```python
# Assessed exercises 6

# Import packages
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import numpy.random as npr

# This week we will use the san-francisco-2018 dataset to test the code. It
# contains salary information for over 40,000 workers in San Francisco. I have
# taken a subset of 500 of the entries from this dataset, and removed some of t
# the entries to create NaN entries. Download the dataset, load it in and have
# a look at the first few entries to see what it looks like.
data = pd.read_csv('san-francisco-2018.csv',index_col='Name')

# Q1 Write a function that will calculate the column means for a given DataFrame
# df and returns the DataFrame with the column means removed (subtracted)
def exercise1(df):
    mean=df.mean()
    return df.sub(mean)
# Suggested test
# Take a small subset of the data, and drop the categorical data
dataQ1 = data.drop(['Job Title','Status'],axis=1).iloc[80:100]
exercise1(dataQ1)
# This should return a DataFrame with 20 rows and 4 columns, with the difference
# from the mean for each measurement, for each person

# Q2 Write a function that computes summary statistics for a DataFrame df. The
# function should return a DataFrame with the summary statistics, mean, standard
# deviation, minimum, maximum, the index for the first minimum and the index
# for the first maximum. The outputted DataFrame should have 6 rows, one for
# each piece of information listed above, labelled 'mean', 'std', 'min', 'max',
# 'minloc', 'maxloc' (in this exact order). The columns should be the same as
# those in the original DataFrame (in the same order). You can assume that df
# does not contain categorical data
def exercise2(df):
    df2=pd.DataFrame(columns=(df.columns[0],df.columns[1],df.columns[2],df.columns[3]))
    df2.loc['mean']=df.mean()
    df2.loc['std']=df.std()
    df2.loc['min']=df.min()
    df2.loc['max']=df.max()
    df2.loc['minloc']=df.idxmin()
    df2.loc['maxloc']=df.idxmax()
    return df2
# Suggested test
# Remove the categorical data. Once you've completed Q4 you should have a
# generalisable way to remove categorical data
dataQ2 = data.drop(['Job Title','Status'],axis=1)
exercise2(dataQ2)
# This should give you back a DataFrame with 6 row and 4 columns, containing
# the mean, std, min, max, minloc, maxloc (in that order) for Base Pay, Overtime
# Pay, Other Pay and Benefits. minloc and maxloc should contain the name of the
# person with the min/max Base Pay, Overtime Pay, etc.


# Q3 Write a function that will return the index of the 3 highest entries for
# each of the columns in a DataFrame df. The function should return a DataFrame,
# where the rows are the columns of the DataFrame df, and the columns labelled
# '1st', '2nd' and '3rd' contain the index label of the 3 highest entries in
# the given column. Again, you can assume that df does not contain categorical
# data
```

```
def exercise3(df):
    series1=df[df.columns[0]].sort_values(ascending=False)
    series2=df[df.columns[1]].sort_values(ascending=False)
    series3=df[df.columns[2]].sort_values(ascending=False)
    series4=df[df.columns[3]].sort_values(ascending=False)
    dfq3=pd.DataFrame(columns=('1st','2nd','3rd'))
    dfq3.loc[df.columns[0]]=[series1[0],series1[1],series1[2]]
    dfq3.loc[df.columns[1]]=[series2[0],series2[1],series2[2]]
    dfq3.loc[df.columns[2]]=[series3[0],series3[1],series3[2]]
    dfq3.loc[df.columns[3]]=[series4[0],series4[1],series4[2]]
    return dfq3
# Suggested test
# We can use the same data from Q2
exercise3(dataQ2)
# This should return a DataFrame with 4 rows and 3 columns, where ther rows are
# Base Pay, Overtime Pay, Other Pay and Benefits, and the columns 1st, 2nd and
# 3rd. The entries should be the names of the employees with the highest, second
# highest and third highest Base Pay, Overtime Pay, etc. Look at the DataFrame
# dataQ2 to ensure the function is returning the correct information.

# Q4 In this question you need to write a function to replace all of the NaNs
# in a DataFrame df with the mean of the column for numeric data and the mode
# of the column for categorical data. If a column of categorical data has more
# than one mode you should use the first one. The function should return df with
# the missing values replaced as outlined above. This function must work for any
# DataFrame, so you cannot use the column names inside the function. You can
# assume that a column won't have both numerical and categorical data in it.
# Hint: You'll need to figure out how to select columns based on their data type
# and then use drop to make the replacements for numeric and categorical data
# separately.
def exercise4(df):
    df=df.fillna(df.select_dtypes(['number']).mean())
    df=df.fillna(df.select_dtypes(['object']).mode().iloc[0])
    return df

# Suggested test
# Take a subset of the data, look at the data and see that there are NaN entries
dataQ4 = data.iloc[150:180]
exercise4(dataQ4)
# This should return a DataFrame with the same dimensions and values as dataQ4,
# with all of the NaN entries filled.
# This function could now be used to replace all of the NaNs for the entire
# DataFrame, or any DataFrame for that matter
```