Nicole Gallo
D599 – Data Preparation and Exploration
June 26, 2025
TCN1 Task 1: Data Cleaning and Profiling

Part I: Data Profiling

A. Review the data dictionary in the attached "Employee Turnover Considerations and Dictionary" document and do the following:

a. Describe the general characteristics of the dataset

The Employee Turnover Dataset contains 10,199 rows and 16 columns of data. Each row represents an individual employee. Other important data attributes include demographic, employment, and compensation (type and annual salary). This data is used to understand the "employee turnover" within the company.

b. Indicate the data type and data subtype for each variable

Below is a list of the variables in the dataset with their corresponding data type and subtype:

Variable Name	Data Type	Subtype
EmployeeNumber	Integer	Nominal (ID)
Age	Integer	Discrete Numeric
Tenure	Integer	Discrete Numeric
Turnover	Object	Nominal (Categorical - Yes/No)
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	Float	Discrete Numeric
AnnualProfessionalDevHrs	Float	Continuous Numeric
PaycheckMethod	Object	Nominal (Categorical)
TextMessageOptIn	Object	Nominal (Categorical - Yes/No)

c. Provide a sample of observable values for each variable

A sample for each variable:

- EmployeeNumber: 5270
- Age: 47
- Tenure: 15
- Turnover: 'No'
- HourlyRate: 41.07
- HoursWeekly: 40
- CompensationType: 'Salary'

- AnnualSalary: 85425.6
- DrivingCommuterDistance: 12
- JobRoleArea: 'Human Resources'
- Gender: 'Male'
- MaritalStatus: 'Divorced'
- NumCompaniesPreviouslyWorked: 8.0
- AnnualProfessionalDevHrs: 11.0
- PaycheckMethod: 'Mail Check'
- TextMessageOptIn: 'Yes'

Part II: Data Cleaning and Plan

B. Inspect the dataset through data cleaning techniques

1. Explain how you inspected the dataset for *each* of the quality issues listed in part B.

   Duplications – used `df.duplicated().sum()` to find the number of duplicate rows
   Missing values – used `df.isnull().sum()` to find the number of missing values
   Formatting errors – standardized text fields using Python to remove any unnecessary symbols and space characters
   Inaccurate data – reviewed unique values in object columns; used Python to confirm inaccurate data, such as negative values *`df['DrivingCommuterDistance'] < 0`*
   Outliers – used IQR (interquartile range) method to calculate upper and lower bounds; values outside of 1.5xIQR were flagged as outliers

2. List your findings for *each* quality issue listed in part B.

   Duplications – 99 duplicate rows were found

```
In [122…  import pandas as pd
          import numpy as np

          # Read in the CSV file
          df=pd.read_csv('Python/Employee_Turnover_Dataset.csv')
```

```
In [123…  # Count how many duplicates in dataset
          df.duplicated().sum()

Out[123…  np.int64(99)
```

   Missing values – NumCompaniesPreviouslyWorked (665), AnnualProfessionalDevHrs (1969), TextMessageOptIn (2266)

```
In [124…  # Count how many missing values in dataset
          df.isnull().sum()

Out[124…  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
```

Formatting errors – HourlyRate contained formatting issues ($ and extra whitespace) as a currency number data type – data type is showing HourlyRate as ‘object’ instead of ‘float’ because of the $ in the entries

[166]:
```python
import pandas as pd
import numpy as np

# Read the dataset
df = pd.read_csv('Python/Employee_Turnover_Dataset.csv')

# Show 'HourlyRate' with $ and whitespace
print(df['HourlyRate '])
```

```
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
```

Inaccurate data – reviewed unique values in object columns; used Python to confirm inaccurate data, such as negative values *df['DrivingCommuterDistance'] < 0*

- Negative Commuting Distances – e.g., *df['DrivingCommuterDistance'] < 0*

In [127…
```python
import pandas as pd
import numpy as np

# Read in the CSV file
df=pd.read_csv('Python/Employee_Turnover_Dataset.csv')

# Show negative DrivingCommuterDistance data
df[df['DrivingCommuterDistance'] < 0]
```

[127]:

	EmployeeNumber	Age	Tenure	Turnover	HourlyRate	HoursWeekly	CompensationType	AnnualSalary	DrivingCommuterDistance	JobRoleArea	Gender	Marit
30	31	30	1	No	$24.50	40	Salary	50960.0	-4	Human_Resources	Female	
31	32	34	2	No	$24.50	40	Salary	50960.0	-4	Human_Resources	Female	
54	55	44	16	No	$31.30	40	Salary	-15896.0	-5	Marketing	Male	
55	56	50	8	No	$31.30	40	Salary	-15896.0	-5	Marketing	Male	
64	65	30	3	Yes	$29.49	40	Salary	-28660.8	-8	Marketing	Female	
...	...	...	...	...	...	...	...	...	...	...	...	
10155	56	50	8	No	$31.30	40	Salary	-15896.0	-5	Marketing	Male	
10164	65	30	3	Yes	$29.49	40	Salary	-28660.8	-8	Marketing	Female	
10165	66	32	5	Yes	$29.49	40	Salary	-28660.8	-8	Marketing	Female	
10190	91	36	9	No	$28.73	40	Salary	59758.4	-7	Laboratory	Female	
10191	92	39	4	No	$28.73	40	Salary	59758.4	-7	Laboratory	Female	

1351 rows × 16 columns

- Inconsistent naming convention - PaycheckMethod “Mail Check” and “Direct Deposit”, and JobRoleArea “Human Resources” and “Information Technology”

[71]:
```python
for col in df.select_dtypes(include='object').columns:
    print(col)
    print(df[col].value_counts(dropna=False))
```

```
Turnover
Turnover
No     5456
Yes    4644
Name: count, dtype: int64
HourlyRate
HourlyRate
$34.28     11
$31.28     10
$33.66     10
$28.83      9
$33.06      9
           ..
$28.37      1
$56.02      1
$89.43      1
$88.05      1
$93.05      1
Name: count, Length: 5244, dtype: int64
CompensationType
CompensationType
Salary    10100
Name: count, dtype: int64
JobRoleArea
JobRoleArea
Research                  2005
Sales                     1988
Marketing                 1093
Manufacturing             1031
Laboratory                1007
Healthcare                1002
Human Resources            909
Information Technology     857
InformationTechnology       80
HumanResources              51
Information_Technology      42
Human_Resources             35
Name: count, dtype: int64
Gender
Gender
Female                  5756
Male                    4208
Prefer Not to Answer     136
Name: count, dtype: int64
MaritalStatus
MaritalStatus
Married     3405
Single      3387
Divorced    3308
Name: count, dtype: int64
PaycheckMethod
PaycheckMethod
Mail Check        4917
Mailed Check      2425
DirectDeposit      988
Direct_Deposit     948
Mail_Check         547
Direct Deposit     226
MailedCheck         49
Name: count, dtype: int64
TextMessageOptIn
TextMessageOptIn
Yes    7299
NaN    2258
No      543
Name: count, dtype: int64
```

- Miscalculation of AnnualSalary, showing negative salary

[329]:
```python
import pandas as pd
import numpy as np

# Read in the CSV file
df=pd.read_csv('Python/Employee_Turnover_Dataset.csv')

# Showing data entries with a negative AnnualSalary,
# which is not accurate and likely due to miscalculation
((df['AnnualSalary']) < 0).sum()
```

[329]:
```
np.int64(57)
```

Outliers – using IQR (interquartile range) to find outliers in DrivingCommuterDistance
`e.g., lower_bound = Q1 - 1.5 * IQR`
`upper_bound = Q3 + 1.5 * IQR`

```
[584]: # Use IQR to find outliers in DrivingCommuterDistance
Q1 = df['DrivingCommuterDistance'].quantile(0.25)
Q3 = df['DrivingCommuterDistance'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Print lower and upper bounds to see range
print("Lower bound is: ", lower_bound)
print("Upper bound is: ", upper_bound)

# Show outlier rows
df[(df['DrivingCommuterDistance'] < lower_bound) | (df['DrivingCommuterDistance'] > upper_bound)]
```

```
Lower bound is:  -69.0
Upper bound is:  155.0
```

[584]:

	EmployeeNumber	Age	Tenure	Turnover	HourlyRate	HoursWeekly	CompensationType	AnnualSalary	DrivingCommuterDistance	JobRoleArea	Gender	M
26	27	32	6	No	31.02	40	Salary	64521.6	910	Research	Female	
27	28	32	5	No	31.02	40	Salary	64521.6	910	Research	Female	
34	35	28	4	No	25.14	40	Salary	52291.2	910	Laboratory	Female	
35	36	35	6	Yes	25.14	40	Salary	52291.2	910	Laboratory	Female	
196	197	30	7	No	30.80	40	Salary	64064.0	950	Information_Technology	Male	
...	...	...	...	...	...	...	...	...	...	...	...	
3456	3457	57	9	No	89.62	40	Salary	186409.6	250	Manufacturing	Female	
3465	3466	59	6	No	33.24	40	Salary	69139.2	250	Manufacturing	Female	
3481	3482	39	7	Yes	51.52	40	Salary	107161.6	250	Research	Male	
3486	3487	49	18	Yes	67.75	40	Salary	140920.0	250	Information Technology	Male	
3491	3492	39	5	No	29.83	40	Salary	62046.4	250	Healthcare	Male	

222 rows × 16 columns

C. Discuss which data cleaning techniques you used

1. Describe how you modified the dataset after identifying each quality issue listed in part B.

Duplications – used `df.drop_duplicates()` to remove 99 duplicated rows

```
[178]: import pandas as pd
import numpy as np

# Read in the CSV file
df=pd.read_csv('Python/Employee_Turnover_Dataset.csv')
```

```
[179]: # Count how many duplicates in dataset
df.duplicated().sum()
```

```
[179]: np.int64(99)
```

```
[180]: df = pd.read_csv('Python/Employee_Turnover_Dataset.csv')
df = df.drop_duplicates()
```

```
[181]: # Count how many duplicates in dataset; after running df.drop_duplicates()
df.duplicated().sum()
```

```
[181]: np.int64(0)
```

Missing values – used `df.isnull().sum()` to find the number of missing values

Before:

```
[234]: # Count how many missing values in dataset
       df.isnull().sum()

[234]: 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 (NumCompaniesPreviouslyWorked):

```
[235]: # Use .fillna() to insert NaN or None into missing values
       df['NumCompaniesPreviouslyWorked'] = df['NumCompaniesPreviouslyWorked'].fillna(df['NumCompaniesPreviouslyWorked'].median())

       # Check missing values
       df.isnull().sum()

[235]: 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       1947
       PaycheckMethod                    0
       TextMessageOptIn               2258
       dtype: int64
```

After (AnnualProfessionalDevHrs):

```
[236]: # Use .fillna() to insert NaN or None into missing values
       df['AnnualProfessionalDevHrs'] = df['AnnualProfessionalDevHrs'].fillna(df['AnnualProfessionalDevHrs'].median())

       # Check missing values
       df.isnull().sum()

[236]: 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               2258
       dtype: int64
```

After (TextMessageOptIn):

```
[237]: # Use .fillna() to insert NaN or None into missing values
df['TextMessageOptIn'] = df['TextMessageOptIn'].fillna(df['TextMessageOptIn'].mode()[0])

# Check missing values
df.isnull().sum()
```

```
[237]: 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
```

Formatting errors – HourlyRate contained formatting issues ($ and extra whitespace) as a currency number data type – e.g., *df.columns = df.columns.str.strip()*

Before:

```
[131]: import pandas as pd
import numpy as np

# Read the dataset
df = pd.read_csv('Python/Employee_Turnover_Dataset.csv')

# Show 'HourlyRate' with $ and whitespace
print(df['HourlyRate '])

# Strip whitespace from column names
df.columns = df.columns.str.strip()
```

```
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
```

After:

```
[132]: # Clean HourlyRate
df['HourlyRate'] = df['HourlyRate'].astype(str).str.replace(r'[^0-9.]', '', regex=True).astype(float)

# Show 'HourlyRate' after cleaning
print(df['HourlyRate'])
```

```
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: float64
```

Inaccurate data – negative commuting distances, PaycheckMethod and JobRoleArea showing the same entry but different version of the word, AnnualSalary not matching HourlyRate x HoursWeekly x 52

- Negative Commuting Distances – `e.g.,` *`df['DrivingCommuterDistance'] < 0`*

```
In [129…  # Change negative DrivingCommuterDistance to positive with .abs()
          df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].abs()

          # Count negative DrivingCommuterDistance data; should now be 0
          ((df['DrivingCommuterDistance']) < 0).sum()

Out[129…  np.int64(0)

In [130…  # Show negative DrivingCommuterDistance data
          df[df['DrivingCommuterDistance'] < 0]

Out[130…    EmployeeNumber  Age  Tenure  Turnover  HourlyRate  HoursWeekly  Compen:
```

- Standardizing Paycheck Method – Inconsistencies between "Mail Check" and "Direct Deposit"
  e.g., *`df['PaycheckMethod'] = df['PaycheckMethod'].replace({...})`*

```
[295…  # Convert to lowercase and strip whitespace to catch all variations
       df['PaycheckMethod'] = df['PaycheckMethod'].str.strip().str.lower()

[302]: # Replace inconsistent entries with standardized versions
       df['PaycheckMethod'] = df['PaycheckMethod'].replace({
           'mailed check': 'Mail Check',
           'mail_check': 'Mail Check',
           'mailedcheck': 'Mail Check',
           'mail check': 'Mail Check',
           'directdeposit': 'Direct Deposit',
           'direct_deposit': 'Direct Deposit',
           'direct deposit': 'Direct Deposit'
       })

[303]: # Print PaycheckMethod with standardized version
       print(df['PaycheckMethod'].value_counts())

PaycheckMethod
Mail Check        7938
Direct Deposit    2162
Name: count, dtype: int64
```

- Standardizing Job Role Area – Inconsistencies between "Human Resources" and "Information Technology"

```
df['JobRoleArea'] = df['JobRoleArea'].replace({
    'Humanresources': 'Human Resources',...
})
```

```
[40]: print(df['JobRoleArea'].value_counts(dropna=False))

JobRoleArea
Research                    2005
Sales                       1988
Marketing                   1093
Manufacturing               1031
Laboratory                  1007
Healthcare                  1002
Human Resources              909
Information Technology       857
InformationTechnology         80
HumanResources                51
Information_Technology        42
Human_Resources               35
Name: count, dtype: int64
```

```
[41]: df['JobRoleArea'] = df['JobRoleArea'].str.strip().str.title()
```

```
[42]: # Replace inconsistent entries with standardized versions
df['JobRoleArea'] = df['JobRoleArea'].replace({
    'Humanresources': 'Human Resources',
    'Human_Resources': 'Human Resources',
    'Information_Technology': 'Information Technology',
    'Informationtechnology': 'Information Technology'
})
```

```
[43]: for col in df.select_dtypes(include='object').columns:
    print(col)
    print(df[col].value_counts(dropna=False))

Turnover
Turnover
No     5456
Yes    4644
Name: count, dtype: int64
HourlyRate
HourlyRate
$34.28    11
$31.28    10
$33.66    10
$28.83     9
$33.06     9
          ..
$28.37     1
$56.02     1
$89.43     1
$88.05     1
$93.05     1
Name: count, Length: 5244, dtype: int64
CompensationType
CompensationType
Salary    10100
Name: count, dtype: int64
JobRoleArea
JobRoleArea
Research                    2005
Sales                       1988
Marketing                   1093
Manufacturing               1031
Laboratory                  1007
Healthcare                  1002
Human Resources              995
Information Technology       979
Name: count, dtype: int64
Gender
Gender
Female                  5756
Male                    4208
Prefer Not to Answer     136
Name: count, dtype: int64
```

- AnnualSalary not matching HourlyRate x HoursWeekly x 52
  e.g., *`df.loc[hourly_mask, 'AnnualSalary'] = df.loc[hourly_mask, 'HourlyRate'] * df.loc[hourly_mask, 'HoursWeekly'] * 52`*

```
•[463…  # Only recalculate for hourly employees
        hourly_mask = df['CompensationType'].str.lower() == 'salary'
        df.loc[hourly_mask, 'AnnualSalary'] = df.loc[hourly_mask, 'HourlyRate'] * df.loc[hourly_mask, 'HoursWeekly'] * 52

        # Round to 2 decimal places
        df['AnnualSalary'] = df['AnnualSalary'].round(2)

        # Print pay metrics to confirm changes
        print(df[['CompensationType', 'HourlyRate', 'HoursWeekly', 'AnnualSalary']].head(10))
```

```
  CompensationType  HourlyRate  HoursWeekly  AnnualSalary
0           Salary       24.37           40       50689.6
1           Salary       24.37           40       50689.6
2           Salary       22.52           40       46841.6
3           Salary       22.52           40       46841.6
4           Salary       88.77           40      184641.6
5           Salary       88.77           40      184641.6
6           Salary       28.43           40       59134.4
7           Salary       28.43           40       59134.4
8           Salary       21.87           40       45489.6
9           Salary       21.87           40       45489.6
```

```
[464]: df[df['AnnualSalary'] < 0]
```

[464]:

EmployeeNumber	Age	Tenure	Turnover	HourlyRate	HoursWeekly	CompensationType	AnnualSalary	DrivingCommuterDistance	JobRoleArea	Gender	MaritalStatus	N

Outliers – removing outliers with outlier capping using *`.clip()`*

```
[95]: # Outlier capping using .clip()
      df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].clip(lower=lower_bound, upper=upper_bound)
```

```
[96]: # Show outlier rows after .clip() method
      df[(df['DrivingCommuterDistance'] < lower_bound) | (df['DrivingCommuterDistance'] > upper_bound)]
```

[96]:

EmployeeNumber	Age	Tenure	Turnover	HourlyRate	HoursWeekly	CompensationType	AnnualSalary	DrivingCommuterDistance	JobRoleArea	Gender	MaritalStatus	N

```
•[97]: # Print lower and upper bounds & Min and Max to see range after .clip() method
       print("Lower bound is: ", lower_bound)
       print("Upper bound is: ", upper_bound)

       print("Min:", df['DrivingCommuterDistance'].min())
       print("Max:", df['DrivingCommuterDistance'].max())
```

```
Lower bound is:  -69.0
Upper bound is:  155.0
Min: 0
Max: 155
```

2. Discuss why you chose the specific data cleaning techniques you used to clean the quality issues listed in part B.
   These specific data cleaning techniques were used to ensure data accuracy and consistency for better analysis:
   - Removing duplicates, `.drop_duplicates()`, eliminates redundancy that could skew analysis
   - Median imputation is less affected by outliers and keeps distributions intact and mode imputation maintains consistency for categorical fields, according to Syed Burhan Ahmed in [When to Use Mean, Median, and Mode for Handling Missing Values in Data?]
   - Text and formatting standardization improves grouping initatives and any future visualization needs.
   - Recalculating AnnualSalary ensures available employee data is accurate, consistent and up-to-date
   - To handle outliers in the DrivingCommuterDistance column, I used the Interquartile Range (IQR) method to identify any extreme values. Instead of removing these rows from the dataset, I chose to apply outlier capping or clipping

(also known as Winsorizing), which caps values above the upper threshold. This method preserves all rows and avoids reducing the dataset size.

3. Describe **two** or more advantages to your data cleaning approach specified in part C1.
    1. It preserves the dataset size by imputing or capping rather than deleting records altogether.
    2. It improves consistency across categorical and standardized fields.
    3. The logical corrections ensure that data reflects realistic values without the need for manual editing.

4. Discuss **two** or more limitations to your data cleaning approach specified in part C1.
    1. Annual salary recalculation will rely on other values being accurate in order to produce an accurate calculation.
    2. Outlier capping may hide extremely high, yet accurate values; for example, long commuter distances or high salaries.

Part III: Submission

D. Submit your findings by doing the following:

1. Provide a data cleaning report as a document file that includes responses to task prompts.
   NicoleGallo_TCN1 Task 1 Data Cleaning and Profiling_attempt2.docx
2. Provide the annotated code you used to detect and mitigate the data quality as an executable script file. R files and Python script files are accepted.
   NicoleGallo_D599_Task1_attempt3.py
3. Provide a copy of the cleaned dataset as a CSV file.
   NicoleGallo_Employee_Turnover_Cleaned.csv
4. Panopto video:
   https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ba031a08-2f2b-4ba3-9f3b-b30e00f5f9e8

Sources

E. The only external source I used beyond WGU resources was the following to learn more about imputation:

Ahmed, S. B. (n.d.). *When to use mean, median, and mode for handling missing values in data?* LinkedIn. https://www.linkedin.com/pulse/when-use-mean-median-mode-handling-missing-values-data-ahmed-tebje/