```python
# Assessed exercises 6

# Import packages
import pandas as pd
import numpy as np

# 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):
    return df.sub(df.mean(axis=0))


# 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
```

```
29
30 # Q2 Write a function that computes summary statistics for a DataFrame df. The
31 # function should return a DataFrame with the summary statistics, mean, standard
32 # deviation, minimum, maximum, the index for the first minimum and the index
33 # for the first maximum. The outputted DataFrame should have 6 rows, one for
34 # each piece of information listed above, labelled 'mean', 'std', 'min', 'max',
35 # 'minloc', 'maxloc' (in this exact order). The columns should be the same as
36 # those in the original DataFrame (in the same order). You can assume that df
37 # does not contain categorical data
38 def exercise2(df):
39     df_mean = df.mean(axis=0)
40     df_std = df.std(axis=0)
41     df_min = df.min(axis=0)
42     df_max = df.max(axis=0)
43     df_minloc = df.idxmin(axis=0)
44     df_maxloc = df.idxmax(axis=0)
45     return pd.DataFrame([df_mean, df_std, df_min, df_max, df_minloc, df_maxloc],
   columns=df.columns)
46
47
48 # Suggested test
49 # Remove the categorical data. Once you've completed Q4 you should have a
50 # generalisable way to remove categorical data
51 dataQ2 = data.drop(['Job Title', 'Status'], axis=1)
52 exercise2(dataQ2)
53
54
55 # This should give you back a DataFrame with 6 row and 4 columns, containing
```

```
56 # the mean, std, min, max, minloc, maxloc (in that order) for Base Pay, Overtime
57 # Pay, Other Pay and Benefits. minloc and maxloc should contain the name of the
58 # person with the min/max Base Pay, Overtime Pay, etc.
59
60
61 # Q3 Write a function that will return the index of the 3 highest entries for
62 # each of the columns in a DataFrame df. The function should return a DataFrame,
63 # where the rows are the columns of the DataFrame df, and the columns labelled
64 # '1st', '2nd' and '3rd' contain the index label of the 3 highest entries in
65 # the given column. Again, you can assume that df does not contain categorical
66 # data
67 def exercise3(df):
68     df_copy = df.copy()
69     st = df_copy.idxmax(axis=0)
70     for i, val in enumerate(st):
71         df_copy.loc[val, st.index[i]] = float('-inf')
72     nd = df_copy.idxmax(axis=0)
73     for i, val in enumerate(nd):
74         df_copy.loc[val, st.index[i]] = float('-inf')
75     rd = df_copy.idxmax(axis=0)
76     for i, val in enumerate(rd):
77         df_copy.loc[val, st.index[i]] = float('-inf')
78     df_new = pd.DataFrame([st, nd, rd]).T
79     df_new.columns = ['1st', '2nd', '3rd']
80     return df_new
81
82
83 # Suggested test
```

```
 84 # We can use the same data from Q2
 85 exercise3(dataQ2)
 86
 87
 88 # This should return a DataFrame with 4 rows and 3 columns, where ther rows are
 89 # Base Pay, Overtime Pay, Other Pay and Benefits, and the columns 1st, 2nd and
 90 # 3rd. The entries should be the names of the employees with the highest, second
 91 # highest and third highest Base Pay, Overtime Pay, etc. Look at the DataFrame
 92 # dataQ2 to ensure the function is returning the correct information.
 93
 94 # Q4 In this question you need to write a function to replace all of the NaNs
 95 # in a DataFrame df with the mean of the column for numeric data and the mode
 96 # of the column for categorical data. If a column of categorical data has more
 97 # than one mode you should use the first one. The function should return df with
 98 # the missing values replaced as outlined above. This function must work for any
 99 # DataFrame, so you cannot use the column names inside the function. You can
100 # assume that a column won't have both numerical and categorical data in it.
101 # Hint: You'll need to figure out how to select columns based on their data type
102 # and then use drop to make the replacements for numeric and categorical data
103 # separately.
104 def exercise4(df):
105     df_copy = df.copy()
106     for i in range(df_copy.shape[1]):
107         if np.issubdtype(df_copy.iloc[:, i].dtype, np.number):
108             df_copy.iloc[:, i].fillna(df_copy.iloc[:, i].mean(), inplace=True)
109         else:
110             df_copy.iloc[:, i].fillna(df_copy.iloc[df_copy.index.get_loc(df.iloc
    [:, i].first_valid_index()), i],
```

```
111                                             inplace=True)
112     return df_copy
113
114
115 # Suggested test
116 # Take a subset of the data, look at the data and see that there are NaN entries
117 dataQ4 = data.iloc[150:180]
118 exercise4(dataQ4)
119 # This should return a DataFrame with the same dimensions and values as dataQ4,
120 # with all of the NaN entries filled.
121 # This function could now be used to replace all of the NaNs for the entire
122 # DataFrame, or any DataFrame for that matter
123
```