Stephen Anderson

Jacquelyn Cleary

Megan Dodson

**Wk. 13 ETL Project Report**

Project Concept: Comparing salary of Data Analyst and Data Scientist jobs to cost of living for job postings in the US.

Extract:

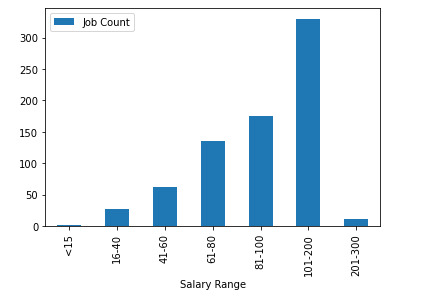
Both of our original data sources were in CSV format and found on Kaggle. The Data Scientist job postings dataset and the cost-of-living dataset can be found respectively at:

* <https://www.kaggle.com/milan400/glassdoordatascientist>
* <https://www.kaggle.com/debdutta/cost-of-living-index-by-country>

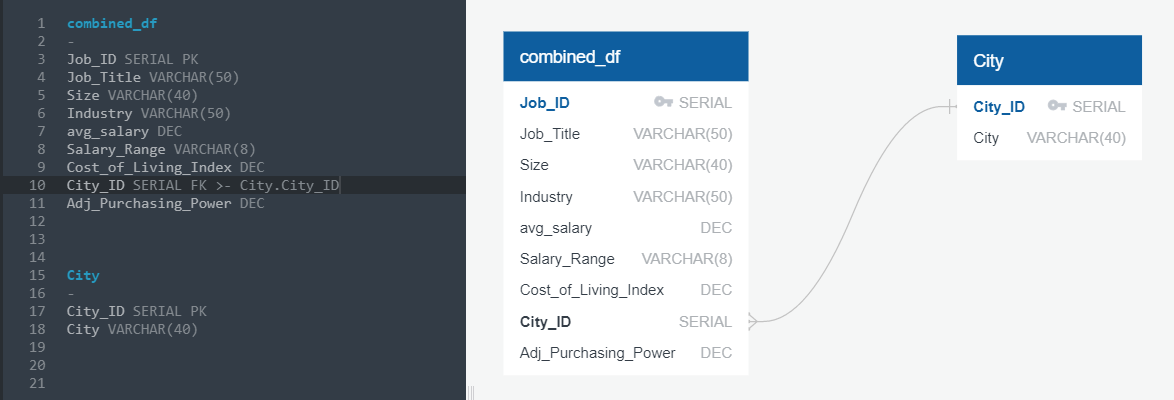
Transform:

The steps we took to clean our data:

* Read CSVs in using pandas
* Reviewed datasets for completeness, N/As, formatting
* Used str.split to clean up our City column that originally came over as City, Country and City, State, Country
* Reviewed job titles for uniqueness to determine if it made sense to combine jobs based on title
  + Determined no because of number of differences between jobs
* Dropped columns that did not add value and were string that would have slowed the SQL query down (ex. Job Descriptions, Company Name)
* Used .describe to understand the spread and nature of salaries in job postings
* Determined multiple entries in data set where repeats however we decided to keep them in as we read it as 2 job postings for 2 open positions instead of a duplicated job posting
* Binned the salaries to see how many jobs fell into each bucket
* Plotted a bar chart to get a visual for the spread of salaries in the job postings



* Created an index for our cities so we could create a child table of city
  + This table would just house City and City ID for SQL efficiency and keeping data integrity
  + This process took a few trials but we finally got it, if we would have had more time we would have done the same thing with Industry, Company Size, Salary Ranges
* Inner merge on Left to be able to add city index to combined\_df to ensure parent child would work
* Created combined\_df the parent, and the City\_df as a template to start optimizing query efficiency of the table
* Used pd.isna to ensure there were no blanks on Cost of Living
* calculated new column Adj\_Purchasing\_Power to showcase salary vs. cost of living relationship for job in that city
* Used Quick DB to create schema and connect the PK and FK (see schema.sql in github)



* Renamed columns to match schema for transfer from pandas to PostgreSQL

Load:

* We loaded the combined\_df and city into PostgreSQL using sqlalchemy
* We used a parent child relational database
* If we had had more time, we would have created more children for data integrity and SQL efficiency