bludb?authSource=admin&replicaSet=repiset"
],
"database": "bludb",
"host_ros": [
    "54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:30592"
],
"hosts": [
    {
        "hostname": "*****",
        "port": 32733
    }
]
}

```

- Use the following format.
- Add security=SSL at the end

**%sql ibm\_db\_sa://my-username:my-password@my-hostname:my-port/my-db-name?  
security=SSL**

In [4]: **%sql ibm\_db\_sa://jjk92789:mUgin2bu22IbDNHE@125f9f61-9715-46f9-9399-c8177b21803b.c1o**

## Tasks

Now write and execute SQL queries to solve the assignment tasks.

### Task 1

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

In [9]: **%sql**

```

select distinct Launch_Site from spacextbl

```

\* ibm\_db\_sa://jjk92789:\*\*\*@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:30426/bludb  
Done.

Out[9]: **launch\_site**

CCAFS LC-40

CCAFS SLC-40

KSC LC-39A

VAFB SLC-4E

## Task 2

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

In [10]: `%%sql`

```
select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;
```

```
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[10]:

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	cust
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	S
2010-08-12	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)	(
2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	



## Task 3

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

In [11]: `%%sql`

```
select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'
```

```
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[11]:

1

45596

## Task 4

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

In [15]: `%%sql`

```
select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';  
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0  
0.databases.appdomain.cloud:30426/bludb  
Done.
```

Out[15]: `1`

```
2928
```

## Task 5

List the date when the first successful landing outcome in ground pad was achieved.

*Hint: Use min function*

In [27]: `%%sql`

```
select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (grou  
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0  
0.databases.appdomain.cloud:30426/bludb  
Done.
```

Out[27]: `min_date`

```
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 [29]: `%%sql`

```
select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_M  
and (Landing__Outcome = 'Success (drone ship)');  
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0  
0.databases.appdomain.cloud:30426/bludb  
Done.
```

Out[29]: `booster_version`

```
F9 FT B1022
```

```
F9 FT B1026
```

```
F9 FT B1021.2
```

```
F9 FT B1031.2
```

## Task 7

List the total number of successful and failure mission outcomes

In [6]:

```
%%sql  
  
select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mi  
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0  
0.databases.appdomain.cloud:30426/bludb  
Done.
```

Out[6]:

mission_outcome	counts
Failure (in flight)	1
Success	99
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]:

```
%%sql  
  
select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ =  
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0  
0.databases.appdomain.cloud:30426/bludb  
Done.
```

Out[32]:

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 failed landing\_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [37]:

```
%%sql
```

```
select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing_
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[37]:

landing_outcome	booster_version	launch_site
-----------------	-----------------	-------------

Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
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 [39]:

```
%%sql
```

```
select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between
group by Landing__Outcome
order by count(*) desc;
```

```
* ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[39]:

landing_outcome	landingcounts
-----------------	---------------

No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Success (ground pad)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	1
Precluded (drone ship)	1