ETL Project 1: Exploring SpaceX’s API

2019 April 18

Emily Constantinidis

Hugo Azevedo

Artem Lobodanov





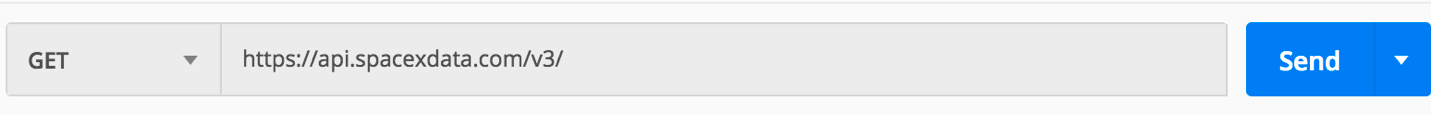
1. Methods

Using ETL processes, the following tasks were completed:

1. Application download of Postman
2. Extraction of rocket data from SpaceX’s open source API
3. Extraction of launch data from SpaceX’s open source API
4. Extraction of launch-pad data from SpaceX’s open source API
5. The creation of tables
6. The cleaning and transformation of the data
7. The load of all data-frames into a SQL database

Application Download of Postman

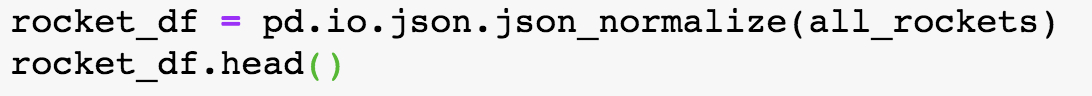
In order to gather data from SpaceX’s API, it was imperative to download a subsequent server/software, Postman. The postman app can be downloaded via their external/online website at <https://www.getpostman.com/downloads/> .

After completion of the postman application was it possible to request data from SpaceX’s open-source API. Through the application, one is instructed to “Enter request URL” and select the action type “GET” to extract any data from server to computer. The available data appears in JSON format on the external Postman Application. The following is the correct URL in order to extract all available API data from SpaceX:

Extraction of rocket data from SpaceX’s open source API

In order to view the rocket data in data frame format, Jupyter Notebook is used to import the dependences (panadas as pd, import requests, import json, from pprint import pprint). Then a GET request is performed and the API’s response is saved within a variable.



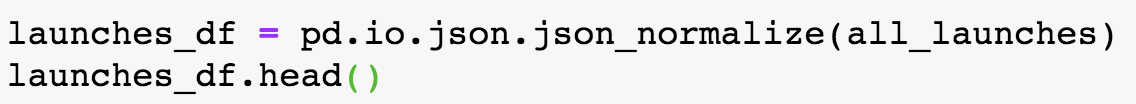
Then the JSON script is normalized and displayed in a data frame with the following code:

Extraction of Launch data from SpaceX’s open source API

In order to view the launch data in data frame format, a GET request is first performed and the API’s response is saved within a variable:



The JSON script is then normalized and displayed in a data frame with the following code:

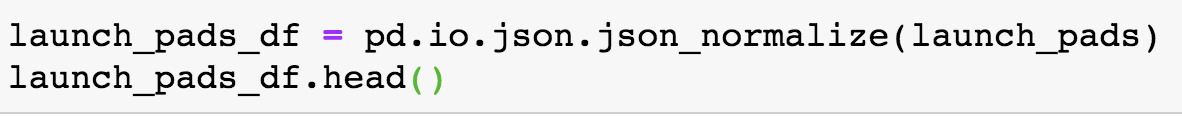


Extraction of Launch PAD data from SpaceX’s open source API

In order to view the launch pad data in a DATAFRAME format, a GET request is performed and the API’s response is saved within a variable.

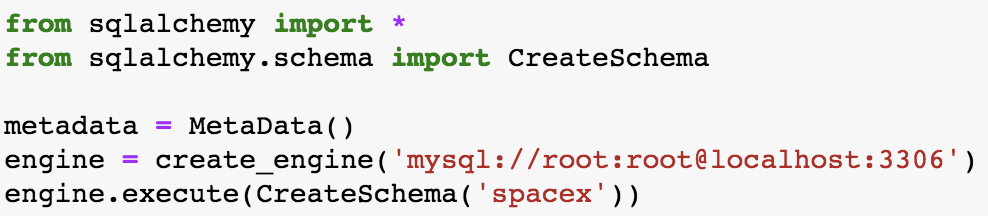


The JSON script is then normalized and displayed in a data frame with the following code:



Creation of TABLES and JOining of DATA

In a separate Jupyter Notebook file, “Create-tables.ipynb”, sqlalchemy is imported in order to describe the databases with metadata. The engine configuration is then used to talk to the database and the schema ‘spacex’ is created. The ‘spacex’ schema is created in order to access the schema “on the fly”.



Using the table construct/configuration, “Rockets” , “Launcher” and “Launchpads” tables are created.

“Rockets” and “Launcher” tables share the column “rocket\_id” and the “Launcher” and “Launchpads” tables share the column “launch\_pads\_id”. These columns will be used to join the three tables in order to further transform our data.

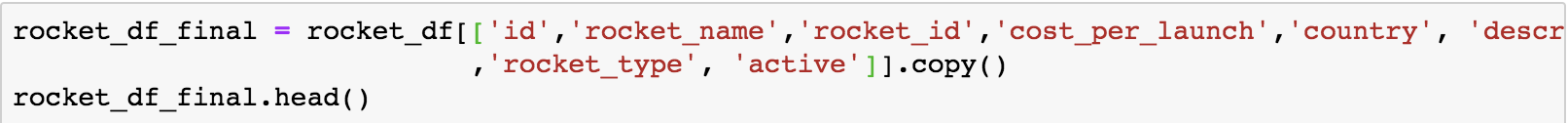
The purpose of this section of the project is to create 5 clean tables that are interlinked and loaded onto MySQL. The 3 tables are: rocket\_data, launch\_data, launchpad\_data.

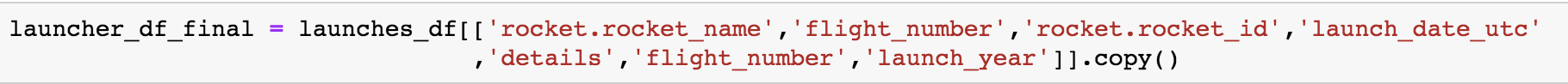
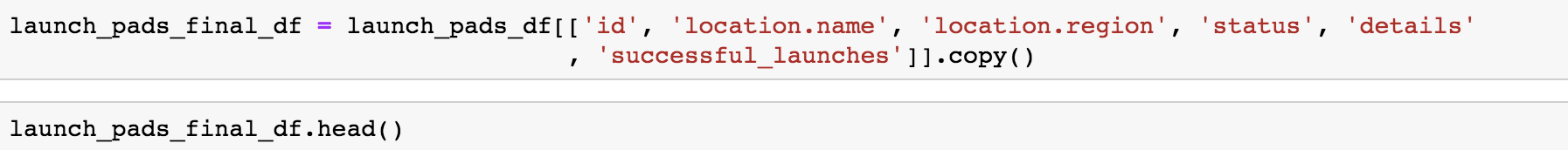
A database is then created to SQL.



Cleaning and Transformation of DATA

In order to condense the data and increase its readability, the removal of over 40\* columns that were not immediately relevant (such as, “site\_name\_long”, Wikipedia, “region”, “latitude”, etc.) was imperative. This was done by selecting/naming the desired columns to display.

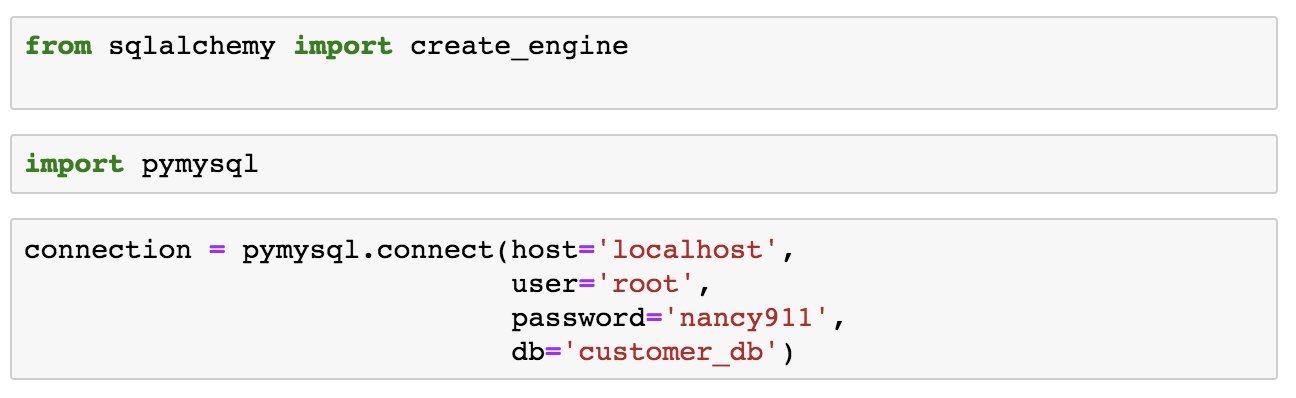
The first table to be transformed is ‘Rockets’. The desired columns from the table are selected to be viewed in the pandas data frame, including, “id, “rocket\_name”, 'id', 'rocket\_name', 'rocket\_id', 'cost\_per\_launch', 'country', 'description', 'rocket\_type' and 'active'. This is done through a subset selection (using square brackets).

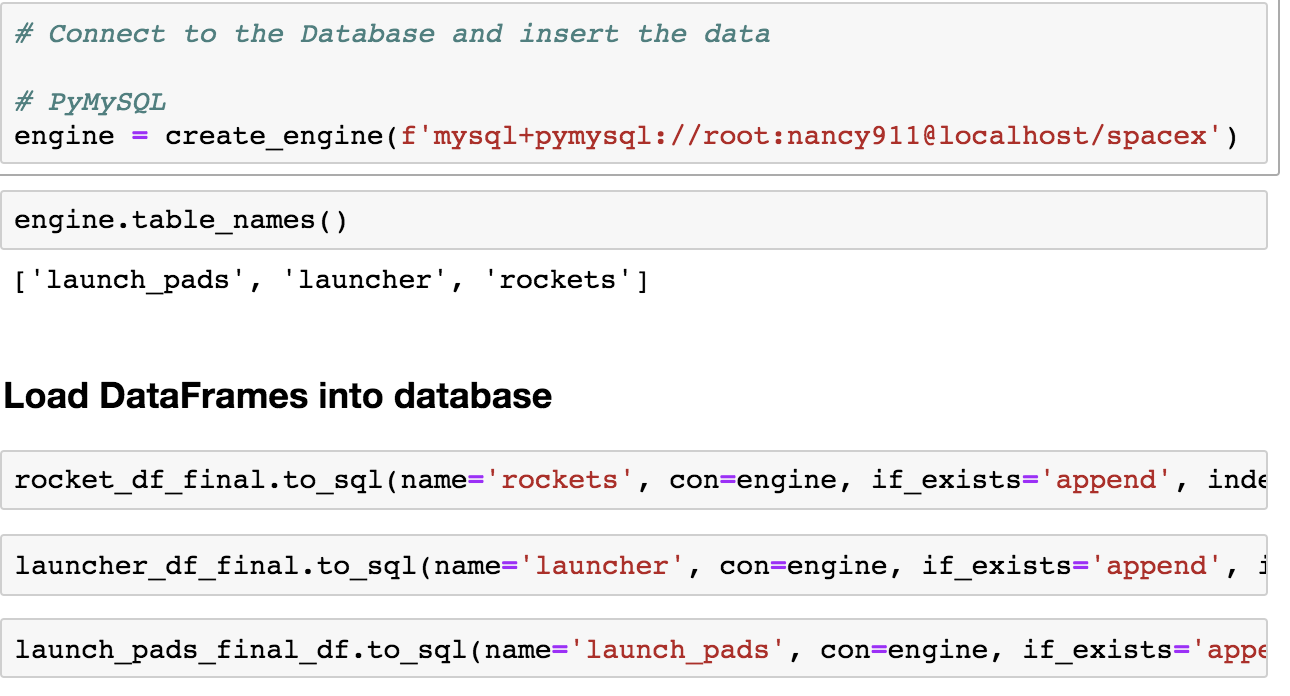
The same steps are repeated for the ‘Launches’ and ‘Launch Pads” tables with respective desired columns.

The columns are then renamed:

Load of all data-Frames into a SQL Database

In order to load the data into a SQL database, we must ensure we are connected to the database using the imported ‘pymysql’ Python database connector so that the program may be able to talk to a SQL server from Jupyter Notebook.

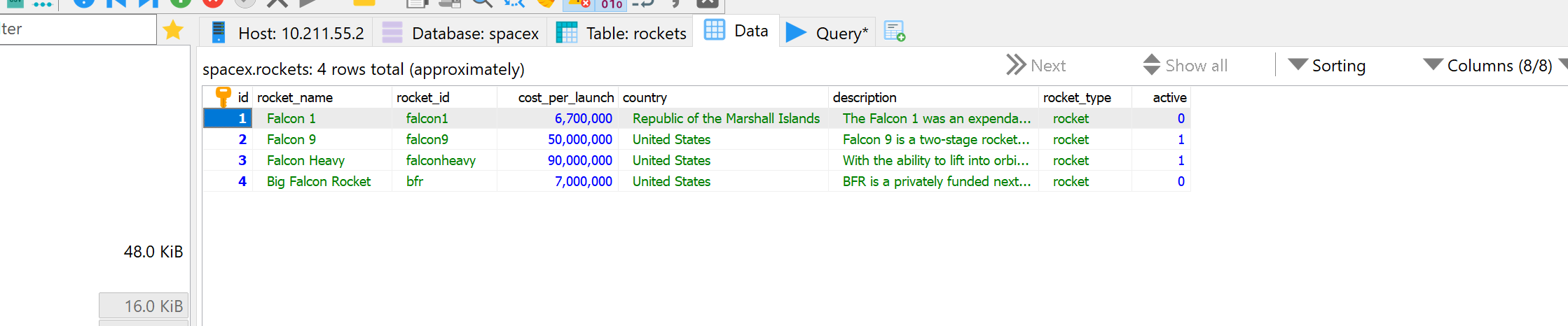
The tables are then loaded to sql using “.to\_sql” which requires specification of the Name of the SQL table, the connection through the engine, how to behave if the table already exists,

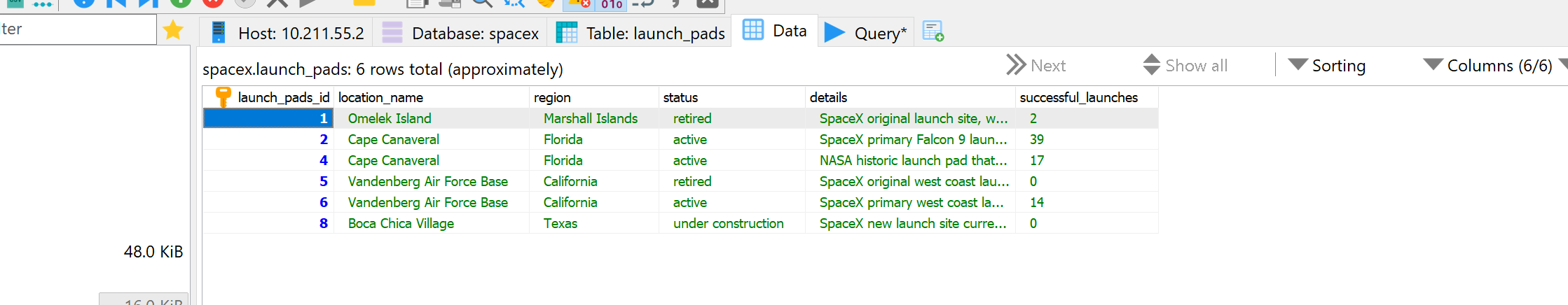
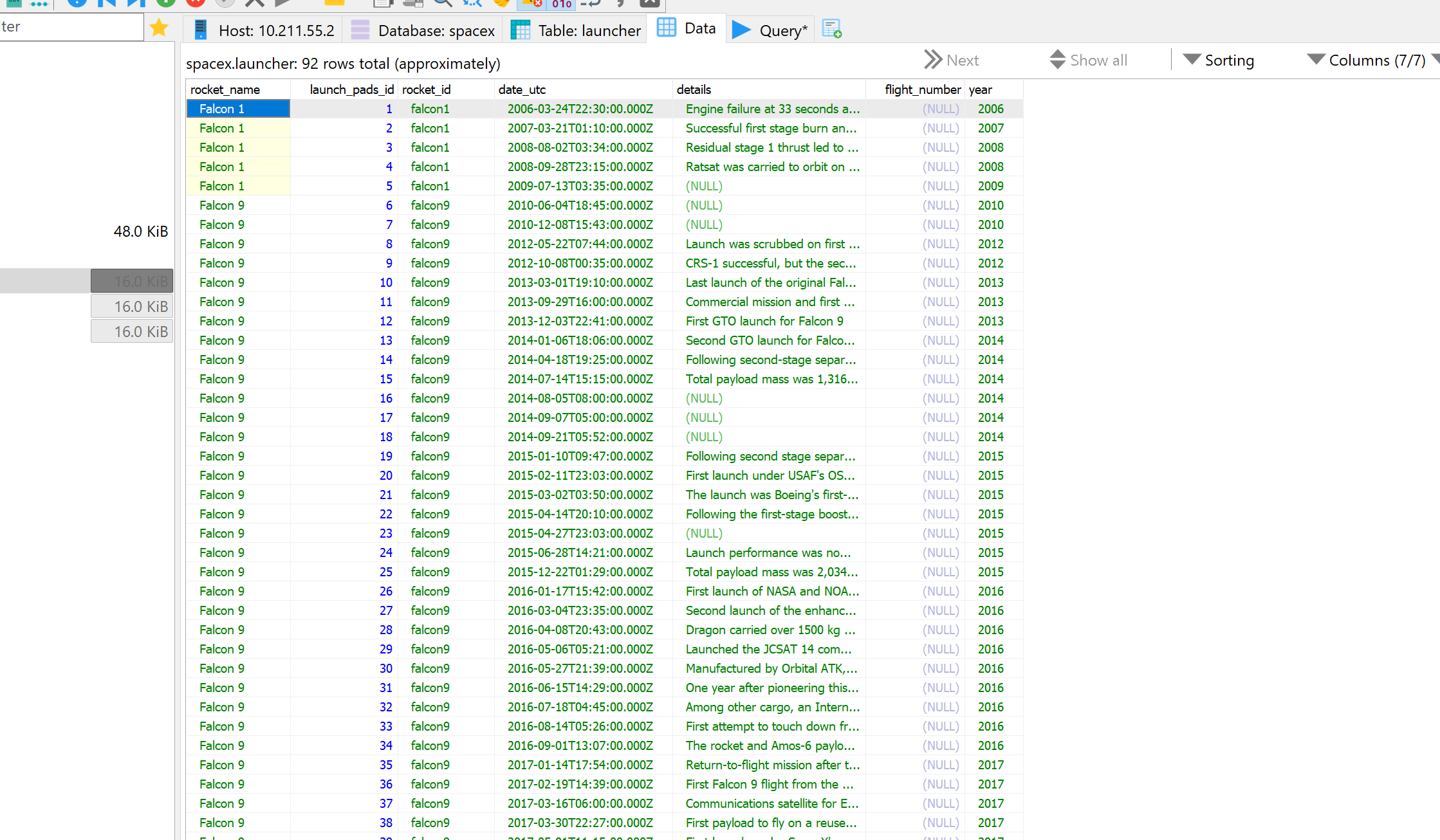
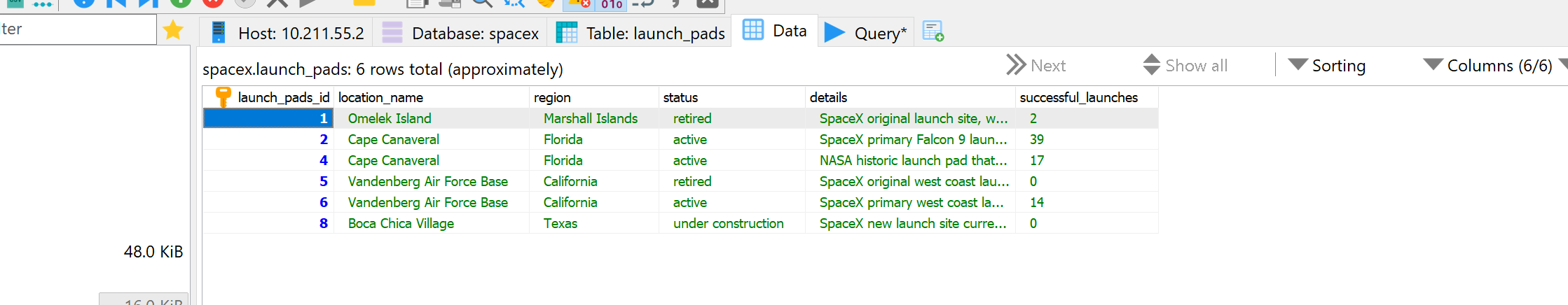


The tables are then joined in SQL:

Both the “Rockets” and “Launcher” table are joined on the column of “rocket\_id” or “id”. The “Launchpads” table is then joined on “launch\_pads\_id”.

FINAL DATABASE

The figure below shows the final database loaded into MySQL



Possible Relationships to Explore

Upon conducting a further analysis, data relationships that could have been explored are:

1. Correlation of cost of rocket with rate of mission/phase success
2. Launch date with rate of success
3. Phase of mission with rate of success
4. Launch pad with rate of success

Visualization of these relationships would aim to establish trends in SpaceX’s operations to show which factors lead to higher rates of success for various rocket missions and launches.