Introduction

In the jupyter notebook, we have analysed a larger data set containing 35,000 records. We conducted similar analysis on this dataset like we did on the census dataset. We cleaned up the data using regular expressions and removed any duplicate rows. Using scatter plots, we try to analyse migration patterns in Scotland and by the use of a 3D plot we tried to analyse migration patterns in Scotland by region and over time.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

The additional data set downloaded from https://statistics.gov.scot/resource?uri=http%3A%2F%2Fstatistics.gov.scot%2Fdata%2Fnet-migration (https://statistics.gov.scot/resource?uri=http%3A%2F%2Fstatistics.gov.scot%2Fdata%2Fnet-migration)

In [ ]:
```
df = pd.read_csv("../data/migrationScotland.csv")
df
```

The data is cleaned up below. This process is similar to the census2011 data set. The regular expressions have been modified in the cleanupMigrationScotland.py file. The refined data is stored in a new csv file in the data/ folder

In [ ]:
```
from cleanupMigrationScotland import cleanup

df_refined = cleanup(df, "../data/migrationScotland_refined.csv")
```

The refined dataset replaces the original dataset below

In [ ]:
```
df = df_refined
df
```

Descriptive Statistics of the migration in scotland data set

counting the records in the data set

In [ ]:
```
len(df.index)
```

different types of data in the dataframe

In [ ]:
```
df.dtypes
```

In [ ]:
```
# loop through the dataframe to display unique values and occurences for each
column
for column in df:
        print (column)
        occ = df[column].value_counts()
        print (occ) # displaying the occurences
```

In [ ]:
```
import ipywidgets as widgets
```

Graph: scatter diagram of the net migration and the year

In the cell below, we have used queries to filter the dataframe and then plot a scatter diagram of the number of females migrating per year across different age groups. The graph has been made interactive using ipywidgets. We added a dropdown list of all the years and we chose to plot a scatter diagram as it can clearly be seen in which years and in which age groups the net migration was positive or negative.

In [ ]:
```
def scatterPlot (year) :
    # query to filter all female migration in scotland by year (argument)
    toPlot = (df['Sex'] == 'Female') & (df['DateCode'] == year)
    # filtering the dataset as a result of the query above
    filtered_df = df[toPlot]
    # plotting a scatter diagram with the columns
    filtered_df.plot.scatter(x= 'Value', y = 'Age', c = 'DarkBlue')
    # adding labels to our plot
    plt.ylabel('Age')
    plt.xlabel('Value')
    plt.title('Net migration in Scotland for females of different ages between
2002-2019')
    plt.show()

# making the widget
dropDown = widgets.Dropdown  (  options = sorted(df['DateCode'].unique()),
                                value = 2002,
                                description = 'Year: ',
                                disabled = False
                                )
# defining the interaction
widgets.interact(scatterPlot, year = dropDown)
```

Groupby and 3D plots

In [ ]:
```
from plot3d import flatten, generate_plot
```

In the cell below, we used groupby to group the migration by year and region. The 'FeatureCode' column represents a region

In [ ]:
```
# migration data for every council in scotland
featureCode_dateCode = df.groupby(['FeatureCode','DateCode']).Value.unique()
featureCode_dateCode_df = flatten(featureCode_dateCode, "FeatureCode", "DateCo
de","Value")
```

In the cell below, we add all the values across different sexes and age groups for each region and every year. We store this data in a different column after dropping the column which had all the values stored in an array. Storing them as integers in a different column allows us to plot them in a 3D plot easily.

In [ ]:
```
values = [] # stores the sum of all the values in a specific year for a refion
in this array
# loop through to find the column
for column in featureCode_dateCode_df :
    if(column == 'Value') :
        # loop through the arrays in the column: 'Value'
        for data in featureCode_dateCode_df['Value']:
            sum = 0 # variable to store the sum of the array
            # loop through values in the array
            for i in range(0,len(data)) :
                sum += data[i]
            values.extend([sum]) # add sum to the values array
# remove the column which had migration data as array
featureCode_dateCode_df = featureCode_dateCode_df.drop(columns = ['Value'])
# add values[] which has the sums stored
featureCode_dateCode_df['Values'] = values
# display the current dataframe
featureCode_dateCode_df
```

In the final cell, we produce a 3D plot of the dataframe produced after using groupby

In [ ]:
```
# generate plot with the x-axis having the regions, y-axis having the year and
the z-axis with the values
generate_plot("Records by FeatureCode and DateCode", featureCode_dateCode_df,
"FeatureCode", "DateCode", "Values")
```

Conclusion

The analysis of the additional data set has mostly been completed. We haven't analysed using maps in this data set due to time constraints. If we had more time, we would have analysed this dataset using maps and improved the display of the final 3D plot.