Assignment 2

Database as a Service

Team 1

Vishal Satam

Manasi Dalvi

DOCKER & Execution Instructions

The required files have been packaged into 2 Docker images and are present on Docker Hub with the image names as vishalsatam1988/wrangleanduploadassignment2 and vishalsatam1988/createdbanduseapi. The manual execution steps are given below.

<u>Note</u> : The docker container runs in the **UTC timezone**.

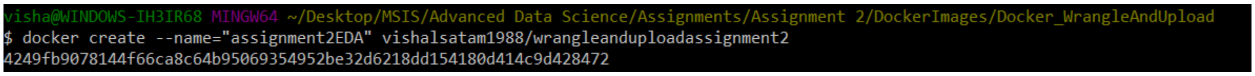
If you encounter any Memory Error issues or segmentation fault. Please increase the RAM of your docker virtual machine to minimum 4 GB.

Docker image - vishalsatam1988/wrangleanduploadassignment2

1. Step 1 : Pull the image
   The docker image is present on the docker hub and is available to pull using the following command

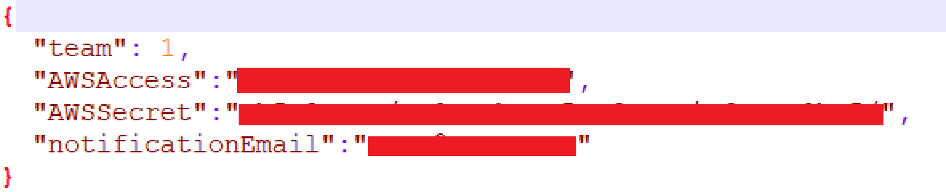
   docker pull vishalsatam1988/wrangleanduploadassignment2

2. Step 2 : Create the container
   docker create --name="assignment2EDA" vishalsatam1988/wrangleanduploadassignment2



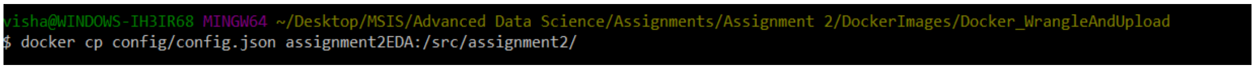
```
visha@WINDOWS-IH3IR68 MINGW64 ~/Desktop/MSIS/Advanced Data Science/Assignments/Assignment 2/DockerImages/Docker_WrangleAndUpload
$ docker create --name="assignment2EDA" vishalsatam1988/wrangleanduploadassignment2
4249fb9078144f66ca8c64b95069354952be32d6218dd154180d414c9d428472
```

3. Copy your config file. (The name of the file has to be **config.json**)
   Sample contents of config.json file



```
{
  "team": 1,
  "AWSAccess":"[redacted]",
  "AWSSecret":"[redacted]",
  "notificationEmail":"[redacted]"
}
```

   docker cp <local file path> <containername>:/src/assignment2/config/
   docker cp <path>/config.json assignment2EDA:/src/assignment2/config/



```
visha@WINDOWS-IH3IR68 MINGW64 ~/Desktop/MSIS/Advanced Data Science/Assignments/Assignment 2/DockerImages/Docker_WrangleAndUpload
$ docker cp config/config.json assignment2EDA:/src/assignment2/
```

4. Start the container
   docker start <containername>
   docker start -i assignment2EDA

5. Commit the container if you want to see logs otherwise invoke the following command to check the jupyter notebooks.

   Password to open jupyter notebook : keras

Command to commit - docker commit assignment2EDA vishalsatam1988/wrangleanduploadassignment2

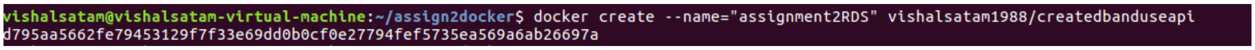
docker run -it -d --name “assignment2EDAjupyter” -p 8888:8888 vishalsatam1988/wrangleanduploadassignment2 /bin/bash -c 'jupyter notebook --no-browser --allow-root --ip=* --NotebookApp.password="$PASSWD" "$@"'

Docker image - vishalsatam1988/createdbanduseapi

1. Step 1 : Pull the image
   The docker image is present on the docker hub and is available to pull using the following command

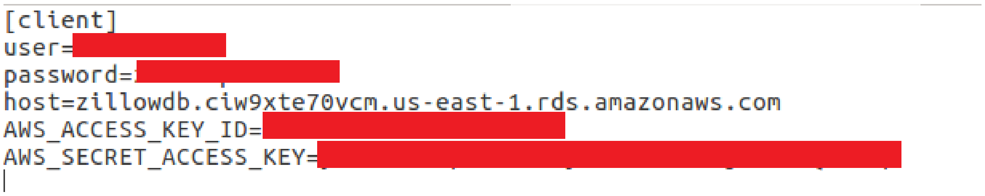
   docker pull vishalsatam1988/createdbanduseapi

2. Step 2 : Create the container
   docker create --name="assignment2RDS" vishalsatam1988/createdbanduseapi



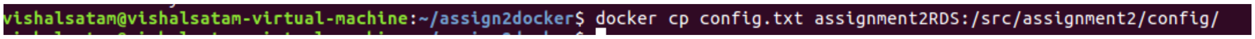
```
vishalsatam@vishalsatam-virtual-machine:~/assign2docker$ docker create --name="assignment2RDS" vishalsatam1988/createdbanduseapi
d795aa5662fe79453129f7f33e69dd0b0cf0e27794fef5735ea569a6ab26697a
```

3. Copy your config file. (The name of the file has to be **config.txt**)
   Sample contents of config.txt file



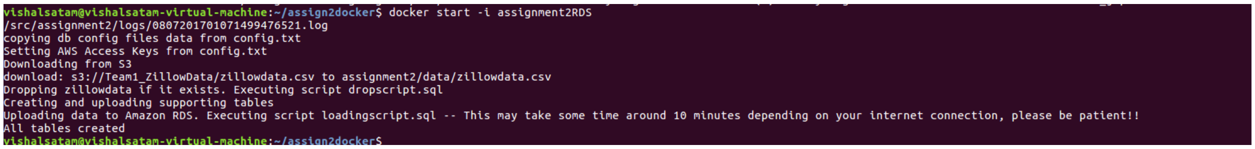
```
[client]
user=[redacted]
password=[redacted]
host=zillowdb.ciw9xte70vcm.us-east-1.rds.amazonaws.com
AWS_ACCESS_KEY_ID=[redacted]
AWS_SECRET_ACCESS_KEY=[redacted]
```

   docker cp <local file path> <containername>:/src/assignment2/config/
   docker cp <path>/config.txt assignment2RDS:/src/assignment2/config/



```
vishalsatam@vishalsatam-virtual-machine:~/assign2docker$ docker cp config.txt assignment2RDS:/src/assignment2/config/
```

4. Start the container
   docker start <containername>
   docker start -i assignment2RDS



```
vishalsatam@vishalsatam-virtual-machine:~/assign2docker$ docker start -i assignment2RDS
/src/assignment2/logs/0807201701071499476521.log
copying db config files data from config.txt
Setting AWS Access Keys from config.txt
Downloading from S3
download: s3://Team1_ZillowData/zillowdata.csv to assignment2/data/zillowdata.csv
Dropping zillowdata if it exists. Executing script dropscript.sql
Creating and uploading supporting tables
Uploading data to Amazon RDS. Executing script loadingscript.sql -- This may take some time around 10 minutes depending on your internet connection, please be patient!!
All tables created
vishalsatam@vishalsatam-virtual-machine:~/assign2docker$
```

5. Commit the container if you want to see logs otherwise invoke the following command to check the jupyter notebooks.

   Command to commit - docker commit assignment2RDS vishalsatam1988/createdbanduseapi

   docker run -it -d --name “assignment2RDS” -p 8888:8888 vishalsatam1988/createdbanduseapi /bin/bash -c 'jupyter notebook --no-browser --allow-root --ip=* --NotebookApp.password="$PASSWD" "$@"'

   Password to open jupyter notebook : keras

Overview :

Zillow has uploaded their properties_2016.csv dataset for a competition and we are using this data to create a Data as a Service application.

The project involves 4 phases.

1. Exploratory Data Analysis on the raw data.
2. Data Wrangling to create clean data
3. Upload data to a Cloud Database (Amazon RDS)
4. Create a REST application provide an API to access this data.

Zillow data consisting of list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data for the year 2016.
Additional description files are provided with their corresponding ID's in the main file. We have created separate tables for the same. The list of tables :
zillowdata - Main table containing the clean data created from the properties file downloaded from kaggle
airconditiontype - Table containing descriptions of Air Condition Types
heatingsystemtypeid - Table containing descriptions of Heating System Types
propertydescid - Table containing Property Type Descriptions

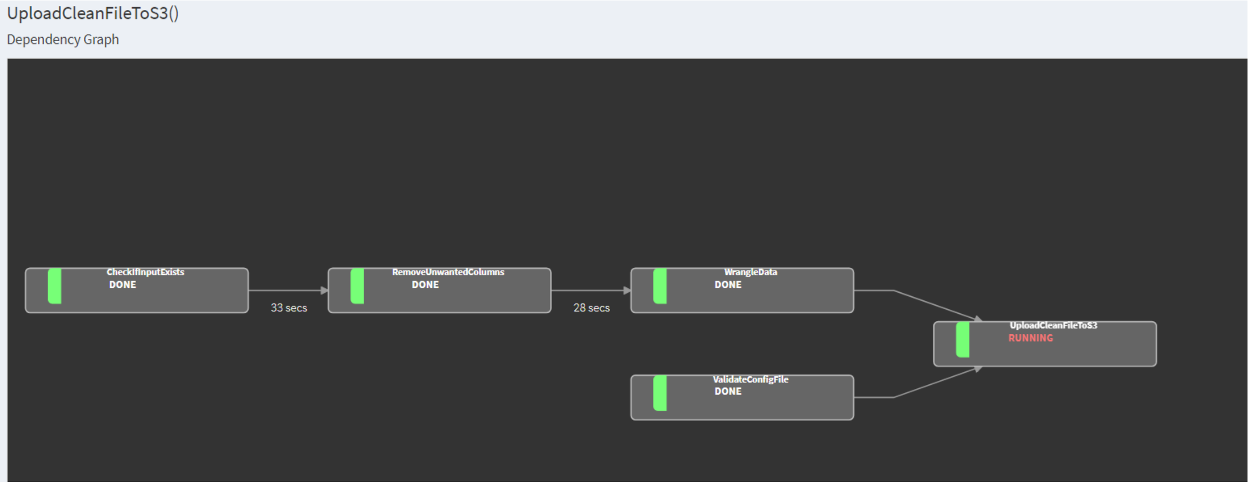
Exploratory Data Analysis

The exploratory data analysis has been covered in detail in the Jupyter notebook ZillowDataEDA.ipynb which is submitted on github as well as included in the Docker image.

We can see that the dataset contains a lot of missing values Few columns are repeated and can be used in place of the other. Mixed Datatypes exist in the data, we have to clean these out.

Data Wrangling

The Data Wrangling is performed using a Data pipeline which has been developed using Luigi as shown in the task dependency graph below.



The pipeline executes in 5 tasks in a sequential manner.

1. **CheckIfInputExists** – This task checks if the raw data file exists in the given path. This is to ensure that we don't run into path errors for the next tasks.
2. **RemoveUnwantedColumns** – This task is responsible for reducing the dataset for processing by the next task. We had seen that this dataset contains many missing values. This task removes all the columns for which 90% or more of the data is missing.
3. **WrangleData –** This is the heart of the wrangling pipeline. This is the main task that performs the data wrangling. The following processing is carried out on the reduced dataset received from the RemoveUnwantedColumns task

   - Replace fireplaceflag by 0 and 1 instead of True and False to make this column numeric. We are also replacing the fireplacecnt to 1 wherever the fireplaceflg was True in order to keep the maintain integrity. After this we remove the fireplaceflag column.
     This is done because we can infer if a fireplace exists or not from the fireplacecnt value. Missing fireplacecnt are kept as Null because almost 80% data is missing. Replacing with anything will affect the integrity of the data.
   - Latitude and Longitude columns have been multiplied by 10^6. We divide the values present in this column by 10^6 to correct these values. Rows for which these values are missing will be removed because we cannot use these values.
   - We group the fields (bathroomcnt, bedroomcnt, roomcnt, heatingorsystemtypeid, buildingqualitytypeid) by the propertylandusetypeid and fips and replace the missing values for the corresponding groups with the means of their groups.
     What this means is that for every missing value, we take the mean value of this column based on the county (fips) and the property type in that county (residential / commercial, etc) and then replace the mean value.

We are doing this because, let's say for instance we have bathroomnt. **A commercial building in a county will not have the bathroomcnt as that of a residential property.** This is why, we decided to take the average based on the property type and the county that the parceled belongs to.

- If there are still missing values for the above columns, we replace them with 0
- Even though the regionidzip contain obfuscated values, we still remove the outliers. As mentioned in the jupyter notebook, there are values that are greater than 99999. These will have to be replaced by some value. Since there are only 421 outlier values, we will replace them by the most common zip which is 96987.

4. **ValidateConfigFile** – This step validates the config file to fetch the AWS credentials and Team which is used by the next task.
5. **UploadCleanFieToS3** – This task collects the cleaned data and uploads it to the S3 bucket.

Upload Dataset to Cloud to create the Database:

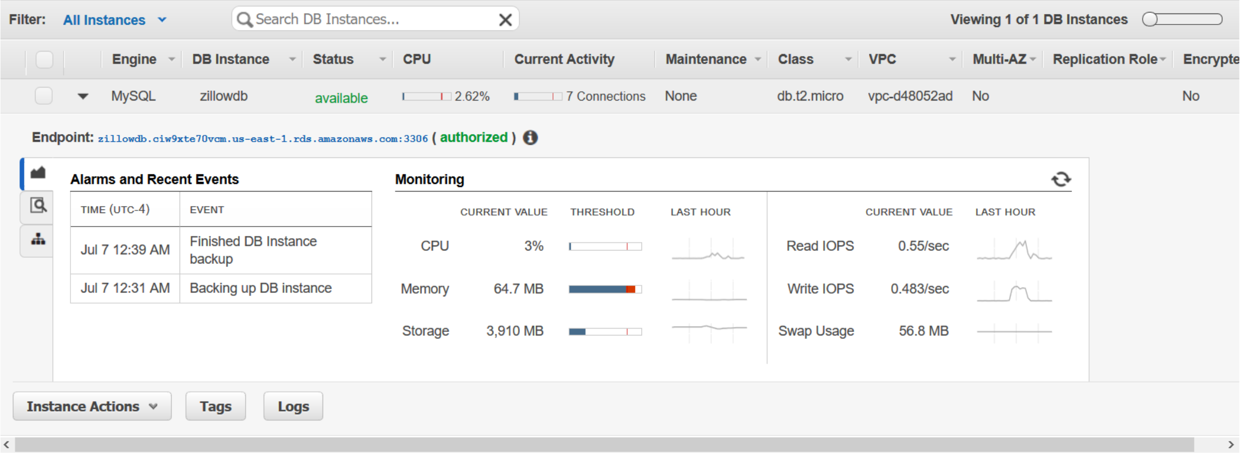
This step uploads our data file which is present on the S3 bucket to the cloud database.

We have been assigned to work on Amazon RDS. A MySQL instance has been created on Amazon RDS and the scripts to create and load the dataset are provided as a docker image.

Creating the database on Amazon RDS:

The database has been created on Amazon RDS as a MySQL instance and is operating on the host URL :

zillowdb.ciw9xte70vcm.us-east-1.rds.amazonaws.com:3306



Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Class	VPC	Multi-AZ	Replication Role	Encrypte
MySQL	zillowdb	available	2.62%	7 Connections	None	db.t2.micro	vpc-d48052ad	No		No

Endpoint: zillowdb.ciw9xte70vcm.us-east-1.rds.amazonaws.com:3306 (authorized)

Alarms and Recent Events

TIME (UTC-4)	EVENT
Jul 7 12:39 AM	Finished DB Instance backup
Jul 7 12:31 AM	Backing up DB instance

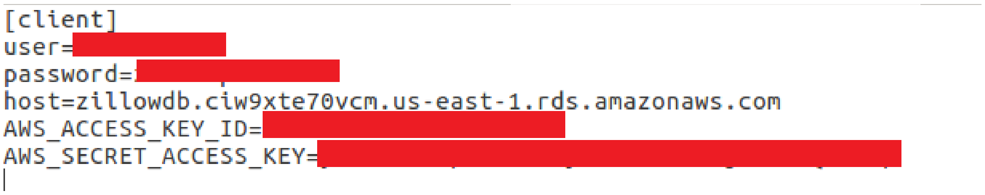
Monitoring

	CURRENT VALUE	THRESHOLD	LAST HOUR
CPU	3%		
Memory	64.7 MB		
Storage	3,910 MB		

	CURRENT VALUE	LAST HOUR
Read IOPS	0.55/sec	
Write IOPS	0.483/sec	
Swap Usage	56.8 MB	

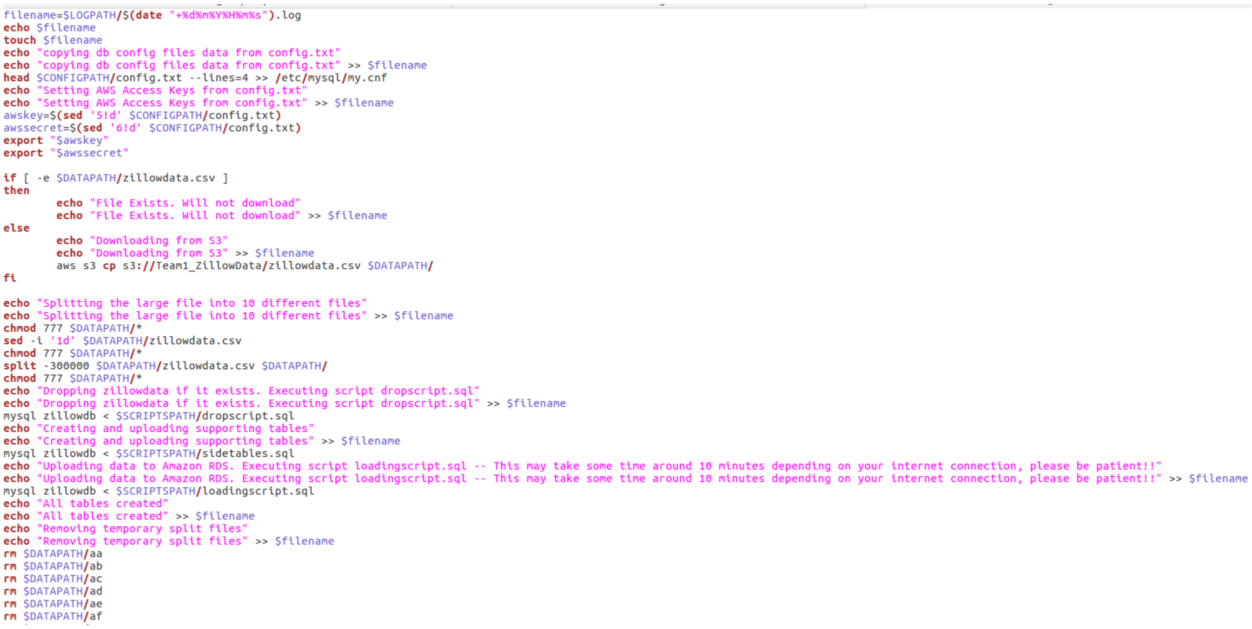
The docker image **vishalsatam1988/createdbanduseapi** contains command line scripts to download the dataset file from s3 using Amazon CLI.

A config.txt file exists in the docker image which provides the credentials to access the database and the amazon s3 bucket. Database credentials have been removed from the docker image intentionally but we can copy our config.txt file into the docker image as explained in the execution steps above.



```
[client]
user=[redacted]
password=[redacted]
host=zillowdb.ciw9xte70vcm.us-east-1.rds.amazonaws.com
AWS_ACCESS_KEY_ID=[redacted]
AWS_SECRET_ACCESS_KEY=[redacted]
```

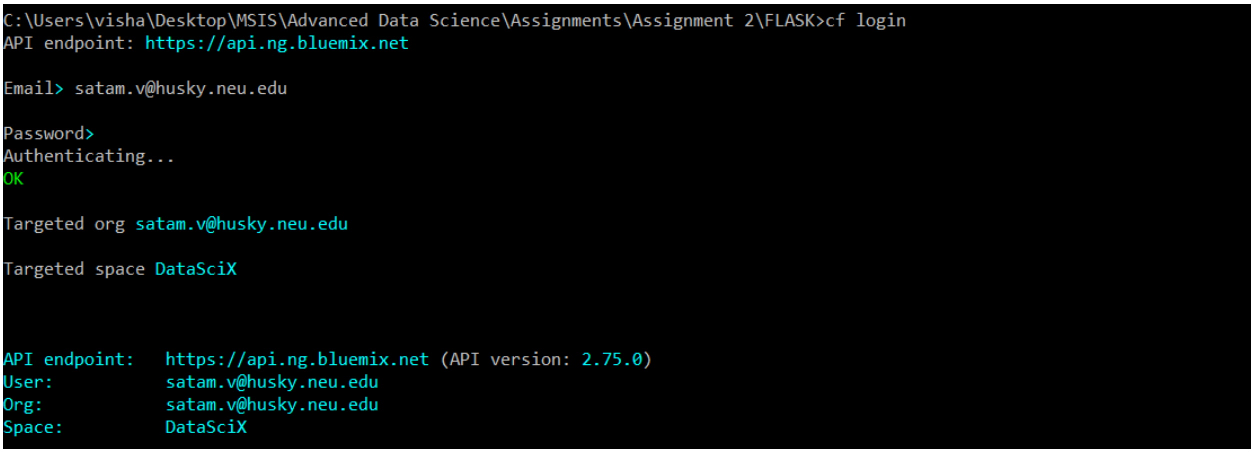
The following shell script file reads the config file and sets environment variables in the docker image which are required to execute the sql files successfully.



```
filename=$LOGPATH/$(date "+%d%m%Y%H%m%s").log
echo $filename
touch $filename
echo "copying db config files data from config.txt"
echo "copying db config files data from config.txt" >> $filename
head $CONFIGPATH/config.txt --lines=4 >> /etc/mysql/my.cnf
echo "Setting AWS Access Keys from config.txt"
echo "Setting AWS Access Keys from config.txt" >> $filename
awskey=$(sed '5!d' $CONFIGPATH/config.txt)
awssecret=$(sed '6!d' $CONFIGPATH/config.txt)
export "$awskey"
export "$awssecret"

if [ -e $DATAPATH/zillowdata.csv ]
then
        echo "File Exists. Will not download"
        echo "File Exists. Will not download" >> $filename
else
        echo "Downloading from S3"
        echo "Downloading from S3" >> $filename
        aws s3 cp s3://Team1_ZillowData/zillowdata.csv $DATAPATH/
fi

echo "Splitting the large file into 10 different files"
echo "Splitting the large file into 10 different files" >> $filename
chmod 777 $DATAPATH/*
sed -i '1d' $DATAPATH/zillowdata.csv
chmod 777 $DATAPATH/*
split -300000 $DATAPATH/zillowdata.csv $DATAPATH/
chmod 777 $DATAPATH/*
echo "Dropping zillowdata if it exists. Executing script dropscript.sql"
echo "Dropping zillowdata if it exists. Executing script dropscript.sql" >> $filename
mysql zillowdb < $SCRIPTSPATH/dropscript.sql
echo "Creating and uploading supporting tables"
echo "Creating and uploading supporting tables" >> $filename
mysql zillowdb < $SCRIPTSPATH/sidetables.sql
echo "Uploading data to Amazon RDS. Executing script loadingscript.sql -- This may take some time around 10 minutes depending on your internet connection, please be patient!!"
echo "Uploading data to Amazon RDS. Executing script loadingscript.sql -- This may take some time around 10 minutes depending on your internet connection, please be patient!!" >> $filename
mysql zillowdb < $SCRIPTSPATH/loadingscript.sql
echo "All tables created"
echo "All tables created" >> $filename
echo "Removing temporary split files"
echo "Removing temporary split files" >> $filename
rm $DATAPATH/aa
rm $DATAPATH/ab
rm $DATAPATH/ac
rm $DATAPATH/ad
rm $DATAPATH/ae
rm $DATAPATH/af
```

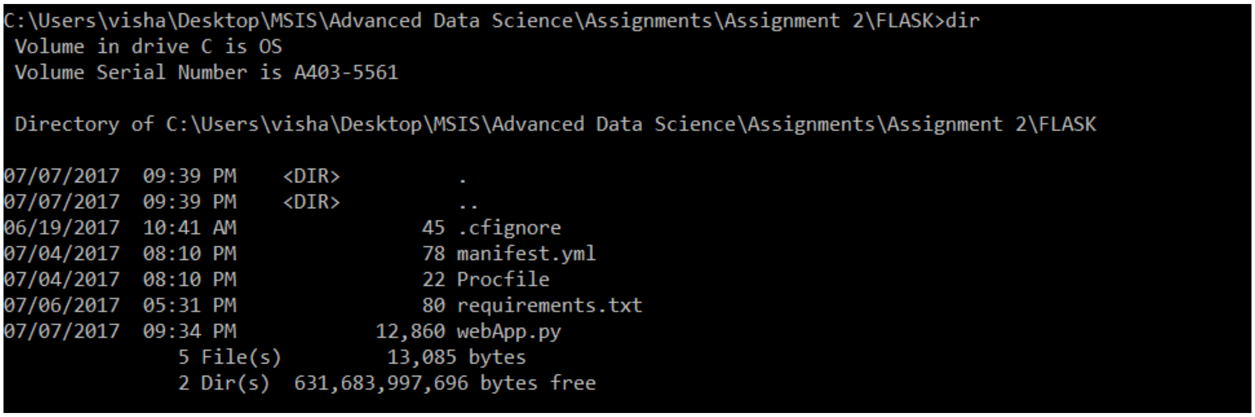
Deploying FLASK Rest API on IBM Bluemix using Cloudfoundry CLI

Logging in



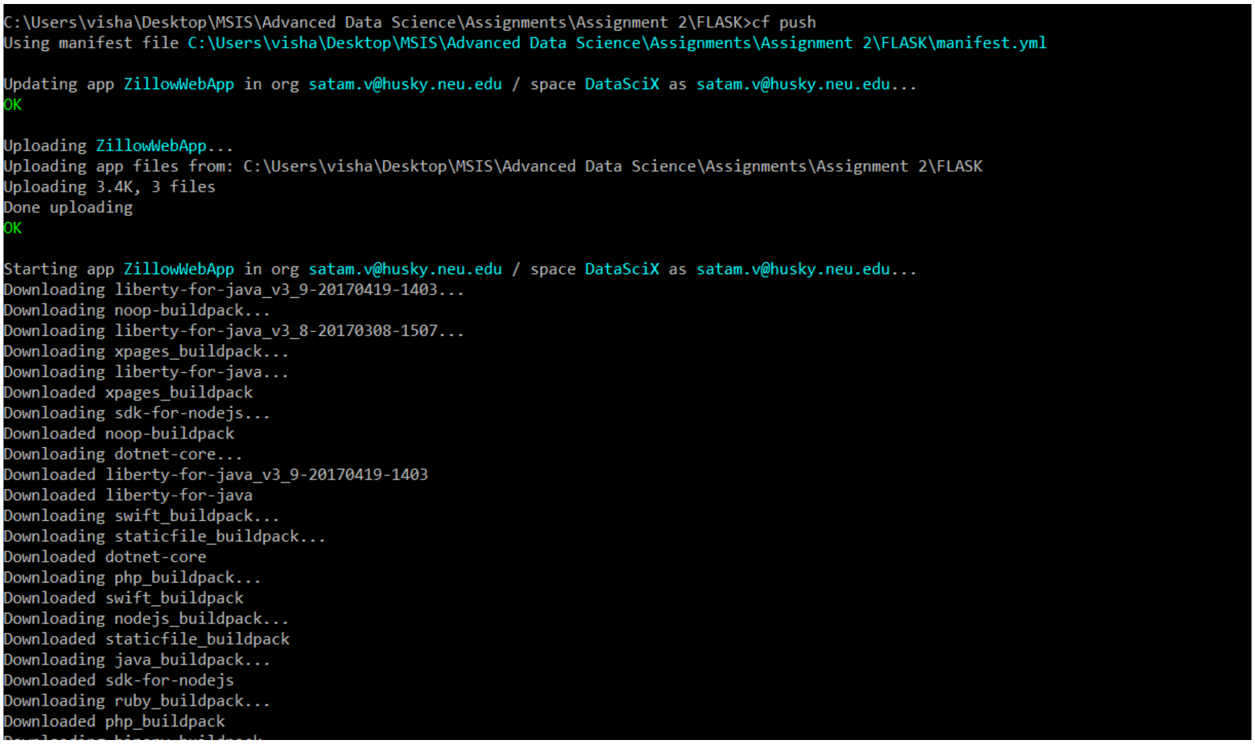
```
C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK>cf login
API endpoint: https://api.ng.bluemix.net

Email> satam.v@husky.neu.edu

Password>
Authenticating...
OK

Targeted org satam.v@husky.neu.edu

Targeted space DataSciX



API endpoint:   https://api.ng.bluemix.net (API version: 2.75.0)
User:           satam.v@husky.neu.edu
Org:            satam.v@husky.neu.edu
Space:          DataSciX
```

Directory Structure

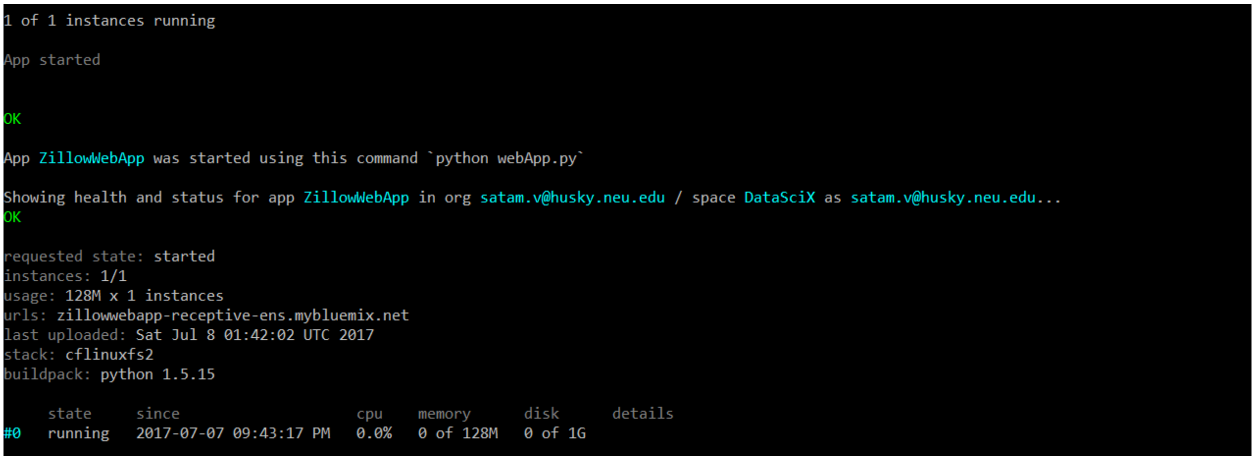


```
C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK>dir
 Volume in drive C is OS
 Volume Serial Number is A403-5561

 Directory of C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK

07/07/2017  09:39 PM    <DIR>          .
07/07/2017  09:39 PM    <DIR>          ..
06/19/2017  10:41 AM                45 .cfignore
07/04/2017  08:10 PM                78 manifest.yml
07/04/2017  08:10 PM                22 Procfile
07/06/2017  05:31 PM                80 requirements.txt
07/07/2017  09:34 PM            12,860 webApp.py
               5 File(s)         13,085 bytes
               2 Dir(s)  631,683,997,696 bytes free
```

Pushing app to bluemix



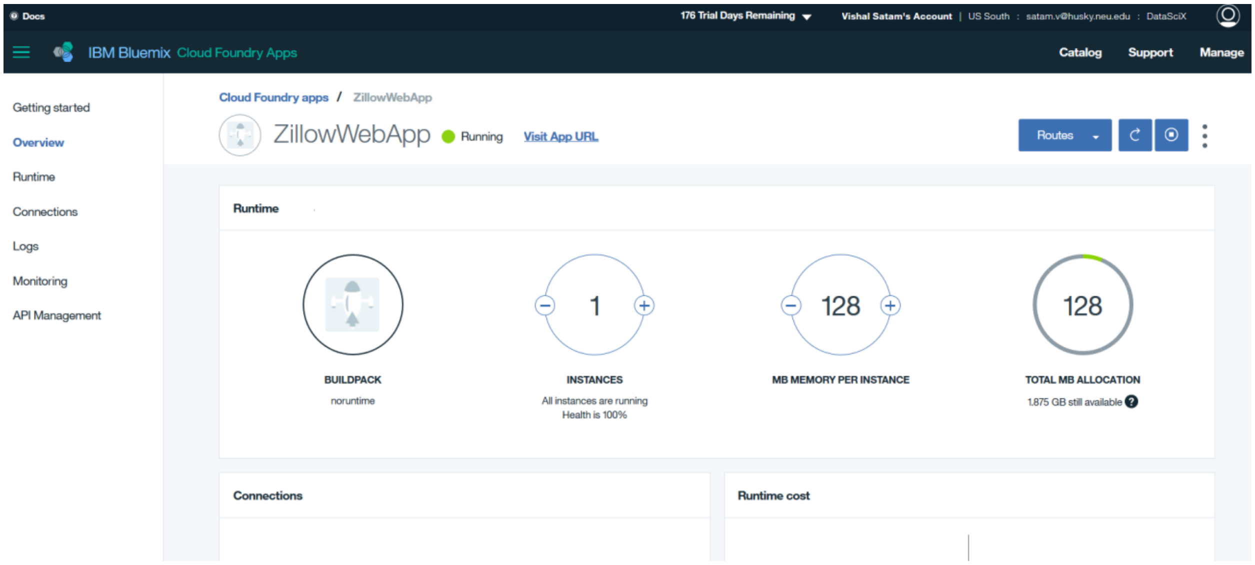
```
C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK>cf push
Using manifest file C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK\manifest.yml

Updating app ZillowWebApp in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
OK

Uploading ZillowWebApp...
Uploading app files from: C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment 2\FLASK
Uploading 3.4K, 3 files
Done uploading
OK

Starting app ZillowWebApp in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
Downloading liberty-for-java_v3_9-20170419-1403...
Downloading noop-buildpack...
Downloading liberty-for-java_v3_8-20170308-1507...
Downloading xpages_buildpack...
Downloading liberty-for-java...
Downloaded xpages_buildpack
Downloading sdk-for-nodejs...
Downloaded noop-buildpack
Downloading dotnet-core...
Downloaded liberty-for-java_v3_9-20170419-1403
Downloaded liberty-for-java
Downloading swift_buildpack...
Downloading staticfile_buildpack...
Downloaded dotnet-core
Downloading php_buildpack...
Downloaded swift_buildpack
Downloading nodejs_buildpack...
Downloaded staticfile_buildpack
Downloading java_buildpack...
Downloaded sdk-for-nodejs
Downloading ruby_buildpack...
Downloaded php_buildpack
Downloading binary_buildpack...
```



```
1 of 1 instances running

App started


OK

App ZillowWebApp was started using this command `python webApp.py`

Showing health and status for app ZillowWebApp in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
OK

requested state: started
instances: 1/1
usage: 128M x 1 instances
urls: zillowwebapp-receptive-ens.mybluemix.net
last uploaded: Sat Jul 8 01:42:02 UTC 2017
stack: cflinuxfs2
buildpack: python 1.5.15

     state     since                    cpu    memory      disk      details
#0   running   2017-07-07 09:43:17 PM   0.0%   0 of 128M   0 of 1G
```

The flask application is deployed and running on the URL :

https://zillowwebapp-receptive-ens.mybluemix.net



BUILDPACK	INSTANCES	MB MEMORY PER INSTANCE	TOTAL MB ALLOCATION
noruntime	1	128	128
	All instances are running Health is 100%		1.875 GB still available

API Details

Search Service API

Search parameter:
http://zillowwebapp-receptive-ens.mybluemix.net/search?zipcode=98698

We can query the api using the search url to perform a search based on the following parameters.

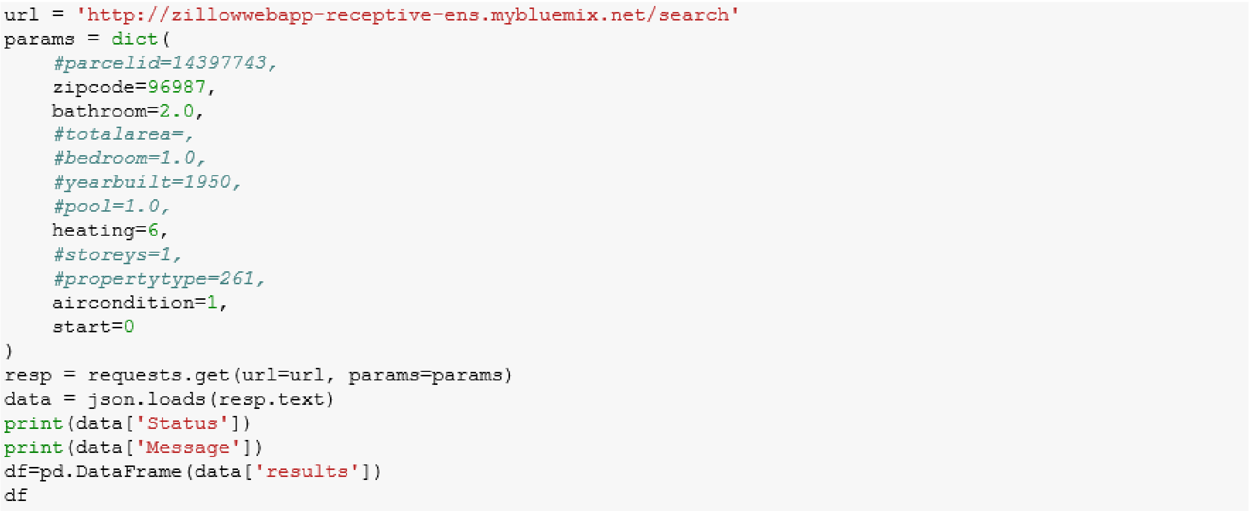
API URL
Parameters For the API (Atleast one parameter is required)

- zipcode (int)
- parcelid (int)
- bathroom (int/float)
- totalarea (float)
- bedroom (int/float)
- yearbuilt (int)
- pool (int)
- heating (int)
- storeys (int)
- propertytype (int)
- aircondition (int)
- start (int) greater than 0 - This is the offset and indicates the index for the next set of results.

Values are returned in json and can be used directly in a dataframe.

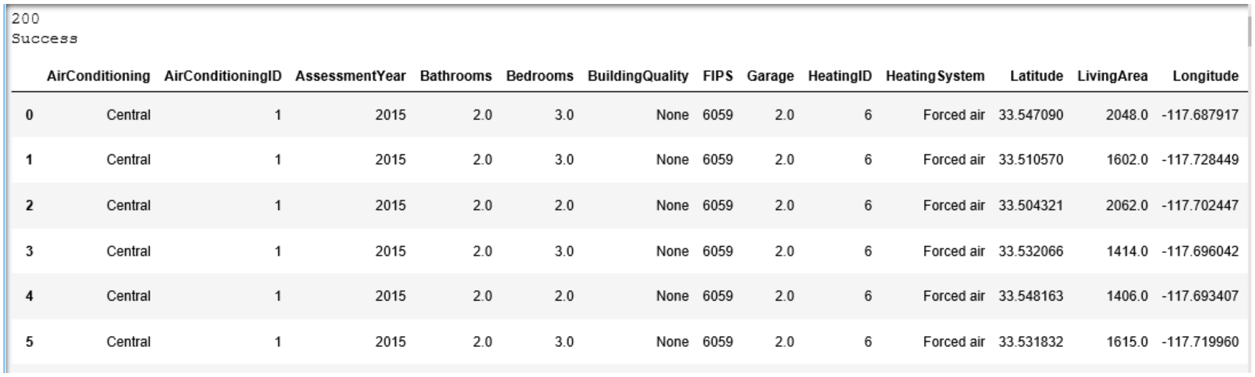
Example 1: query using zipcode, bathroomcount, heating:

The list of search parameters is provided as ‘params’ dictionary and can be modified. At least one parameter is mandatory. Since the dataset size is huge, this search API only returns 100 records at a time. You can request for the next set of 100 records using the start parameter which acts as the offset in the database query.



```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/search'
params = dict(
    #parcelid=14397743,
    zipcode=96987,
    bathroom=2.0,
    #totalarea=,
    #bedroom=1.0,
    #yearbuilt=1950,
    #pool=1.0,
    heating=6,
    #storeys=1,
    #propertytype=261,
    aircondition=1,
    start=0
)
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

The result (show below) of the above search query is returned directly in a dataframe and can be used for further analysis.

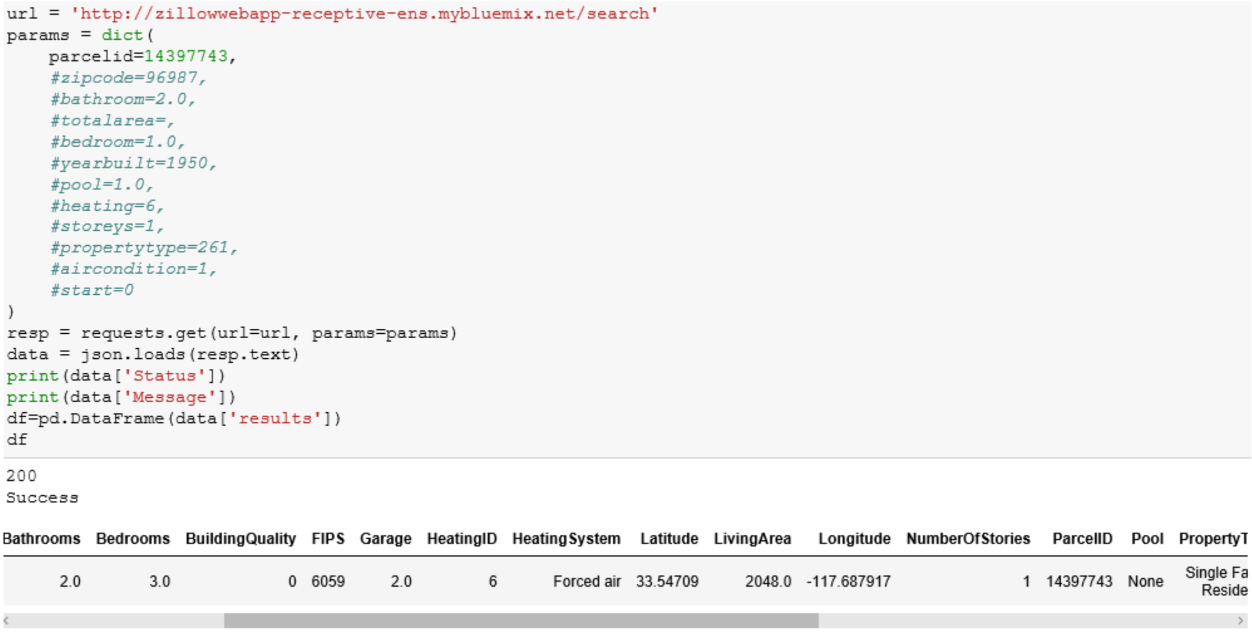


```
200
Success
```

	AirConditioning	AirConditioningID	AssessmentYear	Bathrooms	Bedrooms	BuildingQuality	FIPS	Garage	HeatingID	HeatingSystem	Latitude	LivingArea	Longitude
0	Central	1	2015	2.0	3.0	None	6059	2.0	6	Forced air	33.547090	2048.0	-117.687917
1	Central	1	2015	2.0	3.0	None	6059	2.0	6	Forced air	33.510570	1602.0	-117.728449
2	Central	1	2015	2.0	2.0	None	6059	2.0	6	Forced air	33.504321	2062.0	-117.702447
3	Central	1	2015	2.0	3.0	None	6059	2.0	6	Forced air	33.532066	1414.0	-117.696042
4	Central	1	2015	2.0	2.0	None	6059	2.0	6	Forced air	33.548163	1406.0	-117.693407
5	Central	1	2015	2.0	3.0	None	6059	2.0	6	Forced air	33.531832	1615.0	-117.719960

Example 2: **Query on parceled (property id):**

Return a single row corresponding to the inputted parceled.



```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/search'
params = dict(
    parcelid=14397743,
    #zipcode=96987,
    #bathroom=2.0,
    #totalarea=,
    #bedroom=1.0,
    #yearbuilt=1950,
    #pool=1.0,
    #heating=6,
    #storeys=1,
    #propertytype=261,
    #aircondition=1,
    #start=0
)
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

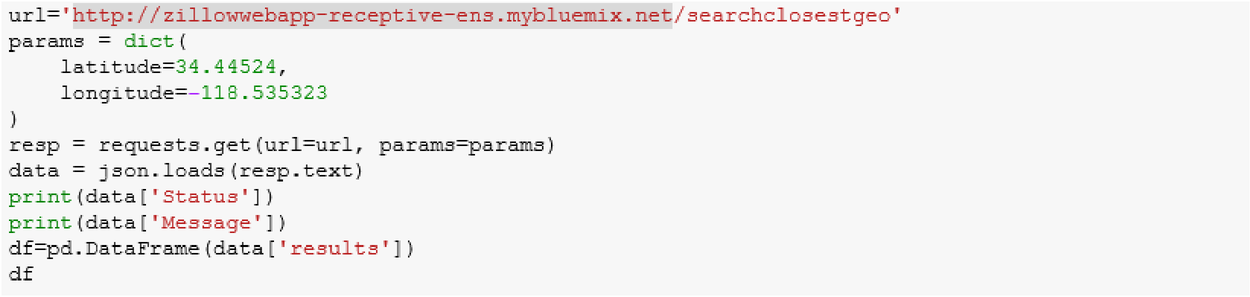
```
200
Success
```

Bathrooms	Bedrooms	BuildingQuality	FIPS	Garage	HeatingID	HeatingSystem	Latitude	LivingArea	Longitude	NumberOfStories	ParcelID	Pool	PropertyT
2.0	3.0	0	6059	2.0	6	Forced air	33.54709	2048.0	-117.687917	1	14397743	None	Single Fa Reside

Geospatial search for ten closest homes

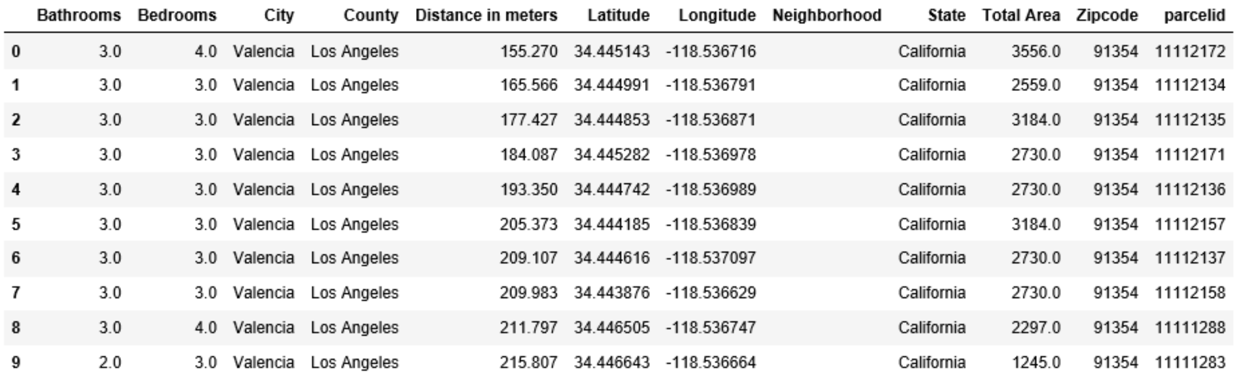
To display 10 closest properties from a given location requires latitude and longitude of the location. The required parameters are given as a dictionary and can be modified for searches.

Following image shows geo search for latitude = 34.44524 and longitude -118.535323



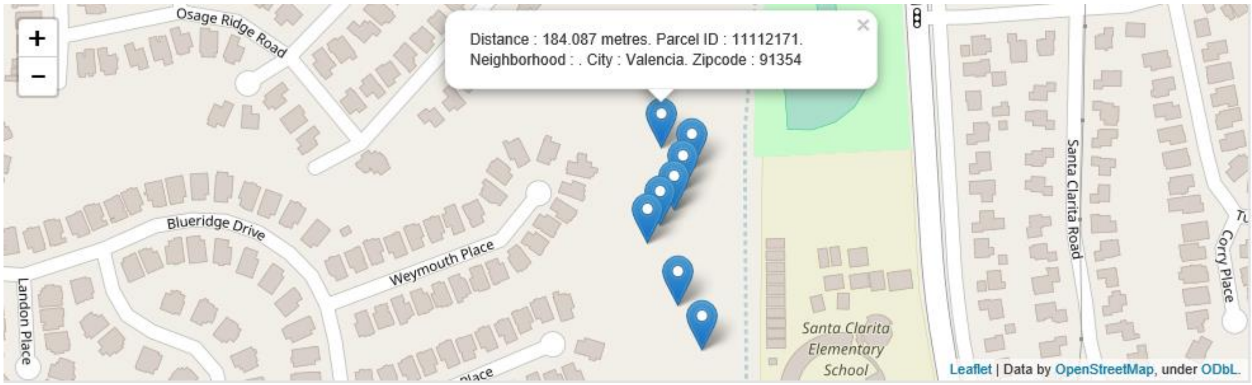
```
url='http://zillowwebapp-receptive-ens.mybluemix.net/searchclosestgeo'
params = dict(
    latitude=34.44524,
    longitude=-118.535323
)
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

The following image shows the ten closest properties for the above given location sorted by distance from the entered location.



	Bathrooms	Bedrooms	City	County	Distance in meters	Latitude	Longitude	Neighborhood	State	Total Area	Zipcode	parcelid
0	3.0	4.0	Valencia	Los Angeles	155.270	34.445143	-118.536716		California	3556.0	91354	11112172
1	3.0	3.0	Valencia	Los Angeles	165.566	34.444991	-118.536791		California	2559.0	91354	11112134
2	3.0	3.0	Valencia	Los Angeles	177.427	34.444853	-118.536871		California	3184.0	91354	11112135
3	3.0	3.0	Valencia	Los Angeles	184.087	34.445282	-118.536978		California	2730.0	91354	11112171
4	3.0	3.0	Valencia	Los Angeles	193.350	34.444742	-118.536989		California	2730.0	91354	11112136
5	3.0	3.0	Valencia	Los Angeles	205.373	34.444185	-118.536839		California	3184.0	91354	11112157
6	3.0	3.0	Valencia	Los Angeles	209.107	34.444616	-118.537097		California	2730.0	91354	11112137
7	3.0	3.0	Valencia	Los Angeles	209.983	34.443876	-118.536629		California	2730.0	91354	11112158
8	3.0	4.0	Valencia	Los Angeles	211.797	34.446505	-118.536747		California	2297.0	91354	11111288
9	2.0	3.0	Valencia	Los Angeles	215.807	34.446643	-118.536664		California	1245.0	91354	11111283

The result can be seen on an interactive leaflet map. Python Folium and Vincent/Vega libraries are used for visualizations.



Distance : 184.087 metres. Parcel ID : 11112171. Neighborhood : . City : Valencia. Zipcode : 91354

Map plotting the ten properties closest from a given location

Analytics Services

The columns AirconditiontypeID, PropertyLandDescriptionTypeID,HeatingSystemTypeID have corresponding description columns which are uploaded as additional tables.

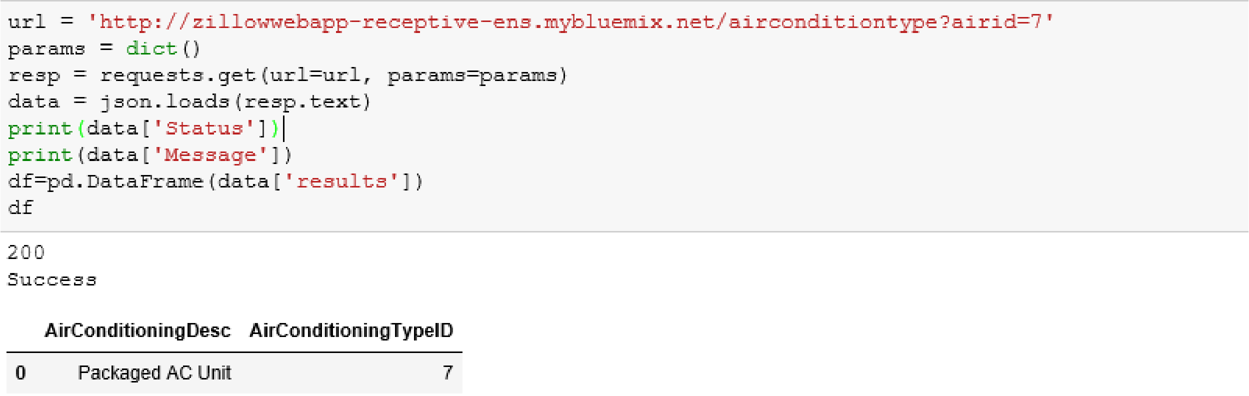
Types and Description Api :

Provides search results for the mentioned features with its description from the corresponding tables. It can be queried with a single ID parameter for specific search or can return the entire set of values for all available IDs.

Example: Air-condition system search

Search by ID returning a single result:
Search parameters : http://zillowwebapp-receptive-ens.mybluemix.net/airconditiontype?airid=7

We can query for a single result for specific type searches based on ID.



```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/airconditiontype?airid=7'
params = dict()
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

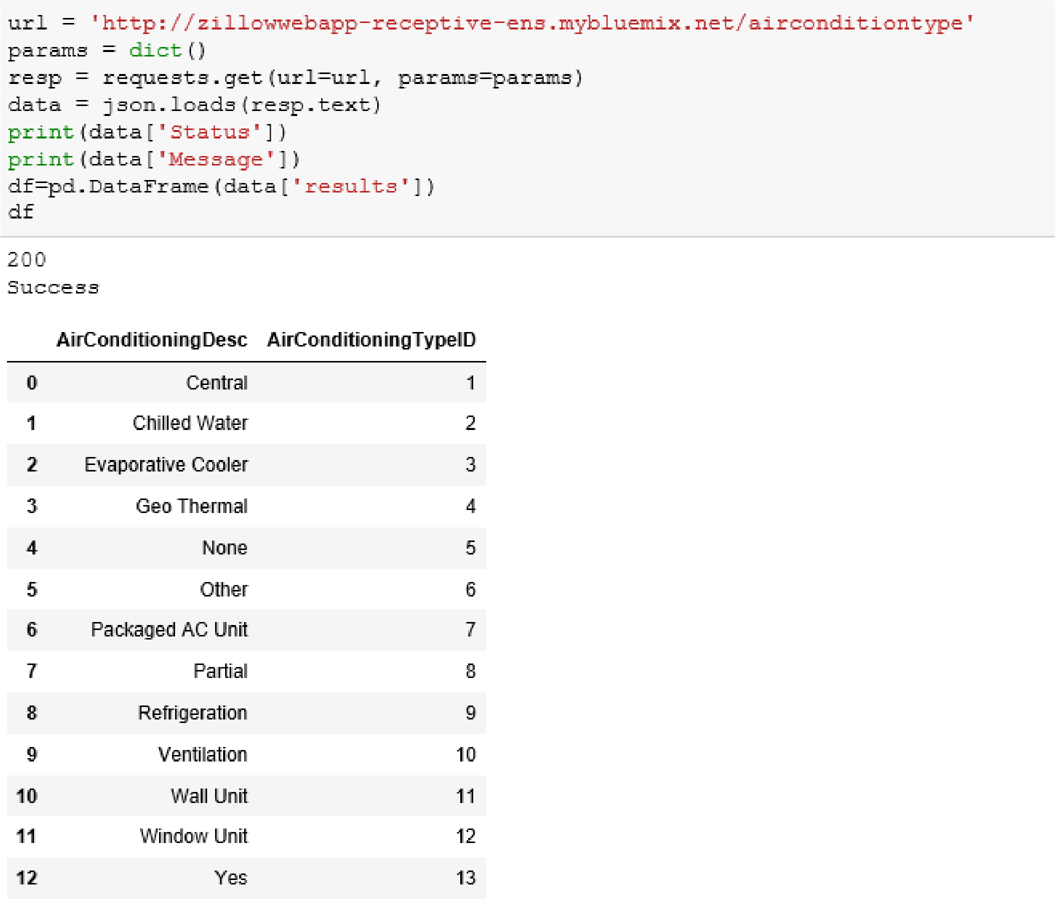
```
200
Success
```

	AirConditioningDesc	AirConditioningTypeID
0	Packaged AC Unit	7

Query for single value

Search by all values:

Search parameters: http://zillowwebapp-receptive-ens.mybluemix.net/airconditiontype
The api returns all air-condition id's with its respective description.



```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/airconditiontype'
params = dict()
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

```
200
Success
```

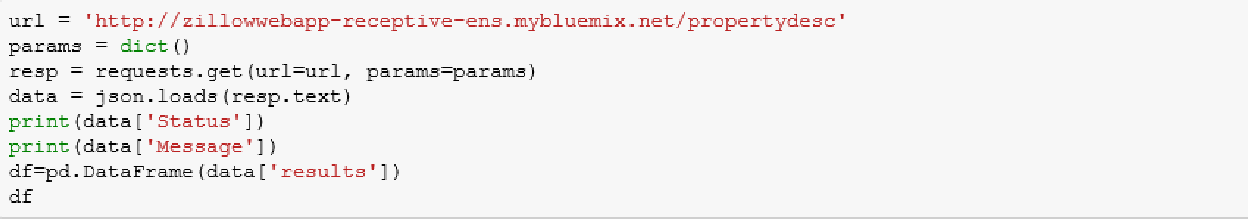
	AirConditioningDesc	AirConditioningTypeID
0	Central	1
1	Chilled Water	2
2	Evaporative Cooler	3
3	Geo Thermal	4
4	None	5
5	Other	6
6	Packaged AC Unit	7
7	Partial	8
8	Refrigeration	9
9	Ventilation	10
10	Wall Unit	11
11	Window Unit	12
12	Yes	13

Query for searching all types or air-condition systems and result

Example: Property type description search:

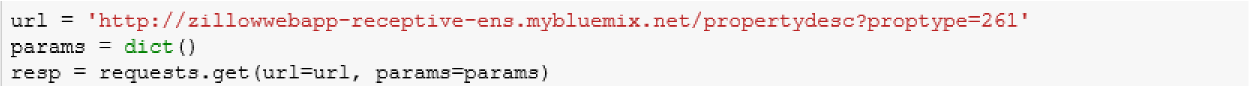
Search parameter: http://zillowwebapp-receptive-ens.mybluemix.net/propertydesc

We can query for types of properties available and get the description and its associated ID
Following query returns all values as no specific id is provided.



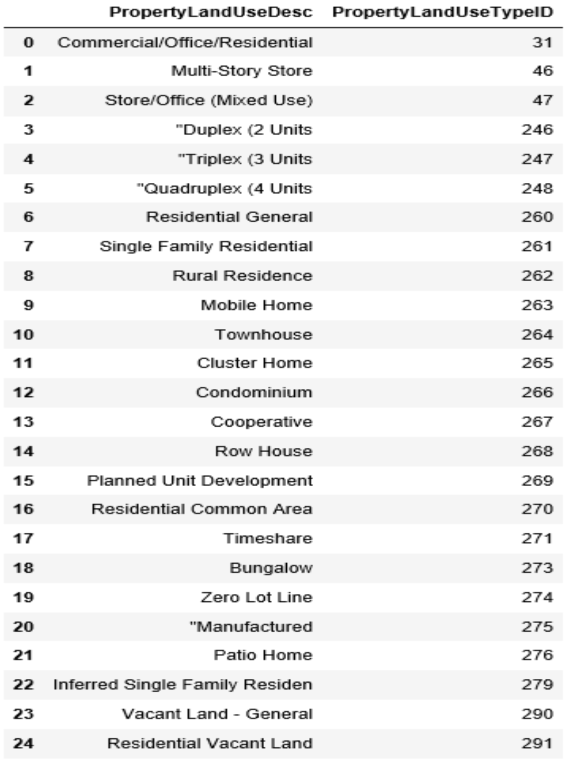
```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/propertydesc'
params = dict()
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

For specific searches provide ID parameter as follows:



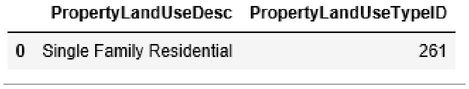
```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/propertydesc?proptype=261'
params = dict()
resp = requests.get(url=url, params=params)
```

The results:



	PropertyLandUseDesc	PropertyLandUseTypeID
0	Commercial/Office/Residential	31
1	Multi-Story Store	46
2	Store/Office (Mixed Use)	47
3	"Duplex (2 Units	246
4	"Triplex (3 Units	247
5	"Quadruplex (4 Units	248
6	Residential General	260
7	Single Family Residential	261
8	Rural Residence	262
9	Mobile Home	263
10	Townhouse	264
11	Cluster Home	265
12	Condominium	266
13	Cooperative	267
14	Row House	268
15	Planned Unit Development	269
16	Residential Common Area	270
17	Timeshare	271
18	Bungalow	273
19	Zero Lot Line	274
20	"Manufactured	275
21	Patio Home	276
22	Inferred Single Family Residen	279
23	Vacant Land - General	290
24	Residential Vacant Land	291

Without Id parameter



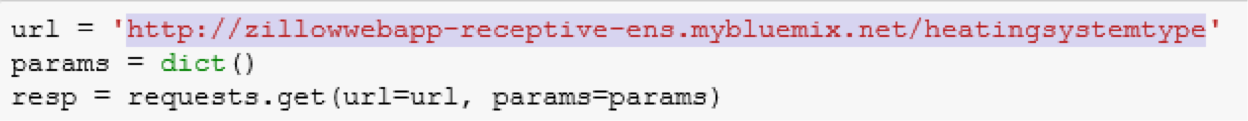
	PropertyLandUseDesc	PropertyLandUseTypeID
0	Single Family Residential	261

Single result with id parameter

Example: Heating System Description

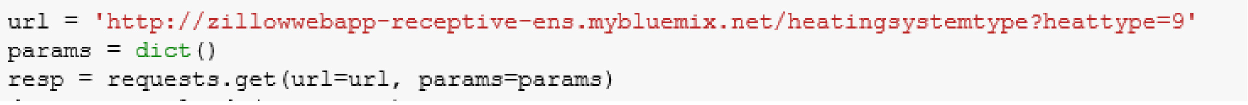
Search parameter: http://zillowwebapp-receptive-ens.mybluemix.net/heatingsystemtype
We can query for the type of heating systems installed by providing the type id or accessing all the types of heating systems available.

For all types:



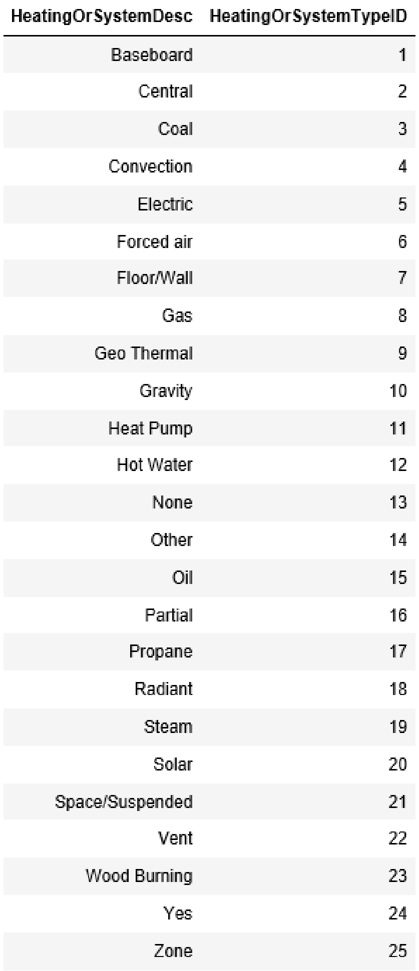
```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/heatingsystemtype'
params = dict()
resp = requests.get(url=url, params=params)
```

For specific type:



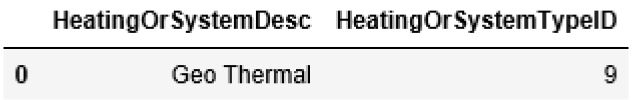
```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/heatingsystemtype?heattype=9'
params = dict()
resp = requests.get(url=url, params=params)
```

Results:



HeatingOrSystemDesc	HeatingOrSystemTypeID
Baseboard	1
Central	2
Coal	3
Convection	4
Electric	5
Forced air	6
Floor/Wall	7
Gas	8
Geo Thermal	9
Gravity	10
Heat Pump	11
Hot Water	12
None	13
Other	14
Oil	15
Partial	16
Propane	17
Radiant	18
Steam	19
Solar	20
Space/Suspended	21
Vent	22
Wood Burning	23
Yes	24
Zone	25

All available types



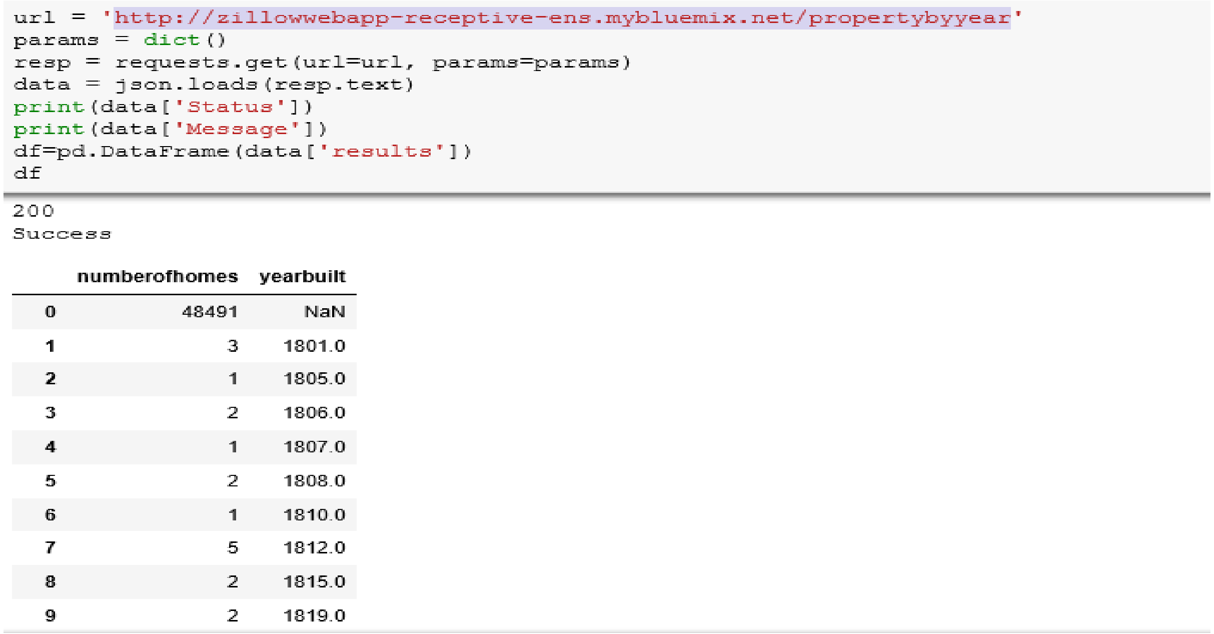
	HeatingOrSystemDesc	HeatingOrSystemTypeID
0	Geo Thermal	9

For specific type ID

Properties built by year

Search parameters:
http://zillowwebapp-receptive-ens.mybluemix.net/propertybyyear

Returns the count of all properties grouped by the year they were built in. To specifically search for properties built, the Search API can be used.

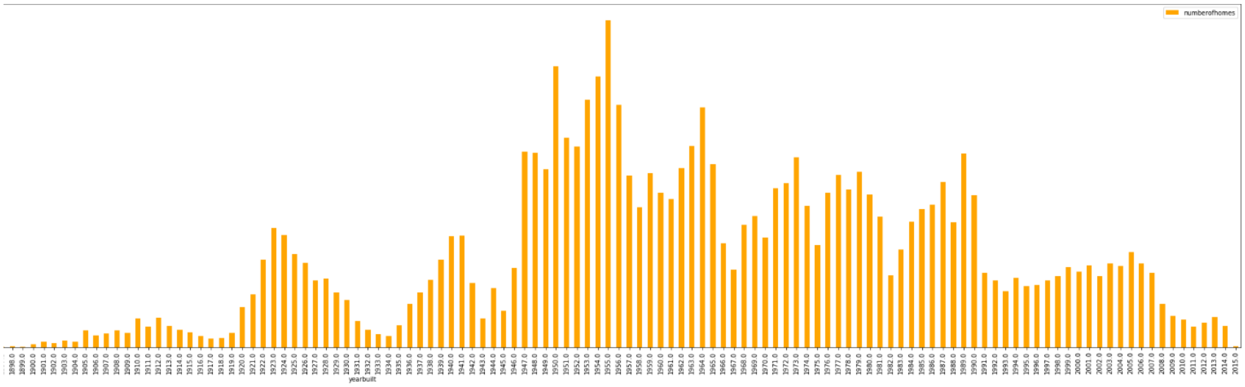


```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/propertybyyear'
params = dict()
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

```
200
Success
```

	numberofhomes	yearbuilt
0	48491	NaN
1	3	1801.0
2	1	1805.0
3	2	1806.0
4	1	1807.0
5	2	1808.0
6	1	1810.0
7	5	1812.0
8	2	1815.0
9	2	1819.0

Provides a distribution of the properties by the year they were built in. The search api mentioned above provides search by year facility. The distribution shows most constructions are between the years 1950 to 1965, with maximum in 1955.

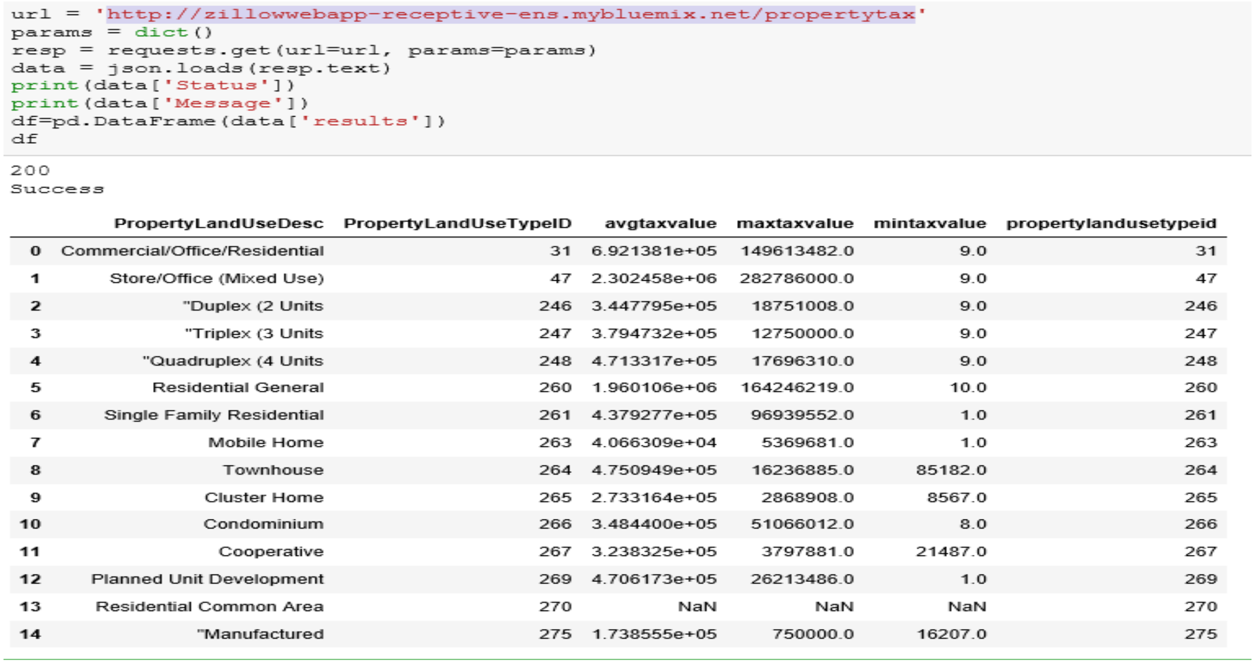


Property distribution by year built

Tax value for type of properties

Search parameters:
http://zillowwebapp-receptive-ens.mybluemix.net/propertytax

Provides the tax value associated by the type of property available. It returns the minimum tax, maximum tax and average tax amount based on the property type.

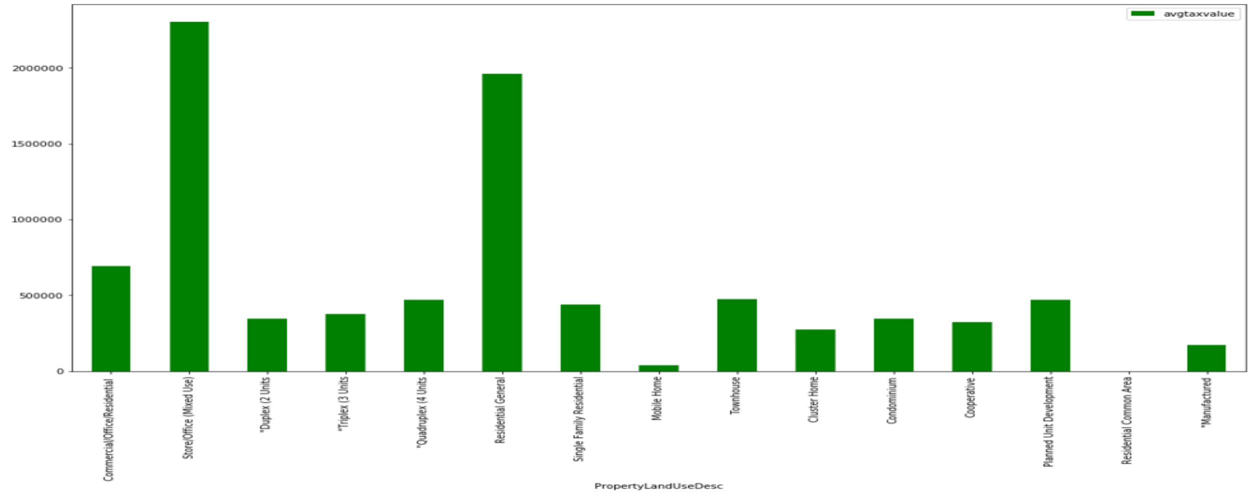


```
url = 'http://zillowwebapp-receptive-ens.mybluemix.net/propertytax'
params = dict()
resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print(data['Status'])
print(data['Message'])
df=pd.DataFrame(data['results'])
df
```

```
200
Success
```

	PropertyLandUseDesc	PropertyLandUseTypeID	avgtaxvalue	maxtaxvalue	mintaxvalue	propertylandusetypeid
0	Commercial/Office/Residential	31	6.921381e+05	149613482.0	9.0	31
1	Store/Office (Mixed Use)	47	2.302458e+06	282786000.0	9.0	47
2	"Duplex (2 Units	246	3.447795e+05	18751008.0	9.0	246
3	"Triplex (3 Units	247	3.794732e+05	12750000.0	9.0	247
4	"Quadruplex (4 Units	248	4.713317e+05	17696310.0	9.0	248
5	Residential General	260	1.960106e+06	164246219.0	10.0	260
6	Single Family Residential	261	4.379277e+05	96939552.0	1.0	261
7	Mobile Home	263	4.066309e+04	5369681.0	1.0	263
8	Townhouse	264	4.750949e+05	16236885.0	85182.0	264
9	Cluster Home	265	2.733164e+05	2868908.0	8567.0	265
10	Condominium	266	3.484400e+05	51066012.0	8.0	266
11	Cooperative	267	3.238325e+05	3797881.0	21487.0	267
12	Planned Unit Development	269	4.706173e+05	26213486.0	1.0	269
13	Residential Common Area	270	NaN	NaN	NaN	270
14	"Manufactured	275	1.738555e+05	750000.0	16207.0	275

Query with results for tax value for every property type



Tax amount by type of property