Data Preparation for Data Science

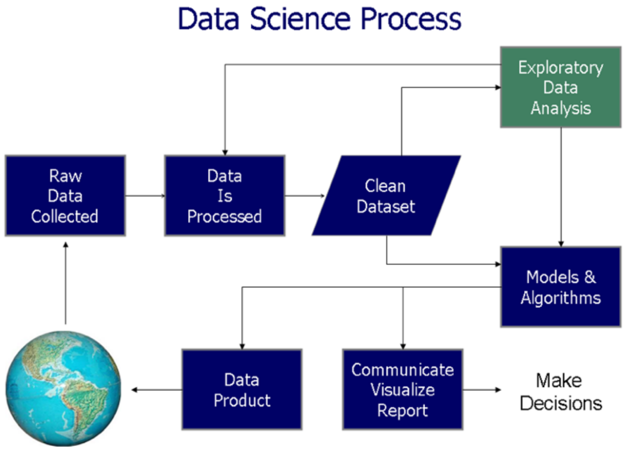
1. Outline

- Data Acumen
  - Data Science Process
  - Data Quality
  - Data Source
  - Data File Format
  - Data Types
- Data Cleaning
  - Missing Data
  - Invalid Data
  - Feature Extraction
  - Demo
- Web Data Preparation
  - Understanding the HTML Page Structure
  - Python and Regular Expressions to Clean Data
  - Python and Beautiful Soup to Collect Data
  - Demo

2. Introduction:

2.1 Data Science Process

- Problem Statement
- Data Collection & Storage
- Data Preparation
  - Access Data
  - Clean Data
  - Transform Data
- Date Analysis & Visualization
- Modeling
- Presentation or Productize



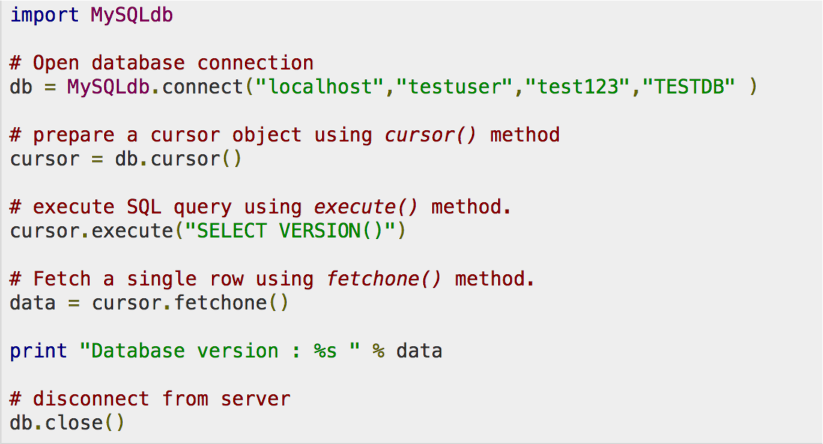
2.2 Data Quality Issues

When we do any Data Science project, the data quality is really important. There is a famous saying: “Garbage in, Garbage out.” And we will introduce some main data quality issues in the following:

- Incorrect/Invalid Entry
    - age = 203; gender = X; price = -100; weekday=8
- Missing Data
    - N/A; Null; “ ” ; Unknown
- Unstructured Data
    - merged cell; double header; html
- Conflicting Data
    - revenue =1000; unit = 0
- Duplicates
    - double loading; double counting
- Outlier
    - House Price = $ 1B

2.3 Data Source

- **RDBMS** stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.



```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print "Database version : %s " % data

# disconnect from server
db.close()
```

2.4 Data File

- Excel:
    - Most common; most problematic
- Delimited format

  - Most common; most preferred
  - Common delimited (csv); tab delimited(tsv); "|" delimited
  - Problem: delimiter in data field. E.g. Los Angles, CA
  - Problem: encoding
- Fixed length
  - Every column has fixed length
  - Problem: Oversized column
- JSON
  - JavaScript Object Notation
  - Semi- Structured
  - Attributes are on the left-hand side of colon
  - Values are on the right-hand side of colon
  - Attributes are separated by a comma
  - Multi-value attributes are as hierarchical values
- XML
  - Extensible Markup Language
  - Semi-Structured
  - Most common for data exchange
- Parquet
  - Column Store
  - Spark

2.6 Web Data

- HTML- Unstructured: Unstructured data (or unstructured information) is information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text-heavy, but may contain data such as dates, numbers, and facts as well.

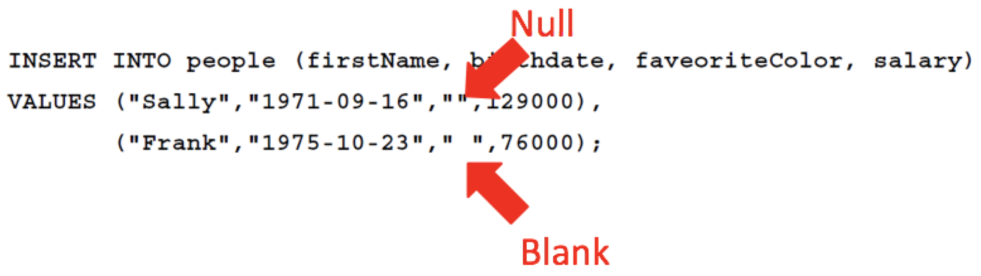
2.7 Big Data Platform

- HDFS – Hive
  - Text file and table
- Spark – RDD
  - Resilient Distributed Datasets
  - RDD is a read-only, partitioned collection of records
- Amazon -- S3

  - Cloud Data storage
  - File can be in any format

2.8 Data Type:

- Numeric
  - Discrete: Count; Rating; Grade; Fibonacci Series
  - Continuous: Revenue; Distance; Home Value
  - Watch out: data range!
- Binary (Dummy)
  - Special case of numeric
  - E.g.: IsMale; HasCar; Pass
- Categorical
  - Usually contains characters: Gender, Product, Geo, etc.
  - Can be consist of pure numbers: SSN, Zipcode, Phone Number
  - Watch out: Valid Values
- Dates and Time
  - Date, Time, Datetime, Timestamp
  - Watch out: Time Zone! UTC=Coordinated Universal Time = GMT = Greenwich Mean Time
- Missing
  - Null
    - Absence of everything; missing; empty
  - Blank
    - “ ” or “   ” or any invisible characters
    - can mean missing
    - can mean “N/A”
  - N/A
    - Can mean “not available” : e.g. Age
    - Can mean “not applicable” : e.g. Middle Name
    - Can mean “no answer” : e.g. Customer Satisfaction Rating on a Questionnaire



```
INSERT INTO people (firstName, birthdate, faveoriteColor, salary)
VALUES ("Sally","1971-09-16","",129000),
       ("Frank","1975-10-23"," ",76000);
```

3 Data Preparation Best Practice

3.1 Data Preparation Steps

- Data Cleansing
  - **Integrate (mapping)**: integrate various data sources into one dataset. E.g. sales units, sales revenue, price
  - **Conform**: Conform the inconsistent values. E.g. Na, n/a => missing; Los Angeles, L.A. => LA
  - **Filter:** Filter out the columns and rows not needed for modeling
  - **Extract:** Extract new column/feature from existing columns. E.g. month from date
  - **Group**: Group many categorical values into less buckets
  - **Aggregate**: Aggregate/Disaggregate date to the desired granularity
  - **Derived feature**: Calculate new metrics based on existing metrics. E.g. Price =Revenue/Units
- Handle Missing Data
- Identity Outlier
- Transform Data
  - One hot encoding: categorical to numerical
  - Normalization/Standardization
  - Log transformation

3.2 Data Cleansing: Regex 101

- a single character of: a, b or c
  [abc]
- a character except: a, b or c
  [^abc]
- a character in the range: a-z
  [a-z]
- a character not in the range: a-z
  [^a-z]
- a character in the range: a-z or A-Z
  [a-zA-Z]
- any single character .
- any whitespace character
  \s
- any non-whitespace character
  \S
- any digit \d
- any non-digit
  \D
- any word character
  \w

- any non-word character \W
- capture everything enclosed (...)
- match either a or b (a|b)
- zero or one of a a?
- zero or more of a a*

- one or more of a a+
- exactly 3 of a a{3}
- 3 or more of a a{3,}
- between 3 and 6 of a a{3,6}
- start of string ^
- end of string $

More information link: www.regex101.com

3.3 Data Cleansing: Useful Regex

- Replace
  - Reverse last name and first name: San, Zhang => Zhang San
  - Regex=/([a-zA-Z]+),\s*([a-zA-Z]+)/, Replace = $2 $1
- Extract
  - Extract url from html: <a href="http://www.amghezi.com">amgheziName</a>
  - Regex = /href=/"([^"]*)/, Replace = $1
- Validation
  - Validate a valid email
  - Regex =/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/i

3.4 Missing Data: Types

- Missing completely at random: MCAR
  - Roll a dice
  - Lottery number
- Not missing at random: NMAR
  - missing values are systematic
  - Income: higher income is less likely to respond
  - Weight: higher weight is less likely to respond
  - Smoking
- Missing at random: MAR
  - Most Common
  - Missing values can somewhat be predicted by known info
  - Know height, missing weight
  - Know # of rooms, missing sqrt

3.5 Missing Data: Handling

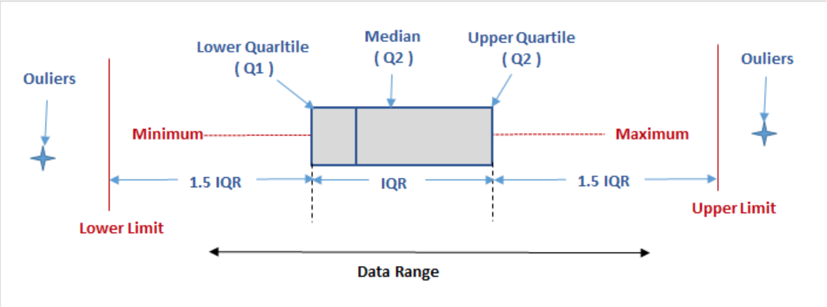
- Impute from other attributes
- Impute from other observations
  - Majority vote (categorical)
  - Mean of same/similar group (numerical)
  - Carry last value (time series)
  - Linear fill (time series)
  - Carry same trend (time series)
- “Missing” Category (not missing at random)
- Extra indicator
- Logical estimation
- Remove row or column

3.6 Outliers:

- 1.5 IQR

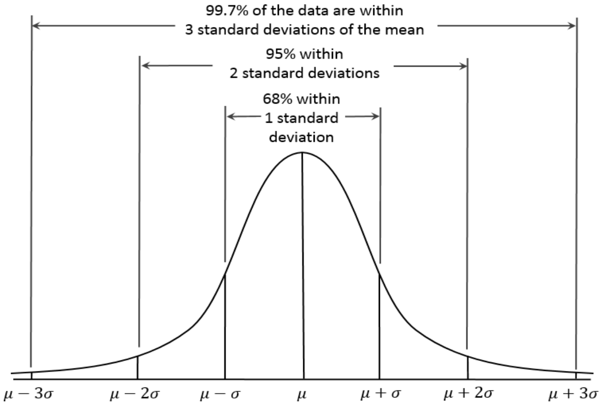
Check the frequency distribution of the data.

Box-plot: An outlier is a point of data that lies over 1.5 IQRs below the first quartile (Q1) or above third quartile (Q3) in a given data set.



- Normal Distribution

Outlier: the range of 2 or 3 STD from mean is limited. If else, there are outliers.



- Other Technics
    - Univariable Outlier:
        - Median Absolute Deviation
    - Multivariate Outlier
        - Mahalanobis Distance

3.7 Data Transformation: Normalization vs. Standardization

	Normalization	Standardization
Formula	$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$	$x_{new} = \frac{x - \mu}{\sigma}$
Pro	•Bounded (-1,1) •Apply to all distribution	•Works well for normal distribution
Con	•Make outliers “normal”	•Unbounded •Only works well for normal distribution

- Normalization
    - Linear Model
        - Recommended
        - Doesn’ t change model accuracy
        - Easier to compare coefficient: larger coefficient, larger impact
        - Intercept well interpreted: the expected value of Yi when the predictors are set to their means
        - Avoid coefficient like 10^-9 when one variable has a very large scale
        - More difficult to interpret the model in terms of on unit change in Xi
    - Tree Model
        - Not necessary as the scale is irrelevant
    - Logistic Regression
        - Typically not needed
    - SVM
        - Recommended
        - Help with faster converge
- log

- Linear Model; Skewed Data
- Log Predictor

$$y = e^{ax} + b \xrightarrow{\log x} y = ax' + b$$

- Log Outcome

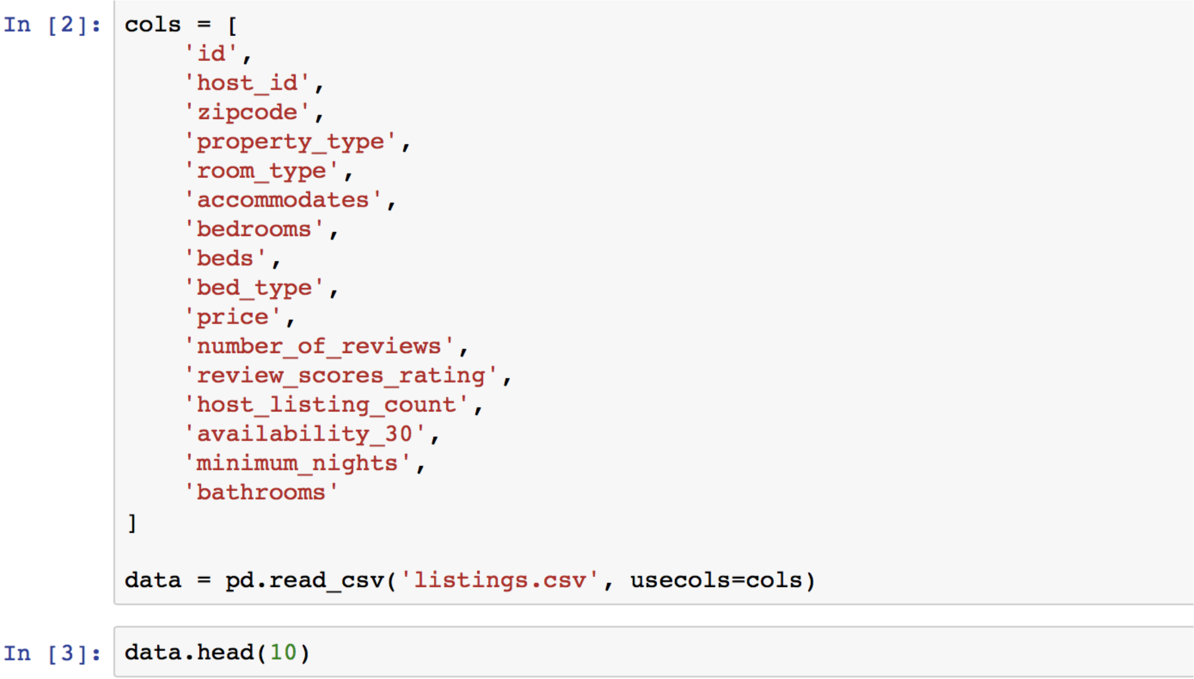
$$y = \ln(ax + b) \xrightarrow{\log y} y' = ax + b$$

- Log both

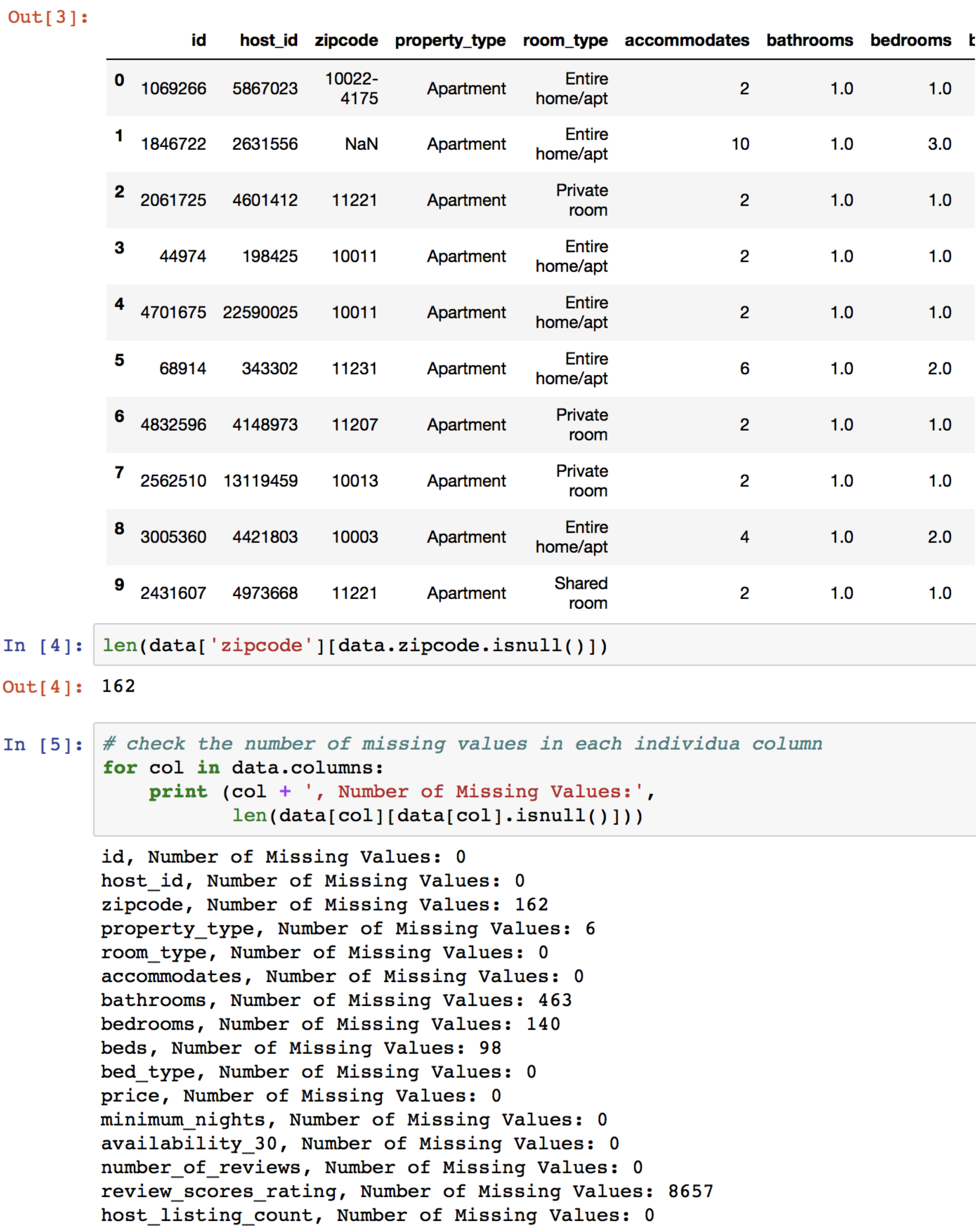
$$y = e^{c} * x_1^{a} * x_2^{b} \xrightarrow{yields} lny = c + ax_1 + bx_2$$

3.8 Demo

Use Python to clean Aribnb listings data (from file)



```
In [2]: cols = [
            'id',
            'host_id',
            'zipcode',
            'property_type',
            'room_type',
            'accommodates',
            'bedrooms',
            'beds',
            'bed_type',
            'price',
            'number_of_reviews',
            'review_scores_rating',
            'host_listing_count',
            'availability_30',
            'minimum_nights',
            'bathrooms'
        ]

        data = pd.read_csv('listings.csv', usecols=cols)

In [3]: data.head(10)
```



Out[3]:

	id	host_id	zipcode	property_type	room_type	accommodates	bathrooms	bedrooms	b
0	1069266	5867023	10022-4175	Apartment	Entire home/apt	2	1.0	1.0	
1	1846722	2631556	NaN	Apartment	Entire home/apt	10	1.0	3.0	
2	2061725	4601412	11221	Apartment	Private room	2	1.0	1.0	
3	44974	198425	10011	Apartment	Entire home/apt	2	1.0	1.0	
4	4701675	22590025	10011	Apartment	Entire home/apt	2	1.0	1.0	
5	68914	343302	11231	Apartment	Entire home/apt	6	1.0	2.0	
6	4832596	4148973	11207	Apartment	Private room	2	1.0	1.0	
7	2562510	13119459	10013	Apartment	Private room	2	1.0	1.0	
8	3005360	4421803	10003	Apartment	Entire home/apt	4	1.0	2.0	
9	2431607	4973668	11221	Apartment	Shared room	2	1.0	1.0	

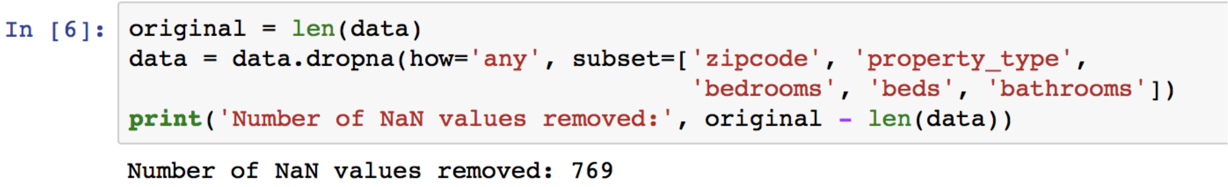
In [4]:
```
len(data['zipcode'][data.zipcode.isnull()])
```

Out[4]:
```
162
```

In [5]:
```
# check the number of missing values in each individua column
for col in data.columns:
    print (col + ', Number of Missing Values:',
           len(data[col][data[col].isnull()]))
```

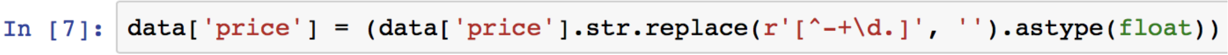
```
id, Number of Missing Values: 0
host_id, Number of Missing Values: 0
zipcode, Number of Missing Values: 162
property_type, Number of Missing Values: 6
room_type, Number of Missing Values: 0
accommodates, Number of Missing Values: 0
bathrooms, Number of Missing Values: 463
bedrooms, Number of Missing Values: 140
beds, Number of Missing Values: 98
bed_type, Number of Missing Values: 0
price, Number of Missing Values: 0
minimum_nights, Number of Missing Values: 0
availability_30, Number of Missing Values: 0
number_of_reviews, Number of Missing Values: 0
review_scores_rating, Number of Missing Values: 8657
host_listing_count, Number of Missing Values: 0
```

Remove NaN values from dataframe except review_scores_rating



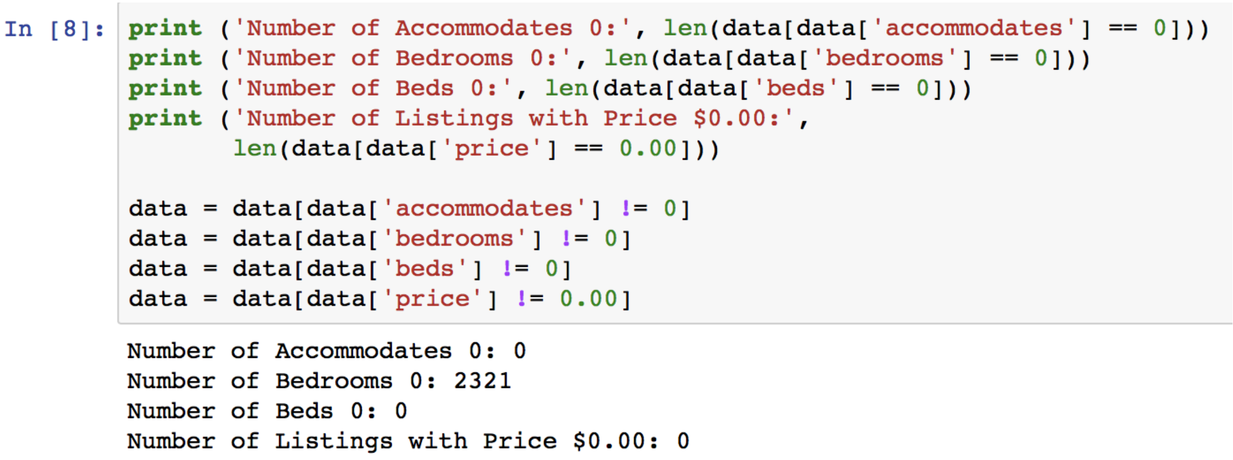
```
In [6]: original = len(data)
        data = data.dropna(how='any', subset=['zipcode', 'property_type',
                                              'bedrooms', 'beds', 'bathrooms'])
        print('Number of NaN values removed:', original - len(data))

        Number of NaN values removed: 769
```

Convert formatting for price from $1.00 into a float of 1.00



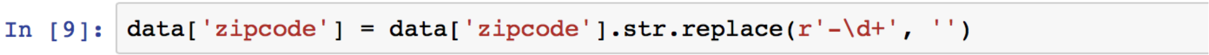
```
In [7]: data['price'] = (data['price'].str.replace(r'[^-+\d.]', '').astype(float))
```

Drop any invalid values

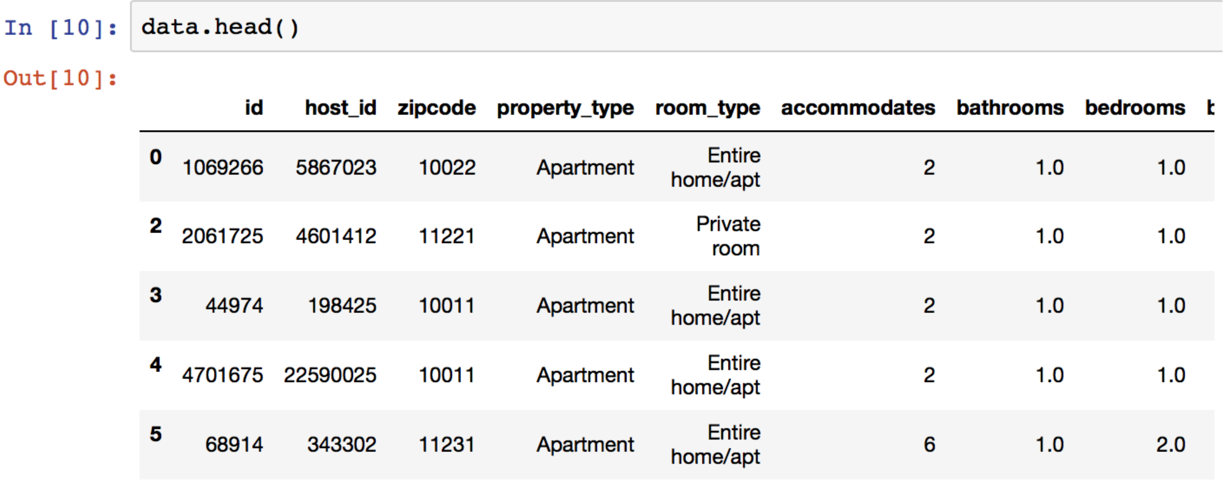


```
In [8]: print ('Number of Accommodates 0:', len(data[data['accommodates'] == 0]))
        print ('Number of Bedrooms 0:', len(data[data['bedrooms'] == 0]))
        print ('Number of Beds 0:', len(data[data['beds'] == 0]))
        print ('Number of Listings with Price $0.00:',
               len(data[data['price'] == 0.00]))

        data = data[data['accommodates'] != 0]
        data = data[data['bedrooms'] != 0]
        data = data[data['beds'] != 0]
        data = data[data['price'] != 0.00]

        Number of Accommodates 0: 0
        Number of Bedrooms 0: 2321
        Number of Beds 0: 0
        Number of Listings with Price $0.00: 0
```

Convert Zipcode to 5 digits



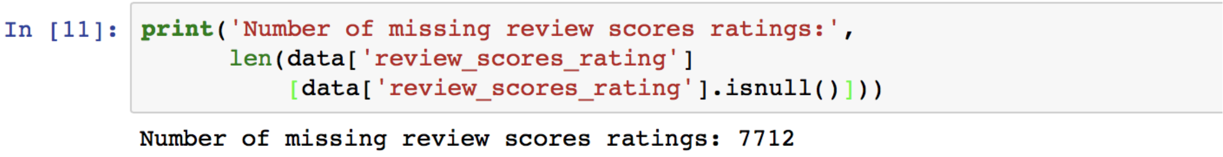
```
In [9]: data['zipcode'] = data['zipcode'].str.replace(r'-\d+', '')
```



```
In [10]: data.head()
```

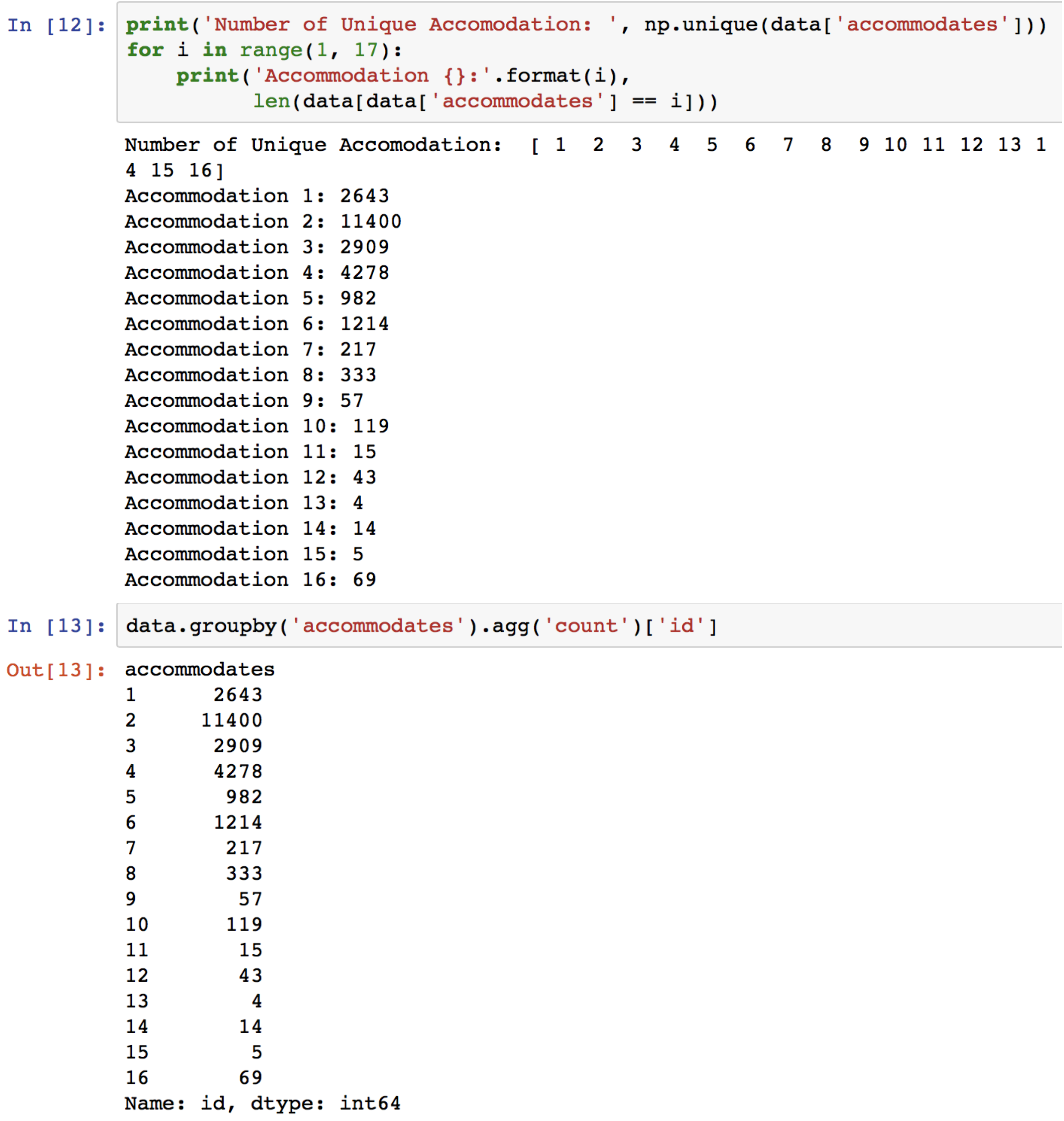
Out[10]:

	id	host_id	zipcode	property_type	room_type	accommodates	bathrooms	bedrooms	b
0	1069266	5867023	10022	Apartment	Entire home/apt	2	1.0	1.0	
2	2061725	4601412	11221	Apartment	Private room	2	1.0	1.0	
3	44974	198425	10011	Apartment	Entire home/apt	2	1.0	1.0	
4	4701675	22590025	10011	Apartment	Entire home/apt	2	1.0	1.0	
5	68914	343302	11231	Apartment	Entire home/apt	6	1.0	2.0	



```
In [11]: print('Number of missing review scores ratings:',
               len(data['review_scores_rating']
                   [data['review_scores_rating'].isnull()]))

         Number of missing review scores ratings: 7712
```

Let's explore distribution of accommodates



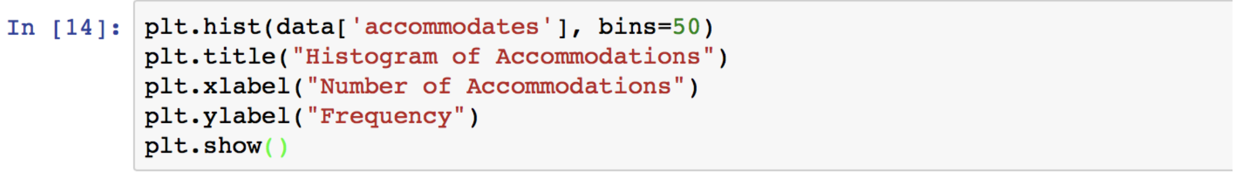
```
In [12]: print('Number of Unique Accomodation: ', np.unique(data['accommodates']))
         for i in range(1, 17):
             print('Accommodation {}:'.format(i),
                   len(data[data['accommodates'] == i]))
```

```
Number of Unique Accomodation:  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 1
4 15 16]
Accommodation 1: 2643
Accommodation 2: 11400
Accommodation 3: 2909
Accommodation 4: 4278
Accommodation 5: 982
Accommodation 6: 1214
Accommodation 7: 217
Accommodation 8: 333
Accommodation 9: 57
Accommodation 10: 119
Accommodation 11: 15
Accommodation 12: 43
Accommodation 13: 4
Accommodation 14: 14
Accommodation 15: 5
Accommodation 16: 69
```

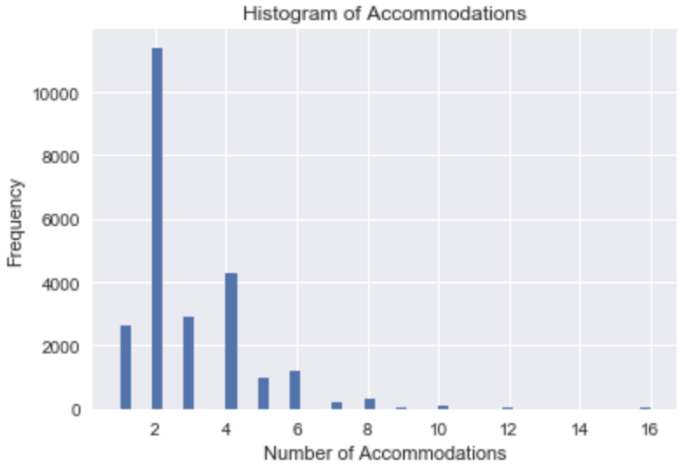
```
In [13]: data.groupby('accommodates').agg('count')['id']
```

```
Out[13]: accommodates
         1       2643
         2      11400
         3       2909
         4       4278
         5        982
         6       1214
         7        217
         8        333
         9         57
         10       119
         11        15
         12        43
         13         4
         14        14
         15         5
         16        69
         Name: id, dtype: int64
```

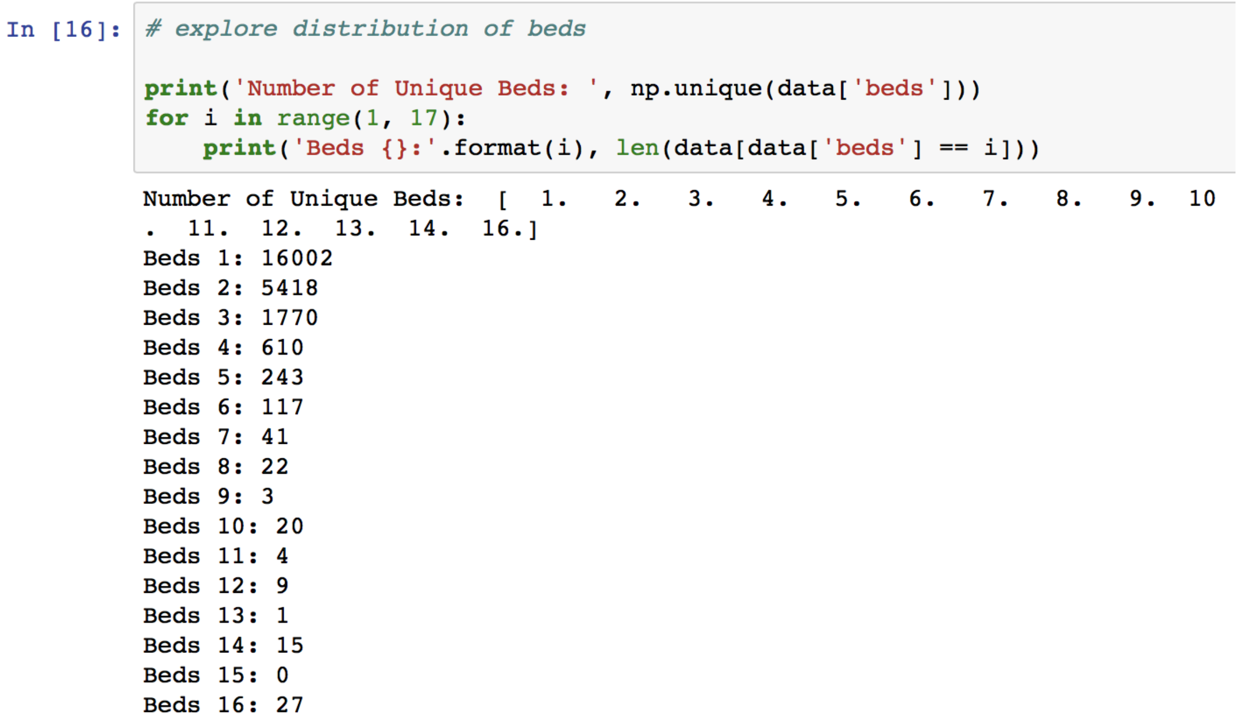
Visualize distribution of price, accommdations, beds, and review_scores_rating respectively



```
In [14]: plt.hist(data['accommodates'], bins=50)
         plt.title("Histogram of Accommodations")
         plt.xlabel("Number of Accommodations")
         plt.ylabel("Frequency")
         plt.show()
```

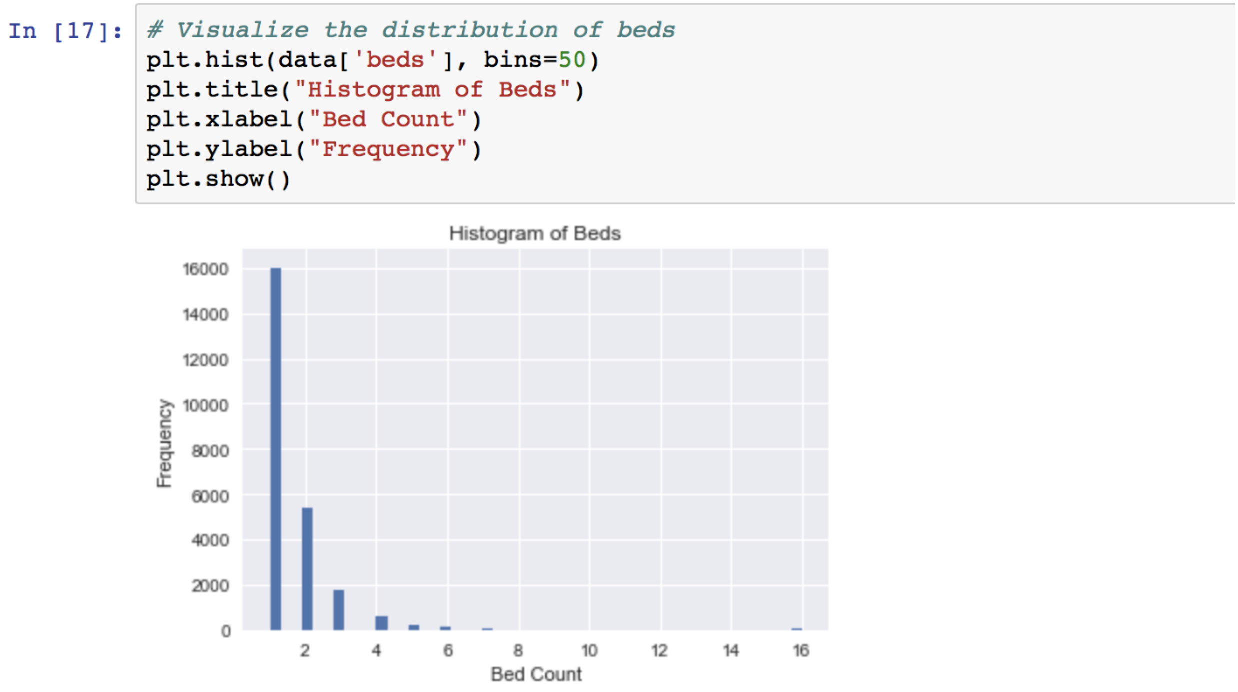


We see that a majority of listings have accomodations for 1-4 people. 1 bed typically accomodates 2 individuals, so let's plot beds instead to analyze how many of the listings are single bedroom listings.

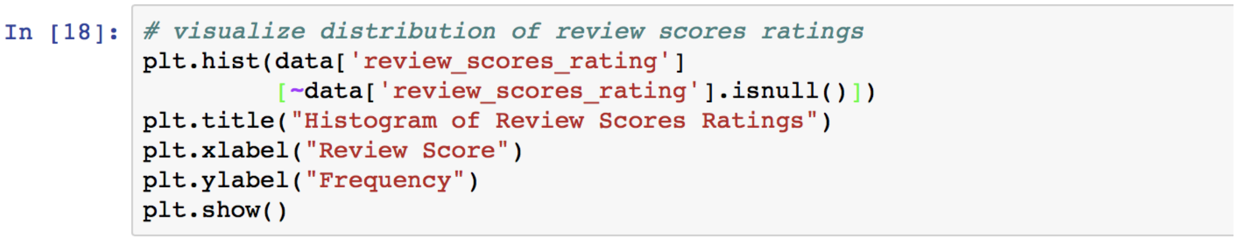


```
In [16]: # explore distribution of beds

         print('Number of Unique Beds: ', np.unique(data['beds']))
         for i in range(1, 17):
             print('Beds {}:'.format(i), len(data[data['beds'] == i]))
```

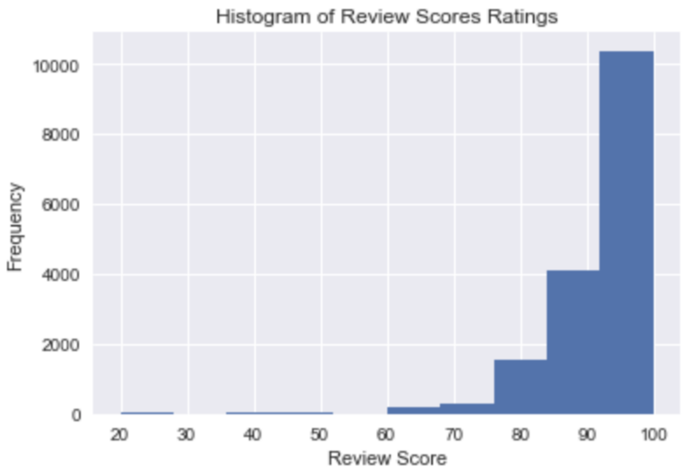
```
Number of Unique Beds:  [  1.   2.   3.   4.   5.   6.   7.   8.   9.  10
.  11.  12.  13.  14.  16.]
Beds 1: 16002
Beds 2: 5418
Beds 3: 1770
Beds 4: 610
Beds 5: 243
Beds 6: 117
Beds 7: 41
Beds 8: 22
Beds 9: 3
Beds 10: 20
Beds 11: 4
Beds 12: 9
Beds 13: 1
Beds 14: 15
Beds 15: 0
Beds 16: 27
```



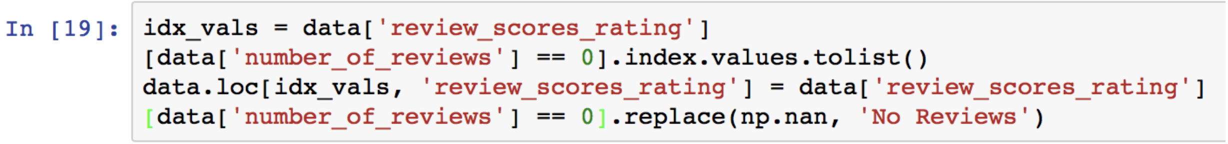
```
In [17]: # Visualize the distribution of beds
         plt.hist(data['beds'], bins=50)
         plt.title("Histogram of Beds")
         plt.xlabel("Bed Count")
         plt.ylabel("Frequency")
         plt.show()
```



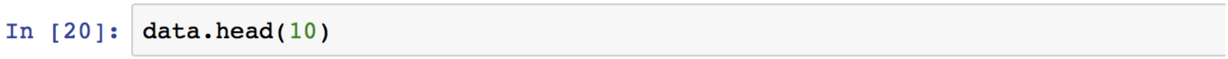
```
In [18]: # visualize distribution of review scores ratings
         plt.hist(data['review_scores_rating']
                  [~data['review_scores_rating'].isnull()])
         plt.title("Histogram of Review Scores Ratings")
         plt.xlabel("Review Score")
         plt.ylabel("Frequency")
         plt.show()
```



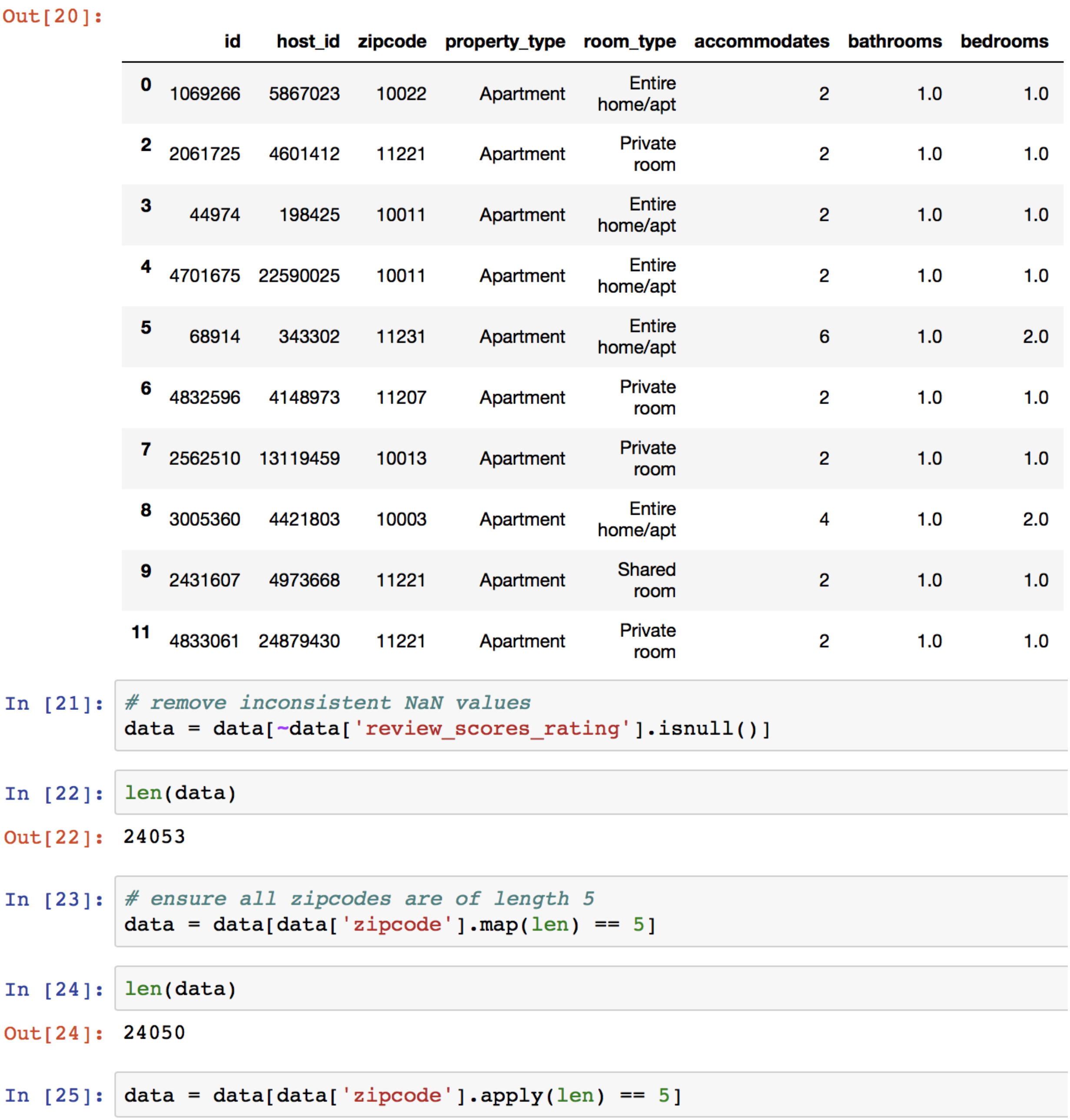
Convert NaN scores with 0 reviews into 'No Reviews'



```
In [19]: idx_vals = data['review_scores_rating']
         [data['number_of_reviews'] == 0].index.values.tolist()
         data.loc[idx_vals, 'review_scores_rating'] = data['review_scores_rating']
         [data['number_of_reviews'] == 0].replace(np.nan, 'No Reviews')
```



```
In [20]: data.head(10)
```



Out[20]:

	id	host_id	zipcode	property_type	room_type	accommodates	bathrooms	bedrooms
0	1069266	5867023	10022	Apartment	Entire home/apt	2	1.0	1.0
2	2061725	4601412	11221	Apartment	Private room	2	1.0	1.0
3	44974	198425	10011	Apartment	Entire home/apt	2	1.0	1.0
4	4701675	22590025	10011	Apartment	Entire home/apt	2	1.0	1.0
5	68914	343302	11231	Apartment	Entire home/apt	6	1.0	2.0
6	4832596	4148973	11207	Apartment	Private room	2	1.0	1.0
7	2562510	13119459	10013	Apartment	Private room	2	1.0	1.0
8	3005360	4421803	10003	Apartment	Entire home/apt	4	1.0	2.0
9	2431607	4973668	11221	Apartment	Shared room	2	1.0	1.0
11	4833061	24879430	11221	Apartment	Private room	2	1.0	1.0

```
In [21]: # remove inconsistent NaN values
         data = data[~data['review_scores_rating'].isnull()]
```

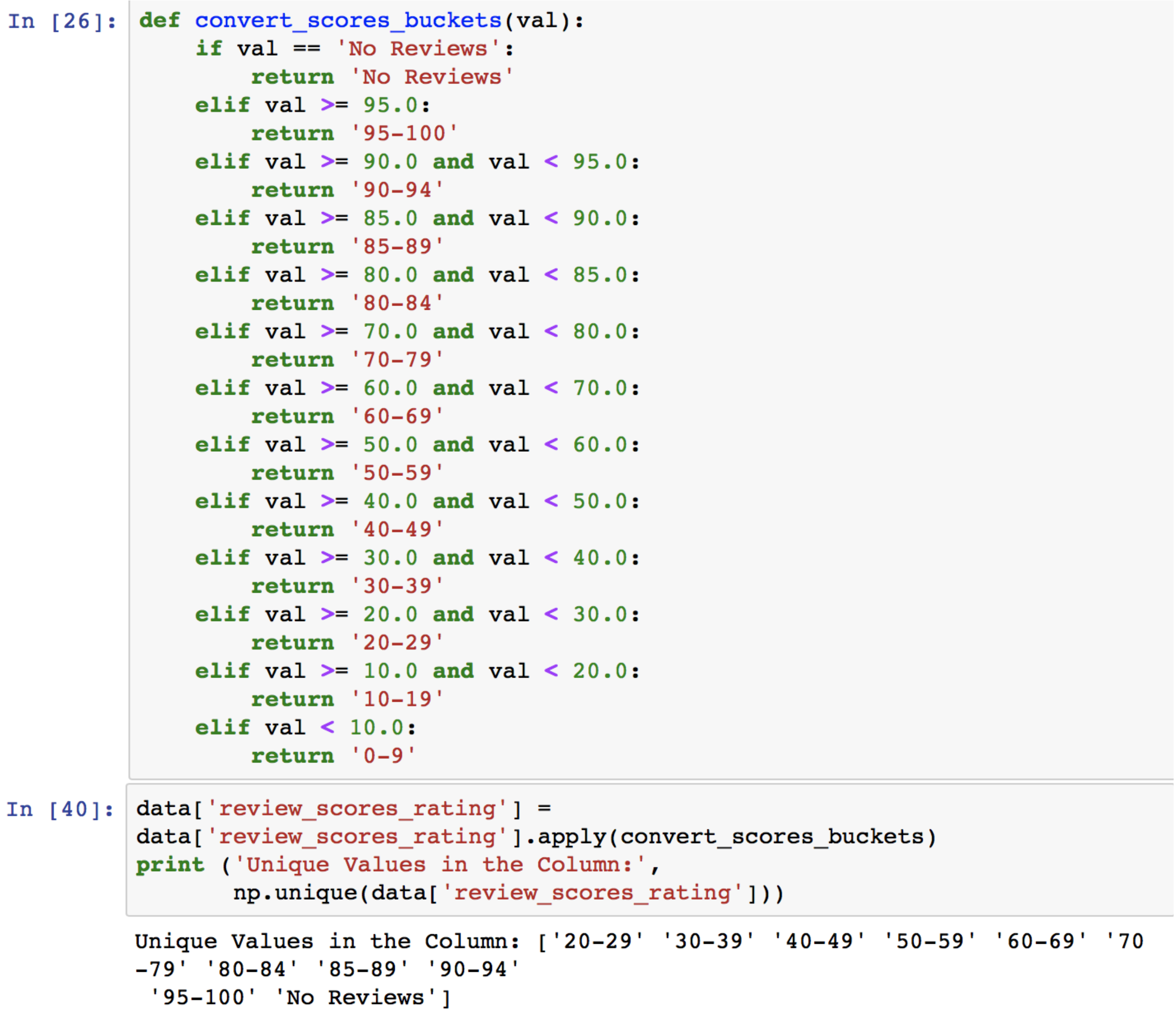
```
In [22]: len(data)

Out[22]: 24053
```

```
In [23]: # ensure all zipcodes are of length 5
         data = data[data['zipcode'].map(len) == 5]
```

```
In [24]: len(data)

Out[24]: 24050
```

```
In [25]: data = data[data['zipcode'].apply(len) == 5]
```

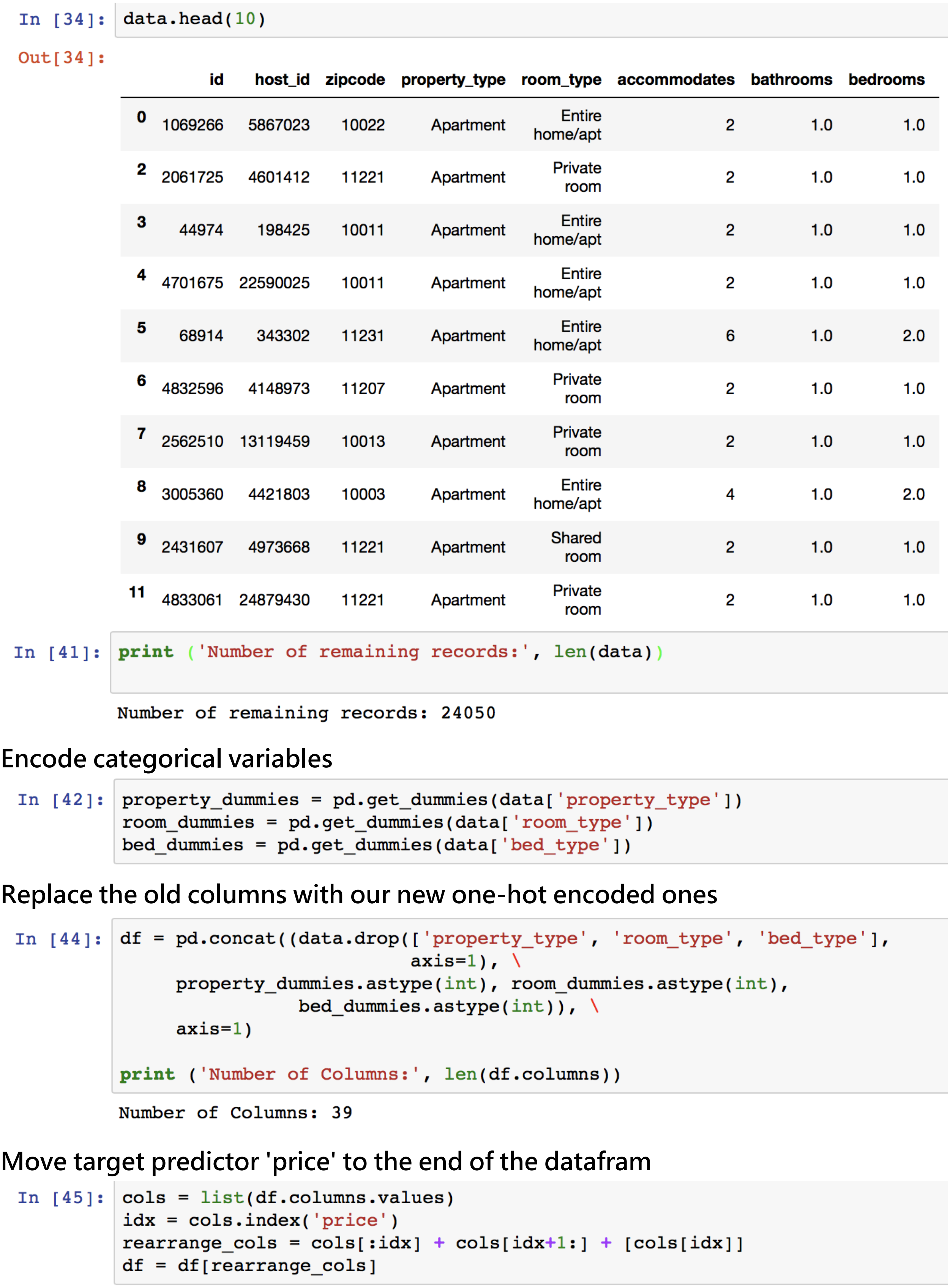
Convert review_scores_rating into different buckets



```
In [26]: def convert_scores_buckets(val):
             if val == 'No Reviews':
                 return 'No Reviews'
             elif val >= 95.0:
                 return '95-100'
             elif val >= 90.0 and val < 95.0:
                 return '90-94'
             elif val >= 85.0 and val < 90.0:
                 return '85-89'
             elif val >= 80.0 and val < 85.0:
                 return '80-84'
             elif val >= 70.0 and val < 80.0:
                 return '70-79'
             elif val >= 60.0 and val < 70.0:
                 return '60-69'
             elif val >= 50.0 and val < 60.0:
                 return '50-59'
             elif val >= 40.0 and val < 50.0:
                 return '40-49'
             elif val >= 30.0 and val < 40.0:
                 return '30-39'
             elif val >= 20.0 and val < 30.0:
                 return '20-29'
             elif val >= 10.0 and val < 20.0:
                 return '10-19'
             elif val < 10.0:
                 return '0-9'
```

```
In [40]: data['review_scores_rating'] =
         data['review_scores_rating'].apply(convert_scores_buckets)
         print ('Unique Values in the Column:',
                np.unique(data['review_scores_rating']))
```

```
Unique Values in the Column: ['20-29' '30-39' '40-49' '50-59' '60-69' '70
-79' '80-84' '85-89' '90-94'
 '95-100' 'No Reviews']
```



```
In [34]: data.head(10)
```

Out[34]:

	id	host_id	zipcode	property_type	room_type	accommodates	bathrooms	bedrooms
0	1069266	5867023	10022	Apartment	Entire home/apt	2	1.0	1.0
2	2061725	4601412	11221	Apartment	Private room	2	1.0	1.0
3	44974	198425	10011	Apartment	Entire home/apt	2	1.0	1.0
4	4701675	22590025	10011	Apartment	Entire home/apt	2	1.0	1.0
5	68914	343302	11231	Apartment	Entire home/apt	6	1.0	2.0
6	4832596	4148973	11207	Apartment	Private room	2	1.0	1.0
7	2562510	13119459	10013	Apartment	Private room	2	1.0	1.0
8	3005360	4421803	10003	Apartment	Entire home/apt	4	1.0	2.0
9	2431607	4973668	11221	Apartment	Shared room	2	1.0	1.0
11	4833061	24879430	11221	Apartment	Private room	2	1.0	1.0

```
In [41]: print ('Number of remaining records:', len(data))

Number of remaining records: 24050
```

Encode categorical variables

```
In [42]: property_dummies = pd.get_dummies(data['property_type'])
         room_dummies = pd.get_dummies(data['room_type'])
         bed_dummies = pd.get_dummies(data['bed_type'])
```

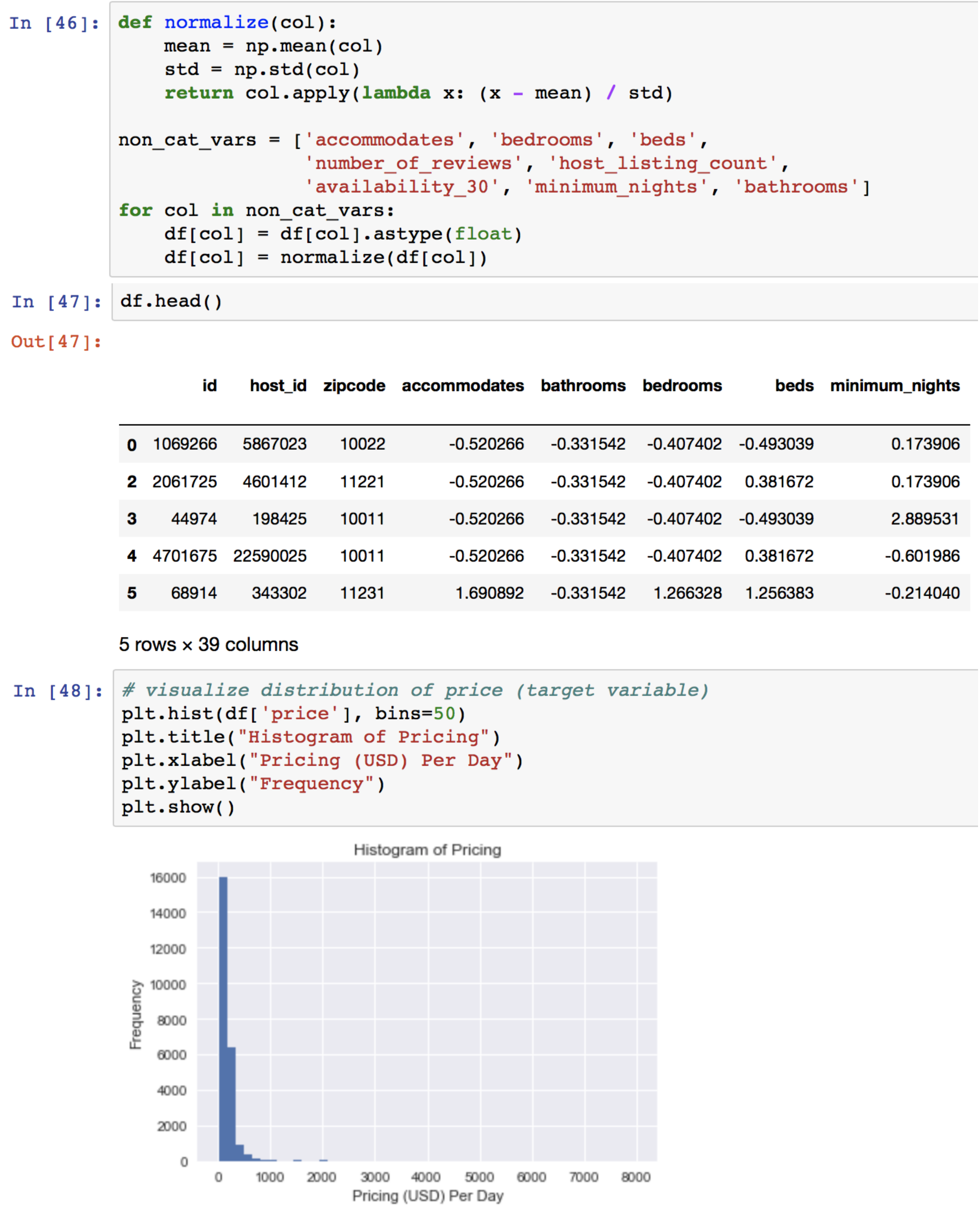
Replace the old columns with our new one-hot encoded ones

```
In [44]: df = pd.concat((data.drop(['property_type', 'room_type', 'bed_type'],
                                   axis=1), \
              property_dummies.astype(int), room_dummies.astype(int),
                         bed_dummies.astype(int)), \
              axis=1)

         print ('Number of Columns:', len(df.columns))

Number of Columns: 39
```

Move target predictor 'price' to the end of the datafram

```
In [45]: cols = list(df.columns.values)
         idx = cols.index('price')
         rearrange_cols = cols[:idx] + cols[idx+1:] + [cols[idx]]
         df = df[rearrange_cols]
```

Convert non-categorical variables to floats and normalize



```
In [46]: def normalize(col):
             mean = np.mean(col)
             std = np.std(col)
             return col.apply(lambda x: (x - mean) / std)

         non_cat_vars = ['accommodates', 'bedrooms', 'beds',
                         'number_of_reviews', 'host_listing_count',
                         'availability_30', 'minimum_nights', 'bathrooms']
         for col in non_cat_vars:
             df[col] = df[col].astype(float)
             df[col] = normalize(df[col])
```

```
In [47]: df.head()
```

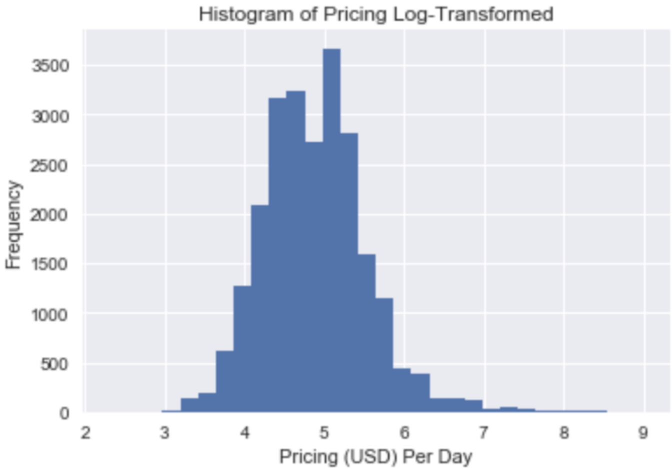
Out[47]:

	id	host_id	zipcode	accommodates	bathrooms	bedrooms	beds	minimum_nights
0	1069266	5867023	10022	-0.520266	-0.331542	-0.407402	-0.493039	0.173906
2	2061725	4601412	11221	-0.520266	-0.331542	-0.407402	0.381672	0.173906
3	44974	198425	10011	-0.520266	-0.331542	-0.407402	-0.493039	2.889531
4	4701675	22590025	10011	-0.520266	-0.331542	-0.407402	0.381672	-0.601986
5	68914	343302	11231	1.690892	-0.331542	1.266328	1.256383	-0.214040

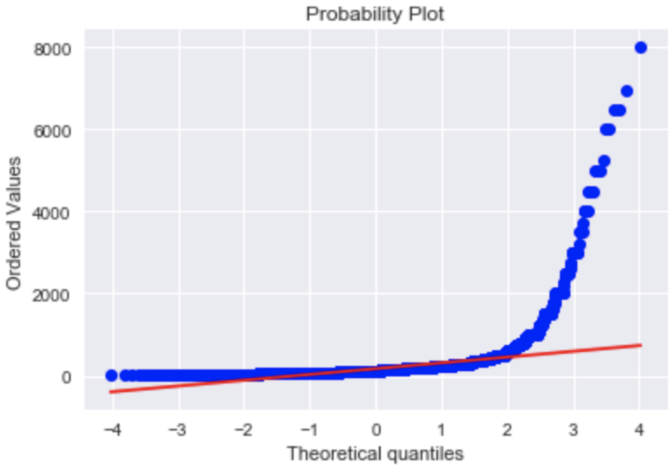
5 rows × 39 columns

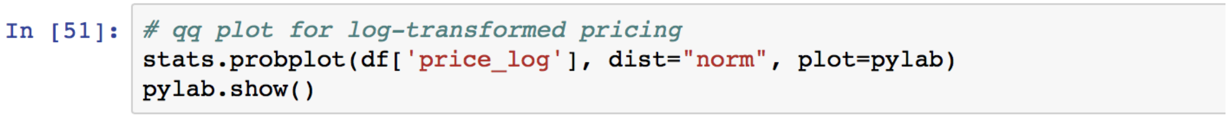
```
In [48]: # visualize distribution of price (target variable)
         plt.hist(df['price'], bins=50)
         plt.title("Histogram of Pricing")
         plt.xlabel("Pricing (USD) Per Day")
         plt.ylabel("Frequency")
         plt.show()
```

```
In [49]: # log transform the response 'price'
         df['price_log'] = df['price'].apply(lambda x: math.log(x))

         plt.hist(df['price_log'], bins=30)
         plt.title("Histogram of Pricing Log-Transformed")
         plt.xlabel("Pricing (USD) Per Day")
         plt.ylabel("Frequency")
         plt.show()
```

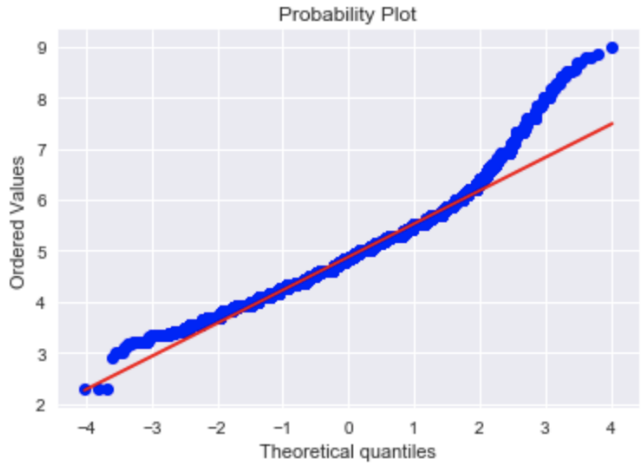


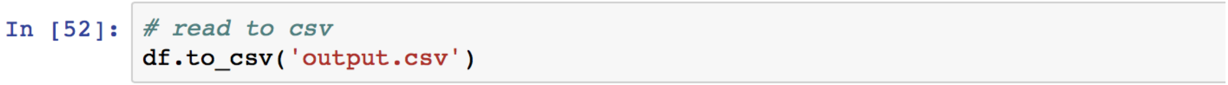
```
In [50]: # qq plot for log-transformed pricing
         stats.probplot(df['price'], dist="norm", plot=pylab)
         pylab.show()
```





```
In [51]: # qq plot for log-transformed pricing
         stats.probplot(df['price_log'], dist="norm", plot=pylab)
         pylab.show()
```





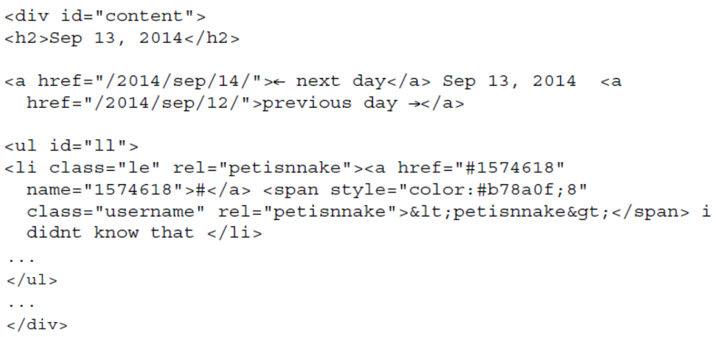
```
In [52]: # read to csv
         df.to_csv('output.csv')
```

4 Web Data Preparation

4.1 Web Data Raw Format: HTML

Understanding the HTML Page Structure

- HTML can be parsed in two ways:
  - The line-by-line delimiter model
  - The tree structure model

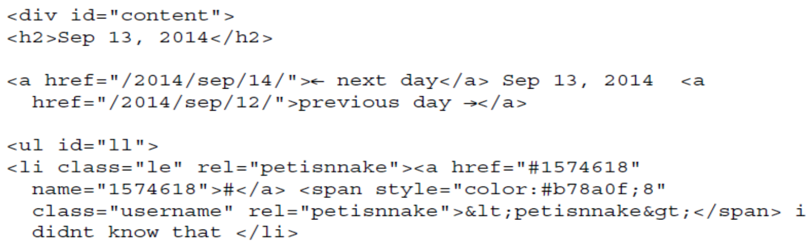


```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
...
</ul>
...
</div>
```

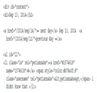
4.2 Web Scraping: Line by Line

The line-by-line delimiter model

- <h2></h2> tags as delimiters to extract the date
- <li></li> tags as delimiters to extract text
- Rel= “” as delimiters to extract user name
- From the end of </span> to the beginning of </li> is the actual line message

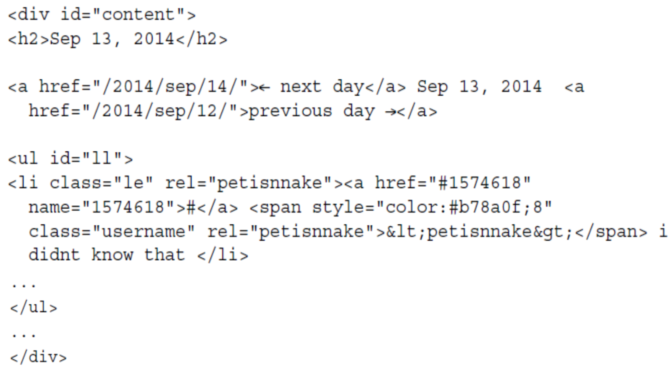


```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
```

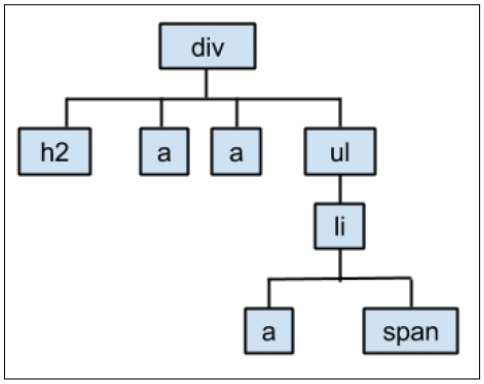


4.3 Web Scraping: Tree Model

The tree structure model: we can consider the structure of HTML as a tree structure. We can use BeautifulSoup to analysis HTML structure, and get the data.

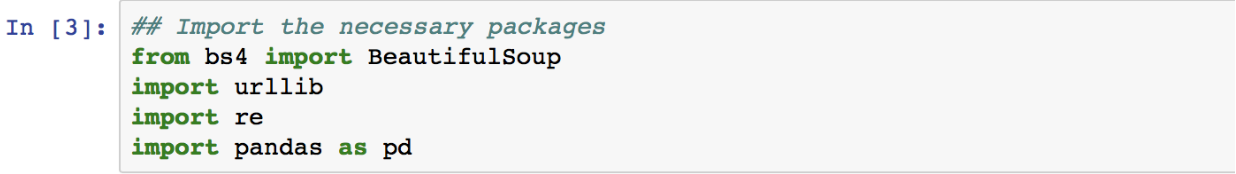


```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
...
</ul>
...
</div>
```



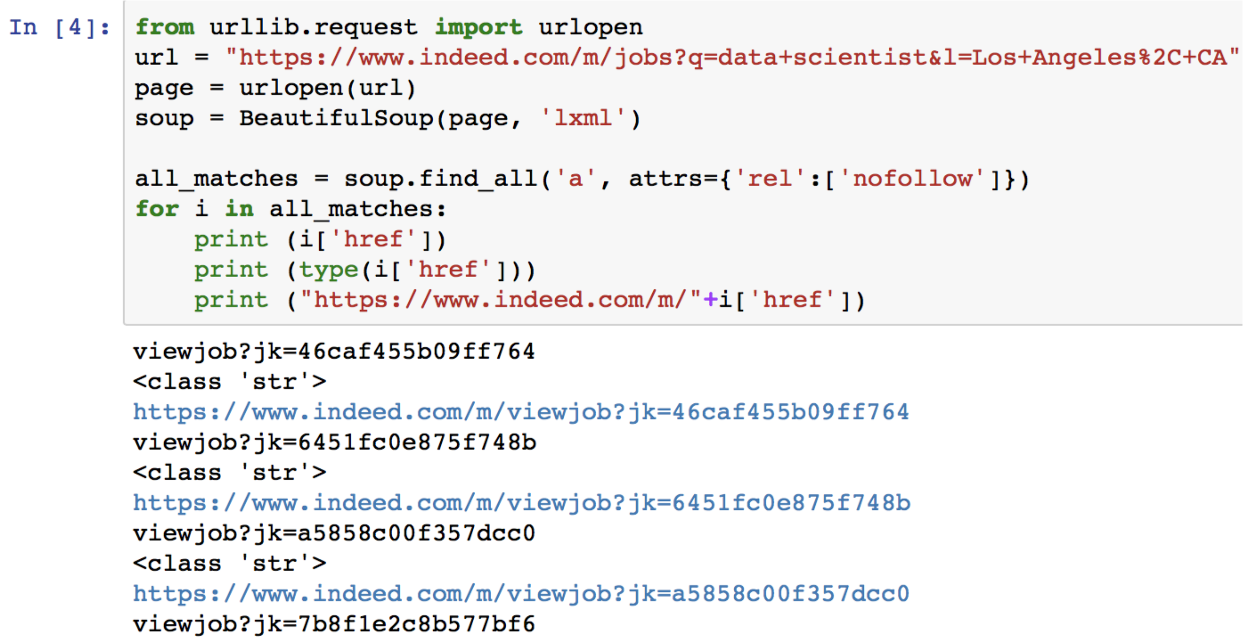
4.4 Demo

Use Python BeautifulSoup to collect and clean job listing data from indeed.com



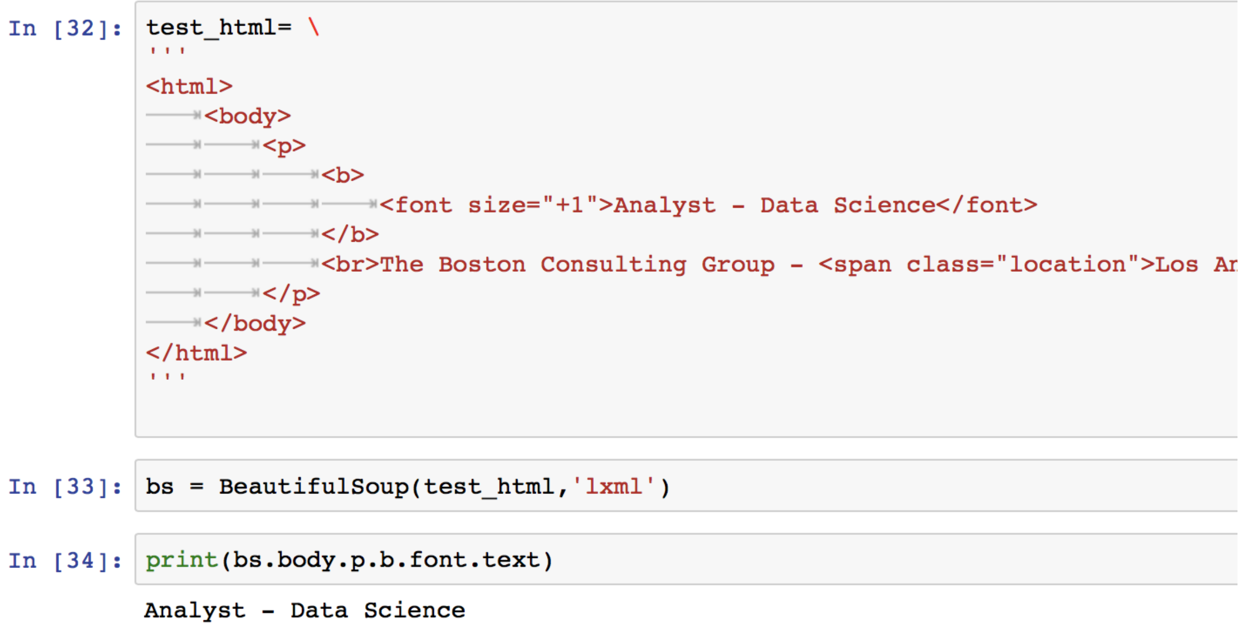
```
In [3]: ## Import the necessary packages
        from bs4 import BeautifulSoup
        import urllib
        import re
        import pandas as pd
```

Reach the link of jobs first

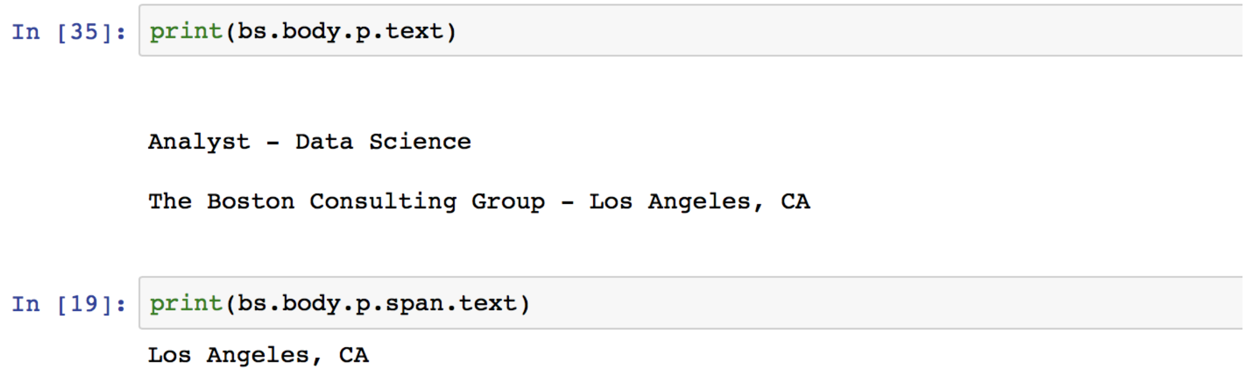


```
In [4]: from urllib.request import urlopen
        url = "https://www.indeed.com/m/jobs?q=data+scientist&l=Los+Angeles%2C+CA"
        page = urlopen(url)
        soup = BeautifulSoup(page, 'lxml')

        all_matches = soup.find_all('a', attrs={'rel':['nofollow']})
        for i in all_matches:
            print (i['href'])
            print (type(i['href']))
            print ("https://www.indeed.com/m/"+i['href'])

        viewjob?jk=46caf455b09ff764
        <class 'str'>
        https://www.indeed.com/m/viewjob?jk=46caf455b09ff764
        viewjob?jk=6451fc0e875f748b
        <class 'str'>
        https://www.indeed.com/m/viewjob?jk=6451fc0e875f748b
        viewjob?jk=a5858c00f357dcc0
        <class 'str'>
        https://www.indeed.com/m/viewjob?jk=a5858c00f357dcc0
        viewjob?jk=7b8f1e2c8b577bf6
```

Find the title, company, location and detailed job description for each job

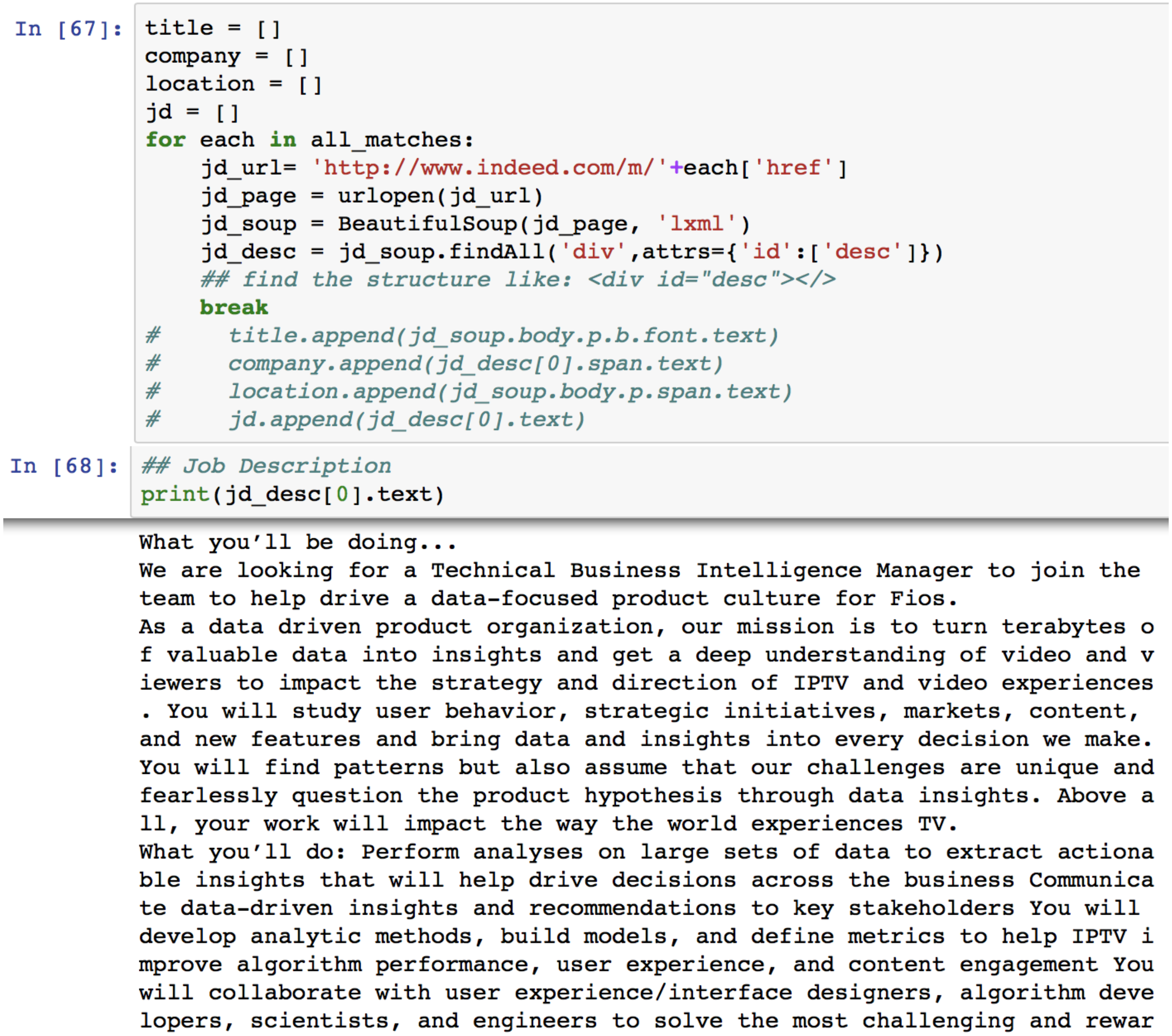


```
In [32]: test_html= \
         '''
         <html>
             <body>
                 <p>
                     <b>
                         <font size="+1">Analyst - Data Science</font>
                     </b>
                     <br>The Boston Consulting Group - <span class="location">Los An
                 </p>
             </body>
         </html>
         '''

In [33]: bs = BeautifulSoup(test_html,'lxml')

In [34]: print(bs.body.p.b.font.text)

         Analyst - Data Science
```

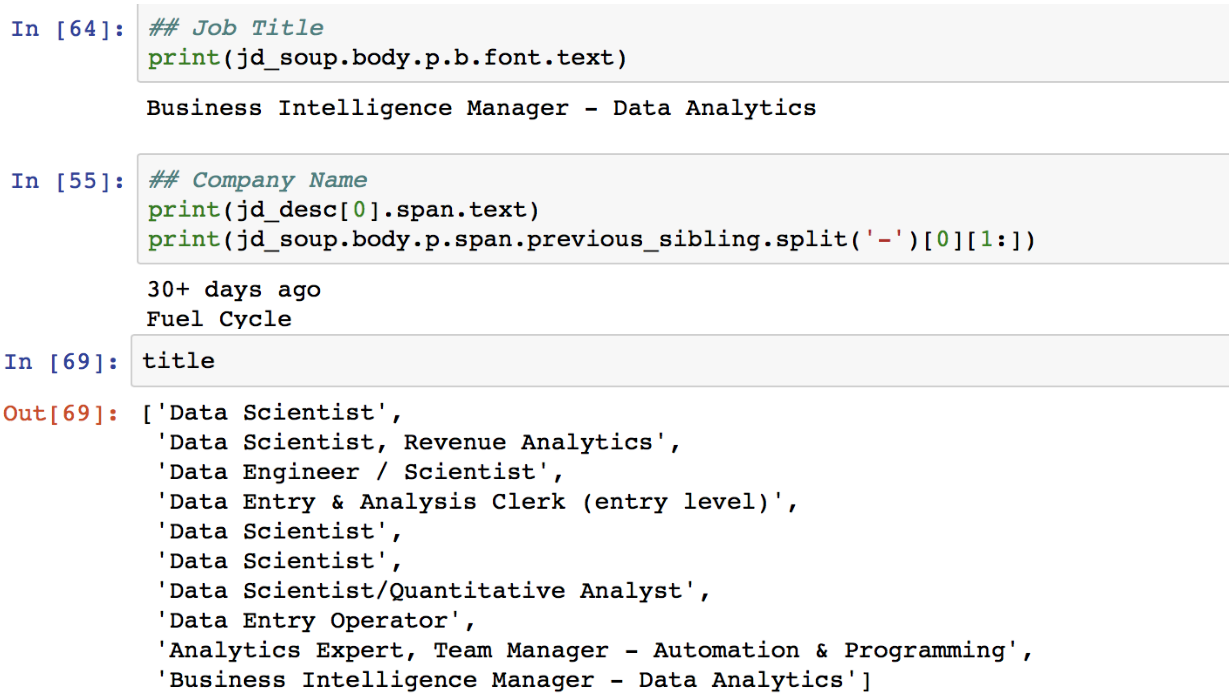


```
In [35]: print(bs.body.p.text)


         Analyst - Data Science

         The Boston Consulting Group - Los Angeles, CA


In [19]: print(bs.body.p.span.text)

         Los Angeles, CA
```

Find title, company, location and job description for one position

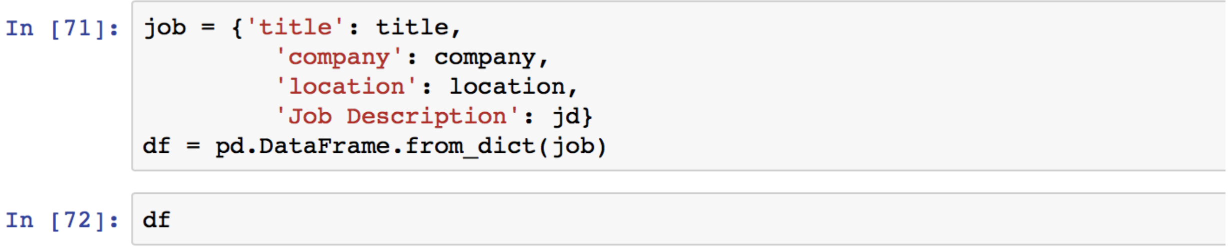


```
In [67]: title = []
         company = []
         location = []
         jd = []
         for each in all_matches:
             jd_url= 'http://www.indeed.com/m/'+each['href']
             jd_page = urlopen(jd_url)
             jd_soup = BeautifulSoup(jd_page, 'lxml')
             jd_desc = jd_soup.findAll('div',attrs={'id':['desc']})
             ## find the structure like: <div id="desc"></>
             break
         #     title.append(jd_soup.body.p.b.font.text)
         #     company.append(jd_desc[0].span.text)
         #     location.append(jd_soup.body.p.span.text)
         #     jd.append(jd_desc[0].text)

In [68]: ## Job Description
         print(jd_desc[0].text)

         What you’ll be doing...
         We are looking for a Technical Business Intelligence Manager to join the
         team to help drive a data-focused product culture for Fios.
         As a data driven product organization, our mission is to turn terabytes o
         f valuable data into insights and get a deep understanding of video and v
         iewers to impact the strategy and direction of IPTV and video experiences
         . You will study user behavior, strategic initiatives, markets, content,
         and new features and bring data and insights into every decision we make.
         You will find patterns but also assume that our challenges are unique and
         fearlessly question the product hypothesis through data insights. Above a
         ll, your work will impact the way the world experiences TV.
         What you’ll do: Perform analyses on large sets of data to extract actiona
         ble insights that will help drive decisions across the business Communica
         te data-driven insights and recommendations to key stakeholders You will
         develop analytic methods, build models, and define metrics to help IPTV i
         mprove algorithm performance, user experience, and content engagement You
         will collaborate with user experience/interface designers, algorithm deve
         lopers, scientists, and engineers to solve the most challenging and rewar
```

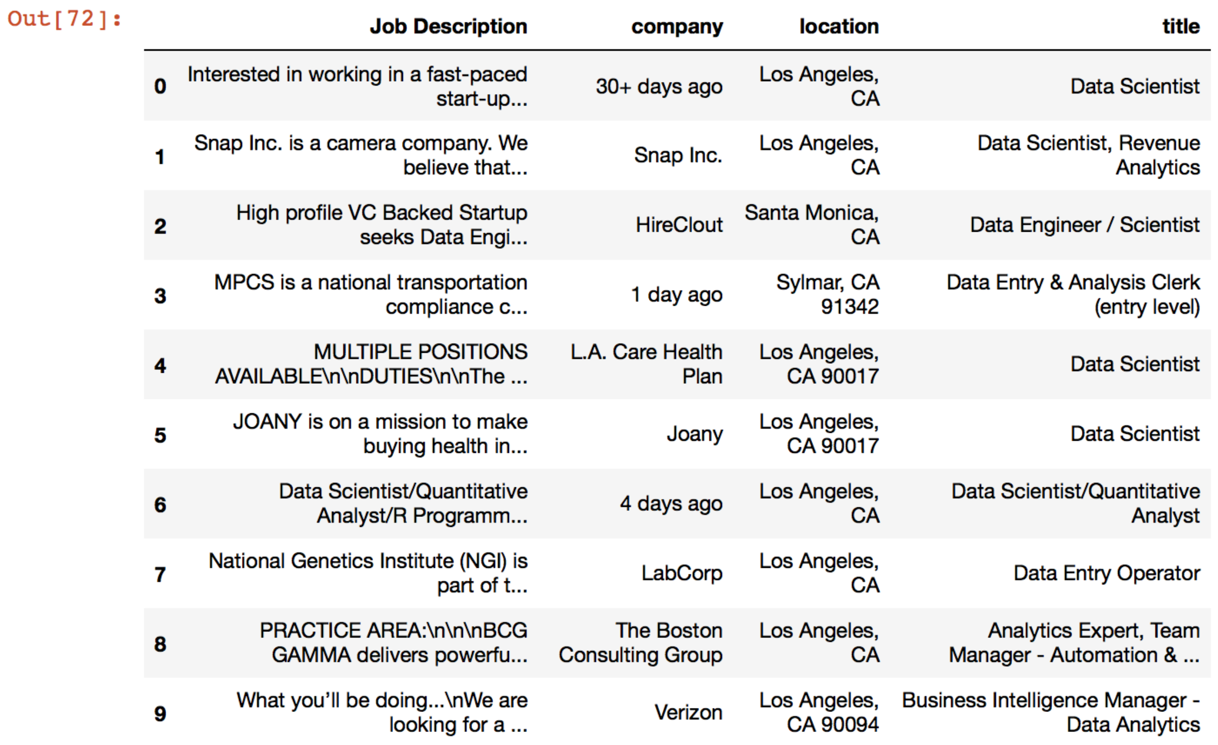


```
In [64]: ## Job Title
         print(jd_soup.body.p.b.font.text)

         Business Intelligence Manager - Data Analytics

In [55]: ## Company Name
         print(jd_desc[0].span.text)
         print(jd_soup.body.p.span.previous_sibling.split('-')[0][1:])

         30+ days ago
         Fuel Cycle
In [69]: title

Out[69]: ['Data Scientist',
          'Data Scientist, Revenue Analytics',
          'Data Engineer / Scientist',
          'Data Entry & Analysis Clerk (entry level)',
          'Data Scientist',
          'Data Scientist',
          'Data Scientist/Quantitative Analyst',
          'Data Entry Operator',
          'Analytics Expert, Team Manager - Automation & Programming',
          'Business Intelligence Manager - Data Analytics']
```

Save the data into Data Frame



```
In [71]: job = {'title': title,
                'company': company,
                'location': location,
                'Job Description': jd}
         df = pd.DataFrame.from_dict(job)

In [72]: df
```

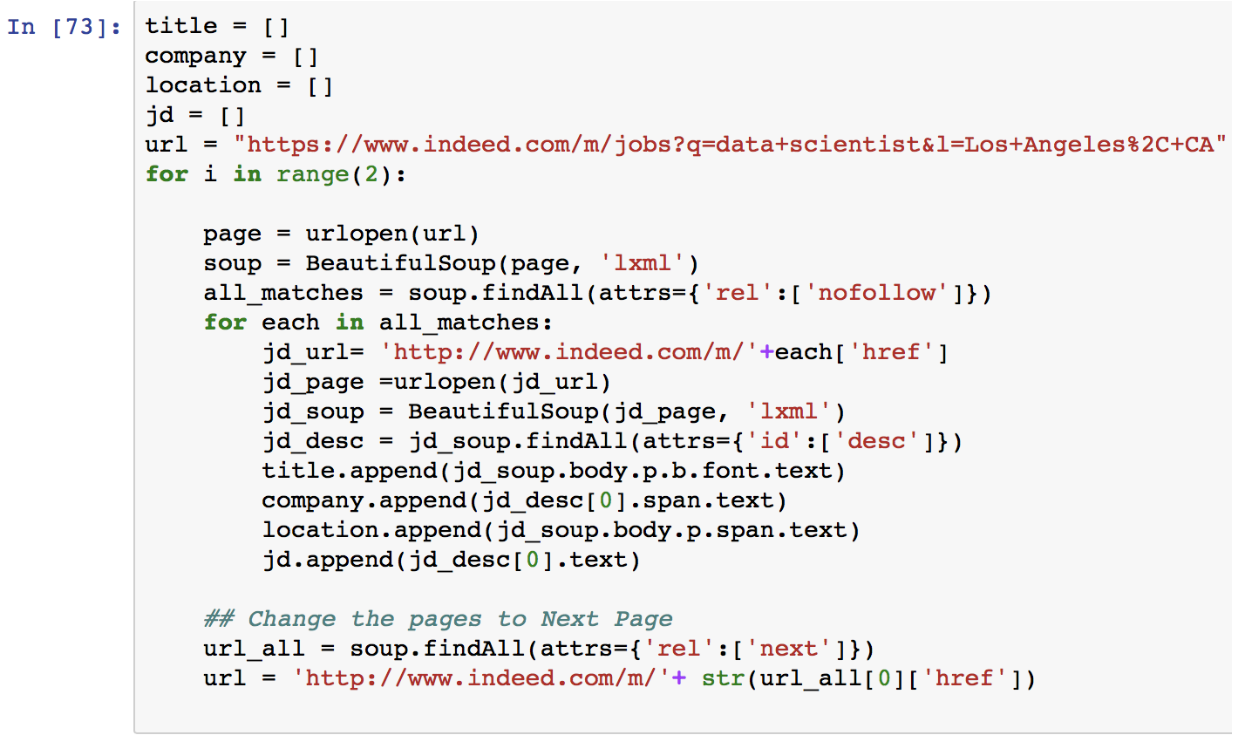


Out[72]:

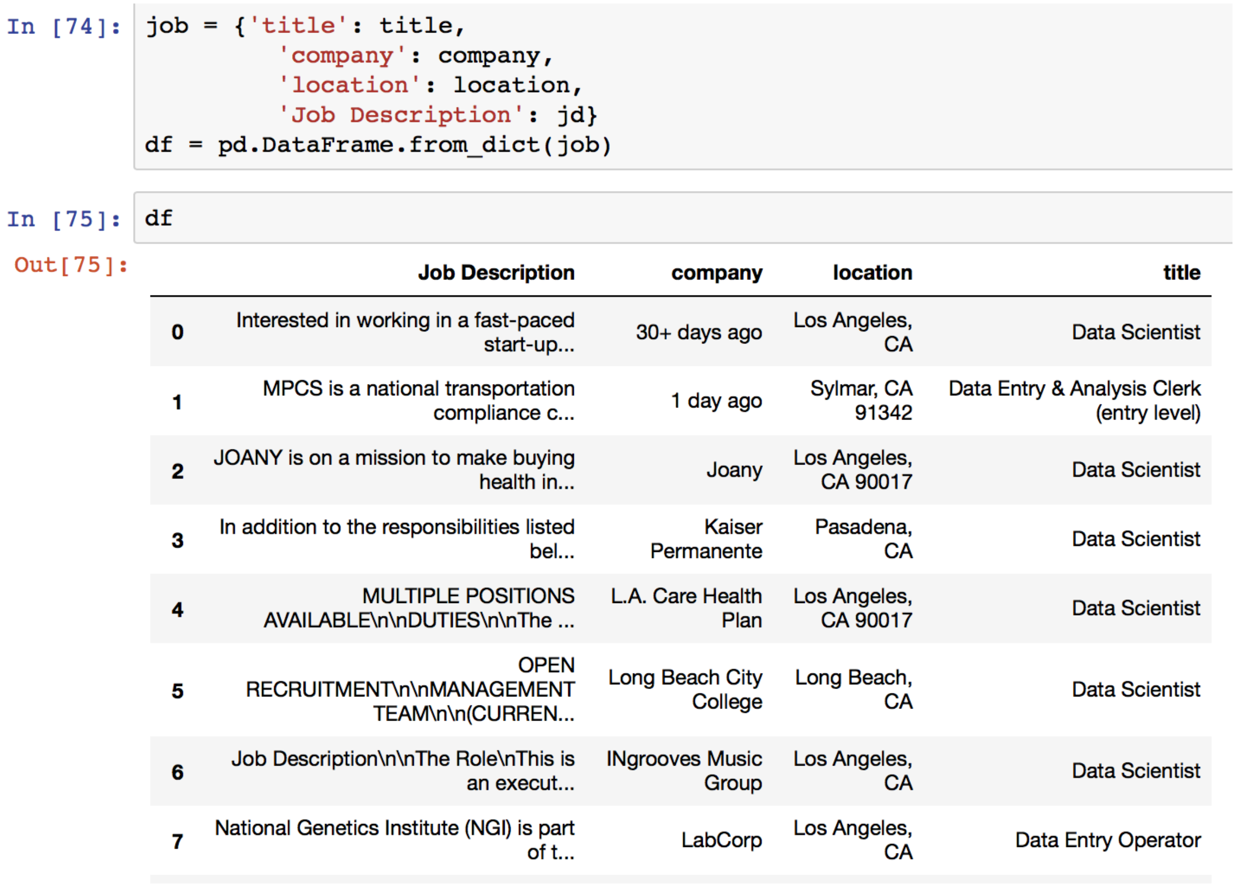
	Job Description	company	location	title
0	Interested in working in a fast-paced start-up...	30+ days ago	Los Angeles, CA	Data Scientist
1	Snap Inc. is a camera company. We believe that...	Snap Inc.	Los Angeles, CA	Data Scientist, Revenue Analytics
2	High profile VC Backed Startup seeks Data Engi...	HireClout	Santa Monica, CA	Data Engineer / Scientist
3	MPCS is a national transportation compliance c...	1 day ago	Sylmar, CA 91342	Data Entry & Analysis Clerk (entry level)
4	MULTIPLE POSITIONS AVAILABLE\n\nDUTIES\n\nThe ...	L.A. Care Health Plan	Los Angeles, CA 90017	Data Scientist
5	JOANY is on a mission to make buying health in...	Joany	Los Angeles, CA 90017	Data Scientist
6	Data Scientist/Quantitative Analyst/R Programm...	4 days ago	Los Angeles, CA	Data Scientist/Quantitative Analyst
7	National Genetics Institute (NGI) is part of t...	LabCorp	Los Angeles, CA	Data Entry Operator
8	PRACTICE AREA:\n\n\nBCG GAMMA delivers powerfu...	The Boston Consulting Group	Los Angeles, CA	Analytics Expert, Team Manager - Automation & ...
9	What you'll be doing...\nWe are looking for a ...	Verizon	Los Angeles, CA 90094	Business Intelligence Manager - Data Analytics

If we don't break the loop above, we can crawl all the job information from one page.

Change Pages Automatically



```
In [73]: title = []
         company = []
         location = []
         jd = []
         url = "https://www.indeed.com/m/jobs?q=data+scientist&l=Los+Angeles%2C+CA"
         for i in range(2):

             page = urlopen(url)
             soup = BeautifulSoup(page, 'lxml')
             all_matches = soup.findAll(attrs={'rel':['nofollow']})
             for each in all_matches:
                 jd_url= 'http://www.indeed.com/m/'+each['href']
                 jd_page =urlopen(jd_url)
                 jd_soup = BeautifulSoup(jd_page, 'lxml')
                 jd_desc = jd_soup.findAll(attrs={'id':['desc']})
                 title.append(jd_soup.body.p.b.font.text)
                 company.append(jd_desc[0].span.text)
                 location.append(jd_soup.body.p.span.text)
                 jd.append(jd_desc[0].text)

             ## Change the pages to Next Page
             url_all = soup.findAll(attrs={'rel':['next']})
             url = 'http://www.indeed.com/m/'+ str(url_all[0]['href'])
```



```
In [74]: job = {'title': title,
                'company': company,
                'location': location,
                'Job Description': jd}
         df = pd.DataFrame.from_dict(job)
```

```
In [75]: df
```

Out[75]:

	Job Description	company	location	title
0	Interested in working in a fast-paced start-up...	30+ days ago	Los Angeles, CA	Data Scientist
1	MPCS is a national transportation compliance c...	1 day ago	Sylmar, CA 91342	Data Entry & Analysis Clerk (entry level)
2	JOANY is on a mission to make buying health in...	Joany	Los Angeles, CA 90017	Data Scientist
3	In addition to the responsibilities listed bel...	Kaiser Permanente	Pasadena, CA	Data Scientist
4	MULTIPLE POSITIONS AVAILABLE\n\nDUTIES\n\nThe ...	L.A. Care Health Plan	Los Angeles, CA 90017	Data Scientist
5	OPEN RECRUITMENT\n\nMANAGEMENT TEAM\n\n(CURREN...	Long Beach City College	Long Beach, CA	Data Scientist
6	Job Description\n\nThe Role\nThis is an execut...	INgrooves Music Group	Los Angeles, CA	Data Scientist
7	National Genetics Institute (NGI) is part of t...	LabCorp	Los Angeles, CA	Data Entry Operator