D599 Task 1

Part A: *Review the data dictionary in the attached "Employee Turnover Considerations and Dictionary" document and provide profile data by doing the following within a document:*

1. *Identify the dataset's number of records (i.e., rows) and number of variables (i.e., columns).*

The dataset has 10,199 rows and 16 columns. Each row represents a different employee and each column represents a different attribute.

2. *List each variable and indicate the variable's data type (quantitative/numerical or qualitative/categorical) and data subtype (i.e., continuous/discrete or nominal/ordinal).*

Below is a chart defining each variable in the dataset, its data type, and its subtype

Variable	Data Type	Data Subtype
EmployeeNumber	Integer	Nominal (ID)
Age	Integer	Discrete Numeric
Tenure	Integer	Discrete Numeric
Turnover	Object	Nominal (Categorical)
HourlyRate	Float	Continuous Numeric
HoursWeekly	Integer	Discrete Numeric
CompensationType	Object	Nominal (Categorical)
AnnualSalary	Float	Continuous Numeric
DrivingCommuterDistance	Integer	Discrete Numeric
JobRoleArea	Object	Nominal (Categorical)
Gender	Object	Nominal (Categorical)
MaritalStatus	Object	Nominal (Categorical)
NumCompaniesPreviouslyWorked	Integer	Discrete Numeric
AnnualProfessionalDevHours	Integer	Continuous Numeric

PaycheckMethod	Object	Nominal (Categorical)
TextMessageOptIn	Object	Nominal (Categorical)

3. *Identify a sample of observable values for each variable.*

EmployeeNumber:	738
Age:	61
Tenure:	14
Turnover:	No
HourlyRate:	$82.96
HoursWeekly:	40
CompensationType:	Salary
AnnualSalary:	172556.8
DrivingCommuterDistance:	10
JobRoleArea:	InformationTechnology
Gender:	Male
MaritalStatus:	Divorced
NumCompaniesWorkedPreviously:	9
AnnualProfessionalDevHours:	25
PaycheckMethod:	Mail Check
TextMessageOptIn:	Yes

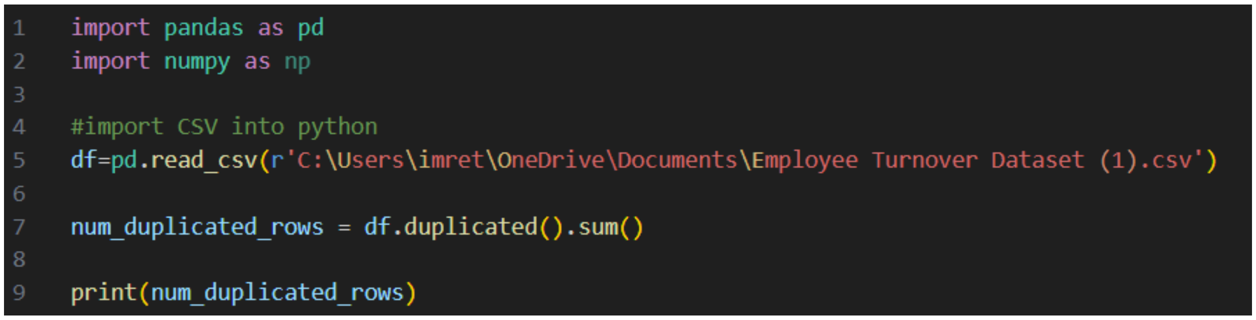
Part B: *Using Python or R code data cleaning techniques do the following:*

1. *Explain how you inspected the dataset to detect the following data quality issues:*

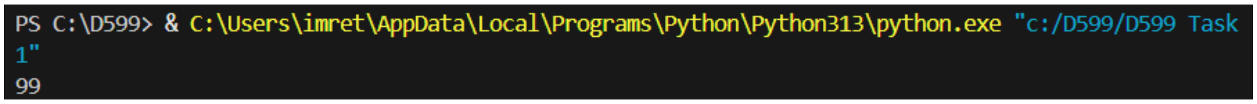
Duplicate Entries: Used df.duplicated().sum() to count all duplicated rows in dataset
Missing Values: Used df.isna().sum() to count how many empty cells are in each column
Inconsistent Entries: Used Python to standardize text fields by removing unnecessary symbol and space characters. Also identified several unusual data points, such as negative commute distances and negative annual salaries. Removed the negative sign from these negative values
Formatting Errors: Used Python to ensure data types matched the desired data types listed in Chart from A2.
Outliers: Used pandas and numpy to identify what values in the table were greater than 3 standard deviations away from the columns mean. Looking at the data, it seems as if these outliers simply had a 0 stuck on the end of them at some point, so I divided the outliers by 10.

2. *Discuss your findings for each quality issue listed in part B1.*

Duplicates: 99 duplicate rows were found in the data set.

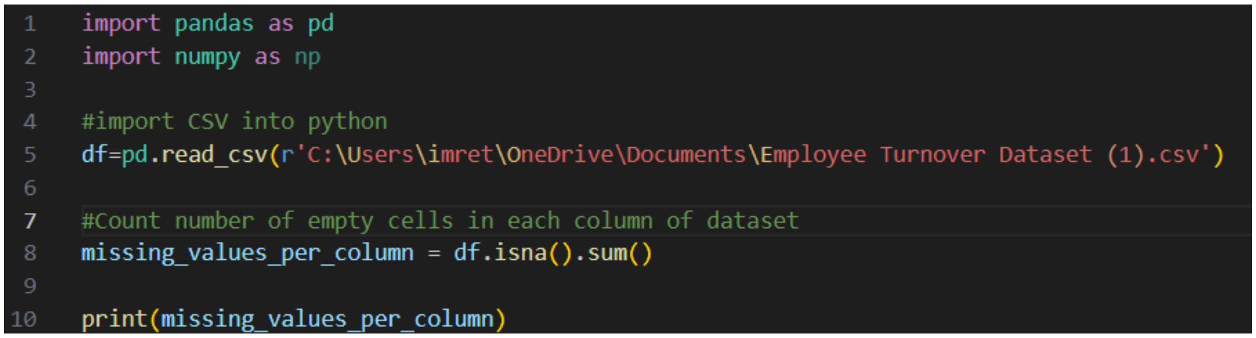


```python
import pandas as pd
import numpy as np

#import CSV into python
df=pd.read_csv(r'C:\Users\imret\OneDrive\Documents\Employee Turnover Dataset (1).csv')

num_duplicated_rows = df.duplicated().sum()

print(num_duplicated_rows)
```

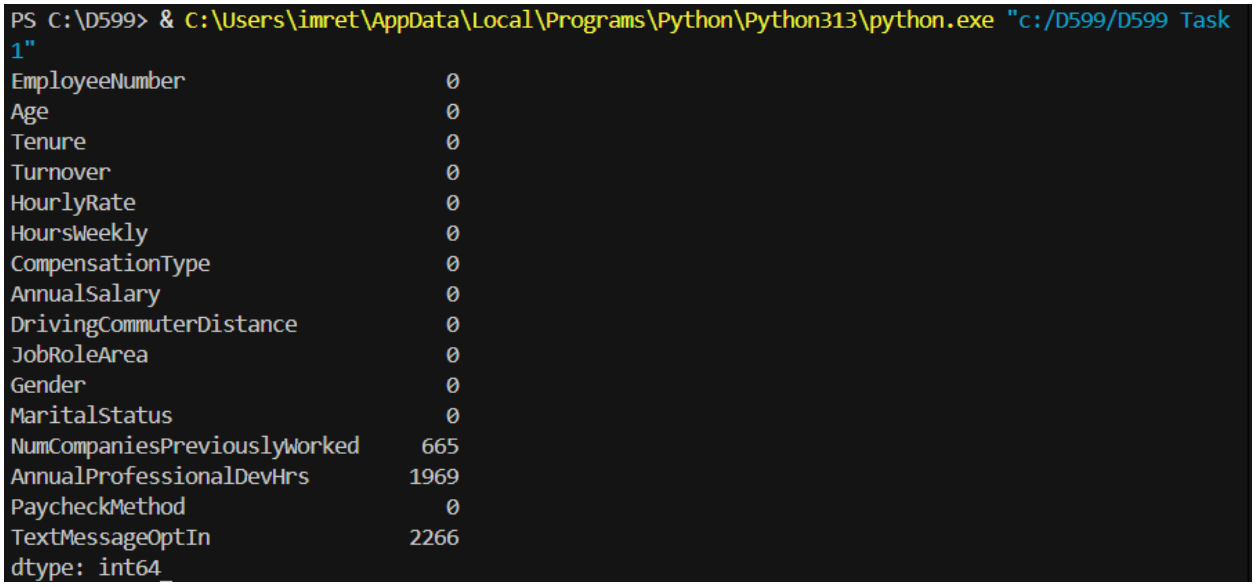


```
PS C:\D599> & C:\Users\imret\AppData\Local\Programs\Python\Python313\python.exe "c:/D599/D599 Task
1"
99
```

Missing Values: 665 empty cells were found in the NumCompaniesPreviouslyWorked column. 1969 empty cells were found in the AnnualProfessionalDevHrs column. 2266 empty cells were found in the TextMessageOptIn column.

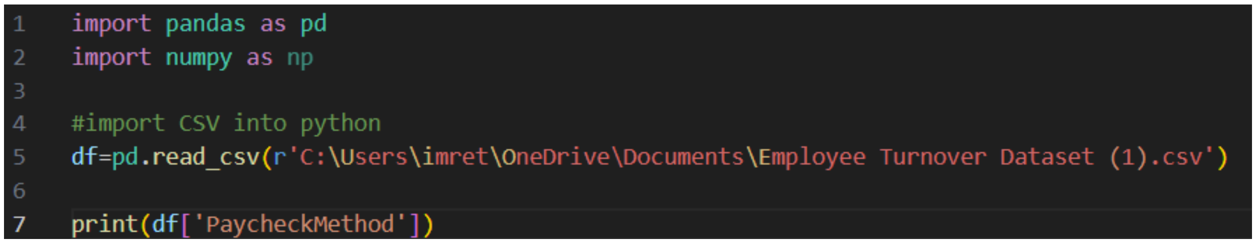


```python
import pandas as pd
import numpy as np

#import CSV into python
df=pd.read_csv(r'C:\Users\imret\OneDrive\Documents\Employee Turnover Dataset (1).csv')

#Count number of empty cells in each column of dataset
missing_values_per_column = df.isna().sum()

print(missing_values_per_column)
```

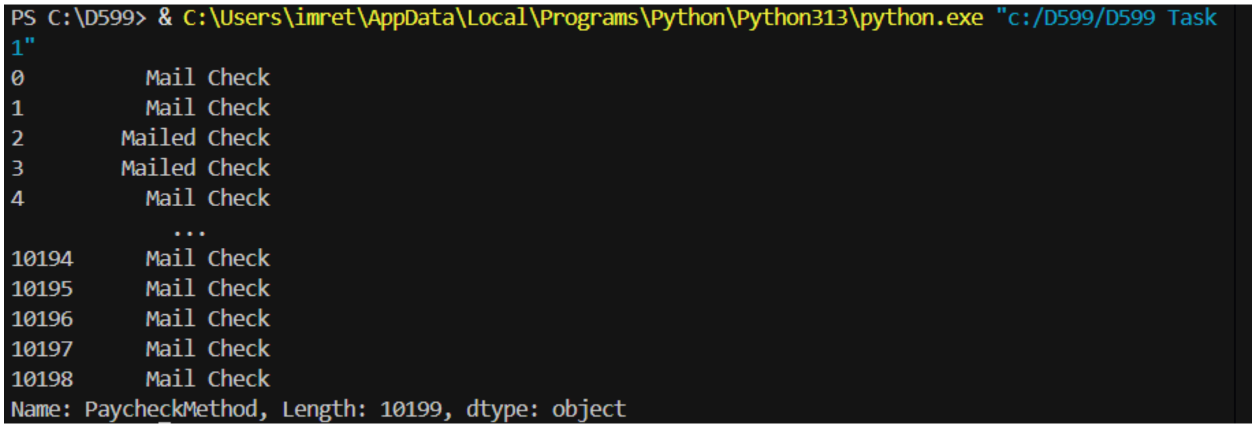


```
PS C:\D599> & C:\Users\imret\AppData\Local\Programs\Python\Python313\python.exe "c:/D599/D599 Task
1"
EmployeeNumber                    0
Age                               0
Tenure                            0
Turnover                          0
HourlyRate                        0
HoursWeekly                       0
CompensationType                  0
AnnualSalary                      0
DrivingCommuterDistance           0
JobRoleArea                       0
Gender                            0
MaritalStatus                     0
NumCompaniesPreviouslyWorked    665
AnnualProfessionalDevHrs       1969
PaycheckMethod                    0
TextMessageOptIn               2266
dtype: int64
```

Inconsistent Entries: Inconsistencies were found in the PaycheckMethod column and the JobRoleArea column. PaycheckMethod column included Mail Check, Mailed Check, Mail_Check, and MailedCheck, as well as Direct Deposit, Direct_Deposit, and DirectDeposit. JobRoleArea column included HumanResources, Human_Resources, and Human Resources as well as InformationTechnology, Information_Technology, and Information Technology.

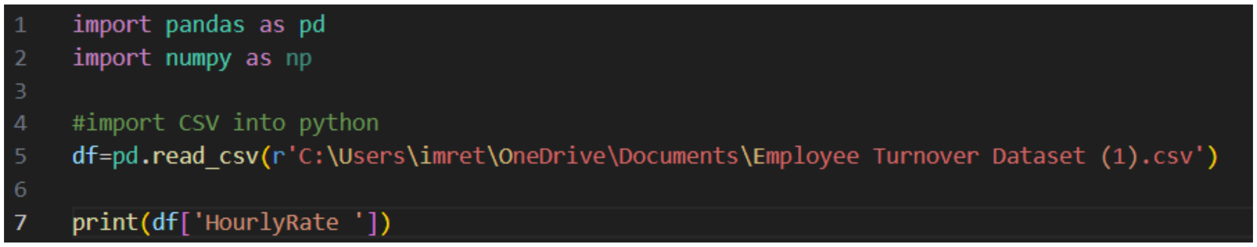


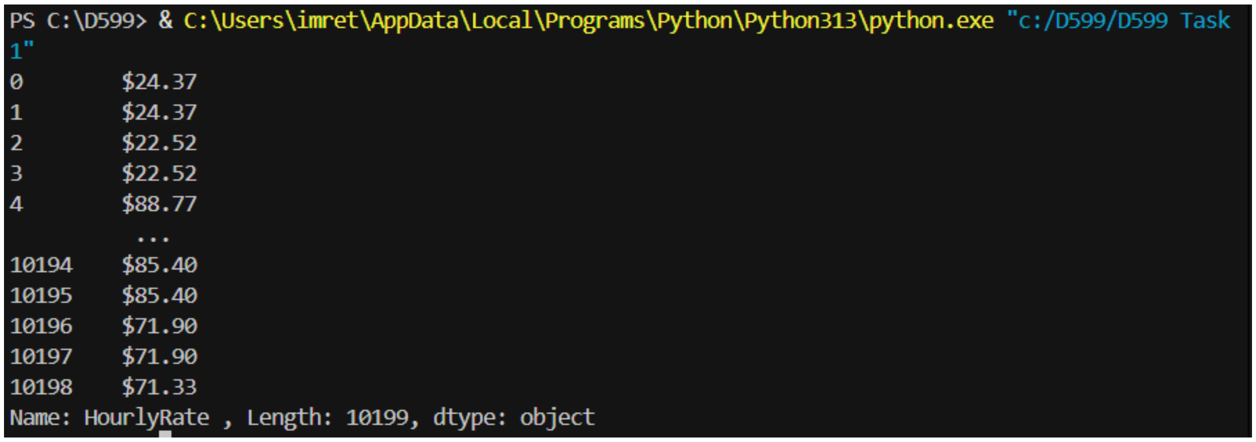
```python
import pandas as pd
import numpy as np

#import CSV into python
df=pd.read_csv(r'C:\Users\imret\OneDrive\Documents\Employee Turnover Dataset (1).csv')

print(df['PaycheckMethod'])
```



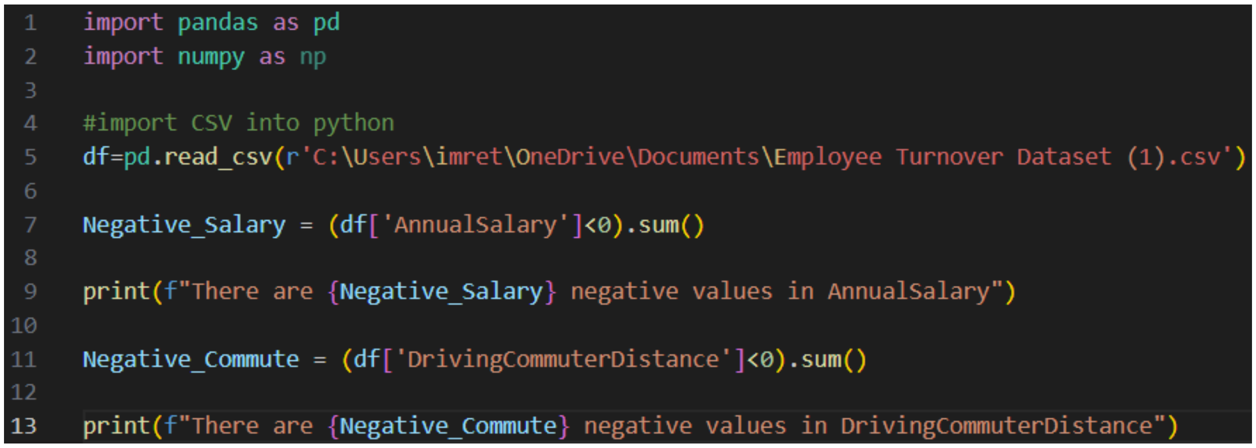
```
PS C:\D599> & C:\Users\imret\AppData\Local\Programs\Python\Python313\python.exe "c:/D599/D599 Task
1"
0          Mail Check
1          Mail Check
2        Mailed Check
3        Mailed Check
4          Mail Check
              ...
10194      Mail Check
10195      Mail Check
10196      Mail Check
10197      Mail Check
10198      Mail Check
Name: PaycheckMethod, Length: 10199, dtype: object
```

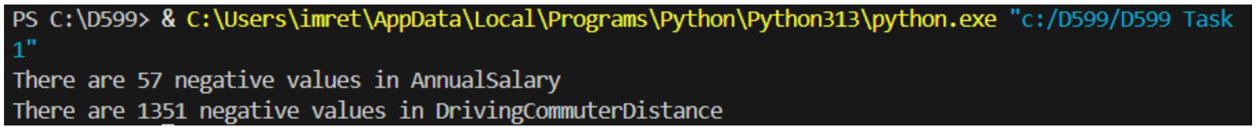
Formatting Errors: Use of $ and white space in HourlyRate column resulted in the data being stored as an object dtype instead of a float dtype.


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

#import CSV into python
df=pd.read_csv(r'C:\Users\imret\OneDrive\Documents\Employee Turnover Dataset (1).csv')

print(df['HourlyRate '])
```


```
PS C:\D599> & C:\Users\imret\AppData\Local\Programs\Python\Python313\python.exe "c:/D599/D599 Task
1"
0        $24.37
1        $24.37
2        $22.52
3        $22.52
4        $88.77
          ...
10194    $85.40
10195    $85.40
10196    $71.90
10197    $71.90
10198    $71.33
Name: HourlyRate , Length: 10199, dtype: object
```

Outliers: There were 57 negative values found in the AnnualSalary column and 1353 negative values found in the DrivingCommuterDistance column.

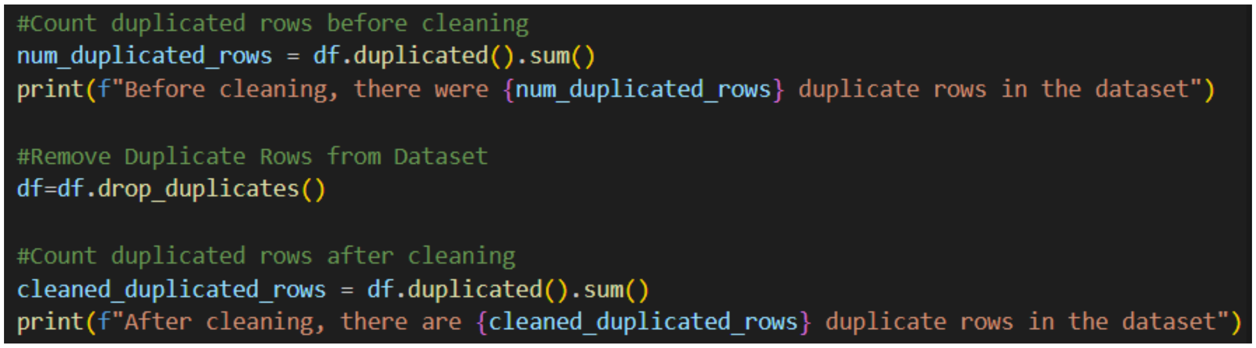

```python
import pandas as pd
import numpy as np

#import CSV into python
df=pd.read_csv(r'C:\Users\imret\OneDrive\Documents\Employee Turnover Dataset (1).csv')

Negative_Salary = (df['AnnualSalary']<0).sum()

print(f"There are {Negative_Salary} negative values in AnnualSalary")

Negative_Commute = (df['DrivingCommuterDistance']<0).sum()

print(f"There are {Negative_Commute} negative values in DrivingCommuterDistance")
```


```
PS C:\D599> & C:\Users\imret\AppData\Local\Programs\Python\Python313\python.exe "c:/D599/D599 Task
1"
There are 57 negative values in AnnualSalary
There are 1351 negative values in DrivingCommuterDistance
```

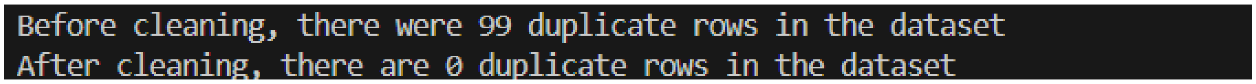
Part C: *Discuss which data cleaning techniques you used to correct all the data quality issues you identified by doing the following:*

1. *Describe how you modified the dataset using Python or R code after identifying each quality issue listed in part B1*

Duplicates: Used .drop_duplicates() to remove all duplicate rows from the data set

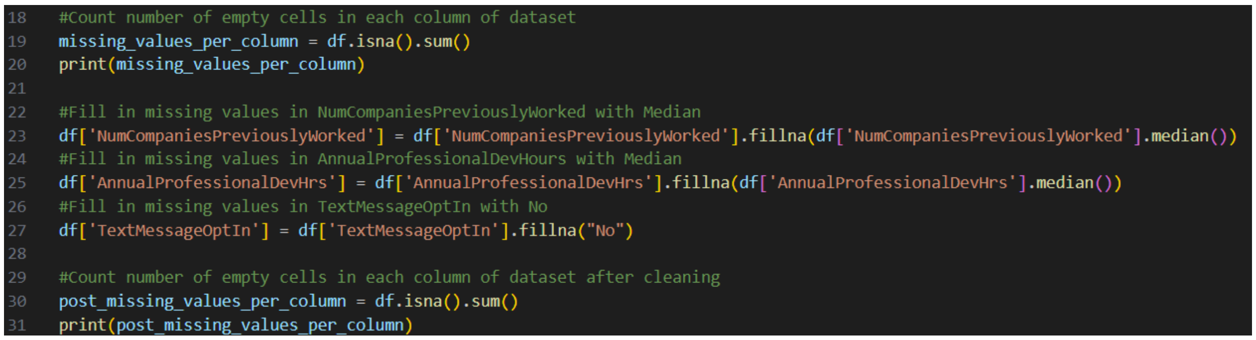


```
#Count duplicated rows before cleaning
num_duplicated_rows = df.duplicated().sum()
print(f"Before cleaning, there were {num_duplicated_rows} duplicate rows in the dataset")

#Remove Duplicate Rows from Dataset
df=df.drop_duplicates()

#Count duplicated rows after cleaning
cleaned_duplicated_rows = df.duplicated().sum()
print(f"After cleaning, there are {cleaned_duplicated_rows} duplicate rows in the dataset")
```



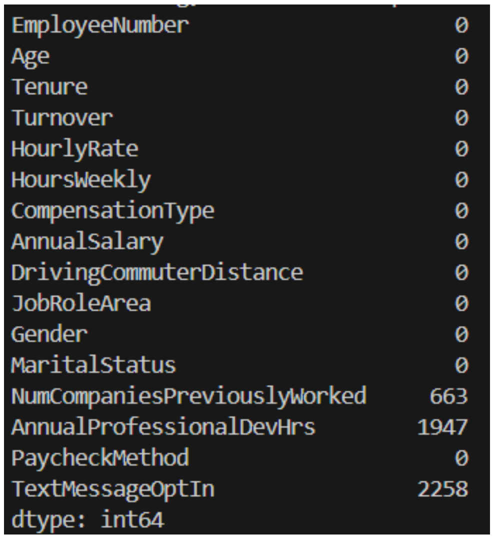
```
Before cleaning, there were 99 duplicate rows in the dataset
After cleaning, there are 0 duplicate rows in the dataset
```

Missing Values: Used .fillna() to substitute the median in for all missing values in the NumCompaniesPreviouslyWorked and AnnualProfessionalDevHrs columns and to substitute 'No' in for all missing values in the TextMessageOptIn.



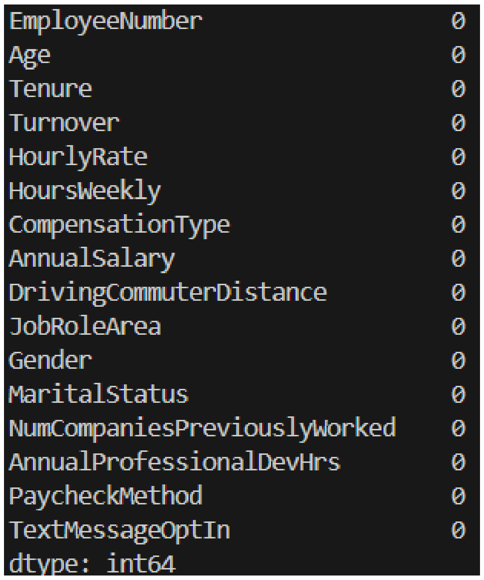
```
#Count number of empty cells in each column of dataset
missing_values_per_column = df.isna().sum()
print(missing_values_per_column)

#Fill in missing values in NumCompaniesPreviouslyWorked with Median
df['NumCompaniesPreviouslyWorked'] = df['NumCompaniesPreviouslyWorked'].fillna(df['NumCompaniesPreviouslyWorked'].median())
#Fill in missing values in AnnualProfessionalDevHours with Median
df['AnnualProfessionalDevHrs'] = df['AnnualProfessionalDevHrs'].fillna(df['AnnualProfessionalDevHrs'].median())
#Fill in missing values in TextMessageOptIn with No
df['TextMessageOptIn'] = df['TextMessageOptIn'].fillna("No")

#Count number of empty cells in each column of dataset after cleaning
post_missing_values_per_column = df.isna().sum()
print(post_missing_values_per_column)
```

Before:



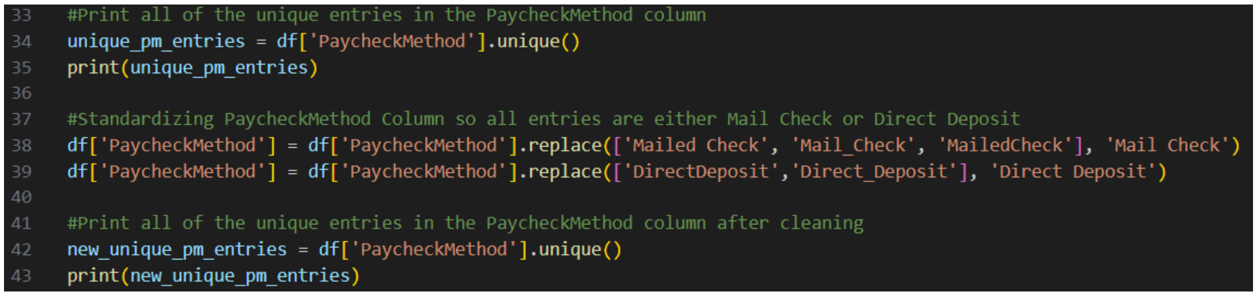
```
EmployeeNumber                  0
Age                             0
Tenure                          0
Turnover                        0
HourlyRate                      0
HoursWeekly                     0
CompensationType                0
AnnualSalary                    0
DrivingCommuterDistance         0
JobRoleArea                     0
Gender                          0
MaritalStatus                   0
NumCompaniesPreviouslyWorked  663
AnnualProfessionalDevHrs     1947
PaycheckMethod                  0
TextMessageOptIn             2258
dtype: int64
```

After:



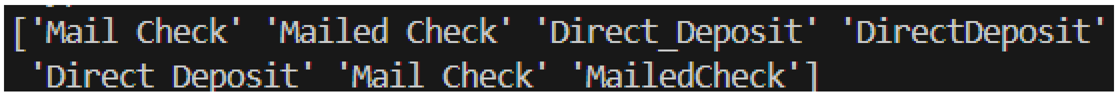
```
EmployeeNumber                  0
Age                             0
Tenure                          0
Turnover                        0
HourlyRate                      0
HoursWeekly                     0
CompensationType                0
AnnualSalary                    0
DrivingCommuterDistance         0
JobRoleArea                     0
Gender                          0
MaritalStatus                   0
NumCompaniesPreviouslyWorked    0
AnnualProfessionalDevHrs        0
PaycheckMethod                  0
TextMessageOptIn                0
dtype: int64
```

Inconsistent Entries: Used .replace() to change all variations in PaycheckMethod column to either Mail Check or Direct Deposit.



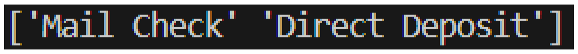
```python
#Print all of the unique entries in the PaycheckMethod column
unique_pm_entries = df['PaycheckMethod'].unique()
print(unique_pm_entries)

#Standardizing PaycheckMethod Column so all entries are either Mail Check or Direct Deposit
df['PaycheckMethod'] = df['PaycheckMethod'].replace(['Mailed Check', 'Mail_Check', 'MailedCheck'], 'Mail Check')
df['PaycheckMethod'] = df['PaycheckMethod'].replace(['DirectDeposit','Direct_Deposit'], 'Direct Deposit')

#Print all of the unique entries in the PaycheckMethod column after cleaning
new_unique_pm_entries = df['PaycheckMethod'].unique()
print(new_unique_pm_entries)
```

Before:



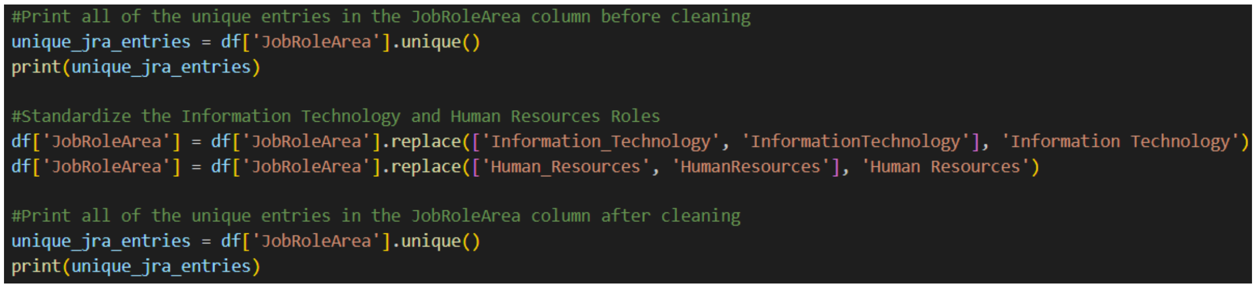
```
['Mail Check' 'Mailed Check' 'Direct_Deposit' 'DirectDeposit'
 'Direct Deposit' 'Mail_Check' 'MailedCheck']
```

After:



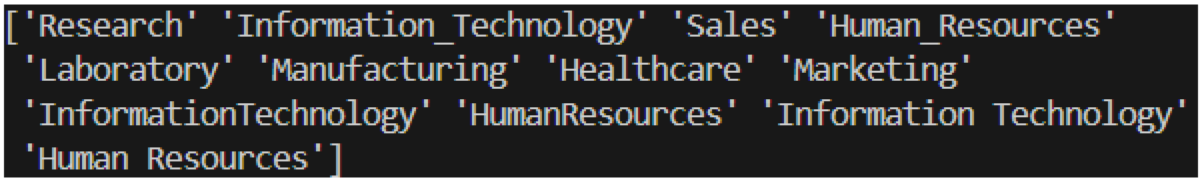
```
['Mail Check' 'Direct Deposit']
```

Used .replace() to change all inconsistent variations in JobRoleArea column to either Human Resources or Information Technology



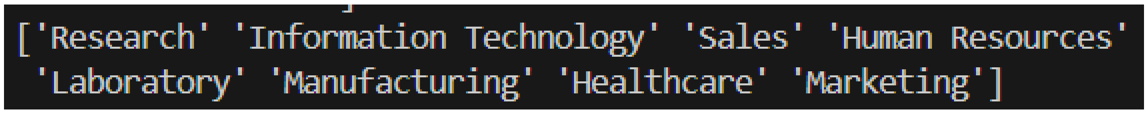
```python
#Print all of the unique entries in the JobRoleArea column before cleaning
unique_jra_entries = df['JobRoleArea'].unique()
print(unique_jra_entries)

#Standardize the Information Technology and Human Resources Roles
df['JobRoleArea'] = df['JobRoleArea'].replace(['Information_Technology', 'InformationTechnology'], 'Information Technology')
df['JobRoleArea'] = df['JobRoleArea'].replace(['Human_Resources', 'HumanResources'], 'Human Resources')

#Print all of the unique entries in the JobRoleArea column after cleaning
unique_jra_entries = df['JobRoleArea'].unique()
print(unique_jra_entries)
```

Before:



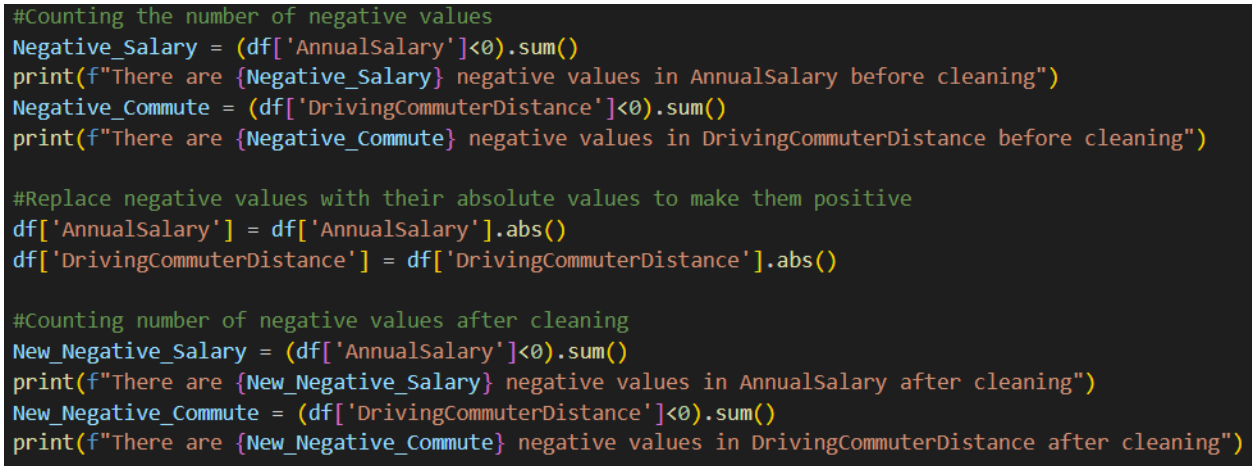
```
['Research' 'Information_Technology' 'Sales' 'Human_Resources'
 'Laboratory' 'Manufacturing' 'Healthcare' 'Marketing'
 'InformationTechnology' 'HumanResources' 'Information Technology'
 'Human Resources']
```

After:



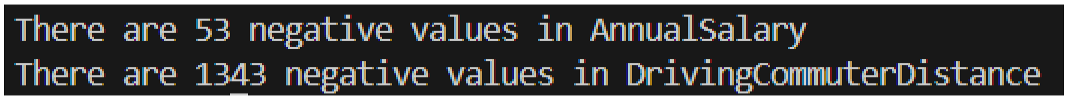
```
['Research' 'Information Technology' 'Sales' 'Human Resources'
 'Laboratory' 'Manufacturing' 'Healthcare' 'Marketing']
```

Also, I identified the negative values in the AnnualSalary column and the DrivingCommuterDistance columns. Determining the exact cause of this error in the real world would require further investigation, but provided no other information, I took the absolute value of these columns to switch the negatives to positive.



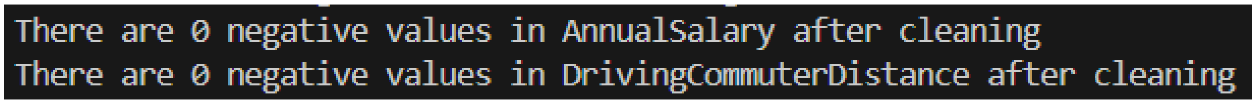
```python
#Counting the number of negative values
Negative_Salary = (df['AnnualSalary']<0).sum()
print(f"There are {Negative_Salary} negative values in AnnualSalary before cleaning")
Negative_Commute = (df['DrivingCommuterDistance']<0).sum()
print(f"There are {Negative_Commute} negative values in DrivingCommuterDistance before cleaning")

#Replace negative values with their absolute values to make them positive
df['AnnualSalary'] = df['AnnualSalary'].abs()
df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].abs()

#Counting number of negative values after cleaning
New_Negative_Salary = (df['AnnualSalary']<0).sum()
print(f"There are {New_Negative_Salary} negative values in AnnualSalary after cleaning")
New_Negative_Commute = (df['DrivingCommuterDistance']<0).sum()
print(f"There are {New_Negative_Commute} negative values in DrivingCommuterDistance after cleaning")
```

Before:



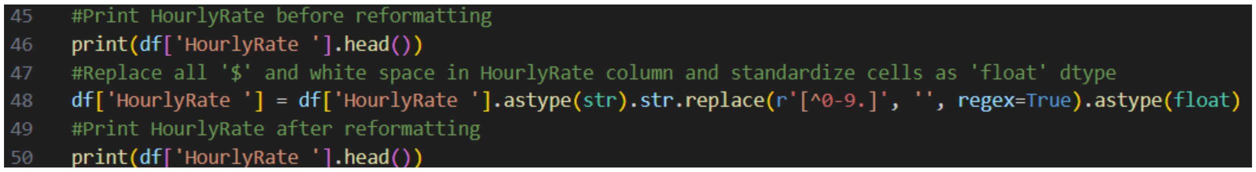
```
There are 53 negative values in AnnualSalary
There are 1343 negative values in DrivingCommuterDistance
```

After:



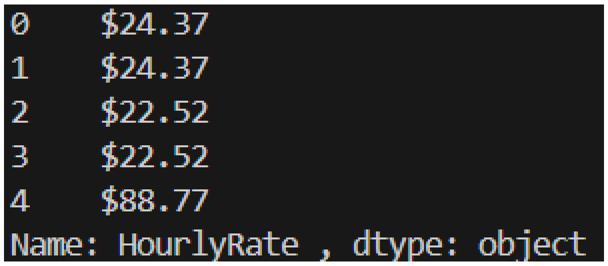
```
There are 0 negative values in AnnualSalary after cleaning
There are 0 negative values in DrivingCommuterDistance after cleaning
```

Formatting Errors: Use .str.replace() to remove all ‘$’ and white space characters from the HourlyRate column and convert all entries from object dtype to float dtype.



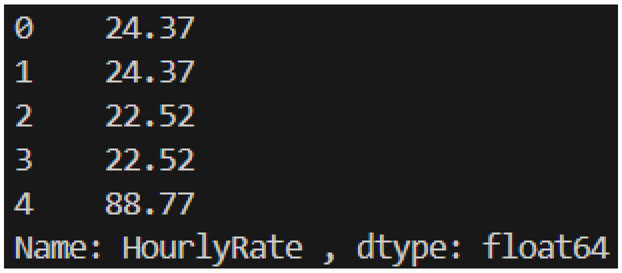
```python
#Print HourlyRate before reformatting
print(df['HourlyRate '].head())
#Replace all '$' and white space in HourlyRate column and standardize cells as 'float' dtype
df['HourlyRate '] = df['HourlyRate '].astype(str).str.replace(r'[^0-9.]', '', regex=True).astype(float)
#Print HourlyRate after reformatting
print(df['HourlyRate '].head())
```

Before:



```
0    $24.37
1    $24.37
2    $22.52
3    $22.52
4    $88.77
Name: HourlyRate , dtype: object
```

After:



```
0    24.37
1    24.37
2    22.52
3    22.52
4    88.77
Name: HourlyRate , dtype: float64
```

Outliers: After looking at the outliers present in the DrivingCommuterDistance Column, I noticed that all of the outliers were one of five distinct values: 910, 950, 250, 275, or 322. This leads me to believe they are a data entry mistake. Because of this I replaced them with the column median to preserve the rest of the data.

2. *Discuss why you chose the specific data cleaning techniques you used to clean the quality issues listed in part B1.*

Duplicates: .drop_duplicates() was used because removing all duplicate rows eliminates redundancy, which can provide a bias during analysis

Missing Values: The median was substituted into the NumCompaniesPreviouslyWorked and AnnualProfessionalDevHrs because these entries have the potential to vary widely, and the median is less likely to be significantly affected by potential outliers. “No” was substituted into the TextMessageOptIn column because it is safer to assume an employee did not opt in, than to assume they did opt in.

Inconsistent Entries: Standardizing all strings in the PaycheckMethod and JobRoleArea column will make future attempts at grouping or visualization much easier and cleaner. Replacing the negative values in AnnualSalary and DrivingCommuterDistance will allow any analysis of the data to be more accurate.

Formatting Errors: Changing HourlyWage from an object dtype to a float dtype makes numerical analysis with the data possible.

Outliers: Changing the outliers in DrivingCommuterDistance to the column’s median will eliminate the outliers while still preserving the rest of the data in their rows.

3. *Describe two advantages to your data cleaning approach specified in part C1.*

First, other than duplicate removal, all data points were imputed or retained which preserves the size of the dataset. Secondly, elimination of empty cells and standardization of formatting ensure that any future analysis will be much simpler, without the need for manual editing.

4. *Discuss two limitations to your data cleaning approach specified in part C1.*

If the negative values in AnnualSalary and DrivingCommuterDistance were more than simple misplaced negative signs, my decision to simply take the absolute value could lead to bad data being included in analysis. Secondly, imputing the median into empty cells has the potential to bias data towards that median during future analysis, especially in situations like the AnnualProfessionalDevHrs column which had empty cells in over 10% of rows.

Part E. *Acknowledge reference sources used to support the Python or R code application. All references listed should also include an in-text citation in the code annotation. Be sure the sources are reliable. If no sources were used for coding, state, "No sources used."*

No sources used.

Part F. *Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.*

No sources used.