

**ADVANCES IN DATA SCIENCE/ARCHITECTURE**

**ASSIGNMENT-2**

**REPORT**

**Submitted By:**

**MEGHA SINGH**

**SNEHA MALSHETTI**

Table of Contents

1. Overview3

Assignment Requirements and detailed overview4

1. Objective Flow of the Project5

Flowchart of Processing of Data Analysis6

1. Data Ingestion ,Exploratory Data Analysis and Data Wrangling11

Steps for Data cleansing and EDA15

1. MongoDB on EC2 instance19

Creating the MongoDb Database 20

Creating MongoDB instance on AWS EC2………………………………………………………………………………………25

1. Create the REST API to serve the Data24

Using Flask To connect with MongoDB 24

Using Flask to create REST API on Cloud26

1. Geospatial Search through the REST API Data24
2. Citations4

**Overview**

1. **Objective Flow of the Project**
2. Language Used: Python
3. Data Ingestion, EDA, Data Wrangling: Jupyter NoteBook, Boto3 connection, Amazon S3 bucket
4. Cloud: Amazon EC3 instance with MongoDB.
5. REST API for Daas: Flask with MongoDB
6. Database as service used: MongoDB (NOSQL)
   1. **Flowchart of Processing of Data Analysis**

1. DATA INGESTION, EDA and Data Wrangling

**DATA INGESTION**

Descriptive Steps and script for DATA INGESTION

1. Download the data from Zillow. (<https://www.kaggle.com/c/zillow-prize-1>)

Use the Jupyter Notebook for using the config

{ "AWSAccess":"access key",

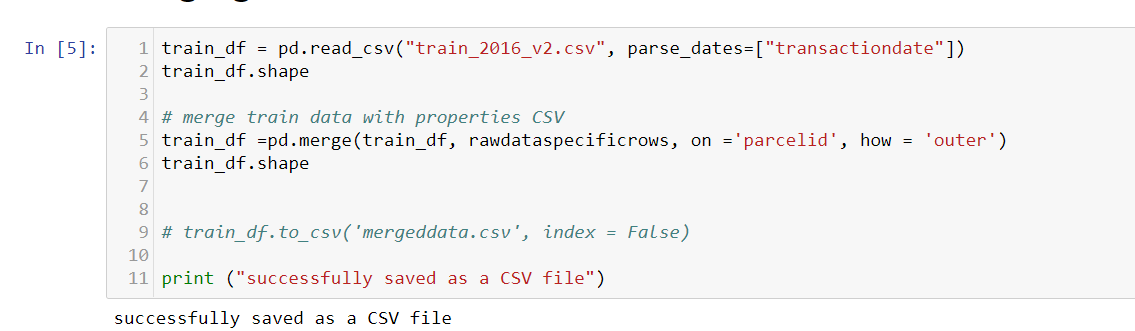
"AWSSecret":"secret key"} to upload the cleaned data on S3 bucket.

**2. Create an IPYB notebook and Conduct an in-depth EDA**

**Exploratory Data Analysis and DATA WRANGLING**

**Filling the Missing data**

1. First merge the train Csv and properties csv to get all the available data and covert it into a dataframe using pandas



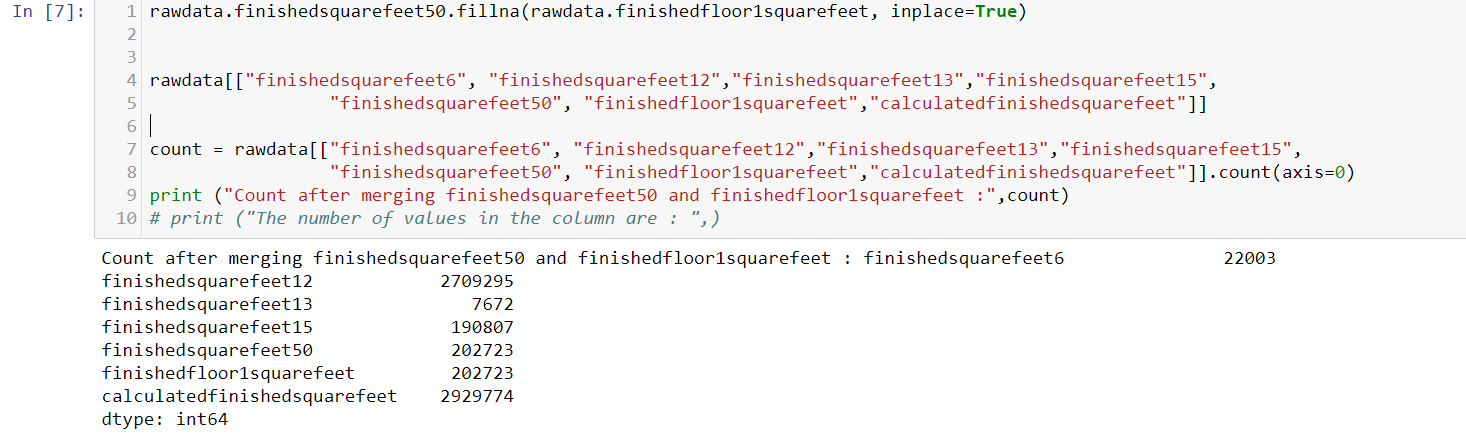
1. From the data Set we have made the analysis

Column : finishedsquarefeet50 and finishedfloor1squarefeet have all the same values so we have merged column into one to get the single column with complete values of the finished squarefeet

Similarly,

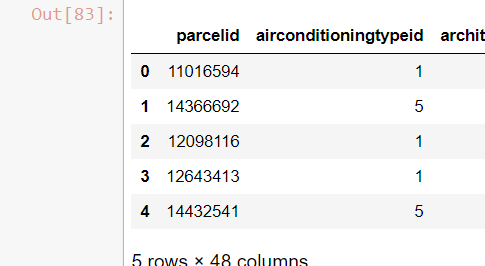
Columns finishedsquarefeet6","finishedsquarefeet12","finishedsquarefeet13",

"finishedsquarefeet15", "finishedsquarefeet50", "finishedfloor1squarefeet","calculatedfinishedsquarefeet" All have the same values after merging them we get the one column value.

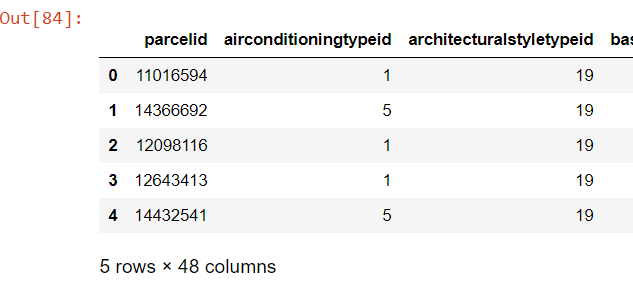


1. Replacing “**airconditioningtypeid”** to 5 default type for none

As the data dictionary has explained the apartments with no AC or other: having column valus as 5. So we replace all the missing values with 5.

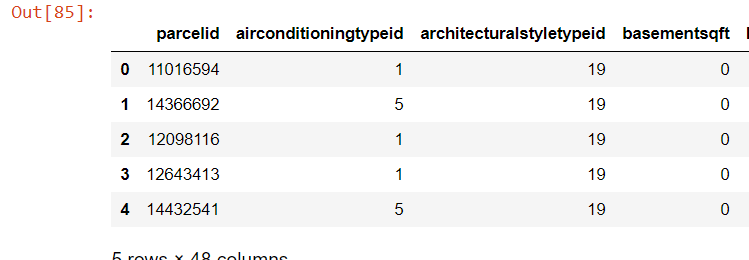


1. Similarly Replacing “**architecturalstyletypeid**” to 19 default type for other for filling the missing or blank data.

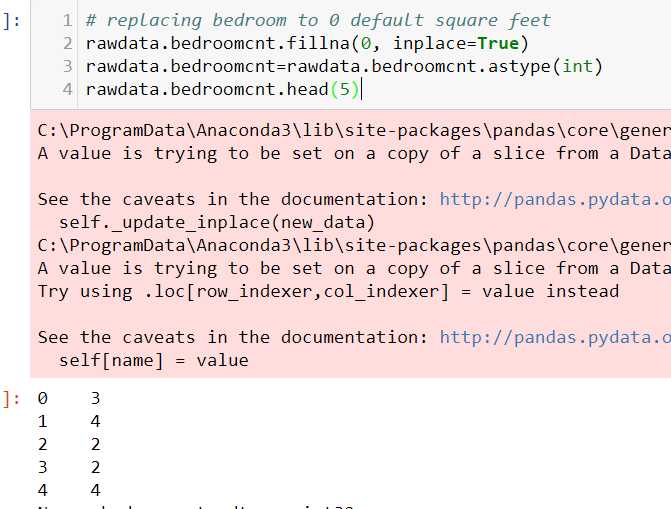


1. Replacing basementsqft to 0 default square feet

Which shows the apartment is not having the basement and converting the column value to integer type



1. Replacing the bedroom missing count with 0 as no data is available for the same.

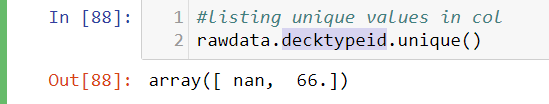


1. Replacing “**buildingclasstypeid”** missing values to 5: default other

The building default type is the structure type of the building so if the data is missing we replace it with other value.

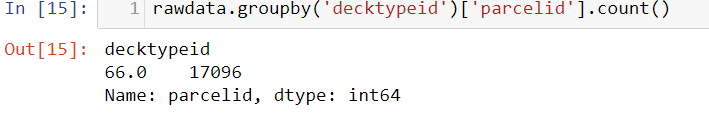
1. Now to check id the **decktypeid** is having any unique values of count for the same

Listing unique values in col



The column is giving the information about the deck availibilty, so we can only put a flag as TRUE or FALSE in the column.

So , We can remove the integer value as True and rest black or 0 as False



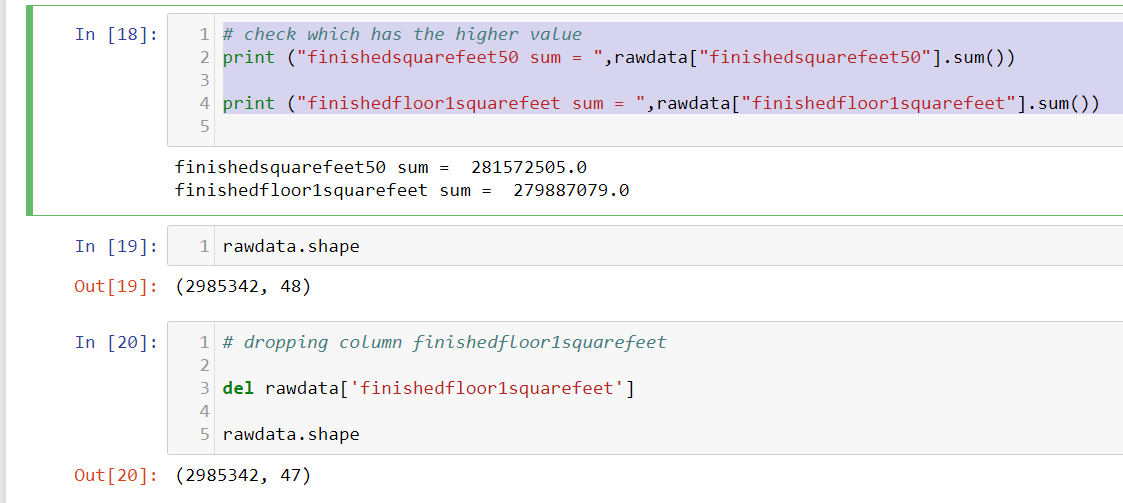
1. We want to calculate the available square feet of the property

But for the same we have still two columns So We compare the higher value from both column as that will be the final or finished available square feet including the utilities such as deck, pool and other facilities:

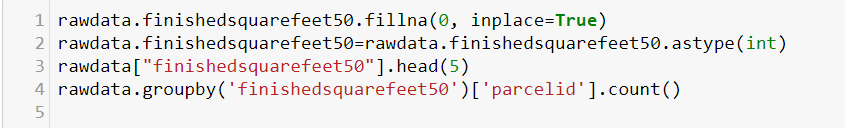
So after comparing Finishedsquarefet50 is having the higher value as it inclused the **deckquarefeet** and **poolsizesquarefeet** as well.

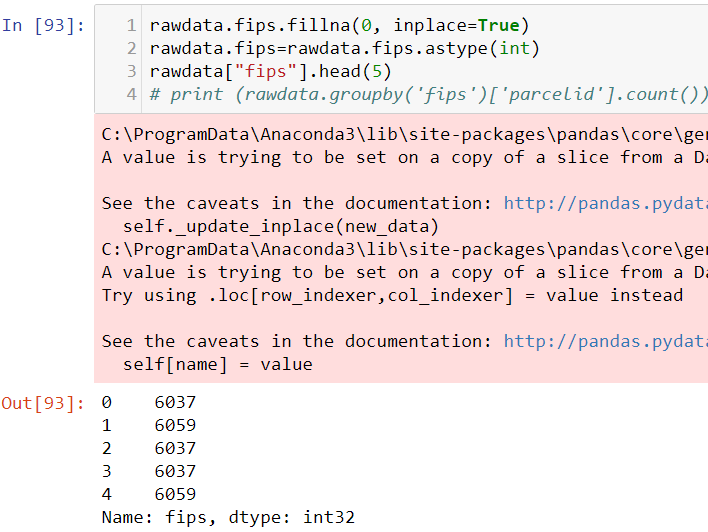
So further analysis we will use this column for the and to fill the missing column value we can merge the finished floorsquarefeet column.

With this we will get the square feet area of properties. And than delete the column

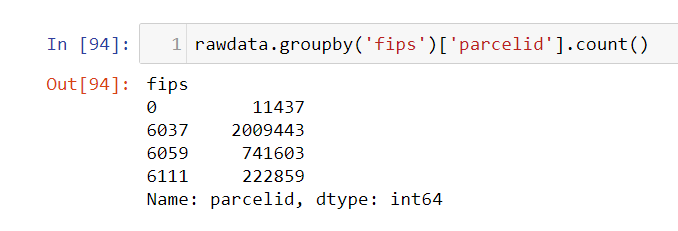


1. Converting the Squarefeetvalue of the properties into integer type

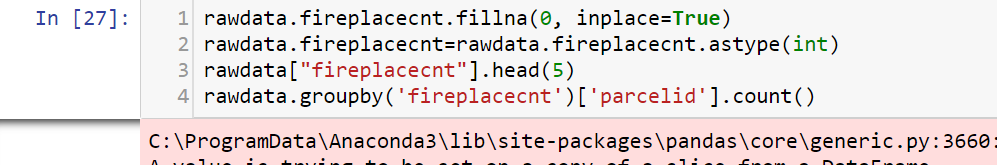


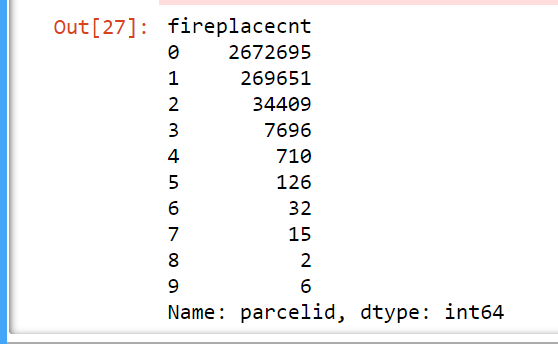


1. Now we can calculate if the column is having relevant number of values else we can remove it from data set

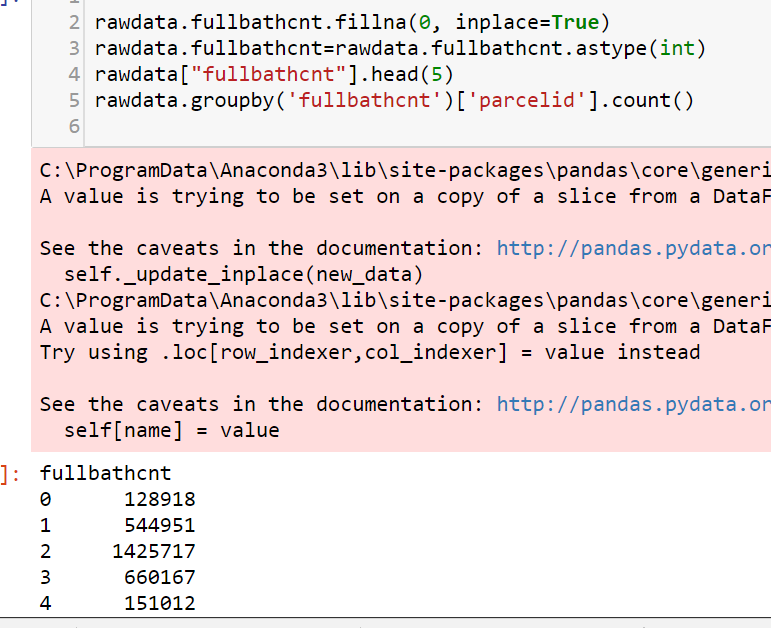


1. Check the number of properties with Fireplace availibity

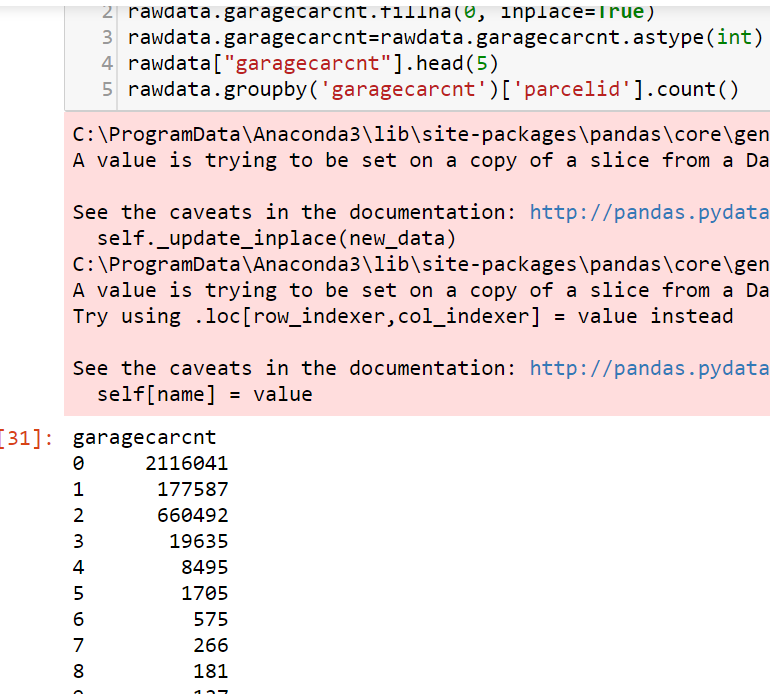




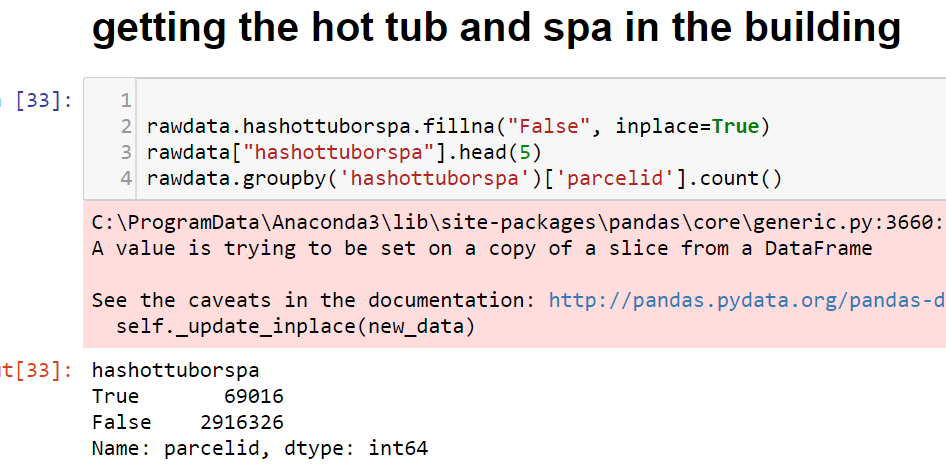
1. Check the number of ¾ baths and filling the missing values



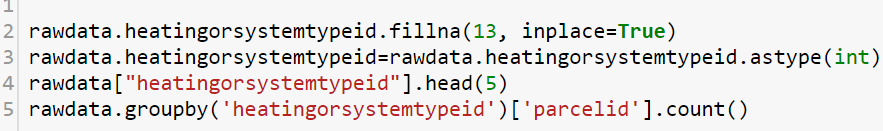
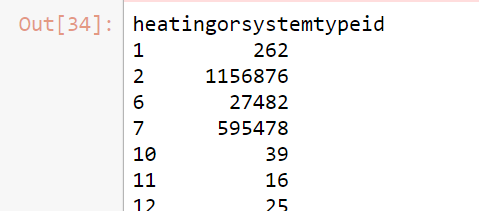
1. Getting the number of properties with Garage Count



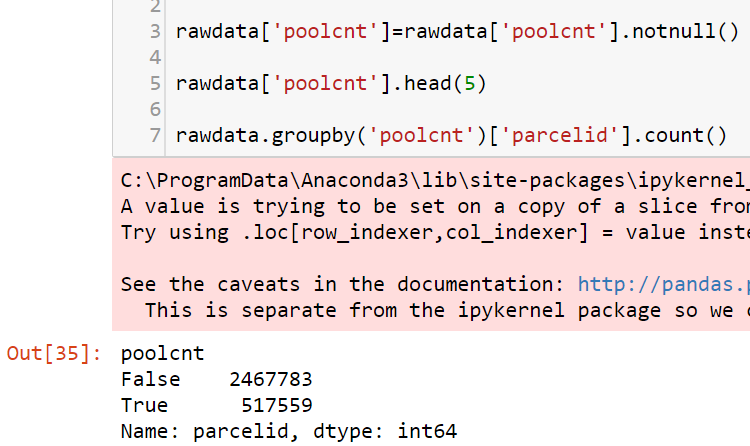
1. Number of hot tub and spa And for better analysis changing it to flag value as True or false.. ie number of total properties with or without



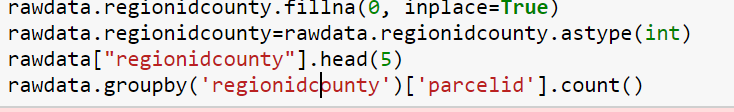
1. Getting the heating systems availibilty else filling the missing value as 13(other )

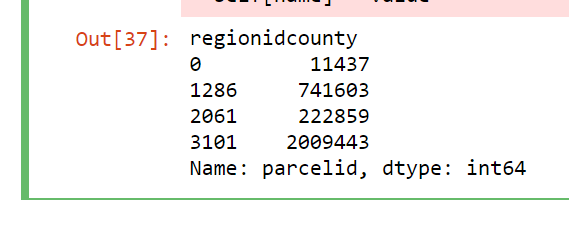
 

1. Getting the numer properties with pool count and changing it to flag

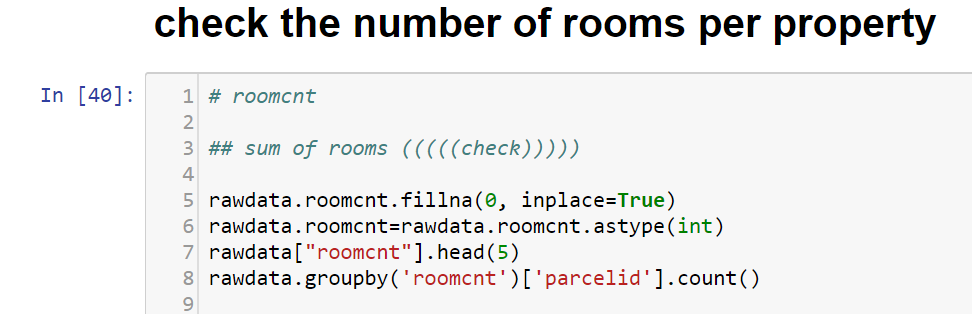


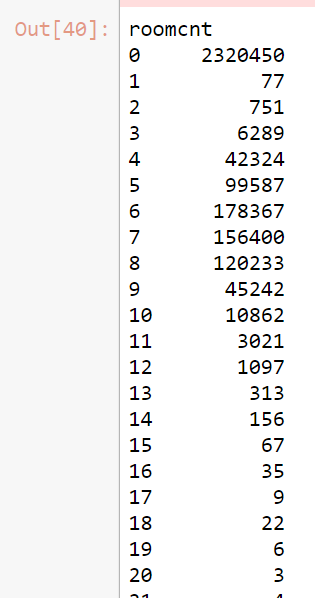
1. Checking the region with no region defined

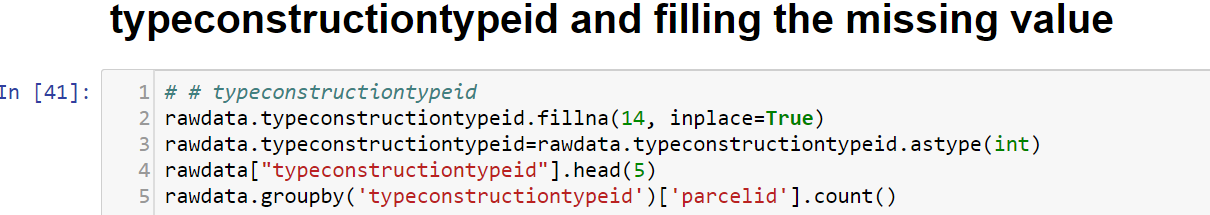


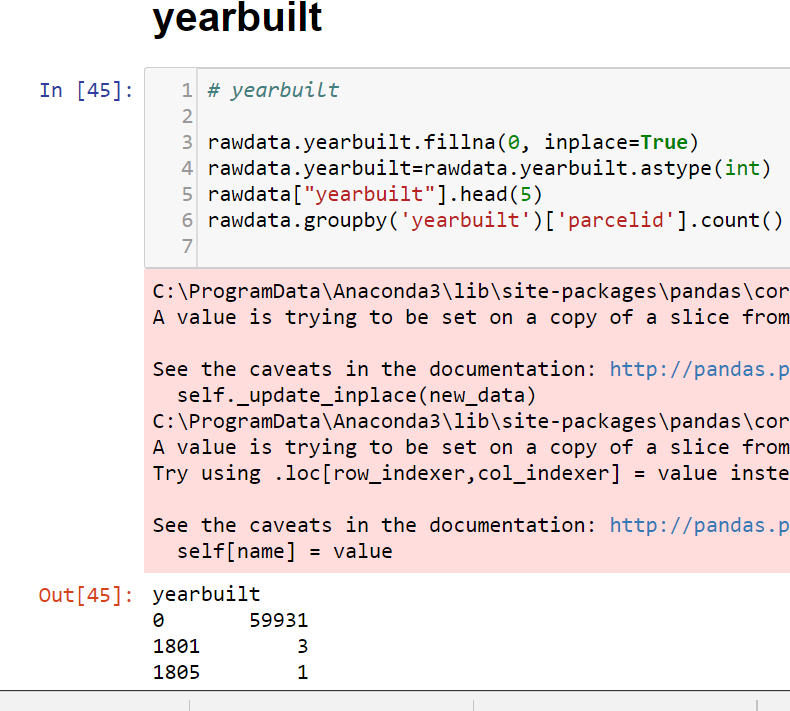


20 . Check the number of rooms per property

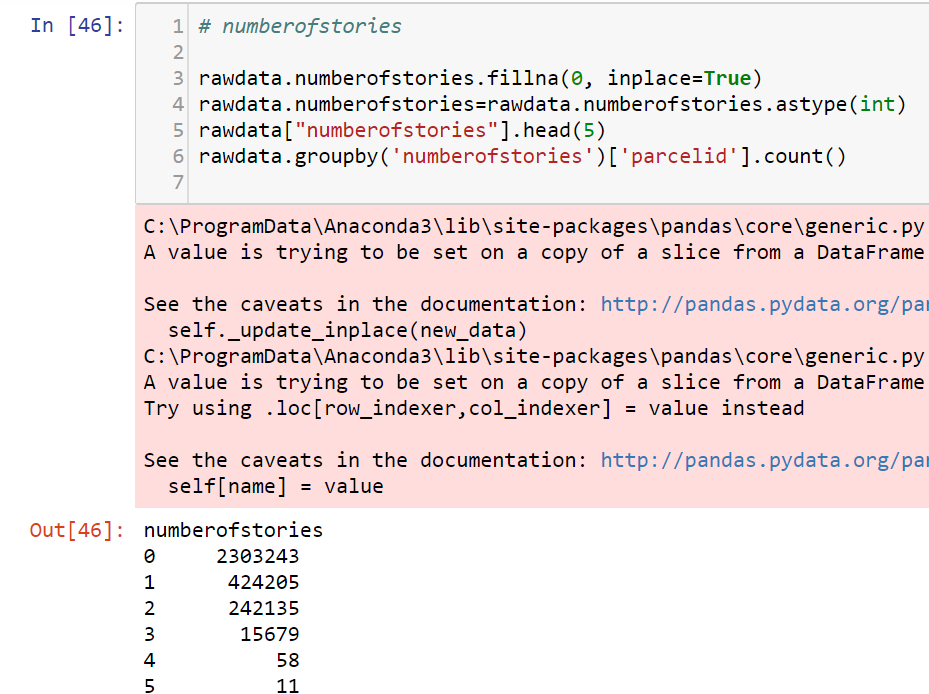




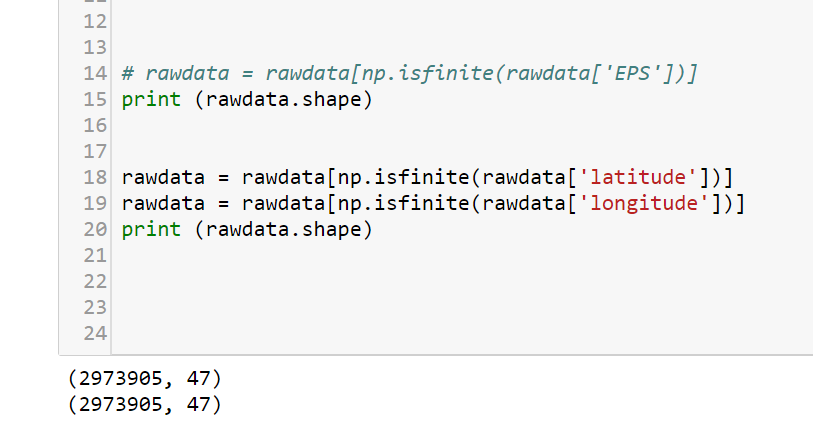
1. 
2. Checking the year built of the house



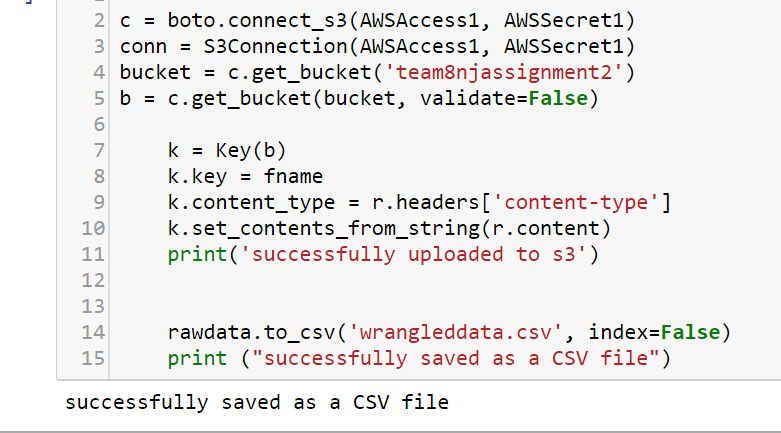
1. Checking the number of stories in properties and filling the missing values



1. Checking the values with zero values of latitude and ongitude and removing them



1. Save the file using pandas into CSV format on local as well on S3 bucket



**Exploratory Data Analysis**

1. MongoDB on EC2 instance

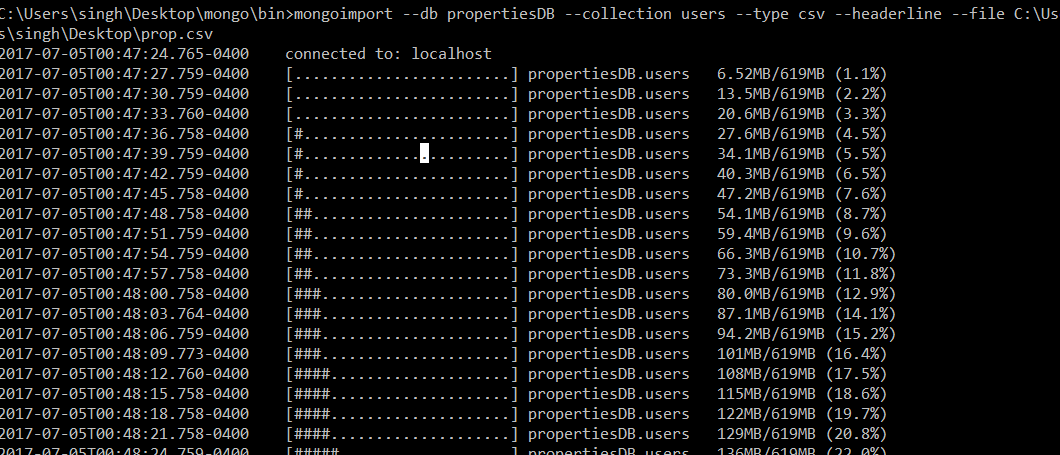
Working with mongoDB and creating the collection of dataset

* Install Mongodb, Then using the CMD in the mongo bin directory
* Run the command:

mongod

* Import the CSV file in mongo
* Run the command to open CSV file using CSV

mongoimport --db propertiesDB --collection users --type csv --headerline --file C:\Users\singh\Desktop\prop.csv



1. Create the REST API to serve the Data

Using Flask To connect with MongoDB 24

Using Flask to create REST API on Cloud26

1. Geospatial Search through the REST API Data24

**Using mongodb e to search nearest values for lattitude and longitude**

For same we No SQL query to get the result:

For exact value of longitude and latitude :

Db.wrangleddata.find({ $or:[ {“latitude”:” ” }, {longitude:””}] })

For 10 nearby properties based on lat and long:

Db.wrangleddata.find({“lat”: {$gte: 20} } # this will give greater than or equal}).limit(5)

Or

Db.wrangleddata.find(

{“lat”: {$lte: 20} } # this will give less than or equal}).limit(5)

Or just want only few columns

Db.wrangleddata.find({“lat”: {$lte: 20} }{“parcelid”:1}}).limit(5)

db.properties.find({"latitude":34280990} ,{"parcelid":1,latitude:1,longitude:1,\_id:0, bedroomcnt:1}).limit(1)

**CITATIONS**