RUT-SOM-DATA-PT-06-2020-U-C Douglas High

ETL Project August 31, 2020

**Summary:** Download a csv file of exoplanet information, extract certain parts, manipulate some of the data, find extra data on other web sites and through manual creation, create postgres db and tables, load tables.

**Programs**: **main.ipynb** – all automated processing done here.

**Web pages**

<https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PSCompPars> – composite table of all known exoplanets. Downloaded csv file as main data source for program.

<https://exoplanetarchive.ipac.caltech.edu/docs/API_PSCompPars_columns.html> - descriptive data for column names used in composite table. Scraped data to add to my column documentation table.

<https://maps.googleapis.com/maps/api/geocode/json> - base url used to get api’s of location information for my discovery facilities table.

**Files**

**data\_in/planet\_composite\_data.csv** – csv extract from exoplanet website.

**data\_in/column\_descriptions.xlsx** - manually extracted column names and basic information from top of composite data csv.

**data\_in/new\_column\_names.xlsx** - manually created list of column names to match names in db.

**data\_in/new\_columns.xlsx** - manually created rows for documentation table of new columns created in this program.

**data\_in/glossary.xlsx** – manually created table of key terms, directly loaded to db table.

**data\_in/table\_definitions.sql** – sql code to create db tables.

**data\_out/column documentation.csv** -csv of column documentation table.

**data\_out/planets.csv** – csv of planets table.

**data\_out/stars.csv** – csv of stars table.

**data\_out/facilities.csv** – csv of facilities table (discovery\_df).

**data\_out/glossary.csv** – does not exist, data was not changed from *data\_in/glossary.xlsx*

**Images/planets\_db\_ERD** – entity relationship diagram of planetary database.

**Database:** postgresql://postgres:@localhost:5432/**planetary** – manually created using pgAdmin.

**Tables:**  created via running sql that was mostly generated using quickDBD. Loaded from dataframes via python.

**planets** - information about known exoplanets.

**stars** – information about the orbited stars/solar systems of known exoplanets.

**Facilities** – information about facilities that discovered known exoplanets.

**Column\_document** – information on columns used in above three tables.

**glossary** – information on key terms.

**Details of process**

1) found dataset of known exoplanets at <https://data.world/markmarkoh/kepler-confirmed-planets/workspace/file?filename=planets.csv>, traced it back to Nasa exoplanet website.

2) downloaded csv of exoplanetary information from <https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PSCompPars>.

3) manually extracted some of the column names from the top of the downloaded csv file and put them in an xl file. This is the start of my column documentation table.

4) start of processing within main.ipynb - created main\_df from csv file, skipping non table data at top of dataset and dropping unwanted columns.

5) created column documentation df to be used to load db table ‘column\_document’.

* took column xl file (from 3 above) and merged it with a table created by scraping the following website <https://exoplanetarchive.ipac.caltech.edu/docs/API_PSCompPars_columns.html>.
* manually created xl sheet of column descriptions for new columns I created.
* appended above xl file to merged df.

6) created planet df to be used to load db table ‘planets’.

* took columns from main\_df related to planets.
* modified various columns.
* transformed data and created new columns.
* re-arranged order of columns

7) created discovery facility df to load db table ‘facilities’.

* took columns from main\_df related to discovery facilities.
* dropped duplicates, keeping each facility for a key value.
* called google maps to get location information via api.
* added new columns with location and geo coordinates.
* added code to prevent erroneous output data (addresses for space bound observatories, and a few facility names that pulled up google locations for non-related information).

8) created star df to load db table ‘stars’.

* took columns from main\_df related to stars and star system.
* dropped duplicates, keeping each star as a key value.
* Renamed and re-ordered columns.

9) manually created glossary xl sheet with key terms related to information on tables, read into dataframe to load onto db table ‘glossary’.

10) using df.to\_sql, load five tables to postgres without index column, replacing if they exist.

**How to re-create**

1. Go to <https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PSCompPars>.
2. Click on ‘download table’; select ‘CSV Format’, ‘Download All Columns’, and ‘Download All Rows’.
3. Copy the downloaded file into data\_in/planet\_composite\_data.csv.
4. Create postgres db called ‘planetary’.
5. Within ‘planetary’ db, open and run data\_in/table\_definitions.sql as query.
6. Create config.\* (I used config.py) with the following two records, with your passwords for postgres and google api

gkey = "*google api key*"

password = "*postgres password*"

1. Go into the main.ipynb program, top of section ’02 Column Description Table’ and change the path for chromedriver.exe ( go to <https://chromedriver.chromium.org/downloads> if you do not have installed). In lieu of chromedriver you may substitute any web browser that splinter allows ( <https://splinter.readthedocs.io/en/latest/browser.html> ), just modify the first two lines of section 02 as needed.
2. Run main.ipynb.
3. There does exist the possibility that the data downloaded in step 1 could change or that the website scraped in main.ipynb section 02 could change, or that the results brought back from goggle maps in main.ipynb section 07 could change. In which case modifications to main.ipynb would need to be made.

**Program Details**

**01 Main Slicing:**  drop unwanted columns from composite data.

In: data\_in/planet\_composite\_data.csv - downloaded csv file of planetary composite data.

Out: main\_df with selected columns.

**02 Column Description Table:**  scrape and transform column documentation data.

In: <https://exoplanetarchive.ipac.caltech.edu/docs/API_PSCompPars_columns.html>

data\_in/column\_descriptions.xlsx - manually extracted column names and basic information from top of composite data csv.

Out: column\_documentation\_df – merged dataframe , added description column from web scrape.

**03 Column Documentation:** create column documentation df to load into db table .

In: data\_in/new\_column\_names.xlsx – manually created list of column names used in db.

data\_in/new\_columns.xlsx – manually created documentation table of new columns created in this program.

Out: col\_doc\_df – column documentation df to load into db table.

data\_out/column\_documentation.csv – csv of col\_doc\_df

**04 Table Split-Planet:** extract columns from main\_df to create planet\_df, transform and re-order.

In: main\_df – output from section 01 Main Slicing.

Out: planet\_df – planet df to load into db table.

data\_out/planets.csv – csv of planet\_df.

**05 Table Split-Discovery Facility:** extract columns from main\_df to create discovery\_df, drop duplicates, rename columns, call google maps api to get location information, add as new columns.

In: main\_df – output from section 01 Main Slicing.

<https://maps.googleapis.com/maps/api/geocode/json> - base url for calls.

Out: discovery\_df – discovery facility df to load into db table.

data\_out/facilities.csv - csv of discovery\_df.

**06 Table Split-Stars:** extract columns from main\_df to create star\_df, drop duplicates, rename columns.

In: main\_df – output from section 01 Main Slicing.

Out: star\_df – star df to load into db table.

data\_out/stars.csv – csv of star\_df.

**07 Load Postgres:** read glossary xl file, load five db tables from df’s.

In: data\_in/glossary.xlsx -manually created glossary of key terms 🡪 glossary.df**.**

col\_doc\_df, star\_df, planet\_df, discovery\_df, (glossary\_df).

out: postgres ‘planetary’ db tables: column\_document, stars, planets, facilities, glossary.

**08.n Verify Table Loads**: five sections, to verify data was loaded to postgres tables.

**Further Considerations**

1. Star table, ‘type’ column contains 3 separate codes (letter, letter/number, and Roman numeral) which describe characteristics about the star. More tables could be added with this information relating back to the star.

Need to look into possibly redefining ‘type’ field into multiple fields or since the data stored in this column can sometimes have one, two, or three of these values and also can contain duplicates of the same value type (i.e. “M5.5/M6”), the table data would need to be programmatically parsed into two or three separate columns, possibly containing a list.

Three likely websites to scrape this information from…

<http://www.star.ucl.ac.uk/~pac/spectral_classification.html>

<https://sites.uni.edu/morgans/astro/course/Notes/section2/spectraltemps.html>

<https://www.enchantedlearning.com/subjects/astronomy/stars/startypes.shtml#:~:text=Stars%20are%20classified%20by%20their,stars%20are%20common%20but%20dim>..

1. With regard to the above, many ‘type’ cells are empty. It may be possible to extrapolate values for these based on other characteristics of the star already included in the table. Research on how stars are classified as ‘types’ is needed.
2. Further information regarding the planets could be looked for, such as type (i.e. rocky, gaseous). Also, maybe calculate length of time for planetary rotation (day) and then calculate number of days in a year since orbital time is known. Current orbital period is expressed in earth days.
3. Location look up for discovery facilities could be refined, could possibly even try to get a blurb of information about the facility or the types of instruments used. This could even branch off into a separate table of instrument types.
4. Expand the glossary.
5. Data\_out folder contains only csv files of dataframes used to load db tables. These files were useful during testing, but seemingly redundant at this point. Should delete data\_out folder and rename data\_in as just data. Would need to modify documentation and main.ipynb to reflect.