Team:Cow System

Name(ID):
Ka Nam Canaan Law (015275299)
Cheuk Hong Ip(015265588)
Shahraiz Qureshi (009551906)

Cow System Project Report

Introduction

The analytics field is on the rise worldwide. In this project, we take a closer look at the US job market about DA jobs. This analysis helps understand different features like which keywords are used in job titles for data related roles and which industry posts the most data jobs, etc. The Datasets from sources were used and the ETL process along with other software was used to analyze and visualize data. The goal of the project was to take data from multiple sources and use different data related software to analyze data and present the finding. We found different sources of data on job opening positions for data analysts,data engineers, and data scientists. The datasets were gathered from different sources such as Kaggle and glassdoor. That data was cleaned and refined using jupyter and python using pandas and numpy. Then the new tables were taken and uploaded to AWS S3 buckets. That data was bulk loaded into MYSQL to run queries. Then tableau was used to visually present the data. There were different datasets containing more than 2000 job listings for data engineer,scientist, analyst and positions, with features such as Salary Estimate, Location, Company Rating (From Glassdoor), Job Description and more. The company rating is based on the employees who work in the company while glassdoor will calculate the ratings from their algorithm. The queries were used to answer important questions like What is the average estimated salary for a data related job in each city around the Bay Area? How many data related jobs are in each state, and other information the analysis could extract.

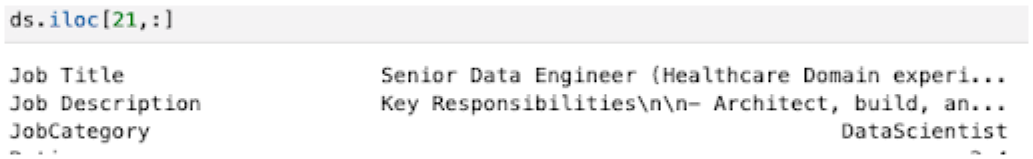
Data-analyst-vs-data-engineer-vs-data-scientist

This article discusses the key difference and similarities between the three data analysts,engineer, and scientist. The article starts out with talking about data analysts mentioning that most entry level professionals start off with the position as Data analysts. The position mostly requires a bachelors and a good understanding of statistical analysis. The primary expectation is that the person understands data handling, modeling, and other reporting techniques along with good understanding of the business. Then there is the data engineer position which requires either a masters in the data-field or plenty of experience as an data analyst. They need to be able to create and integrate API's, understand data pipelines, and lastly continuous performance optimization. Lastly, data scientist this role usually fits a person if they have acquired ample amount of experience in various data science skill sets, which include but are not limited to advanced statistical analysis, complete understanding of machine learning and many others. They basically analyse and interpret complex data. The goal of reading this article was to determine if we take the collected data sets and analyze them will they relate to this article. After analyzing the results it was determined this article was correct. The group did an Analysis on job description keyword searching from each job category determining data analyst need to know data warehousing, some sort of progrogramming primarily python. They need to know SQL data reporting and visualization. Secondly data engineers need to know ETL, Advance programing, data architecture, machine learning knowledge or concepts. Lastly, data scientists need to know advanced statistical analysis, data mining, in-depth programming, data optimization and other decision making skills. In conclusion, what the article stated closely resembled the group's findings.

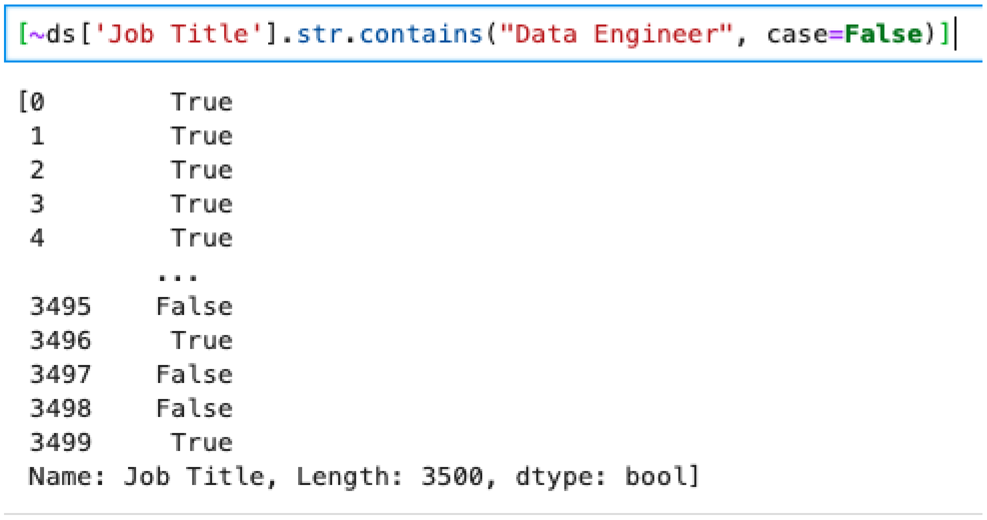
Python for ETL

For the project, we were using python pandas to do ETL, extracted some useful information, cleaned the data, reformatted data values and loaded into a couple databases that we work on by using three datasets which are originally from Glassdoor in kaggle. We mainly focused on jobs that are related to the data field such as data analysis, data engineer and data science. For the ETL, we mainly focused on using python libraries such as pandas and numpy. For extracting some useful information, cleaning the data and formatting data values, we changed the data values by editing or dropping some of the data and splitting useful data values into a new column.

First, we found an issue that the data values were not consistent in some datasets and some of the job titles did not match the job category from python. Therefore, by using the pandas function of str.contains, we can confirm if the text of data engineer appeared in the column of "Job Title". For example, the job title was senior data engineer but it actually appeared in the data science dataset with the job category of data science.

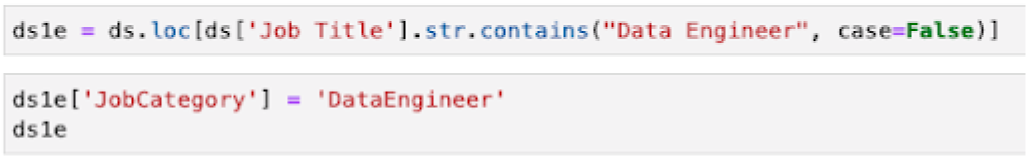


```
ds.iloc[21,:]

Job Title                  Senior Data Engineer (Healthcare Domain experi...
Job Description            Key Responsibilities\n\n- Architect, build, an...
JobCategory                                                    DataScientist
```



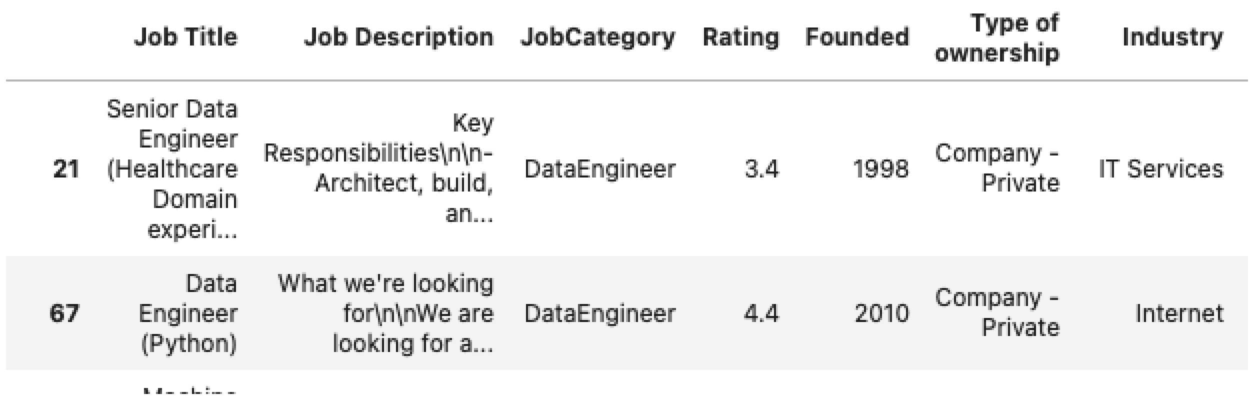
```
[~ds['Job Title'].str.contains("Data Engineer", case=False)]

[0        True
 1        True
 2        True
 3        True
 4        True
        ...
 3495    False
 3496     True
 3497    False
 3498    False
 3499     True
 Name: Job Title, Length: 3500, dtype: bool]
```

In order to solve this issue, we extracted those mismatched rows and insert it into a new data frame, then we updated the job category as data engineer from data science.

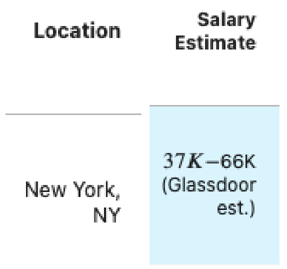


```
ds1e = ds.loc[ds['Job Title'].str.contains("Data Engineer", case=False)]
```

```
ds1e['JobCategory'] = 'DataEngineer'
ds1e
```

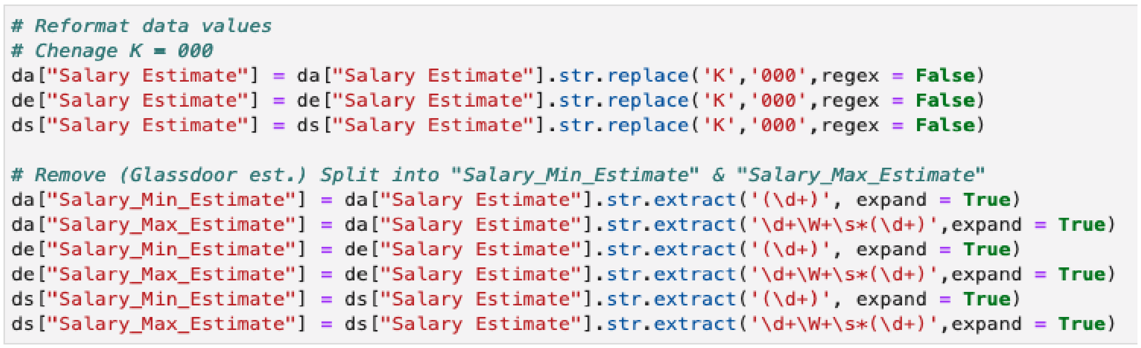


	Job Title	Job Description	JobCategory	Rating	Founded	Type of ownership	Industry
21	Senior Data Engineer (Healthcare Domain experi...	Key Responsibilities\n\n- Architect, build, an...	DataEngineer	3.4	1998	Company - Private	IT Services
67	Data Engineer (Python)	What we're looking for\n\nWe are looking for a...	DataEngineer	4.4	2010	Company - Private	Internet

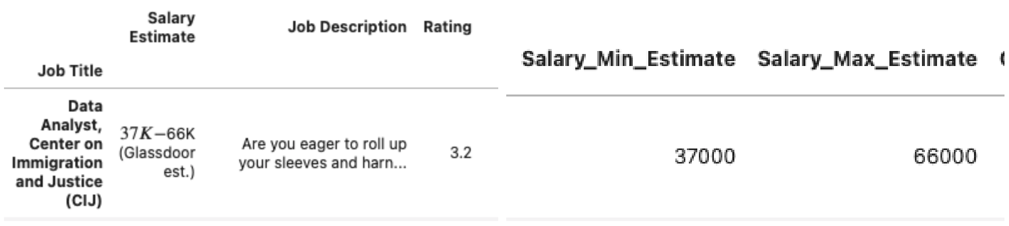


Location	Salary Estimate
New York, NY	$37K-66$K (Glassdoor est.)

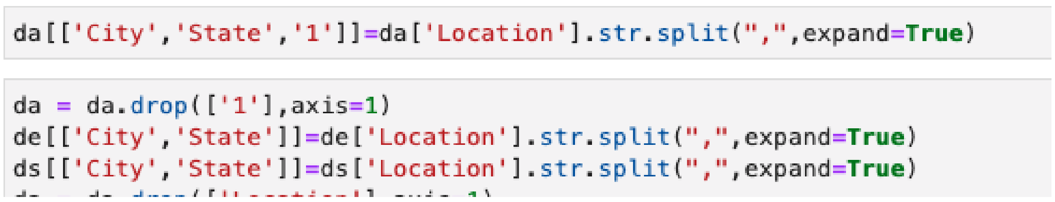
Second, from above graphs, we did not want a column of salary estimates with data values 37K-66K(Glassdoor est.) that was not able to compute and we also wanted to separate City and State to make a data becoming useful information for querying in the database system and analyze the data by using some useful coding such as regular expression, df.str.replace , df.str.extract, and df.str.split. We wanted to replace ‘K’ to ‘000’ and extracted the estimated salary into two columns, and we wanted to split location with the data values such as “New York, NY” into two columns named City and State. Here is the example after the formatting and cleaning with the codings.



```
# Reformat data values
# Chenage K = 000
da["Salary Estimate"] = da["Salary Estimate"].str.replace('K','000',regex = False)
de["Salary Estimate"] = de["Salary Estimate"].str.replace('K','000',regex = False)
ds["Salary Estimate"] = ds["Salary Estimate"].str.replace('K','000',regex = False)

# Remove (Glassdoor est.) Split into "Salary_Min_Estimate" & "Salary_Max_Estimate"
da["Salary_Min_Estimate"] = da["Salary Estimate"].str.extract('(\d+)', expand = True)
da["Salary_Max_Estimate"] = da["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)
de["Salary_Min_Estimate"] = de["Salary Estimate"].str.extract('(\d+)', expand = True)
de["Salary_Max_Estimate"] = de["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)
ds["Salary_Min_Estimate"] = ds["Salary Estimate"].str.extract('(\d+)', expand = True)
ds["Salary_Max_Estimate"] = ds["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)
```

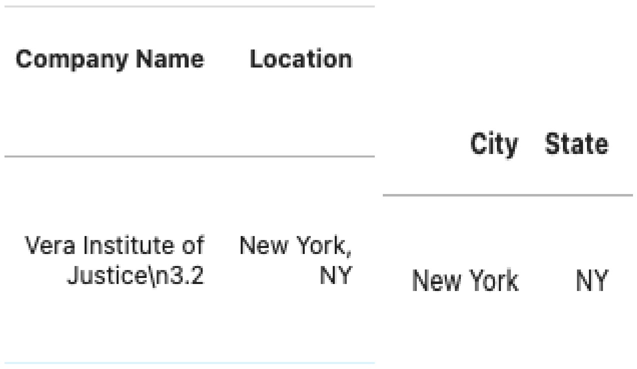


	Salary Estimate	Job Description	Rating	Salary_Min_Estimate	Salary_Max_Estimate
Job Title					
Data Analyst, Center on Immigration and Justice (CIJ)	37K−66K (Glassdoor est.)	Are you eager to roll up your sleeves and harn...	3.2	37000	66000



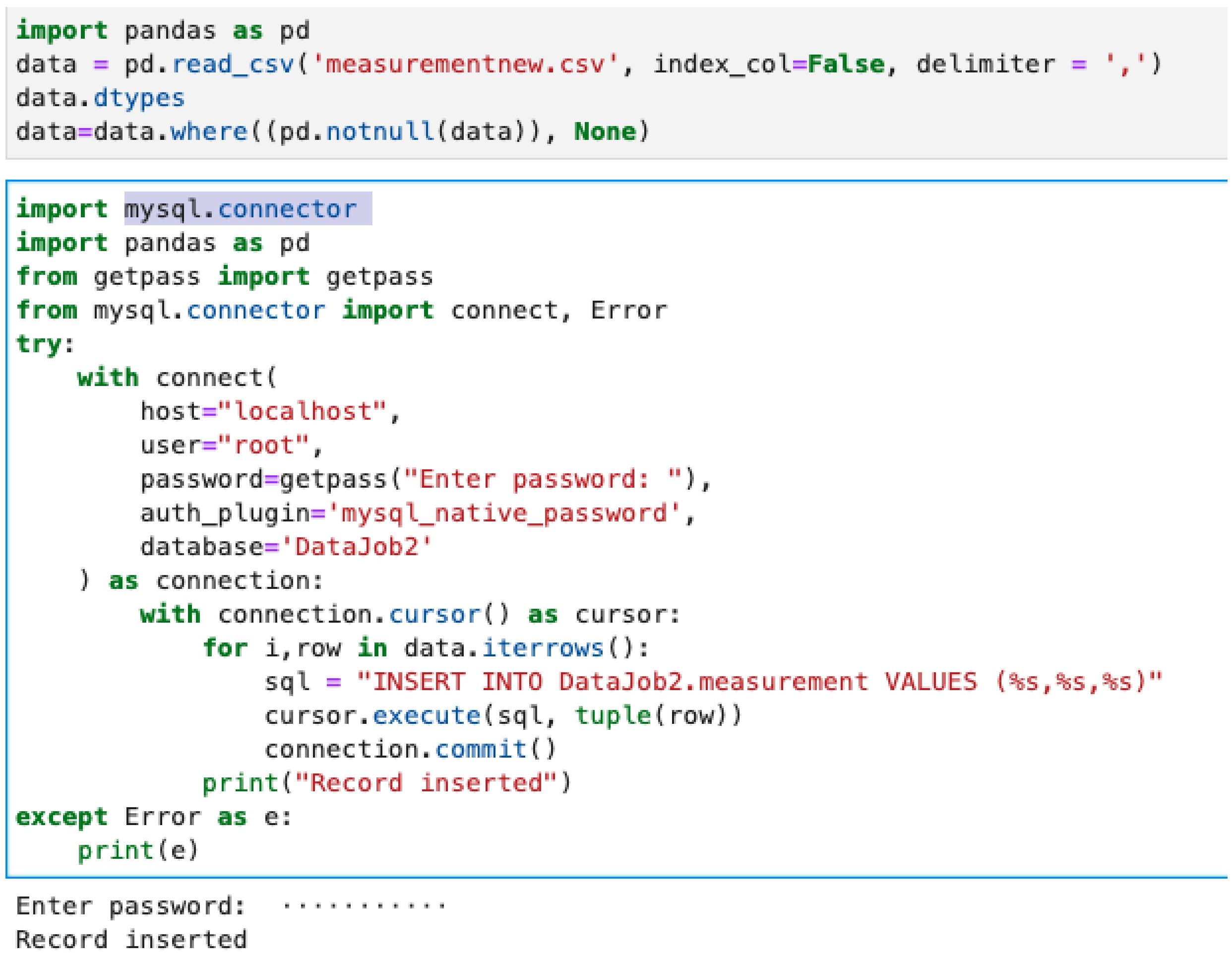
```
da[['City','State','1']]=da['Location'].str.split(",",expand=True)
```

```
da = da.drop(['1'],axis=1)
de[['City','State']]=de['Location'].str.split(",",expand=True)
ds[['City','State']]=ds['Location'].str.split(",",expand=True)
```



Company Name	Location	City	State
Vera Institute of Justice\n3.2	New York, NY	New York	NY

After all the data formatting, extracting and cleaning, we splitted and grouped some columns into its own table or csv file for our ER model, loading into MYSQL database, and our visualization in Tableau. For the last step of ETL, we loaded datasets into the MYSQL database by using a python library called mysql.connector to connect and query to insert the data into the database. Here is the example for python.



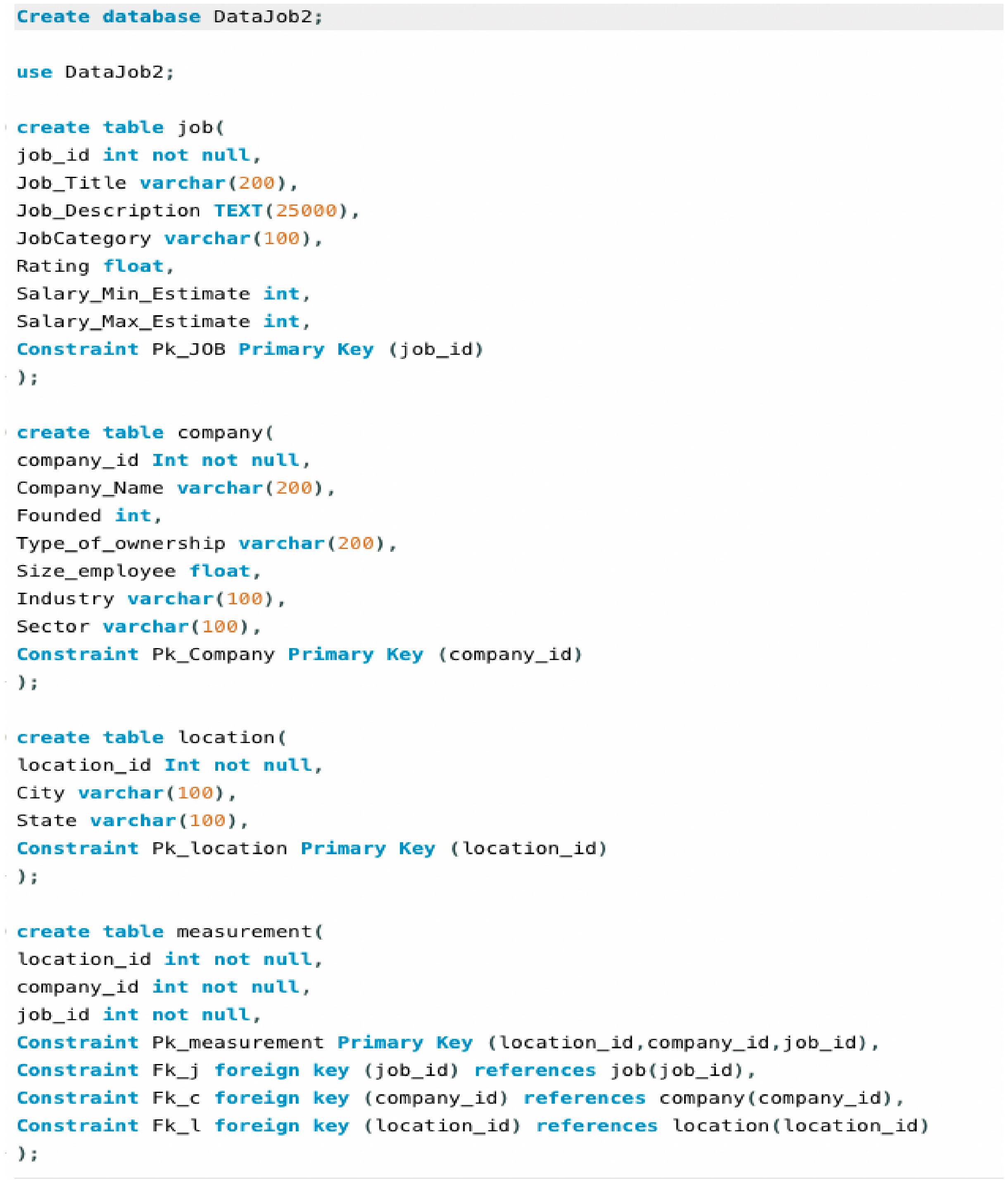
```python
import pandas as pd
data = pd.read_csv('measurementnew.csv', index_col=False, delimiter = ',')
data.dtypes
data=data.where((pd.notnull(data)), None)
```

```python
import mysql.connector
import pandas as pd
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(
        host="localhost",
        user="root",
        password=getpass("Enter password: "),
        auth_plugin='mysql_native_password',
        database='DataJob2'
    ) as connection:
        with connection.cursor() as cursor:
            for i,row in data.iterrows():
                sql = "INSERT INTO DataJob2.measurement VALUES (%s,%s,%s)"
                cursor.execute(sql, tuple(row))
                connection.commit()
            print("Record inserted")
except Error as e:
    print(e)
```

```
Enter password:  ···········
Record inserted
```

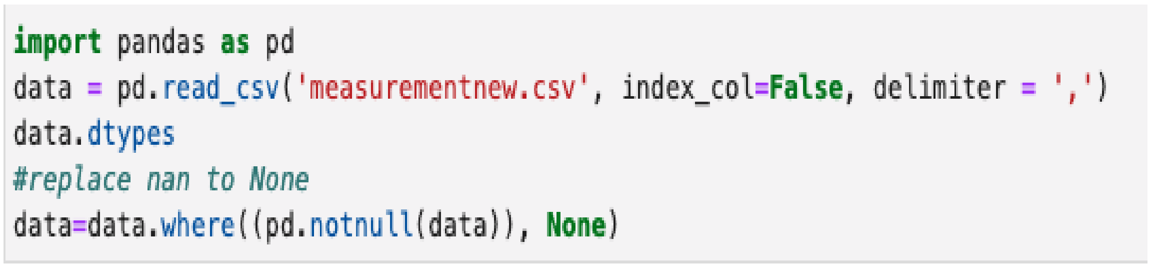
Create Table and Load for Database Used

This is the image of querying for table creation.

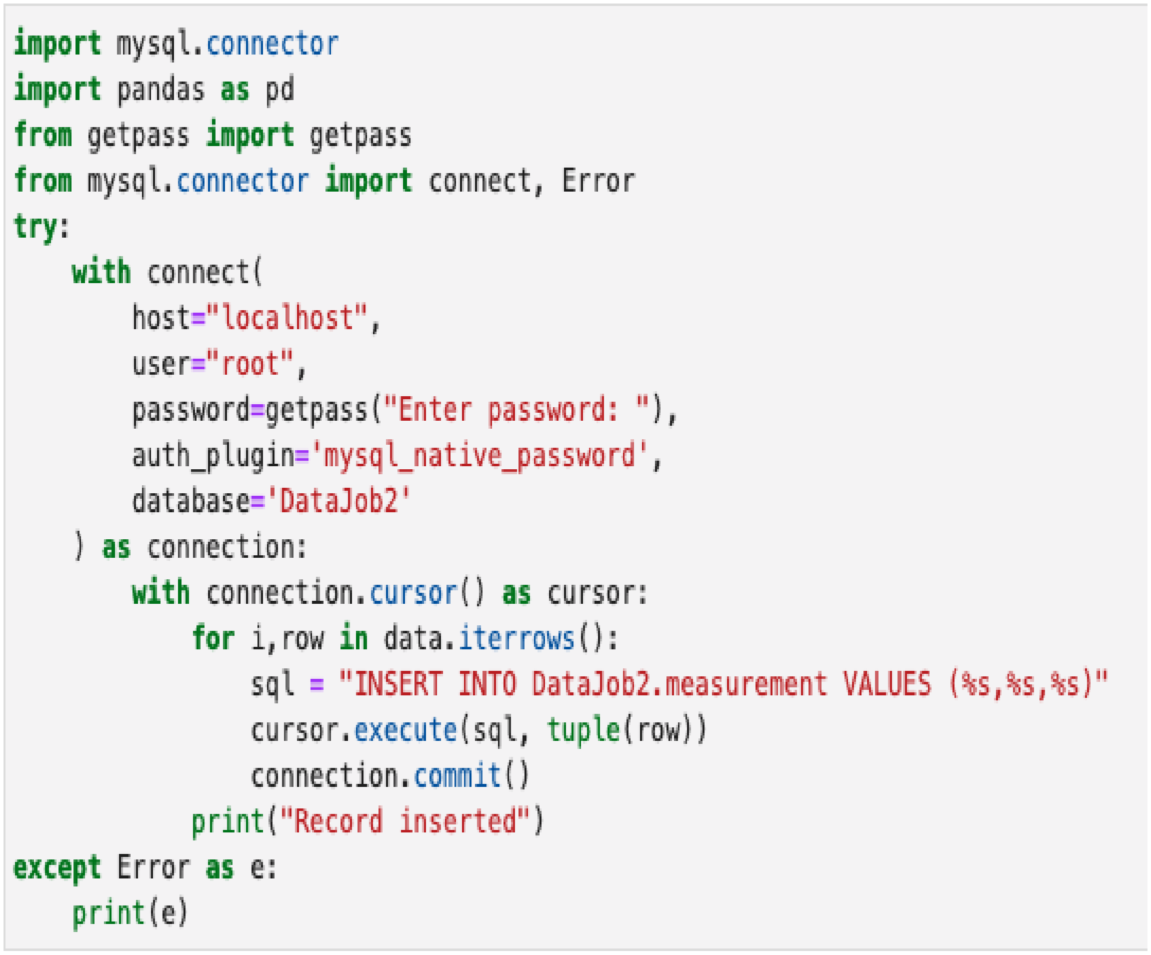


```sql
Create database DataJob2;

use DataJob2;

create table job(
job_id int not null,
Job_Title varchar(200),
Job_Description TEXT(25000),
JobCategory varchar(100),
Rating float,
Salary_Min_Estimate int,
Salary_Max_Estimate int,
Constraint Pk_JOB Primary Key (job_id)
);

create table company(
company_id Int not null,
Company_Name varchar(200),
Founded int,
Type_of_ownership varchar(200),
Size_employee float,
Industry varchar(100),
Sector varchar(100),
Constraint Pk_Company Primary Key (company_id)
);

create table location(
location_id Int not null,
City varchar(100),
State varchar(100),
Constraint Pk_location Primary Key (location_id)
);

create table measurement(
location_id int not null,
company_id int not null,
job_id int not null,
Constraint Pk_measurement Primary Key (location_id,company_id,job_id),
Constraint Fk_j foreign key (job_id) references job(job_id),
Constraint Fk_c foreign key (company_id) references company(company_id),
Constraint Fk_l foreign key (location_id) references location(location_id)
);
```

This is the images of loading the data into MySQL database for querying and ER Model from python with a python library called mysql.connector



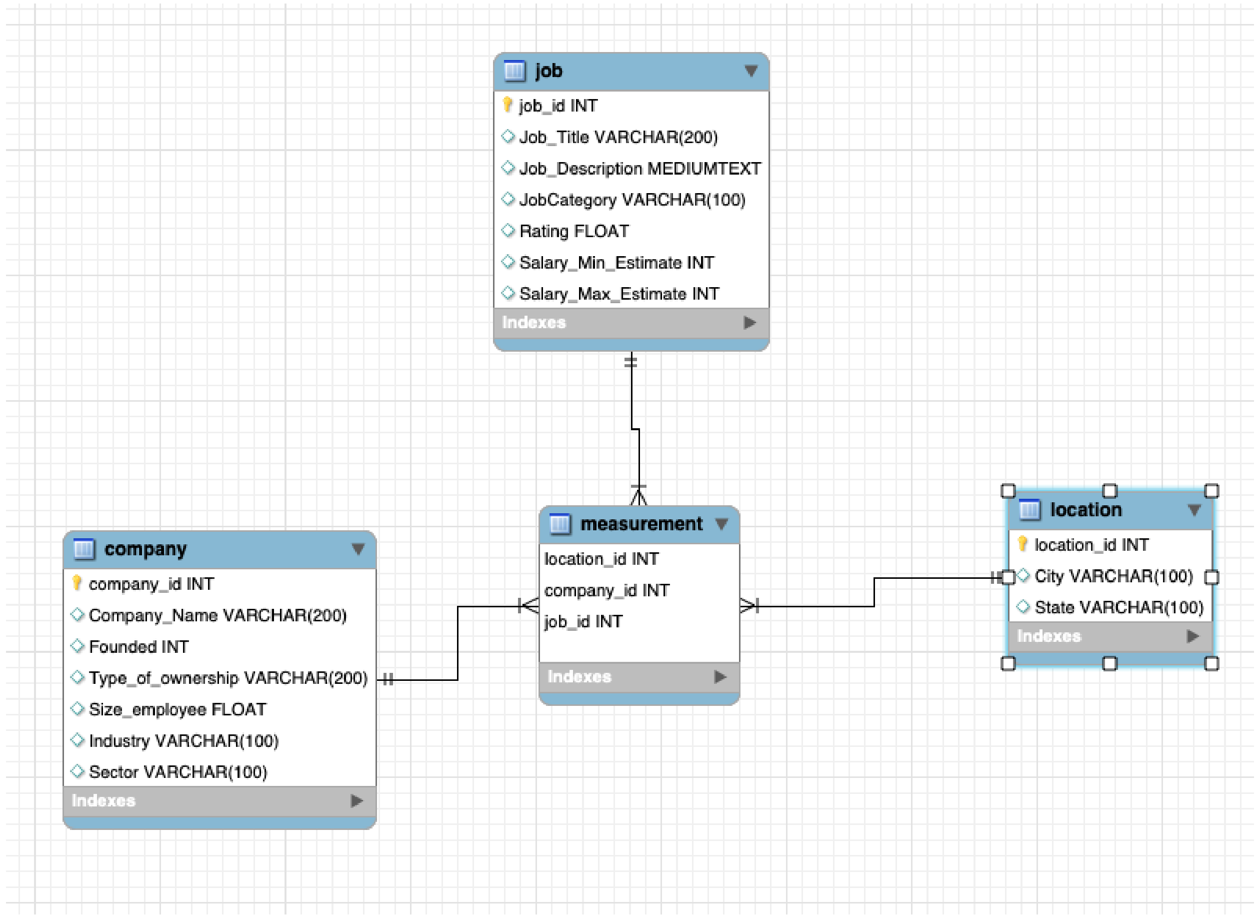
```python
import pandas as pd
data = pd.read_csv('measurementnew.csv', index_col=False, delimiter = ',')
data.dtypes
#replace nan to None
data=data.where((pd.notnull(data)), None)
```



```python
import mysql.connector
import pandas as pd
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(
        host="localhost",
        user="root",
        password=getpass("Enter password: "),
        auth_plugin='mysql_native_password',
        database='DataJob2'
    ) as connection:
        with connection.cursor() as cursor:
            for i,row in data.iterrows():
                sql = "INSERT INTO DataJob2.measurement VALUES (%s,%s,%s)"
                cursor.execute(sql, tuple(row))
                connection.commit()
            print("Record inserted")
except Error as e:
    print(e)
```

ER model from MYSQL Workbench

Based on the dataset that we cleaned from python pandas to do ETL(Extract Transform and Load) , We created a star schema for our data model which contains a factless fact table called measurement with forigen keys(location_id, job_id and comapny_id). We have three dimensional tables which are job, company and location. Each dimensional table contains their id and attributes. The main reason we do not make the model to be a snowflake with higher dimension hierarchy to separate city and state is that we want to have less join in the query and better performance for querying in the relational database without any time out. There is a trade off by using star schema which is data redundancy and disk storage.



Query For The Brief Summary

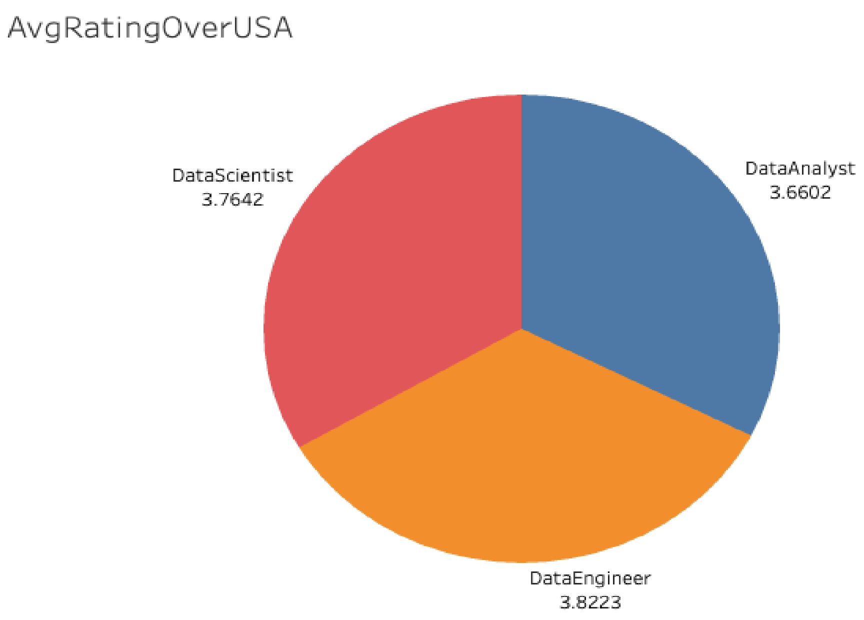
We used the MYSQL database system to query some data and made visualizations in Tableau after we cleaned and created the database and tables.

Avg rating for each Job Category

Finding: The average rating of all three category are similar, but data eng

```sql
Select j.JobCategory , avg(rating)
From job j , measurement m  , location l
where j.job_id = m.job_id and l.location_id = m.location_id
group by j.JobCategory;
```

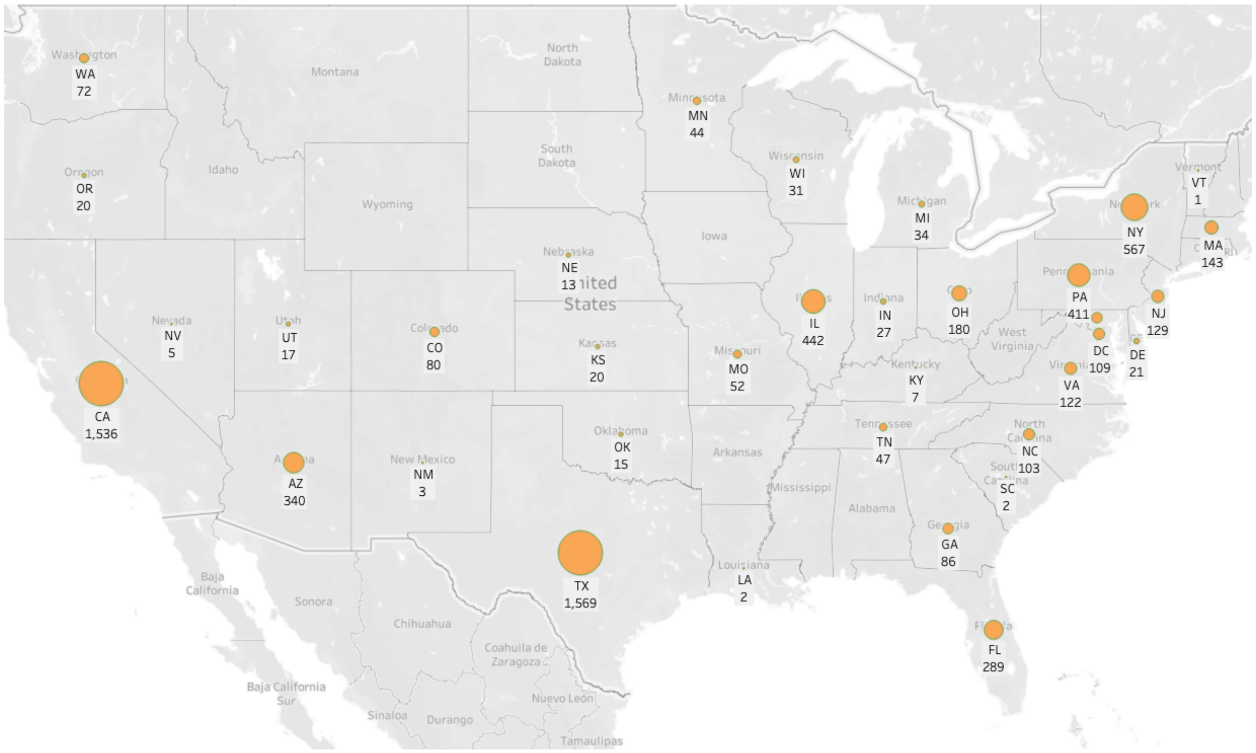
JobCategory	avg(rating)
DataAnalyst	3.660208254465165
DataEngineer	3.8223367633142287
DataScientist	3.7642210105946963

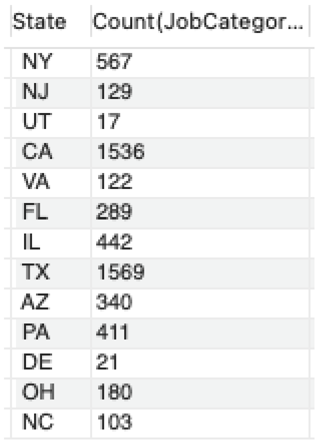


How many data related jobs are in each state?

Finding: There are more data related jobs and opportunities in California and Texas compared to other states.

```
Select State, Count(JobCategory)
from location Natural Join job
Natural Join  measurement Group by State;
```

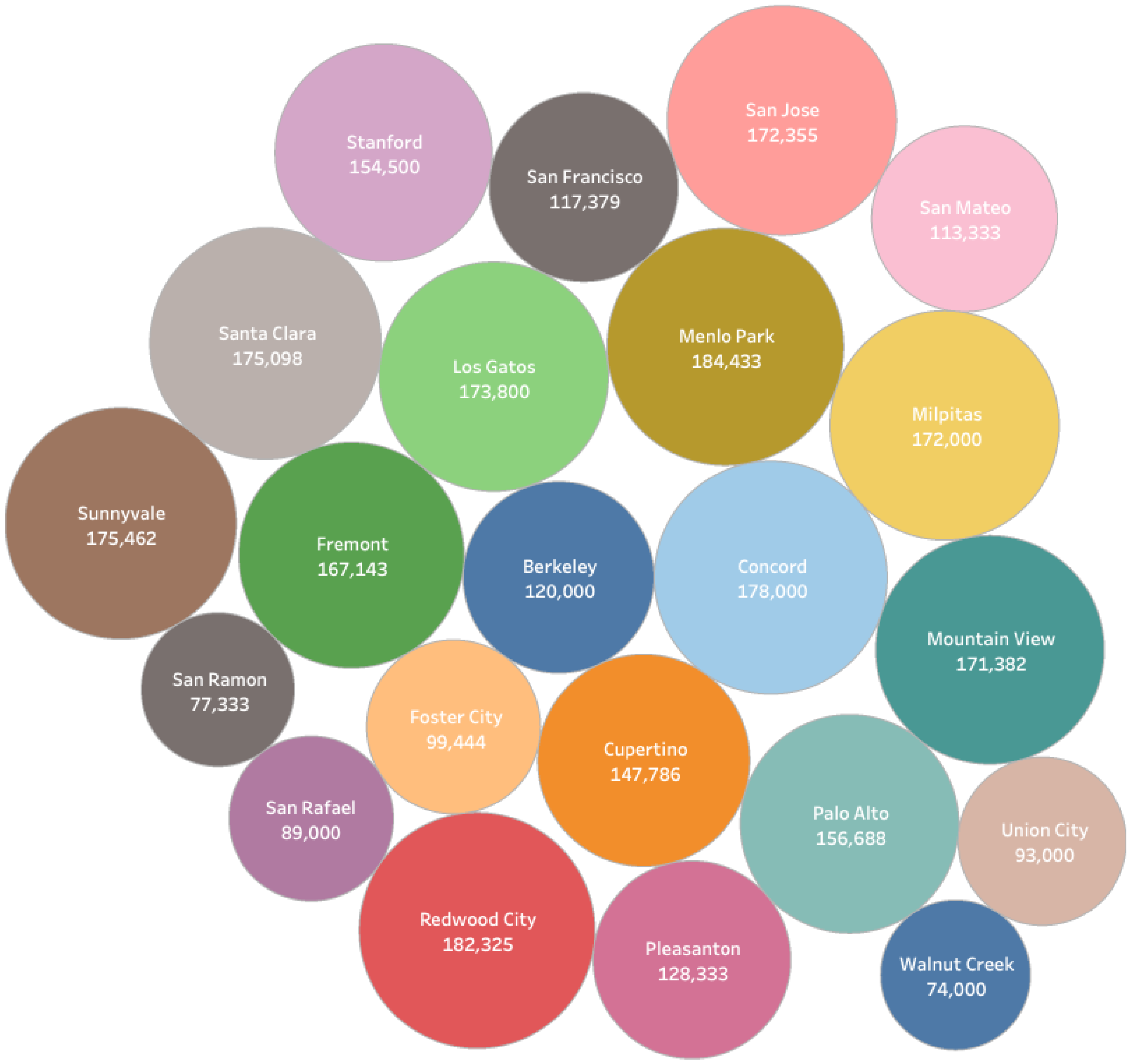




State	Count(JobCategor...
NY	567
NJ	129
UT	17
CA	1536
VA	122
FL	289
IL	442
TX	1569
AZ	340
PA	411
DE	21
OH	180
NC	103

What is the Average estimated salary for a data related job in each city around the Bay Area?

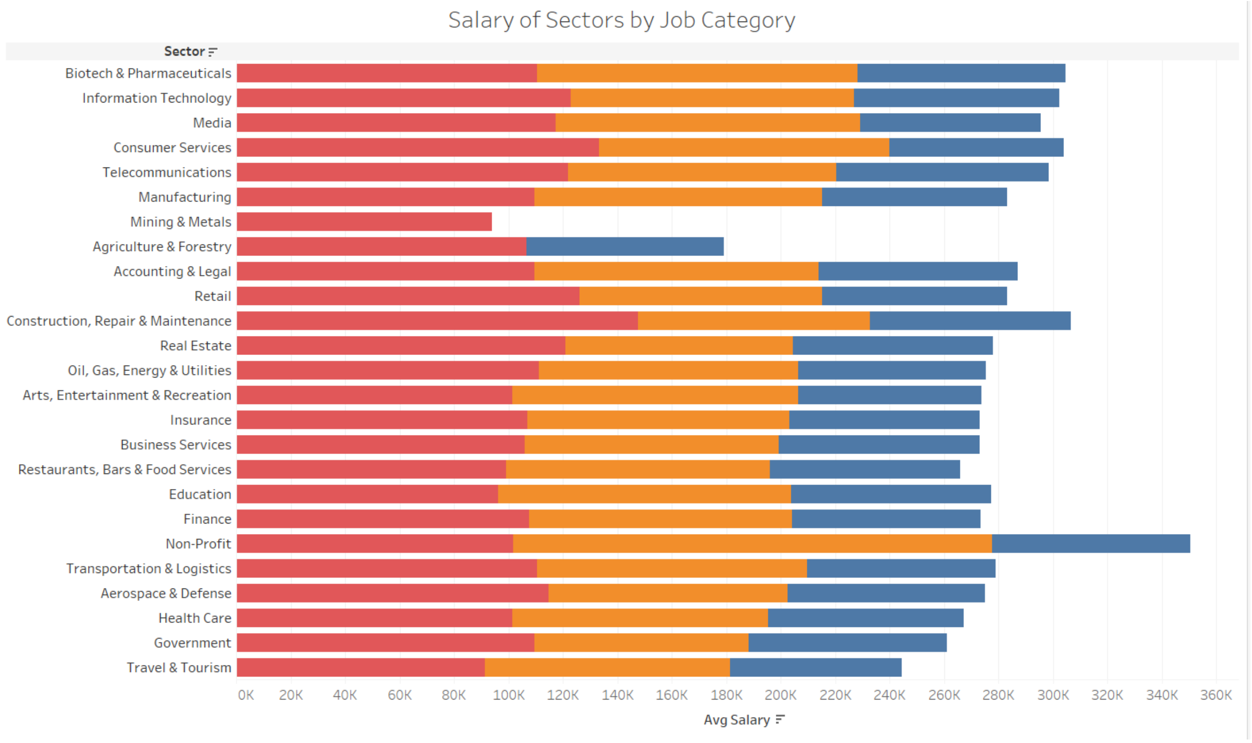
Finding: Although Redwood City and Menlo Park have the highest average estimated salary in the bay area, all the main cities in the bay area have about the same average salary.



Visualization (Tableau)

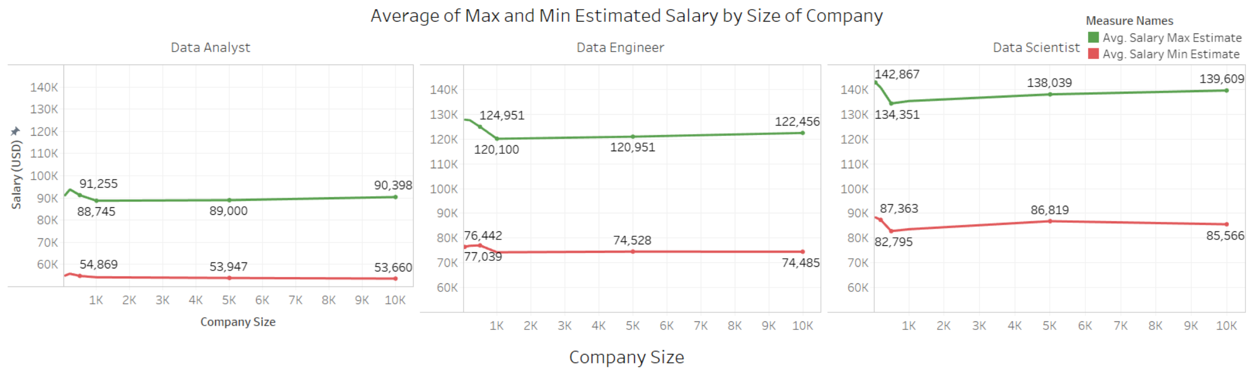
Salary of Sector by Job Category

Finding: Data engineers in the non-profit sector has the highest salary compared to all other sector and job category.

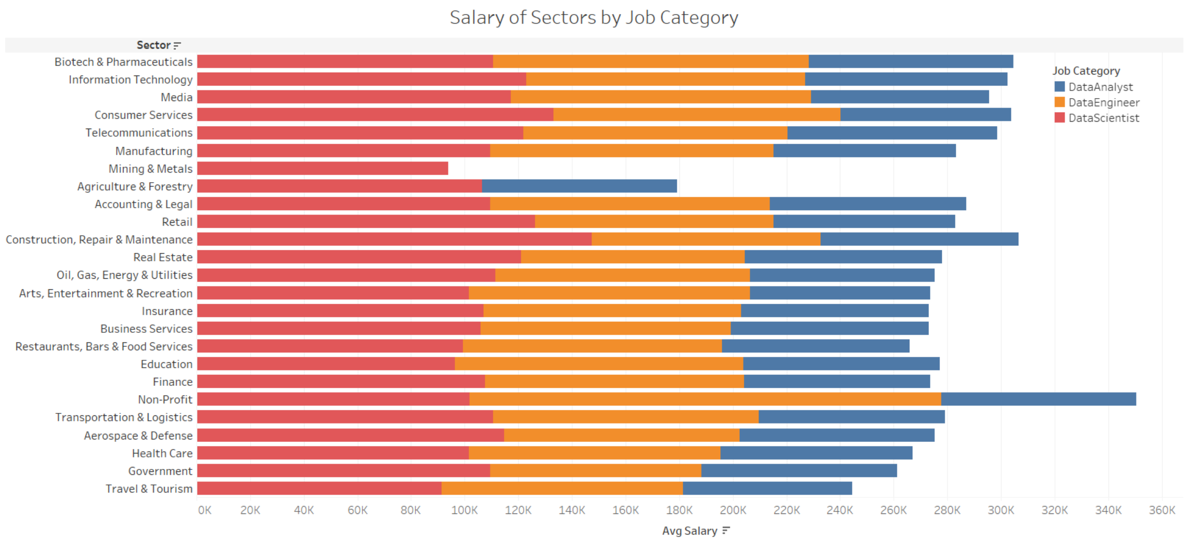


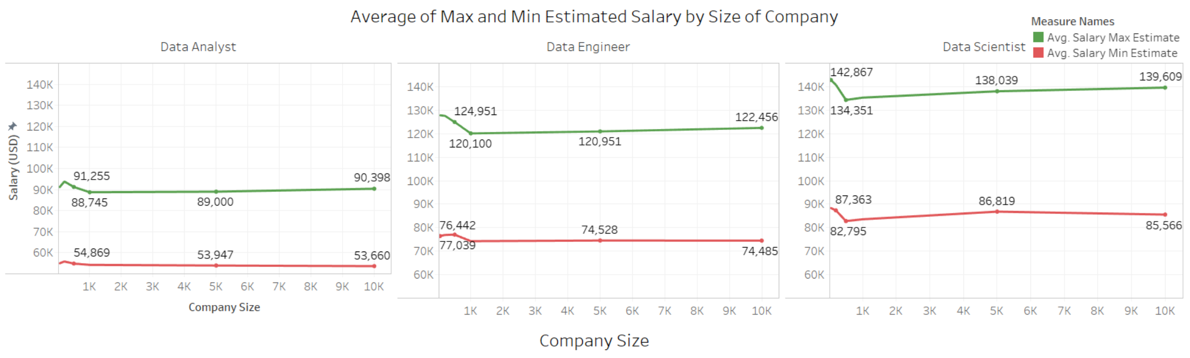
Average of Max and Min Estimated Salary by Size of Company

Finding: Data scientist has the highest salary. Data Engineer is the second highest and Data Analyst is the lowest among the three.



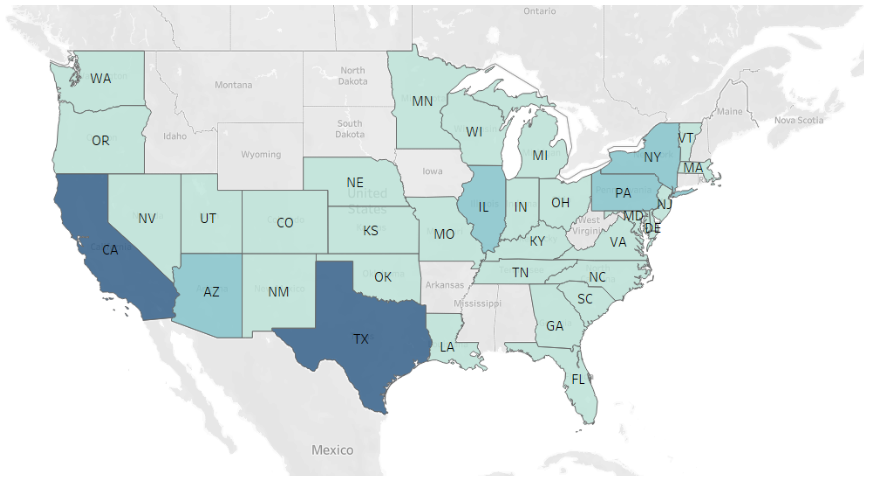
Dashboard 1



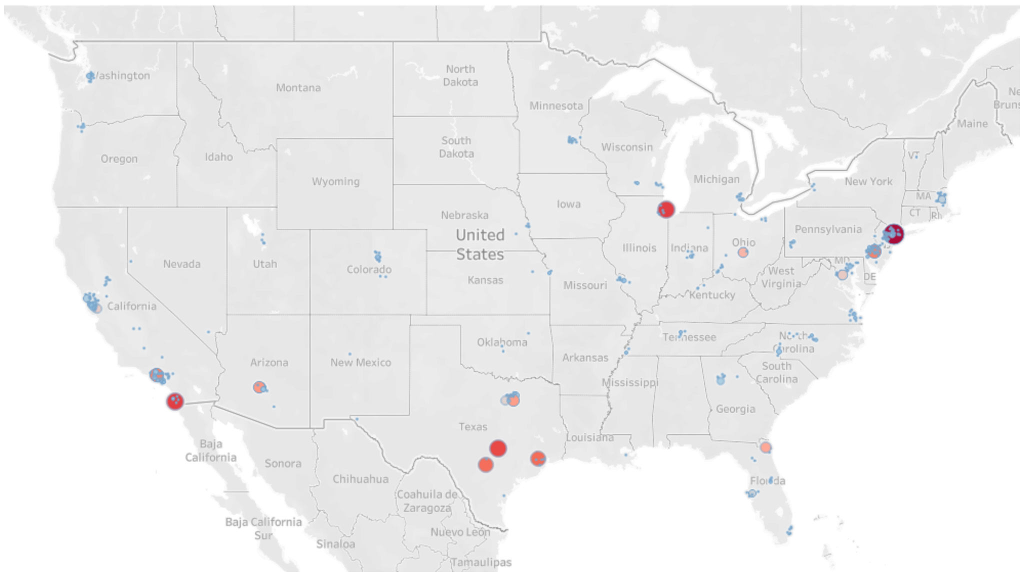


Number of Job Category by State

Finding: California and Texas have the highest Number of Job Categories in the U.S.

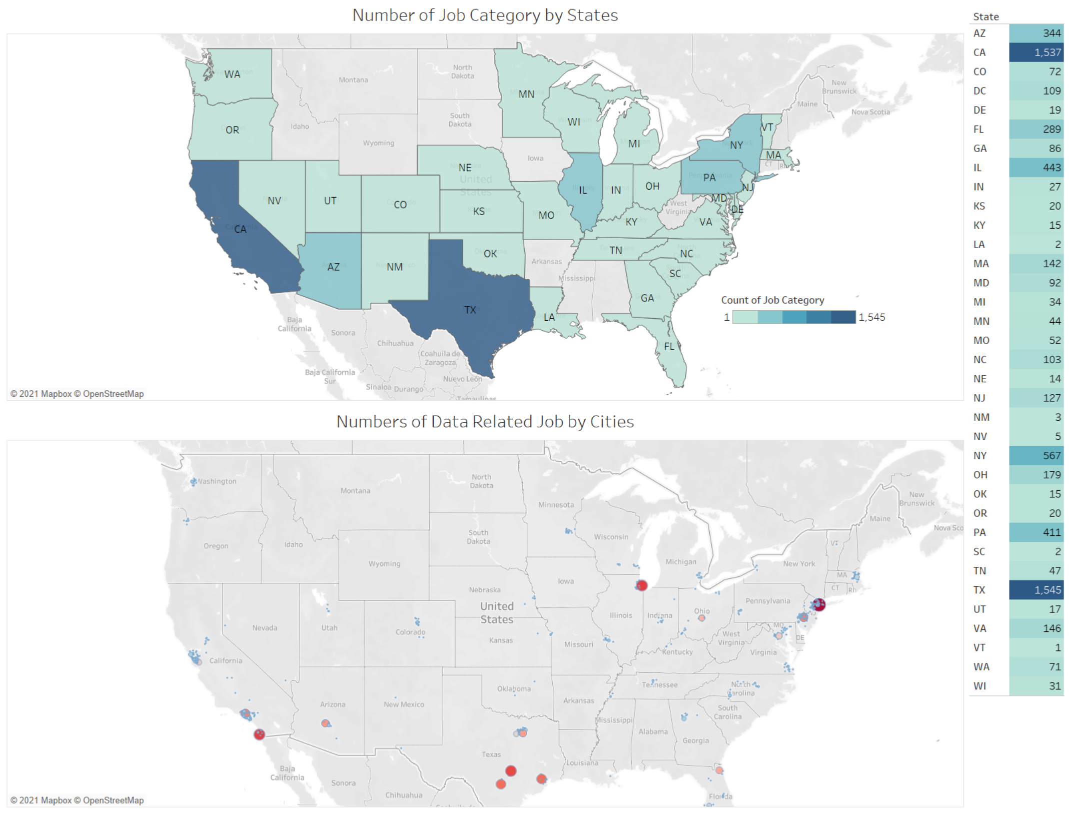


Numbers of Data Related Job by Cities



Dashboard 2

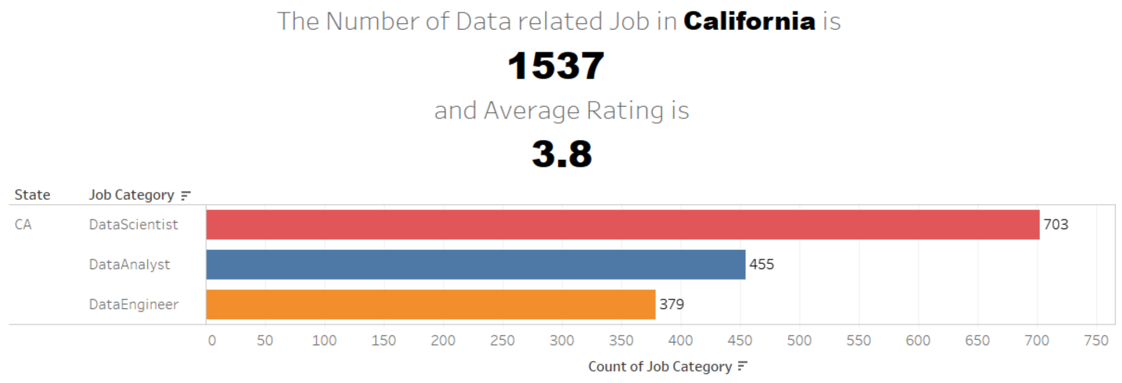
Dashboard 2 shows the number of job categories by each state; it is also drilled into cities. The number of jobs in cities can be judged by the size of the circle.


Number of Job Category by States
Count of Job Category
1
1,545
© 2021 Mapbox © OpenStreetMap
Numbers of Data Related Job by Cities
© 2021 Mapbox © OpenStreetMap

State	
AZ	344
CA	1,537
CO	72
DC	109
DE	19
FL	289
GA	86
IL	443
IN	27
KS	20
KY	15
LA	2
MA	142
MD	92
MI	34
MN	44
MO	52
NC	103
NE	14
NJ	127
NM	3
NV	5
NY	567
OH	179
OK	15
OR	20
PA	411
SC	2
TN	47
TX	1,545
UT	17
VA	146
VT	1
WA	71
WI	31

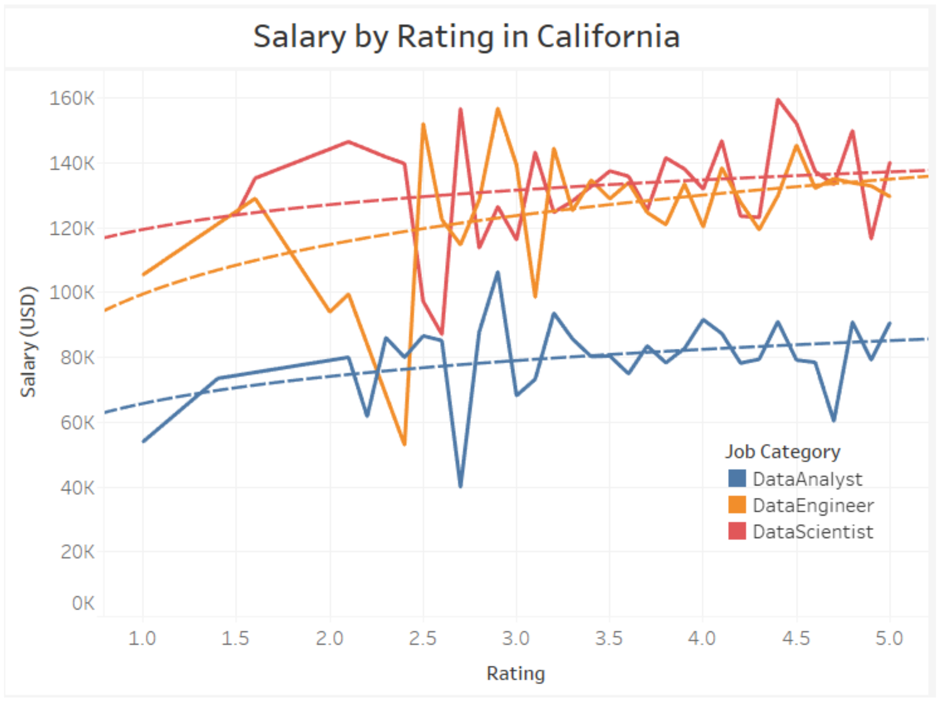
Number of Data Related Jobs in California

Finding: In California, data scientists have the most number of jobs. Data analyst is second and data engineer is the least.



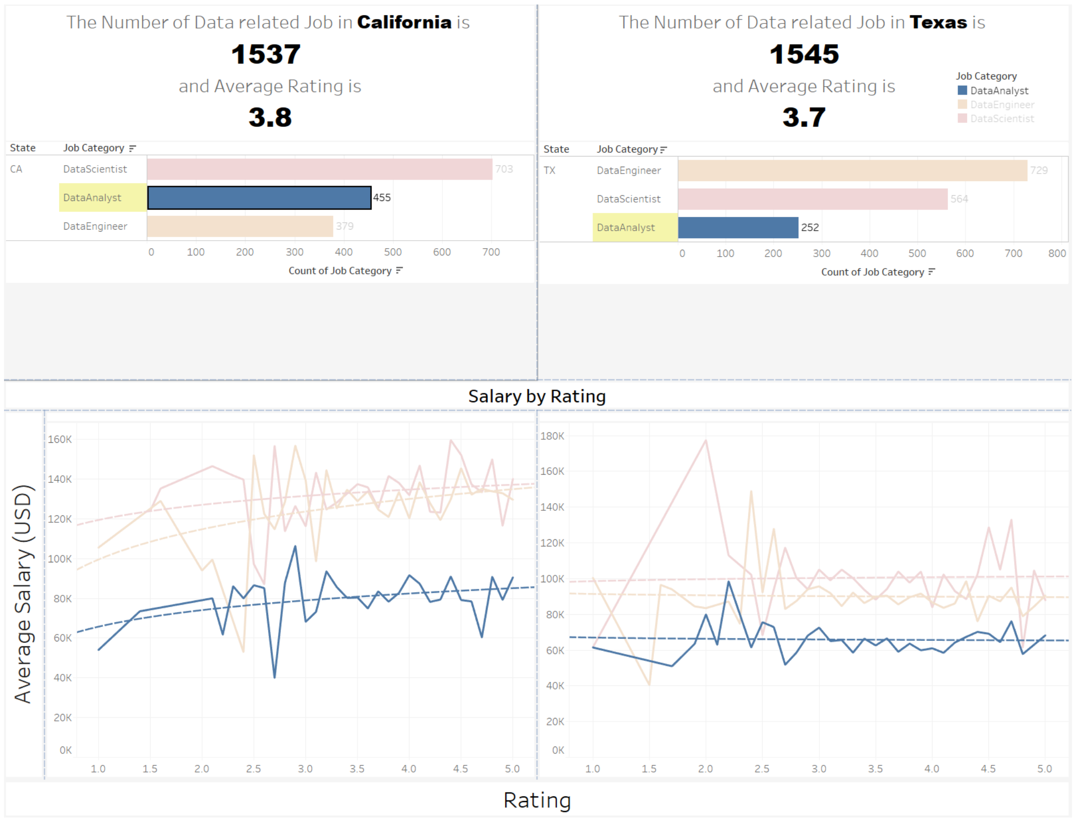
Salary(USD) by Rating

Finding : Salary tend to be higher when the company rating is higher in California



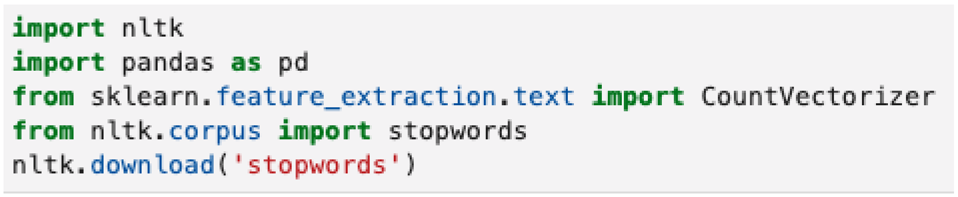
Dashboard 3

This dashboard mainly compares the difference between California and Texas since they have the most data related job in United State.

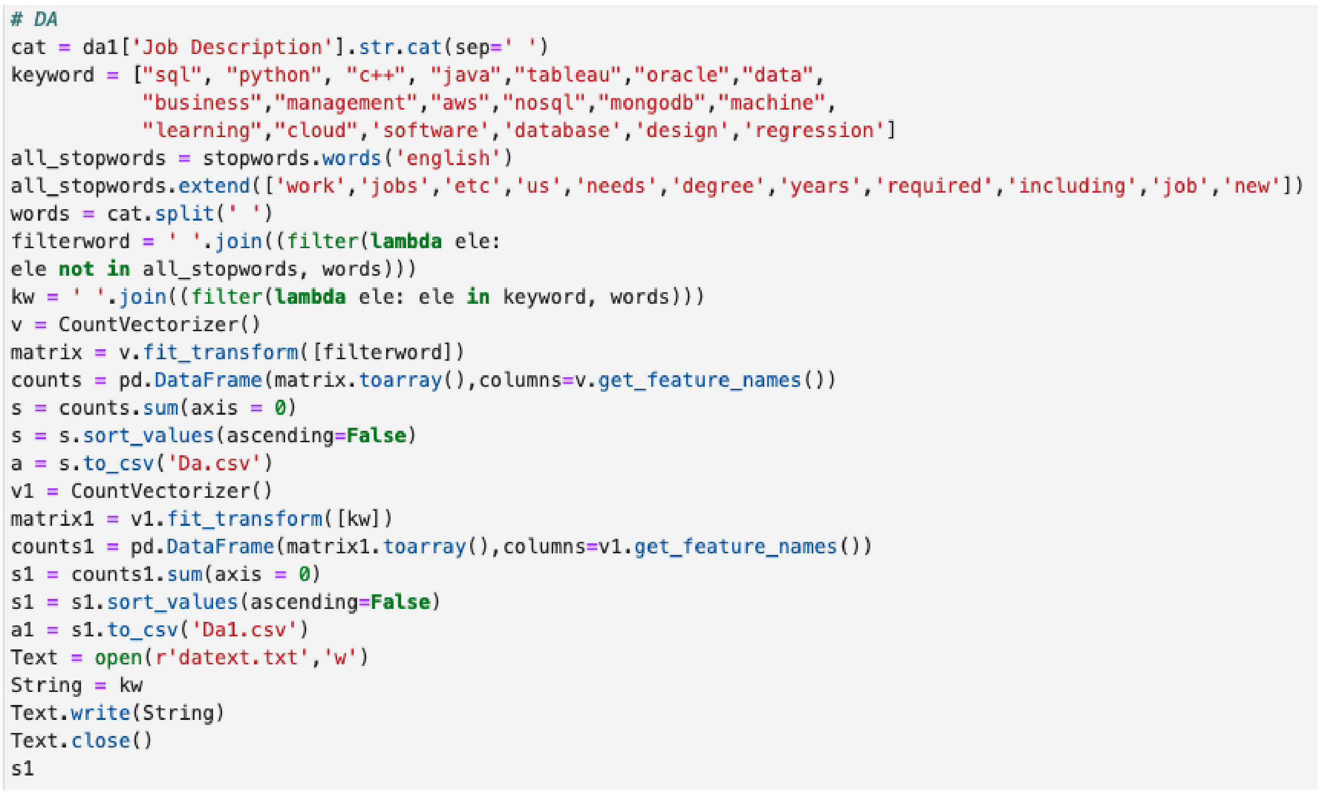


Analysis(Job Description Keyword Searching From Each Job Category)

The reason we applied keyword search and word counting on job descriptions was because it was too complicated to read all the job descriptions that match its respective positions (data analyst, data engineer, and data scientist), to see the requirements that match the skill set needed for each data job position. We used python coding to do a quick keyword search from all the job descriptions, and aggregate the results to figure out how the word pattern in each data job description was tied to its respective job position. After we got the results, we loaded it into a csv file and text files for visualization in Tableau and created graphs and word clouds to display the results. Here is an example of coding by using nltk and sklearn library in python.



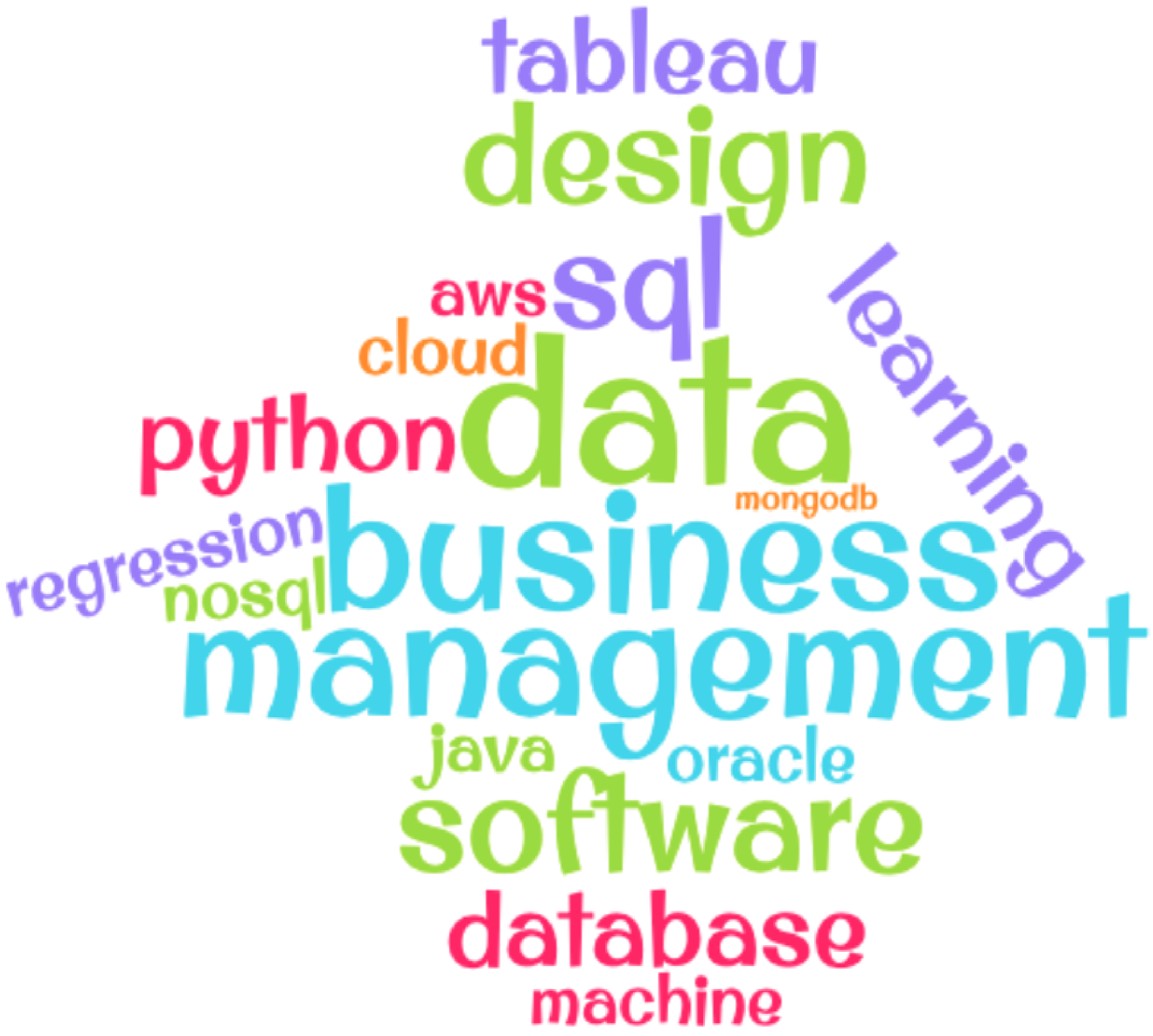
```
import nltk
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
nltk.download('stopwords')
```



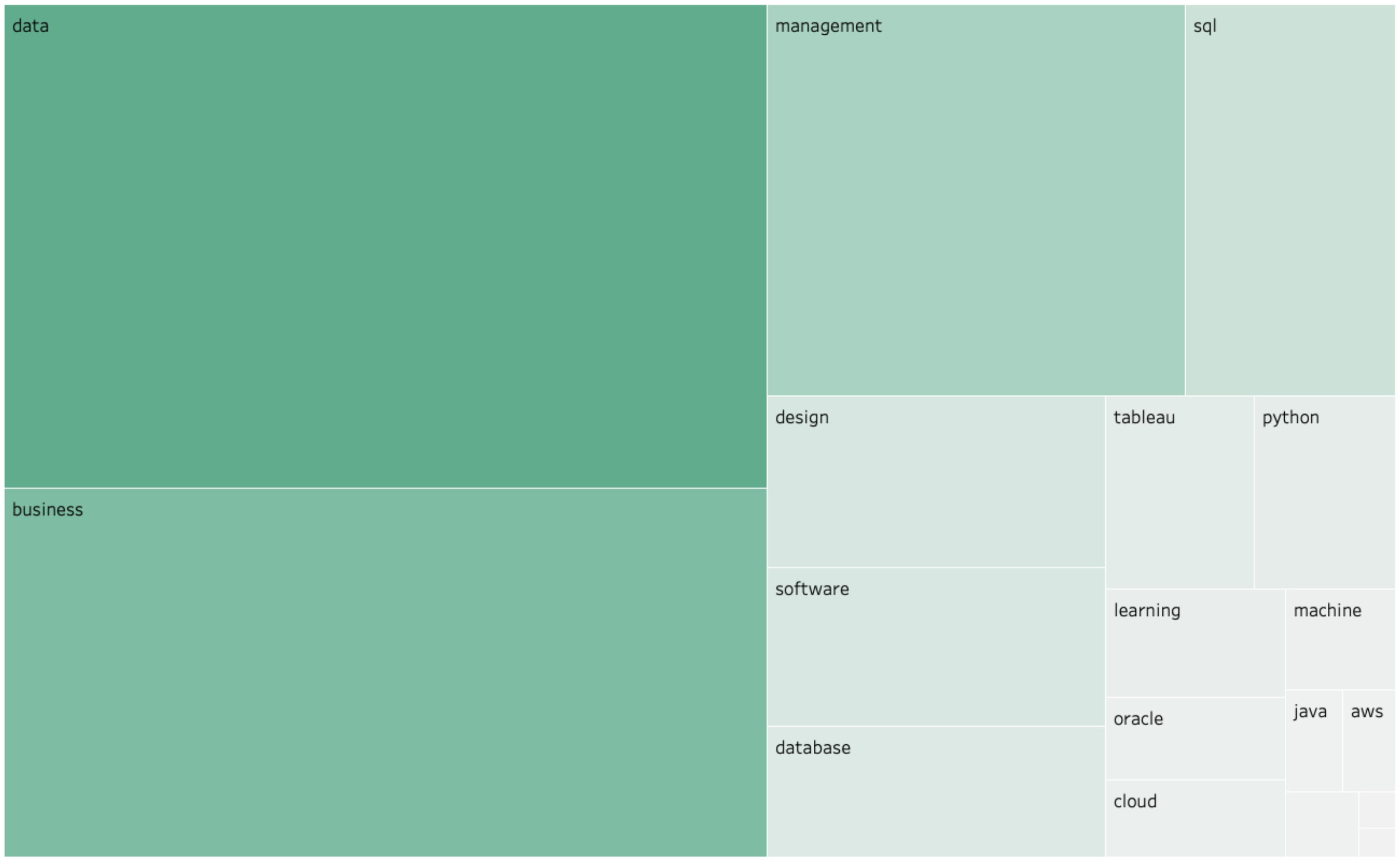
```
# DA
cat = da1['Job Description'].str.cat(sep=' ')
keyword = ["sql", "python", "c++", "java","tableau","oracle","data",
           "business","management","aws","nosql","mongodb","machine",
           "learning","cloud",'software','database','design','regression']
all_stopwords = stopwords.words('english')
all_stopwords.extend(['work','jobs','etc','us','needs','degree','years','required','including','job','new'])
words = cat.split(' ')
filterword = ' '.join((filter(lambda ele:
ele not in all_stopwords, words)))
kw = ' '.join((filter(lambda ele: ele in keyword, words)))
v = CountVectorizer()
matrix = v.fit_transform([filterword])
counts = pd.DataFrame(matrix.toarray(),columns=v.get_feature_names())
s = counts.sum(axis = 0)
s = s.sort_values(ascending=False)
a = s.to_csv('Da.csv')
v1 = CountVectorizer()
matrix1 = v1.fit_transform([kw])
counts1 = pd.DataFrame(matrix1.toarray(),columns=v1.get_feature_names())
s1 = counts1.sum(axis = 0)
s1 = s1.sort_values(ascending=False)
a1 = s1.to_csv('Da1.csv')
Text = open(r'datext.txt','w')
String = kw
Text.write(String)
Text.close()
s1
```

Data Analysis

The graph is showing all the keywords that we learn from our classes. The bigger the font is, the more frequent the keyword is in a job description.



The graph is showing all the keywords that we learn from our classes. The bigger the box is, the more frequent the keyword is in a job description.

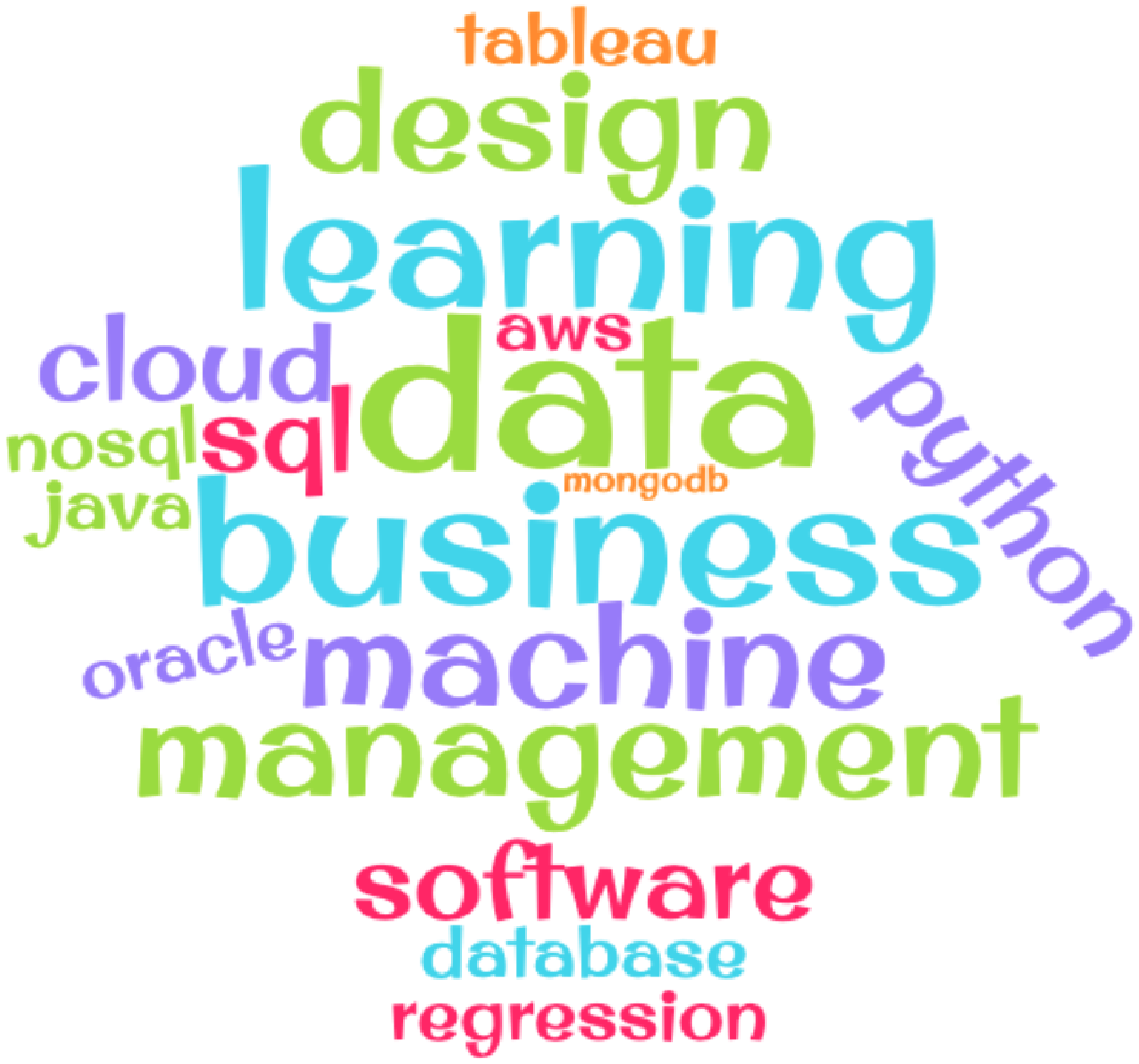


Findings:

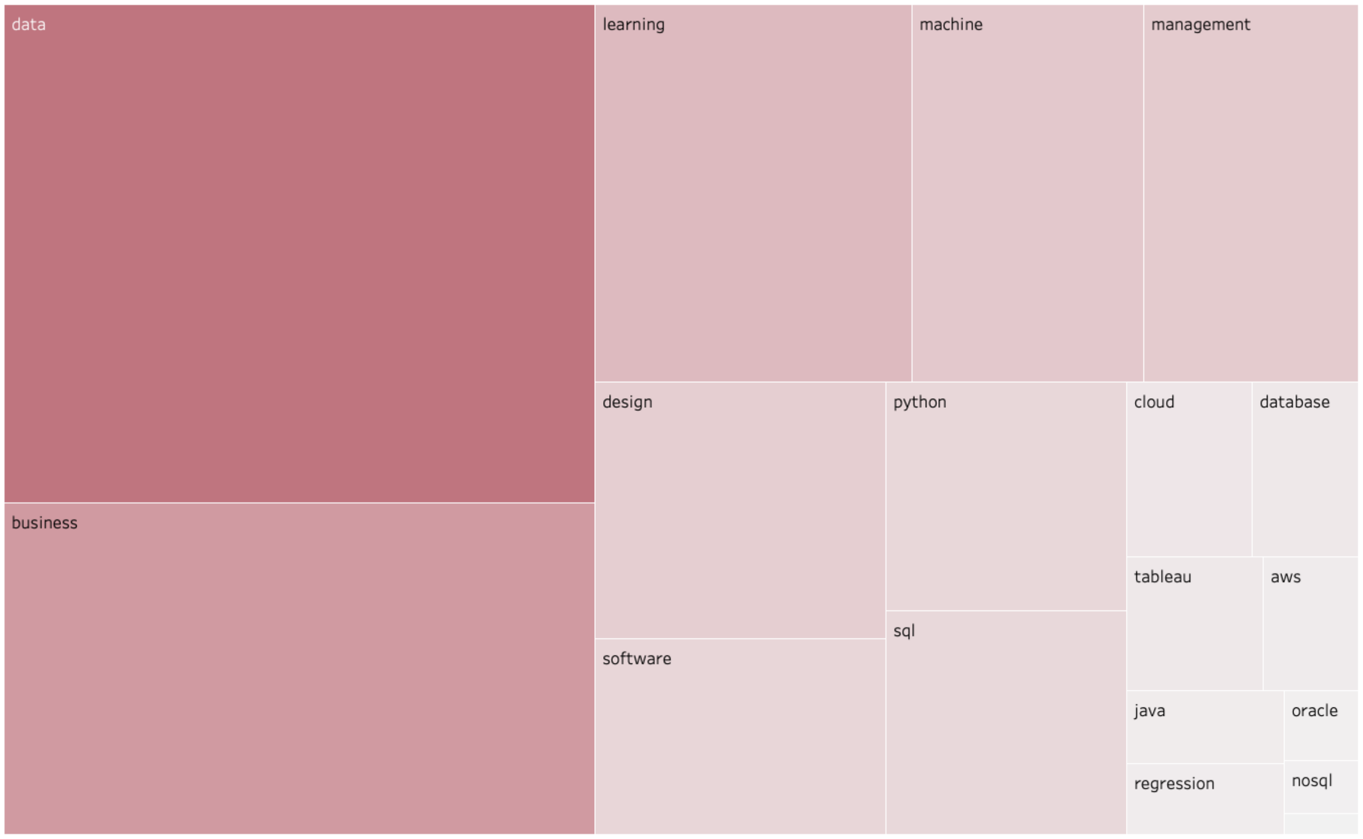
Based on the visualization above, we can see that data analysts need more skills and knowledge on business and management with programming skills like sql and python. On the other hand, data analysts do not need so much skill and knowledge from machine learning and cloud systems to work in the data analysis field.

Data Science

The graph is showing all the keywords that we learn from our classes. The biggest font it is, the most frequent it is in a job description.



The graph is showing all the keywords that we learn from our classes. The bigger the box is, the more frequent the keyword is in a job description.

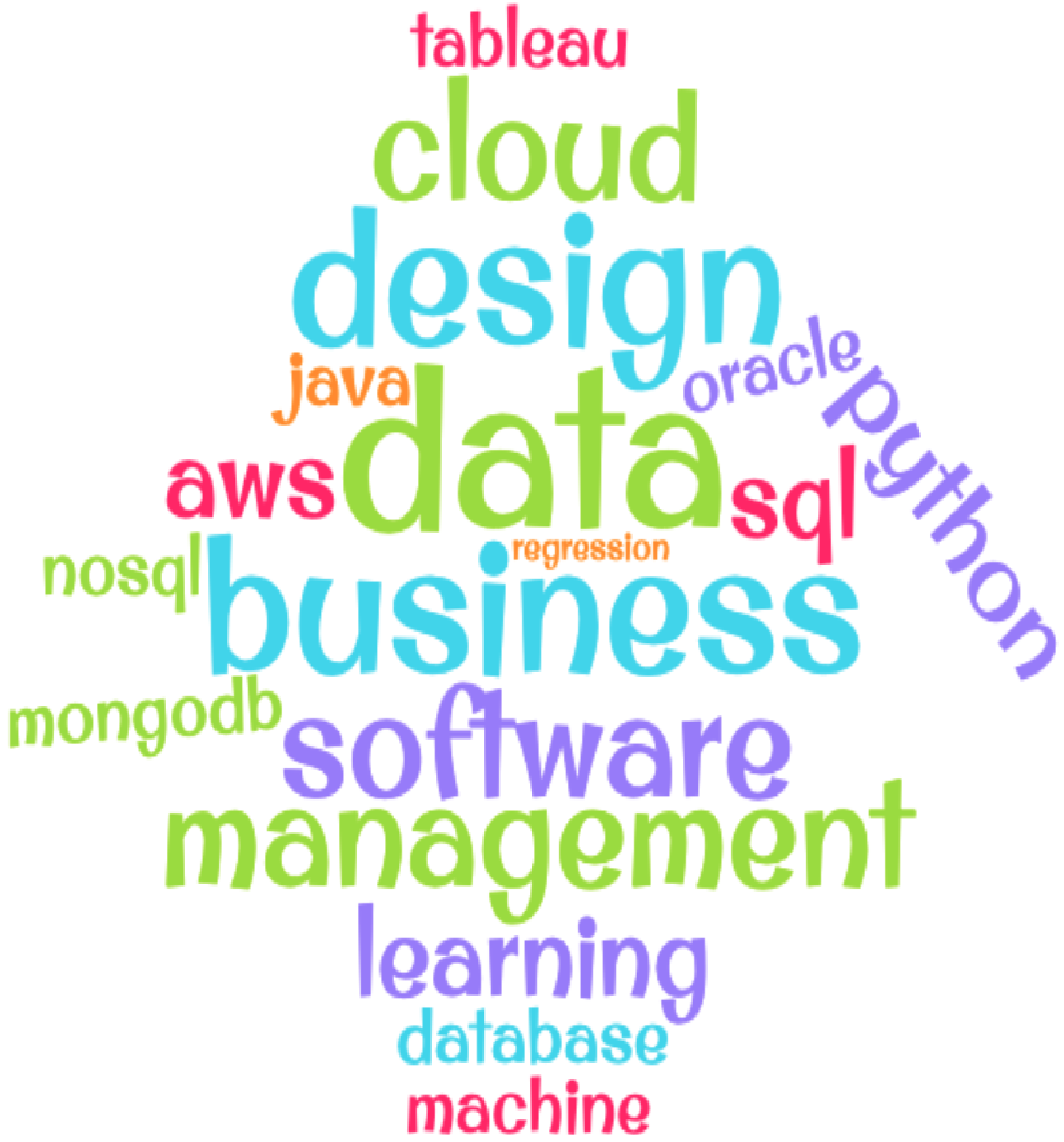


Findings:

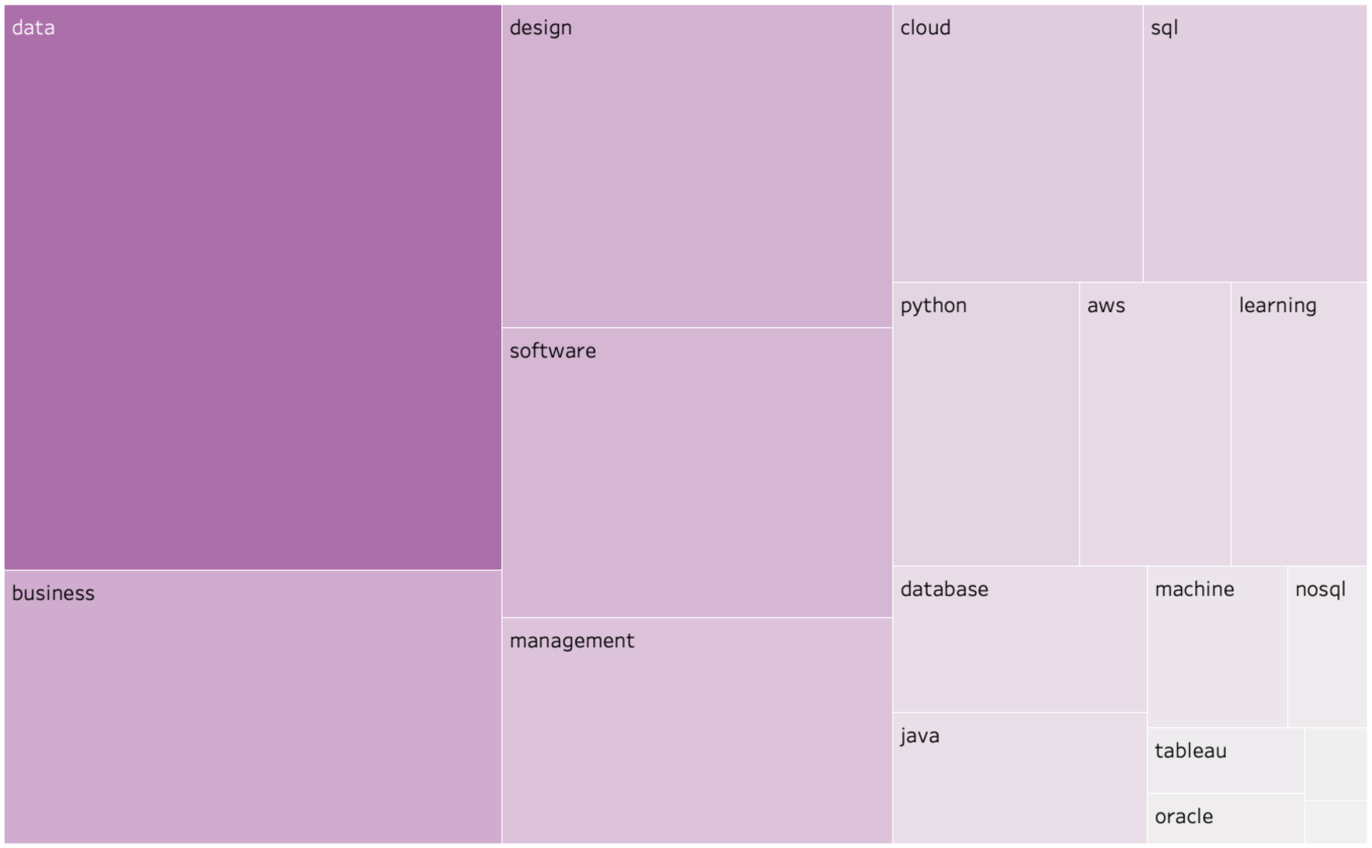
Based on the visualization above, machine learning is moving up from the bottom of the list to top 5 keywords compared to the list in data analysis. As data scientists, they require more knowledge and skills in machine learning along with the programming and software skills such as python and sql.

Data Engineer

The graph is showing all the keywords that we learn from our classes. The biggest font it is, the most frequent it is in a job description.



The graph is showing all the keywords that we learn from our classes. The bigger the box is, the more frequent the keyword is in a job description.



Findings:

Based on the visualization above, we can see that as a data engineer they are required more software skill and software design compared to two previous graphs . They are also required to be able to use or have experience on cloud systems such as AWS. On the other hand, data engineers have less requirements on knowledge of machine learning.

Overall Comparison

In this comparison, it shows what the job requirements are based on different roles. The higher the position goes, the more higher-level skill it requires.

Data Analysis	Data Science	Data Engineer
tableau design aws sql cloud learning python data mongodb regression nosql business management java oracle software database machine	tableau design learning cloud aws nosql sql data java mongodb python business oracle machine management software database regression	tableau cloud design java oracle aws data sql python nosql regression business mongodb software management learning database machine

Data Analysis	Data Science	Data Engineer
• Business	• Machine Learning	• Software Skills
• Management	• Python	• Software design
• Python	• SQL	• AWS
• SQL	• Cloud System	• Cloud System
• Database	• Regression Analysis	• Python

Conclusion

Since we are all students within the MSDA program, the motivation for our project goal learn from the core of every Data Analysis project ie. Using a database! And considering the volume of data being generated every day, we plan to understand the very fundamentals of using a NoSQL database in our day to day life. This research will help our success after graduation to scope what types of jobs are available from the various data sets by allowing us to by understanding the specific requirements companies have. We plan to use this research to assist with what gaps in knowledge we may have and help us progress to the right stages, where obtaining these positions will come easily. Most of our excitement comes from where (LinkedIn, Glassdoor, etc.) we can find these job opportunities, what salary ranges they offer, company culture, and the range of skills we will further develop.

Source:
https://github.com/picklesueat/data_jobs_data

https://www.glassdoor.com/Salaries/index.htm

https://www.kaggle.com/andrewmvd/data-analyst-jobs

https://www.kaggle.com/andrewmvd/data-scientist-jobs

https://www.kaggle.com/andrewmvd/data-engineer-jobs

https://www.edureka.co/blog/data-analyst-vs-data-engineer-vs-data-scientist/

https://www.wordclouds.com/

Our Project and Coding link in GitHub:

https://github.com/cheukhongip/228CowSystemProjectCoding

Query for creating table:

```
create table job(
job_id int not null,
Job_Title varchar(200),
Job_Description TEXT(25000),
JobCategory varchar(100),
Rating float,
Salary_Min_Estimate int,
Salary_Max_Estimate int,
Constraint Pk_JOB Primary Key (job_id)
);
create table company(
company_id Int not null,
Company_Name varchar(200),
Founded int,
Type_of_ownership varchar(200),
Size_employee float,
Industry varchar(100),
Sector varchar(100),
Constraint Pk_Company Primary Key (company_id)
);
```

```
create table location(
location_id Int not null,
City varchar(100),
```

State varchar(100), Constraint Pk_location Primary Key (location_id));
create table measurement(location_id int not null, company_id int not null, job_id int not null, Constraint Pk_measurement Primary Key (location_id,company_id,job_id), Constraint Fk_j foreign key (job_id) references job(job_id), Constraint Fk_c foreign key (company_id) references company(company_id), Constraint Fk_l foreign key (location_id) references location(location_id));

Querying for Summary and visualization	
Avg rating for each Job Category	Select j.JobCategory , avg(rating) From job j , measurement m , location l where j.job_id = m.job_id and l.location_id = m.location_id group by j.JobCategory;
How many data related jobs are in each state?	Select State, Count(JobCategory) from location Natural Join job Natural Join measurement Group by State;
What is the highest estimated salary in each city In CA?	Select City , Round(Avg(Salary_Max_Estimate) ,0) from job Natural Join measurement Natural Join location where Trim(City) = "San Mateo" or Trim(City) = "San Jose" or Trim(City) = "San Ramon" or Trim(City) = "Santa Clara" or Trim(City) = "Stanford" or Trim(City) = "Walnut Creek" or Trim(City) = "Mountain View" or Trim(City) = "Palo Alto" or Trim(City) = "Pleasanton" or Trim(City) = "Redwood City" or Trim(City) = "San Rafael" or Trim(City) = "Sunnyvale" or Trim(City) = "Union City" or Trim(City) = "San Francisco" or Trim(City) = "Los Gatos" or Trim(City) = "Menlo Park" or

	Trim(City) = "Milpitas" or Trim(City) = "Concord" or Trim(City) = "Cupertino" or Trim(City) = "Foster City" or Trim(City) = "Fremont" or Trim(City) = "Berkeley" group by City;

Python Coding for ETL

import pandas as pd
import numpy as np
import string
 	
da = pd.read_csv("DataAnalyst.csv",index_col=0)
de = pd.read_csv("DataEngineer.csv",index_col=0)
ds = pd.read_csv("DataScientist.csv",index_col=0)
 	
#Drop Unused Column and ROW
da['Rating'] = da['Rating'].dropna()
de['Rating'] = de['Rating'].dropna()
ds['Rating'] = ds['Rating'].dropna()
da['Salary Estimate'] = da['Salary Estimate'].dropna()
de['Salary Estimate'] = de['Salary Estimate'].dropna()
ds['Salary Estimate'] = ds['Salary Estimate'].dropna()
print('1',da['Rating'].isnull().values.any())
print('2',de['Rating'].isnull().values.any())
print('3',ds['Rating'].isnull().values.any())
print('4',da['Salary Estimate'].isnull().values.any())
print('5',de['Salary Estimate'].isnull().values.any())
print('6',ds['Salary Estimate'].isnull().values.any())
print('7',da['Rating'].isin([-1.0]).values.any())
print('8',de['Rating'].isin([-1.0]).values.any())
print('9',ds['Rating'].isin([-1.0]).values.any())
 	
#Drop -1.0 in Rating
index_names = da[da['Rating'] == -1.0].index
da.drop(index_names, inplace = True)
index_names1 = de[de['Rating'] == -1.0].index
de.drop(index_names1, inplace = True)
index_names2 = ds[ds['Rating'] == -1.0].index
ds.drop(index_names2, inplace = True)

```
print('7',da['Rating'].isin([-1.0]).values.any())
print('8',de['Rating'].isin([-1.0]).values.any())
print('9',ds['Rating'].isin([-1.0]).values.any())

# Drop Column
da = da.drop(['Revenue','Competitors','Easy Apply'], axis = 1)
de = de.drop(['Revenue','Competitors','Easy Apply'], axis = 1)
ds = ds.drop(['Revenue','Competitors','Easy Apply','index'], axis = 1)

# Reformat data values
# Chenage K = 000
da["Salary Estimate"] = da["Salary Estimate"].str.replace('K','000',regex = False)
de["Salary Estimate"] = de["Salary Estimate"].str.replace('K','000',regex = False)
ds["Salary Estimate"] = ds["Salary Estimate"].str.replace('K','000',regex = False)

# Remove (Glassdoor est.) Split into "Salary_Min_Estimate" & "Salary_Max_Estimate"
da["Salary_Min_Estimate"] = da["Salary Estimate"].str.extract('(\d+)', expand = True)
da["Salary_Max_Estimate"] = da["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)
de["Salary_Min_Estimate"] = de["Salary Estimate"].str.extract('(\d+)', expand = True)
de["Salary_Max_Estimate"] = de["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)
ds["Salary_Min_Estimate"] = ds["Salary Estimate"].str.extract('(\d+)', expand = True)
ds["Salary_Max_Estimate"] = ds["Salary Estimate"].str.extract('\d+\W+\s*(\d+)',expand = True)

da = da.drop(['Salary Estimate'], axis = 1)
de = de.drop(['Salary Estimate'], axis = 1)
ds = ds.drop(['Salary Estimate'], axis = 1)

# Reformat companyname
da["Company_Name"] = da["Company Name"].str.replace('\n\d\.\d','' ,regex = True)
de["Company_Name"] = de["Company Name"].str.replace('\n\d\.\d','' ,regex = True)
ds["Company_Name"] = ds["Company Name"].str.replace('\n\d\.\d','' ,regex = True)
da = da.drop(['Company Name'], axis = 1)
de = de.drop(['Company Name'], axis = 1)
ds = ds.drop(['Company Name'], axis = 1)

# Reset Index
da = da.reset_index()
de = de.reset_index()
#da = da.drop(['index'], axis = 1)

da['Location'] = da['Location'].replace(['Greenwood Village, Arapahoe, CO'], 'Greenwood
Village Arapahoe, CO')
da[['City','State','1']]=da['Location'].str.split(",",expand=True)

da = da.drop(['1'],axis=1)
de[['City','State']]=de['Location'].str.split(",",expand=True)
ds[['City','State']]=ds['Location'].str.split(",",expand=True)
da = da.drop(['Location'],axis=1)
de = de.drop(['Location'],axis=1)
ds = ds.drop(['Location'],axis=1)
```

```
de[['City_Headquarter','State_Headquarter_Country']]=de['Headquarters'].str.split(",",expand=
True)
ds[['City_Headquarter','State_Headquarter_Country','1']]=ds['Headquarters'].str.split(",",expan
d=True)
da[['City_Headquarter','State_Headquarter_Country','1']]=da['Headquarters'].str.split(",",expan
d=True)

da = da.drop(['Headquarters'],axis=1)
de = de.drop(['Headquarters'],axis=1)
ds = ds.drop(['Headquarters'],axis=1)

ds = ds.reset_index()
ds = ds.drop(['index'],axis=1)
da['Size_employee'] = da['Size'].str.extract('\d+\s*\W+\s*(\d+)',expand = True)

ds = ds.drop(['1'],axis=1)
da = da.drop(['1'],axis=1)

da['Size'] = da['Size'].str.replace('+',' to 10000',regex = False)
ds['Size'] = ds['Size'].str.replace('+',' to 10000',regex = False)
de['Size'] = de['Size'].str.replace('+',' to 10000',regex = False)

da['Size_employee'] = da['Size'].str.extract('\d+\s*\w+\s*(\d+)',expand = True)
de['Size_employee'] = de['Size'].str.extract('\d+\s*\w+\s*(\d+)',expand = True)
ds['Size_employee'] = ds['Size'].str.extract('\d+\s*\w+\s*(\d+)',expand = True)

ds = ds.drop(['Size'],axis=1)
da = da.drop(['Size'],axis=1)
de = de.drop(['Size'],axis=1)

da.insert(2,"JobCategory",'DataAnalyst')
de.insert(2,"JobCategory",'DataEngineer')
ds.insert(2,"JobCategory",'DataScientist')
ds[~ds['Job Title'].str.contains("Data Engineer", case=False)]

ds1e = ds.loc[ds['Job Title'].str.contains("Data Engineer", case=False)]
ds1e['JobCategory'] = 'DataEngineer'
ds1e

all_in_one = pd.concat([da, de,ds1e, ds], axis=0)
all_in_one.to_csv('Alltest.csv',index = True)
a = all_in_one.drop_duplicates(subset = ['Job Title', 'Company_Name'],keep =
'first').reset_index(drop = True)

#a['Salary_Max_Estimate']=a['Salary_Max_Estimate'].dropna()
a.dtypes

a['Salary_Min_Estimate'] =  a['Salary_Min_Estimate'].astype(int)
a['Salary_Max_Estimate'] =  a['Salary_Max_Estimate'].fillna(0)
```

```
a['Salary_Max_Estimate'] =  a['Salary_Max_Estimate'].astype(int)
a = a[a['Salary_Min_Estimate'] >= 100]

a = a.to_csv('AllIN.csv',index = False)
```

Split data into each csv files

```
import pandas as pd
import numpy as np

df = pd.read_csv('Allin.csv')
df

df=df.reset_index()

df = df.rename(columns = {'index' : 'job_id'})

df = df.rename(columns = {'Job Title' : 'Job_Title'})
df = df.rename(columns = {'Job Description' : 'Job_Description'})

df.insert(loc=0,column='company_id',value=df['Company_Name'].factorize()[0]+1)
df.insert(loc=0,column='location_id',value=df['City'].factorize()[0]+1)

measurement  = df[['location_id','company_id','job_id',]]
measurement

job =
df[['job_id','Job_Title','Job_Description','JobCategory','Rating','Salary_Min_Estimate','Salary_
Max_Estimate']]
jobs = job.to_csv('jobnew.csv',index=False)
job


company = df[['company_id','Company_Name','Founded','Type of
ownership','Size_employee','Industry','Sector']]

company = company.drop_duplicates(subset = ['company_id'],keep = 'first').reset_index(drop
= True)
company
company = company.to_csv('companynew.csv',index=False)

location = df[['location_id','City','State']]

location = location.drop_duplicates(subset = ['location_id'],keep = 'first').reset_index(drop =
True)

locations = location.to_csv('locationnew.csv',index=False)
```

```
measurement.to_csv('measurementnew.csv',index=False)
```

Loading data into MYSQL from Python

```
import pandas as pd
data = pd.read_csv('measurementnew.csv', index_col=False, delimiter = ',')
data.dtypes
#replace nan to None
data=data.where((pd.notnull(data)), None)
```

```
import mysql.connector
import pandas as pd
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(
        host="localhost",
        user="root",
        password=getpass("Enter password: "),
        auth_plugin='mysql_native_password',
        database='DataJob2'
    ) as connection:
        with connection.cursor() as cursor:
            for i,row in data.iterrows():
                sql = "INSERT INTO DataJob2.measurement VALUES (%s,%s,%s)"
                cursor.execute(sql, tuple(row))
                connection.commit()
            print("Record inserted")
except Error as e:
    print(e)
```

Keyword searching and word counting python coding

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

df = pd.read_csv('Allin.csv')
```

```
df.head()

grouped = df.groupby(df.JobCategory)
da = grouped.get_group("DataAnalyst")
de = grouped.get_group("DataEngineer")
ds = grouped.get_group("DataScientist")

da1 = da[['Job Description']]
de1 = de[['Job Description']]
ds1 = ds[['Job Description']]

da1['Job Description'] = da1['Job Description'].str.lower()
de1['Job Description'] = de1['Job Description'].str.lower()
ds1['Job Description'] = ds1['Job Description'].str.lower()

da1['Job Description'] = da1['Job Description'].str.replace('\n',' ',regex=False)
de1['Job Description'] = de1['Job Description'].str.replace('\n',' ',regex=False)
ds1['Job Description'] = ds1['Job Description'].str.replace('\n',' ',regex=False)

da1['Job Description'] = da1['Job Description'].str.replace('\W',' ',regex=True)
de1['Job Description'] = de1['Job Description'].str.replace('\W',' ',regex=True)
ds1['Job Description'] = ds1['Job Description'].str.replace('\W',' ',regex=True)

import nltk
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
nltk.download('stopwords')

# DA
cat = da1['Job Description'].str.cat(sep=' ')
keyword = ["sql", "python", "c++", "java","tableau","oracle","data",
        "business","management","aws","nosql","mongodb","machine",
        "learning","cloud",'software','database','design','regression']
all_stopwords = stopwords.words('english')
all_stopwords.extend(['work','jobs','etc','us','needs','degree','years','required','including','job','ne
w'])
words = cat.split(' ')
filterword = ' '.join((filter(lambda ele:
ele not in all_stopwords, words)))
kw = ' '.join((filter(lambda ele: ele in keyword, words)))
v = CountVectorizer()
matrix = v.fit_transform([filterword])
counts = pd.DataFrame(matrix.toarray(),columns=v.get_feature_names())
s = counts.sum(axis = 0)
s = s.sort_values(ascending=False)
a = s.to_csv('Da.csv')
v1 = CountVectorizer()
matrix1 = v1.fit_transform([kw])
counts1 = pd.DataFrame(matrix1.toarray(),columns=v1.get_feature_names())
```

```
s1 = counts1.sum(axis = 0)
s1 = s1.sort_values(ascending=False)
a1 = s1.to_csv('Da1.csv')
Text = open(r'datext.txt','w')
String = kw
Text.write(String)
Text.close()
s1

# DS
cat = ds1['Job Description'].str.cat(sep=' ')
keyword = ["sql", "python", "c++", "java","tableau","oracle","data",
          "business","management","aws","nosql","mongodb","machine",
          "learning","cloud",'software','database','design','regression']
all_stopwords = stopwords.words('english')
all_stopwords.extend(['work','jobs','etc','us','needs','degree','years','required','including','job','ne
w'])

words = cat.split(' ')
filterword1 = ' '.join((filter(lambda ele: ele not in all_stopwords, words)))
kw1 = ' '.join((filter(lambda ele: ele in keyword, words)))


v = CountVectorizer()
matrix = v.fit_transform([filterword1])
counts = pd.DataFrame(matrix.toarray(),columns=v.get_feature_names())
s = counts.sum(axis = 0)
s = s.sort_values(ascending=False)
a = s.to_csv('Ds.csv')

v1 = CountVectorizer()
matrix1 = v1.fit_transform([kw1])
counts1 = pd.DataFrame(matrix1.toarray(),columns=v1.get_feature_names())
s1 = counts1.sum(axis = 0)
s1 = s1.sort_values(ascending=False)
a1 = s1.to_csv('Ds1.csv')

Text = open(r'dstext.txt','w')
String = kw1
Text.write(String)
Text.close()

s1

# DE
cat = de1['Job Description'].str.cat(sep=' ')
keyword = ["sql", "python", "c++", "java","tableau","oracle","data",
          "business","management","aws","nosql","mongodb","machine",
          "learning","cloud",'software','database','design','regression']
all_stopwords = stopwords.words('english')
```

```
all_stopwords.extend(['work','jobs','etc','us','needs','degree','years','required','including','job','ne
w'])

words = cat.split(' ')
filterword2 = ' '.join((filter(lambda ele:
ele not in all_stopwords, words)))
kw2 = ' '.join((filter(lambda ele: ele in keyword, words)))


v = CountVectorizer()
matrix = v.fit_transform([filterword2])
counts = pd.DataFrame(matrix.toarray(),columns=v.get_feature_names())
s = counts.sum(axis = 0)
s = s.sort_values(ascending=False)
a = s.to_csv('De.csv')

v1 = CountVectorizer()
matrix1 = v1.fit_transform([kw2])
counts1 = pd.DataFrame(matrix1.toarray(),columns=v1.get_feature_names())
s1 = counts1.sum(axis = 0)
s1 = s1.sort_values(ascending=False)
a1 = s1.to_csv('De1.csv')

Text = open(r'detext.txt','w')
String = kw2
Text.write(String)
Text.close()

s1
```