```
import pandas as pd
import numpy as np
```

```
C:\Users\aakas\AppData\Local\Temp\ipykernel_6692\2162656668.py:1:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type,
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at
https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

```
salary = pd.read_csv("assignment3-part-1-dataset1.csv")

store = pd.read_csv("assignment3-part-1-dataset2.csv")

salary.head()
```

```
   Unnamed: 0                Company Name       Job Title  Salaries
Reported  \
0           0                     Mu Sigma  Data Scientist
105
1           1                         IBM  Data Scientist
95
2           2  Tata Consultancy Services  Data Scientist
66
3           3           Impact Analytics  Data Scientist
40
4           4                   Accenture  Data Scientist
32

    Location     Salary
0  Bangalore   648573.0
1  Bangalore  1191950.0
2  Bangalore   836874.0
3  Bangalore   669578.0
4  Bangalore   944110.0
```

```
salary.replace("?", np.nan,inplace=True)

salary.isnull().sum()
```

```
Unnamed: 0           0
Company Name         0
Job Title            0
Salaries Reported    0
Location             0
```

```
Salary                0
dtype: int64
```

```
salary.groupby("Job Title")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x000001EBAF00B5F0>
```

```
salary.head()
```

```
   Unnamed: 0                Company Name       Job Title  Salaries
Reported  \
0           0                     Mu Sigma  Data Scientist
105
1           1                          IBM  Data Scientist
95
2           2  Tata Consultancy Services  Data Scientist
66
3           3            Impact Analytics  Data Scientist
40
4           4                   Accenture  Data Scientist
32

    Location     Salary
0  Bangalore   648573.0
1  Bangalore  1191950.0
2  Bangalore   836874.0
3  Bangalore   669578.0
4  Bangalore   944110.0
```

```
mean = salary.groupby("Job Title")["Salary"].mean()
print(mean)
```

```
Job Title
Associate Machine Learning Engineer     4.643720e+05
Data Analyst                            6.164699e+05
Data Engineer                           1.309051e+06
Data Science                            3.649053e+05
Data Science Associate                  1.203913e+06
Data Science Consultant                 2.671464e+06
Data Science Lead                       4.068310e+06
Data Science Manager                    4.619021e+06
Data Scientist                          1.411330e+06
Data Scientist - Trainee                6.105120e+05
Junior Data Scientist                   5.963231e+05
Lead Data Scientist                     1.852189e+06
Machine Learning Associate              2.951140e+05
Machine Learning Consultant             7.064010e+05
Machine Learning Data Analyst           3.613780e+05
Machine Learning Data Associate         2.758410e+05
Machine Learning Data Associate I       2.585960e+05
```

```
Machine Learning Data Associate II      3.832130e+05
Machine Learning Developer              5.811190e+05
Machine Learning Engineer               7.971884e+05
Machine Learning Scientist              1.701180e+05
Machine Learning Software Engineer      1.397347e+06
Senior Data Scientist                   1.766130e+06
Senior Machine Learning Engineer        1.473436e+06
Software Engineer - Machine Learning    1.566780e+06
Name: Salary, dtype: float64
```

```
median = salary.groupby("Job Title")["Salary"].median()
print(median)
```

```
Job Title
Associate Machine Learning Engineer      464372.0
Data Analyst                              508150.5
Data Engineer                             792683.0
Data Science                              240780.0
Data Science Associate                   1203913.0
Data Science Consultant                  2671464.0
Data Science Lead                        4068310.0
Data Science Manager                     4619021.0
Data Scientist                            914480.0
Data Scientist - Trainee                  610512.0
Junior Data Scientist                     554963.0
Lead Data Scientist                      1664364.0
Machine Learning Associate                295114.0
Machine Learning Consultant               706401.0
Machine Learning Data Analyst             361378.0
Machine Learning Data Associate           275841.0
Machine Learning Data Associate I         258596.0
Machine Learning Data Associate II        383213.0
Machine Learning Developer                581119.0
Machine Learning Engineer                 627048.5
Machine Learning Scientist                170118.0
Machine Learning Software Engineer       1397347.0
Senior Data Scientist                    1733388.0
Senior Machine Learning Engineer         1335445.0
Software Engineer - Machine Learning     1566780.0
Name: Salary, dtype: float64
```

```
mode = salary.groupby("Job Title")["Salary"].apply(lambda
x:x.mode().iloc[0])
print(mode)

# lambda x: This defines an anonymous function (lambda function) that
takes one argument x.
# In this context, x represents each group of salaries within each job
title.
```

```
# x.mode(): Inside the lambda function, x is a Series object
containing all the salaries within a specific job title group.
# The mode() function is called on this Series object to compute the
mode, i.e., the most frequently occurring value.

# .iloc[0]: After calculating the mode, .iloc[0] is used to retrieve
the first value from the resulting Series.
# This is necessary because the mode() function may return multiple
values if there are ties for the most frequent value.
# By selecting the first value, we ensure that only one mode value is
returned.
```

```
Job Title
Associate Machine Learning Engineer      464372.0
Data Analyst                              338792.0
Data Engineer                             515940.0
Data Science                              180000.0
Data Science Associate                   1203913.0
Data Science Consultant                  2671464.0
Data Science Lead                        4068310.0
Data Science Manager                     4619021.0
Data Scientist                            600000.0
Data Scientist - Trainee                  610512.0
Junior Data Scientist                     616492.0
Lead Data Scientist                      1520967.0
Machine Learning Associate                295114.0
Machine Learning Consultant               186475.0
Machine Learning Data Analyst             361378.0
Machine Learning Data Associate           275841.0
Machine Learning Data Associate I         258596.0
Machine Learning Data Associate II        383213.0
Machine Learning Developer                410952.0
Machine Learning Engineer                 128988.0
Machine Learning Scientist                 62160.0
Machine Learning Software Engineer       1397347.0
Senior Data Scientist                    2474429.0
Senior Machine Learning Engineer          229416.0
Software Engineer - Machine Learning     1521236.0
Name: Salary, dtype: float64
```

```
minimum = salary.groupby("Job Title")["Salary"].min()
print(minimum)
```

```
Job Title
Associate Machine Learning Engineer      464372.0
Data Analyst                               10814.0
Data Engineer                              33120.0
Data Science                               60840.0
Data Science Associate                   1203913.0
Data Science Consultant                  2671464.0
```

```
Data Science Lead                          4068310.0
Data Science Manager                       4619021.0
Data Scientist                               48000.0
Data Scientist - Trainee                    610512.0
Junior Data Scientist                        60840.0
Lead Data Scientist                        1520967.0
Machine Learning Associate                  295114.0
Machine Learning Consultant                 186475.0
Machine Learning Data Analyst               361378.0
Machine Learning Data Associate             275841.0
Machine Learning Data Associate I           258596.0
Machine Learning Data Associate II          383213.0
Machine Learning Developer                  410952.0
Machine Learning Engineer                    21628.0
Machine Learning Scientist                   62160.0
Machine Learning Software Engineer         1397347.0
Senior Data Scientist                       324089.0
Senior Machine Learning Engineer            229416.0
Software Engineer - Machine Learning       1521236.0
Name: Salary, dtype: float64
```

```
maximum = salary.groupby("Job Title")["Salary"].max()
print(maximum)
```

```
Job Title
Associate Machine Learning Engineer     4.643720e+05
Data Analyst                            3.900962e+07
Data Engineer                           1.190400e+08
Data Science                            2.000000e+06
Data Science Associate                  1.203913e+06
Data Science Consultant                 2.671464e+06
Data Science Lead                       4.068310e+06
Data Science Manager                    4.619021e+06
Data Scientist                          1.661404e+08
Data Scientist - Trainee                6.105120e+05
Junior Data Scientist                   1.498750e+06
Lead Data Scientist                     2.839138e+06
Machine Learning Associate              2.951140e+05
Machine Learning Consultant             1.226327e+06
Machine Learning Data Analyst           3.613780e+05
Machine Learning Data Associate         2.758410e+05
Machine Learning Data Associate I       2.585960e+05
Machine Learning Data Associate II      3.832130e+05
Machine Learning Developer              7.512860e+05
Machine Learning Engineer               6.518917e+06
Machine Learning Scientist              2.780760e+05
Machine Learning Software Engineer      1.397347e+06
Senior Data Scientist                   3.654010e+06
Senior Machine Learning Engineer        3.110514e+06
```

```
Software Engineer - Machine Learning    1.612324e+06
Name: Salary, dtype: float64
```

```
std = salary.groupby("Job Title")["Salary"].std()
print(std)
```

```
Job Title
Associate Machine Learning Engineer              NaN
Data Analyst                            1.292116e+06
Data Engineer                           6.009190e+06
Data Science                            3.388020e+05
Data Science Associate                           NaN
Data Science Consultant                          NaN
Data Science Lead                                NaN
Data Science Manager                             NaN
Data Scientist                          5.140558e+06
Data Scientist - Trainee                         NaN
Junior Data Scientist                   3.931792e+05
Lead Data Scientist                     5.017356e+05
Machine Learning Associate                       NaN
Machine Learning Consultant             7.352864e+05
Machine Learning Data Analyst                    NaN
Machine Learning Data Associate                  NaN
Machine Learning Data Associate I                NaN
Machine Learning Data Associate II               NaN
Machine Learning Developer              2.406525e+05
Machine Learning Engineer               7.047460e+05
Machine Learning Scientist              1.526757e+05
Machine Learning Software Engineer               NaN
Senior Data Scientist                   7.833905e+05
Senior Machine Learning Engineer        9.506370e+05
Software Engineer - Machine Learning    6.440894e+04
Name: Salary, dtype: float64
```

```
salary_freqency = salary.groupby("Job Title")["Salary"].value_counts()
salary_frequency_more_1 = salary_freqency[salary_freqency > 1]
print(salary_frequency_more_1)
```

```
Job Title                  Salary
Data Analyst               338792.0     6
                           203280.0     5
                           360000.0     5
                           521474.0     5
                           537441.0     5
                                       ..
Machine Learning Engineer  1244206.0    2
                           1500000.0    2
                           2017951.0    2
                           2063316.0    2
```

```
Senior Data Scientist      2474429.0    2
Name: count, Length: 442, dtype: int64

filtered_salary = salary[salary["Job
Title"].isin(salary_frequency_more_1)]
std_filtered = filtered_salary.groupby("Job Title")["Salary"].std()

print(std_filtered)

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call
last)
File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\indexes\base.py:3802, in Index.get_loc(self, key)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File index.pyx:153, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:182, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Salary'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
Cell In[44], line 1
----> 1 filtered_salary = salary[salary["Job
Title"].isin(salary_frequency_more_1["Salary"])]
      2 std_filtered = filtered_salary.groupby("Job Title")
["Salary"].std()
      4 print(std_filtered)

File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\series.py:1111, in Series.__getitem__(self, key)
   1108     return self._values[key]
   1110 elif key_is_scalar:
-> 1111     return self._get_value(key)
   1113 # Convert generator to list before going through hashable part
   1114 # (We will iterate through the generator there to check for
slices)
   1115 if is_iterator(key):
```

```
File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\series.py:1227, in Series._get_value(self, label,
takeable)
   1224     return self._values[label]
   1226 # Similar to Index.get_value, but we do not fall back to
positional
-> 1227 loc = self.index.get_loc(label)
   1229 if is_integer(loc):
   1230     return self._values[loc]

File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\indexes\multi.py:3040, in
MultiIndex.get_loc(self, key)
   3037     return mask
   3039 if not isinstance(key, tuple):
-> 3040     loc = self._get_level_indexer(key, level=0)
   3041     return _maybe_to_slice(loc)
   3043 keylen = len(key)

File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\indexes\multi.py:3391, in
MultiIndex._get_level_indexer(self, key, level, indexer)
   3388         return slice(i, j, step)
   3390 else:
-> 3391     idx = self._get_loc_single_level_index(level_index, key)
   3393     if level > 0 or self._lexsort_depth == 0:
   3394         # Desired level is not sorted
   3395         if isinstance(idx, slice):
   3396             # test_get_loc_partial_timestamp_multiindex

File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\indexes\multi.py:2980, in
MultiIndex._get_loc_single_level_index(self, level_index, key)
   2978     return -1
   2979 else:
-> 2980     return level_index.get_loc(key)

File c:\Users\aakas\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pandas\core\indexes\base.py:3809, in Index.get_loc(self, key)
   3804     if isinstance(casted_key, slice) or (
   3805         isinstance(casted_key, abc.Iterable)
   3806         and any(isinstance(x, slice) for x in casted_key)
   3807     ):
   3808         raise InvalidIndexError(key)
-> 3809     raise KeyError(key) from err
   3810 except TypeError:
   3811     # If we have a listlike key, _check_indexing_error will
raise
   3812     #  InvalidIndexError. Otherwise we fall through and re-
```

```
raise
   3813     #  the TypeError.
   3814     self._check_indexing_error(key)

KeyError: 'Salary'
```