Task 1:

Data Cleaning and Profiling

Keniyah Chestnut

Data Preparation and Exploration — D599

SID:012601305

A1: PROFILE DATA

The initial dataset is a CSV file containing 10,199 rows and 16 columns of data about employees at a technology company. Each row represents an individual employee record, with columns covering a wide range of attributes, including demographic details, compensation, commuting distance, job roles, work history, and turnover status. This dataset was prepared to support analysis of employee turnover and related factors

A2 and A3: VARIABLE DATA TYPES and OBSERVABLE VALUES

Variable	Type	Subtype	Observable Values
Employee Number	Numeric	Integer	101, 202, 301
Age	Numeric	Integer	24, 30, 49
Tenure	Numeric	Float	1.0, 4.5, 10.0
Turnover	Text	Object	Yes, No
Compensation Type	Text	Object	Salaried, Hourly
Hourly Rate	Numeric	Float	15.0, 30.5, 50.0
Hours Weekly	Numeric	Integer	30, 40, 45
Annual Salary	Numeric	Float	50000, 75000, 150000
Driving Commuter Distance	Numeric	Float	2.0, 10.5, 55.0
Job Role	Text	Object	Data Analyst, Developer
Gender	Text	Object	Male, Female
Marital Status	Text	Object	Single, Married
Number of Companies Worked	Numeric	Integer	1, 3, 5
Annual Professional Development Hours	Numeric	Float	5.0, 40.0, 70.0
Paycheck Method	Text	Object	Direct Deposit, Mail Check
Text Message Opt-In	Text	Object	Yes, No

Additional Observations:

- Although Tenure and Annual Professional Development Hours are recorded as floats, they only contain whole number values and could be stored as integers.
- All categorical columns are stored as text objects (strings), as expected.

Inspection Method:

I used the .dtypes attribute and .head() method in Python Pandas to inspect the datatypes and review observable values for each column.

Part II: Data Cleaning and Plan

B1:DATASET QUALITY ISSUES AND B2: LIST OF QUALITY ISSUES

I started by importing the data into the data frame:

```
[1]: # Import required library
     import pandas as pd

[3]: # Load the dataset
     df = pd.read_csv("Employee Turnover Dataset.csv")

[5]: # Check the initial shape of the dataset
     print("Initial dataset shape:")
     print(df.shape)

     Initial dataset shape:
     (10199, 16)
```

The dataset was inspected for duplicates, missing values, inconsistent entries, formatting errors, and outliers using Python Pandas functions.

Quality Issue 1: Duplicate Entries

First, I checked with the following code for duplicate rows:

```
# ----------------------------
# Quality Issue #1 - Duplicate Entries
# ----------------------------

# Check for duplicated rows
num_duplicates = df.duplicated().sum()
print("\nNumber of duplicate rows:")
print(num_duplicates)
```

```
Number of duplicate rows:
99
```

After identifying duplicate rows, I reviewed them directly in the dataset to verify that they weretrue duplicates before proceeding with their remove

```
# If duplicates exist, display them
if num_duplicates > 0:
    duplicate_rows = df[df.duplicated()]
    print("\nDuplicate rows:")
    print(duplicate_rows)
```

```
Duplicate rows:
       EmployeeNumber  Age  Tenure Turnover HourlyRate   HoursWeekly  \
10100               1   28       6      Yes     $24.37            40
10101               2   33       2      Yes     $24.37            40
10102               3   22       1       No     $22.52            40
10103               4   23       1       No     $22.52            40
10104               5   40       6       No     $88.77            40
...               ...  ...     ...      ...        ...           ...
10194              95   48      13      Yes     $85.40            40
10195              96   54      17       No     $85.40            40
10196              97   44       6       No     $71.90            40
10197              98   58      19       No     $71.90            40
10198              99   48      17      Yes     $71.33            40

      CompensationType  AnnualSalary  DrivingCommuterDistance  \
10100           Salary       50689.6                       89
10101           Salary       50689.6                       89
10102           Salary       46841.6                       35
10103           Salary       46841.6                       35
10104           Salary      284641.6                       12
...                ...           ...                      ...
10194           Salary      177632.0                       31
10195           Salary      177632.0                       31
10196           Salary      149552.0                       32
10197           Salary      149552.0                       32
10198           Salary      148075.2                       50
```


```
      CompensationType  AnnualSalary  DrivingCommuterDistance  \
10100           Salary       50689.6                       89
10101           Salary       50689.6                       89
10102           Salary       46841.6                       35
10103           Salary       46841.6                       35
10104           Salary      284641.6                       12
...                ...           ...                      ...
10194           Salary      177632.0                       31
10195           Salary      177632.0                       31
10196           Salary      149552.0                       32
10197           Salary      149552.0                       32
10198           Salary      148075.2                       50
```


```
                  JobRoleArea                 Gender MaritalStatus  \
10100                Research                 Female       Married
10101                Research                 Female       Married
10102  Information_Technology                 Female        Single
10103  Information_Technology                 Female        Single
10104                   Sales  Prefer Not to Answer         Single
...                       ...                    ...           ...
10194                Research                   Male        Single
10195                Research                   Male        Single
10196               Marketing                   Male       Married
10197               Marketing                   Male       Married
10198                   Sales  Prefer Not to Answer        Married

      NumCompaniesPreviouslyWorked  AnnualProfessionalDevHrs PaycheckMethod  \
10100                          3.0                       7.0     Mail Check
10101                          6.0                       7.0     Mail Check
10102                          1.0                       8.0   Mailed Check
10103                          3.0                       NaN   Mailed Check
10104                          7.0                       NaN     Mail Check
...                            ...                       ...            ...
10194                          7.0                       5.0     Mail Check
10195                          2.0                      25.0     Mail Check
10196                          6.0                       NaN     Mail Check
10197                          5.0                      23.0     Mail Check
10198                          8.0                       8.0     Mail Check
```

```
      TextMessageOptIn
10100              Yes
10101              Yes
10102              Yes
10103              Yes
10104              Yes
...                ...
10194              NaN
10195              Yes
10196              Yes
10197              Yes
10198              Yes

[99 rows x 16 columns]
```

Quality Issue 2: Missing values

To identify missing values, I used Python's .isnull() function combined with .sum() to count the number of null cells in each column. This allowed me to see which columns had missing data.

```
# -----------------------------------------------
# Quality Issue #2 - Missing Values
# -----------------------------------------------

# Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())
```

The Results :

```
Missing values in each column:
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
```

Three of the sixteen columns contained missing data, with TextMessageOptIn having the highest number of missing entries, making it especially problematic. This issue, along with the other missing values, was addressed during the data cleaning process, which is discussed in the following section.

Quality Issues #3 and #4: Inconsistent Entries and Formatting Errors

Before cleaning the inconsistent entries, I used the .unique() function to review the distinct values present in each categorical column.

- This allowed me to identify any inconsistencies, such as:
- Different capitalizations (e.g., "DirectDeposit" vs "Direct Deposit")
- Underscores and combined words (e.g., "Mail_Check" vs "Mail Check")
- Extra spaces or misspellings

This step was essential to ensure that later analysis and models would not treat inconsistent entries as separate categories.

```
for col in categorical_cols:
    # Check unique values before cleaning
    print("\nUnique values in", col, "before cleaning:")
    print(df[col].unique())
```

The results:

```
Unique values in CompensationType before cleaning:
['Salary']

Unique values in PaycheckMethod before cleaning:
['Mail Check' 'Mailed Check' 'Direct_Deposit' 'DirectDeposit'
 'Direct Deposit' 'Mail_Check' 'MailedCheck']

Unique values in JobRoleArea before cleaning:
['Research' 'Information_Technology' 'Sales' 'Human_Resources'
 'Laboratory' 'Manufacturing' 'Healthcare' 'Marketing'
 'InformationTechnology' 'HumanResources' 'Information Technology'
 'Human Resources']

Unique values in Gender before cleaning:
['Female' 'Prefer Not to Answer' 'Male']

Unique values in MaritalStatus before cleaning:
['Married' 'Single' 'Divorced']

Unique values in TextMessageOptIn before cleaning:
['Yes' 'No']
```

By printing the unique values before cleaning, I could easily determine which columns required manual mapping and formatting corrections.

Quality Issue 5: Outliers

I reviewed the summary statistics of the numeric columns to identify outliers, such as unusually high or negative values.

```
[55]: # Outliers check using summary statistics
print("\nSummary statistics for numeric columns (Outlier Check):")
print(df.describe())
```

The Results:

```
Summary statistics for numeric columns (Outlier Check):
       EmployeeNumber           Age        Tenure  HoursWeekly   AnnualSalary  \
count    10100.000000  10100.000000  10100.000000      10100.0   10100.000000
mean      5050.500000     44.078911      9.007624         40.0  120994.773564
std       2915.763193     10.213311      5.512046          0.0   77358.965898
min          1.000000     21.000000      1.000000         40.0  -33326.400000
25%       2525.750000     37.000000      5.000000         40.0   63440.000000
50%       5050.500000     44.000000      8.000000         40.0  101774.400000
75%       7575.250000     53.000000     13.000000         40.0  153717.200000
max      10100.000000     61.000000     20.000000         40.0  339950.400000

       DrivingCommuterDistance  NumCompaniesPreviouslyWorked  \
count             10100.000000                  10100.000000
mean                 45.165743                      3.942970
std                  51.390866                      2.618329
min                -275.000000                      0.000000
25%                  13.000000                      2.000000
50%                  42.000000                      3.000000
75%                  71.000000                      6.000000
max                 950.000000                      9.000000

       AnnualProfessionalDevHrs
count              10100.000000
mean                  14.954455
std                    5.466432
min                    5.000000
25%                   11.000000
50%                   15.000000
75%                   19.000000
max                   25.000000
```

The summary statistics revealed unrealistic outliers, such as an AnnualSalary of -33,326 and a DrivingCommuterDistance of -275. Since negative salaries and commute distances are not possible, these were identified as errors and corrected during the cleaning process.

C1: DATASET MODIFICATION

Below, I will outline the issues identified in the dataset, along with the code used to correct them and the validation steps taken to confirm the corrections. I will also explain why each cleaning method was necessary and how it improved the data quality.

Quality Issue 1: Duplicate Entries

Issue: There were 99 duplicate rows in the dataset, which provided no additional value and needed to be removed.

Solution: These duplicates were deleted to ensure the dataset contained only unique and meaningful records.

```
[13]: # Remove duplicate rows
      df = df.drop_duplicates()
```

```
[15]: # Confirm duplicates removed
      print("\nDataset shape after removing duplicates:")
      print(df.shape)
```

```
Dataset shape after removing duplicates:
(10100, 16)
```

As shown, the dataset now contains 10,100 rows instead of the original 10,199, confirming that the 99 duplicate rows were successfully removed.

Quality Issue 2: Missing Values

Three of the sixteen columns had missing data. TextMessageOptIn was the most problematic, as a large portion of its entries were missing. Since opting in to receive text messages could be a relevant factor in employee retention, I determined that these missing values should not be left blank. Instead, I replaced missing entries with "No" to indicate that these employees did not opt in.

For NumCompaniesPreviouslyWorked, missing values were filled with 0, assuming no previous companies worked at if left blank.

Numeric columns such as AnnualProfessionalDevHrs were filled with the median value to maintain consistency and avoid deleting valuable records.

Solution:

```
[23]: # Fill numeric missing values with median
df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].fillna(df['DrivingCommuterDistance'].median())
df['AnnualProfessionalDevHrs'] = df['AnnualProfessionalDevHrs'].fillna(df['AnnualProfessionalDevHrs'].median())

# Fill TextMessageOptIn with "No"
df['TextMessageOptIn'] = df['TextMessageOptIn'].fillna('No')

# Fill NumCompaniesPreviouslyWorked with 0
df['NumCompaniesPreviouslyWorked'] = pd.to_numeric(df['NumCompaniesPreviouslyWorked'], errors='coerce')
df['NumCompaniesPreviouslyWorked'] = df['NumCompaniesPreviouslyWorked'].fillna(0)
```

Confirmed the missing data had been filled.

```
[21]: # Confirm missing values handled
print("\nMissing values after filling:")
print(df.isnull().sum())
```

```
Missing values after filling:
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
```

Quality Issues 3 & 4: Inconsistent Entries and Formatting Errors

Issue: Some columns had inconsistent and incorrectly formatted values, such as "MailedCheck", "Mail_Check", and "DirectDeposit". These could cause errors by treating the same thing as different categories.

Solution:

```
# Manual mappings for known inconsistent entries

# PaycheckMethod fixes
df['PaycheckMethod'] = df['PaycheckMethod'].replace({
    'Direct_Deposit': 'Direct Deposit',
    'DirectDeposit': 'Direct Deposit',
    'Mail_Check': 'Mail Check',
    'MailedCheck': 'Mail Check',
    'Mailed Check': 'Mail Check'
})

# JobRoleArea fixes
df['JobRoleArea'] = df['JobRoleArea'].replace({
    'Information_Technology': 'Information Technology',
    'InformationTechnology': 'Information Technology',
    'Human_Resources': 'Human Resources',
    'HumanResources': 'Human Resources'
})

# Apply general cleaning (strip and title case)
for col in categorical_cols:
    df[col] = df[col].astype(str).str.strip().str.title()
```

Confirmed the data formatting was fixed:

```
[51]:
# Confirm unique values after cleaning
for col in categorical_cols:
    print("\nUnique values in", col, "after cleaning:")
    print(df[col].unique())
```

```
Unique values in CompensationType after cleaning:
['Salary']

Unique values in PaycheckMethod after cleaning:
['Mail Check' 'Direct Deposit']

Unique values in JobRoleArea after cleaning:
['Research' 'Information Technology' 'Sales' 'Human Resources'
 'Laboratory' 'Manufacturing' 'Healthcare' 'Marketing']

Unique values in Gender after cleaning:
['Female' 'Prefer Not To Answer' 'Male']

Unique values in MaritalStatus after cleaning:
['Married' 'Single' 'Divorced']

Unique values in TextMessageOptIn after cleaning:
['Yes' 'No']
```

Quality Issue 5: Outliers

Issue:

Some numeric columns contained extreme or unusual values. For example, AnnualSalary included a negative number (-33,326.40) and DrivingCommuterDistance had a negative value (-275). These outliers are not realistic and could affect analysis.

Solution:

```
[59]:
import numpy as np


# Fix negative AnnualSalary
# Replace negative salaries with NaN
df['AnnualSalary'] = df['AnnualSalary'].apply(lambda x: x if x >= 0 else np.nan)

# Fill NaN (negative values replaced) with median salary
df['AnnualSalary'] = df['AnnualSalary'].fillna(df['AnnualSalary'].median())

# Fix negative DrivingCommuterDistance
# Replace negative distances with NaN
df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].apply(lambda x: x if x >= 0 else np.nan)

# Fill NaN (negative values replaced) with median distance
df['DrivingCommuterDistance'] = df['DrivingCommuterDistance'].fillna(df['DrivingCommuterDistance'].median())
```

The Results:

```
[61]: # Confirm fixed values
print("\nFixed AnnualSalary summary:")
print(df['AnnualSalary'].describe())

print("\nFixed DrivingCommuterDistance summary:")
print(df['DrivingCommuterDistance'].describe())
```

```
Fixed AnnualSalary summary:
count     10100.000000
mean     121612.107267
std       76735.205764
min        1307.200000
25%       63835.200000
50%      102440.000000
75%      153717.200000
max      339950.400000
Name: AnnualSalary, dtype: float64

Fixed DrivingCommuterDistance summary:
count    10100.000000
mean        53.421683
std         44.483847
min          0.000000
25%         30.000000
50%         49.000000
75%         71.000000
max        950.000000
Name: DrivingCommuterDistance, dtype: float64
```

```
[ ]:
```

C2: DATA CLEANING TECHNIQUES

1. Removing Duplicate Rows

Duplicate records provided no new information and could bias the analysis. These rows were removed to ensure that each record was unique and meaningful.

2. Handling Missing Values

Numeric Columns (e.g., AnnualProfessionalDevHrs):

Missing values were filled using the median. The median is preferred because it is not affected by outliers and gives a reasonable estimate of typical values.

Categorical Columns (e.g., TextMessageOptIn):

Logical default values were applied. For example, missing entries in "TextMessageOptIn" were filled with "No" to maintain consistency.

3. Correcting Inconsistent Entries

Some text entries were inconsistent (e.g., "Mail_Check", "MailedCheck", and "Mailed Check"). These were standardized using the .replace() method to make sure all similar values were uniform.

4. Correcting Formatting Errors

Categorical columns also contained entries with inconsistent capitalization and extra spaces. These were fixed using .str.strip() and .str.title(), ensuring that text entries were properly formatted and consistent.

5. Handling Invalid Numeric Values (Outliers)

Some numeric columns, such as AnnualSalary and DrivingCommuterDistance, contained invalid negative values (e.g., -33,326 and -275). These were replaced with NaN and filled with the median to maintain realistic and logical values for analysis.

These methods were selected to make the data accurate, consistent, and ready for analysis without distorting meaningful patterns.

C3: TECHNIQUE ADVANTAGES

- Improved Data Accuracy and Consistency:

Cleaning inconsistent text values ensures that analysis does not treat identical entries as separate categories. This improves accuracy in aggregations, counts, and models.

- Preservation of Data Integrity:

By filling missing and invalid numeric values with the median, the data remains as close to its original distribution as possible, reducing distortion while still resolving gaps.

- Prevention of Analytical Errors:

Removing duplicates and fixing invalid entries prevents issues like double-counting or incorrect averages, which improves the reliability of future analysis and modeling.

C4: TECHNIQUE LIMITATIONS

- Assumptions Made During Filling Missing Values:

Filling missing numeric data with the median assumes that missing values are random and similar to existing values. If the missingness is not random, this could introduce bias.

- Outliers Were Not Removed Completely:

Although negative values were corrected, other extreme outliers (very high salaries or commute distances) were not removed, as further business review is required. These could still affect some analysis if not handled carefully later.

- Manual Mapping Risk:

The process of manually mapping inconsistent categorical entries depends on human judgment. There is always a small risk of missing some variations or making incorrect mappings.

References

No outside sources were used. All materials and datasets were provided by WGU.