EMPLOYEE TURNOVER CONSIDERATIONS

Data Profiling, Cleaning Plan, and Findings

Table of Contents

EXECUTIVE SUMMARY

Business Problem

This report prepares the employee dataset for turnover analysis, enabling HR to forecast risk, prevent attrition where possible, improve the employee experience, reduce the costs of training and onboarding of new employees, and improve productivity of the business.

Key Insights

Dataset Used: `employee_turnover_considerations.csv`

Data Size: 10,199 Rows | 16 Columns

Unique Primary Key: `EmployeeNumber`

Data Quality: Some missing values in `AnnualProfessionalDevHours`, duplicate entries, and inconsistent text on `PaycheckMethod`. The analysis found a total of 99 duplicates.

Time Range: The stakeholders did not specify the time range. The dataset includes recently hired employees. The maximum tenure is 20 years. The most frequent tenure is 1 year, with a median value of 8 years.

Cleaning Actions Taken

Duplicates: Removed 99 rows (0.97%).

***Standardization**:* Normalized labels (categorical) by removing underscores and extra spaces; aligned derived column values (`HourlyRate`); applied consistent pay formats.

Missing Values: Imputed values in `DrivingCommuterDistance` (medians). Imputed NaN for `AnnualProfessionalDevHours` with flags to preserve transparency.

Outliers/Business Rules: removed negative distances for `DrivingCommuterDistance`. Flagged values > 99th percentile for review and validated `AnnualSalary` computation by applying the business rule (`HourlyPay` x `HoursWeekly` x 52).

Before/After Metrics:

Metric	Before	After
Duplicate ID	99	0
Missing Values Total	4868 (3.01%)	0
Missing Values: `TextMessagesOptIn`	2259 (22.36%)	0
Missing Values: `AnnualProfessionalDevHrs`	1947 (19.27%)	0
Missing Values: `NumCompaniesPreviouslyWorked`	663 (6.56%)	0
Consistency: underscores and whitespaces detected	Detected	Resolved
Validity (out-of-range `DrivingCommuteDistances`***)***	205 Flagged	100% Reviewed
Accuracy of Calculated Column `AnnualSalary`	2% Mismatched	<0.01% Mismatched
Rows	10199	10100 (0.97% removed)
Columns	16	24 (Added 8 flag columns for auditing)

PART I: DATA PROFILING

A1. TOTAL NUMBER OF RECORDS AND VARIABLES

Key Counts

Rows: 10199 | Columns: 16

Timeframe

No specific time range given; the dataset is a snapshot of employee information captured at a single point in time. This dataset includes recently hired employees (with 0 years of tenure). The maximum tenure is 20 years. The most frequent tenure is 1 year, with a median value of 8 years.

Entities

- The primary entity is Employee, uniquely identified by `EmployeeNumber`
- Each Employee has a `JobRole/Area` (12 categories)
- `Gender` (3 categories, including `Prefer Not to Answer`)
- `MaritalStatus` (3 categories)
- `Paycheck Method` (`Direct Deposit`, `Mail Check`, `Mailed Check`, etc., in 7 forms, needs standardization),
- `Turnover` (2 categories),
- `TextMessageOptIn` (2 categories)

Row Grain

One row per unique Employee identified by `EmployeeNumber`.

Primary Keys

`EmployeeNumber` is the primary key. It uniquely identifies each Employee in the dataset.

Foreign Keys

This dataset does not include foreign keys because it is a single flat file.

A2. DISTINCT VARIABLES, DATA TYPES, AND SUBTYPES

Variable	Definition	Data Type	Sub-type
`EmployeeNumber`	Unique employee identifier	Qualitative Categorical	Nominal Identifier/label (not used in math)
`Age`	Employee age	Quantitative Numerical	Continuous Recorded in whole years, but conceptually continuous

`Tenure`	The length of time the employee has worked for the company	Quantitative Numerical	Discrete (years stored as integers)
`Turnover`	Did the employee leave in the last year?	Qualitative Categorical	Nominal (Yes/No)
`HourlyRate`	Pay per hour	Quantitative Numerical	Continuous (Currency)
`HoursWeekly`	Weekly hours worked	Quantitative Numerical	Discrete (full hours)
`CompensationType`	Salaried or hourly employee	Qualitative Categorical	Nominal (hourly/salary)
`AnnualSalary`	Hourly rate multiplied by 52 times hours worked weekly	Quantitative Numerical	Continuous (derived currency)
`DrivingCommuterDistance`	Distance to the office in miles	Quantitative Numerical	Continuous (miles)
`JobRoleArea`	Employee's role	Qualitative Categorical	Nominal (department/role)
`Gender`	Male/female/prefer not to answer	Qualitative Categorical	Nominal
`MaritalStatus`	Single/Divorced/Married	Qualitative Categorical	Nominal
`NumCompaniesPreviouslyWorked`	Employee's work history	Quantitative Numerical	Discrete (count)
`AnnualProfessionalDevHours`	Hours spent on development per year	Quantitative Numerical	Continuous (hours count)
`PaycheckMethod`	How the employee receives pay	Qualitative Categorical	Nominal
`TextMessageOptin`	Whether the employee receives text messages from the employer	Qualitative Categorical	Nominal (Yes/No/ NaN for missing data)

A3. SAMPLES OF OBSERVABLE VALUES FOR EACH VARIABLE

Variable	**Sample Values**				
`EmployeeNumber`	611	1565	69	2157	2884
`Age`	36	48	40	58	39
`Tenure`	1	6	4	14	8

Turnover	Yes	No	Yes	No	Yes
HourlyRate	$27.28	$70.74	$85.33	$32.12	$30.21
HoursWeekly	40	40	40	40	40
CompensationType	Salary	Salary	Salary	Salary	Salary
AnnualSalary	56534.4	147139.2	177486.4	66809.6	62836.8
DrivingCommuter\ Distance	67	58	1	25	42
JobRoleArea	Research	Information Technology	Sales	Research	Research
Gender	Female	Male	Prefer not to answer	Female	Female
MaritalStatus	Divorced	Married	Single	Single	Divorced
NumCompanies\ PreviouslyWorked	5.0	8.0	9.0	3.0	3.0
AnnualProfessional\ DevHours	5.0	8.0	19.0	NaN	7.0
PaycheckMethod	Mail_Check	Mail Check	Mail Check	Mail Check	Direct_Deposit
TextMessageOptin	No	Yes	NaN	Yes	Yes

PART II: DATA CLEANING PLAN

B1. EXPLANATION OF DATA QUALITY INSPECTION METHODS

Duplicate Entries Inspection

Basic Inspection

The basic inspection of the dataset begins with the evaluation of duplicate records in two steps. First, the code scans for exact duplicates and, in the second pass, checks for key duplicates.

First Pass

The first pass targets exact rows where every column matches a column in a different row (Koneshloo, 2022). The df.duplicated() function flags the repeated rows, via df.duplicated().sum() returning a total count of exact row duplicates (here, equal to 99).

Second Pass

The second pass evaluates key duplicates for collisions starting with `df[pk].is_unique`, checking if each ID appears only once. If it returns False, at least one value repeats, and the code counts the total of duplicated rows with `df[pk].duplicated().sum()`. Finally, `exact_dupes = df[df.duplicated()]` isolates the second occurrences of each duplicate for review, and `len(exact_dupes)` confirms the count. Matching counts between exact duplicates and key duplicates (by comparing the first and second pass tests) indicate that key collisions arise from whole row duplicates.

Missing Values Inspection

Column Inspection

The inspection of missing values uses column counts, via `isna().sum()`, to identify incomplete data. This method highlights the presence of missing data and the scale across the dataset. The column inspection reveals three columns with significant gaps: `TextMessageOptIn`, `AnnualProfessionalDevHrs`, and `NumCompaniesPreviouslyWorked`.

Missing Values In Rows

A second check uses the `isna()` function on `df_clean` to find missing values in rows `(axis=1)` and sort them in descending order. This inspection method gives visibility into the records with missing data by the highest count and points to which specific columns need further review.

Scope and Distribution

The row/column inspection methods work together to show the scope (how many missing overall) and distribution (across which rows) of data gaps, which helps narrow down the areas of data that need application of missing value handling methods for three variables (columns representing missing 22.3%, 19.2%, and 6.5% of total values in the data).

Inconsistent Entries Inspection

Inventory of categories within column values

A preview of random data (`df.sample`) pointed to variations in categorical field values. The `PaycheckMethod` variable included 7 unique categorizations for 2s payment types. Payments sent by mail had 4 distinct categories, and direct deposits had 3 separate spelling forms. For checks sent by mail, the values included: `Mail Check`, `Mailed Check`, and `MailedCheck, Mailed_Check. For funds deposited to an employee's bank, the values included Direct_Deposit, DirectDeposit,` and `Direct Deposit.` All 7 values have only 2 distinct meanings, with spelling as the only difference between them. The code scanned and printed the spelling variations using a loop with `.nunique` and `.unique` attribute and printing the unique values with their counts.

Storage of Primary Key (Identifier)

Although storing a primary key in an inappropriate data type is not a direct data consistency issue, this data cleaning document includes PK storage under the consistency section to emphasize that consistently storing the `EmployeeNumber` field supports business rule enforcement and schema integrity. Here, the preliminary inspection revealed that the data stored EmployeeNumber as an integer, which enables arithmetic calculations and accidental modifications to the unique ID.

Standardization of Category Names

As an inspection method applied after standardizing category names (by defining and mapping existing entries to consistent values with the same semantic meaning), the code printed a table. The table provided a view of each variable's category values, printing 'VariableName after mapping:' and the current values after the change. This step diagnosed and double-checked redundancy in equivalent values, confirming consolidation.

Casting of Categorical Variables
Casting cleaned columns to the category dtype locks their valid levels and prevents drift. Each categorical field runs through an allowed values check to validate its domain. `PaycheckMethod` allows only `Direct Deposit` and `Mailed Check`, while `CompensationType` allows only `Salaried` and `Hourly`. This verification served as an inspection method to ensure stability and remove any remaining mismatches between formatting and semantics.

AnnualSalary Derived Column Value
Additionally, the code inspected the consistency of calculating the `AnnualSalary` business rule. The code inspected the derived value by calculating the expected salary as `HourlyRate` x `HoursWeekly` x 52. The script created a Boolean mask for the mismatched values, with a $1 tolerance to allow rounding. The code created a preview of mismatched rows, calculating the difference between the expected value and the stored value.

Primary Key Data Type
The check of data types, using `.info()` attribute (Part I: Data Profiling), revealed that the data frame stored `EmployeeNumber` as `int64`. Since this variable is the primary key, uniquely identifying each employee, this data type storage was not appropriate.

Formatting Errors Inspection

Formatting with whitespaces and underscores
Values like `Mail_Check` and `Mail Check` have the same meaning in business, but the inclusion of an underscore changes the results of data analysis. If left in the current form, Python would treat those as distinct labels. The HourlyPay variable included an extra trailing space, which is common issue in data. To inspect extra space formatting problems, the code used a regular expression scan (regex) via the re library. It scanned through the columns to identify extra whitespaces, trailing spaces, and underscore symbols. The target columns included: `CompensationType`, `JobRoleArea`, `Gender`, `MaritalStatus`, `PaycheckMethod`, `TextMessageOptIn`, and `Turnover`.

Column Names
Accurate analysis results require consistent variable naming, especially when performing calculations that produce derived column values (e.g., hourly pay times hours worked weekly times 52 weeks equals yearly salary). During the diagnostic process, the code displayed a preview of the column names. It revealed a trailing space in `HourlyRate`, which means that calculating the derived column `AnnualSalary` would result in an error unless the code accounts for the extra space in `HourlyRate`.

Numeric Field Structure Check
The pay fields (`HourlyRate`, `HoursWeekly`, and `AnnualSalary`) contain commas, currency symbols, and non-numeric characters that interfere with parsing. The diagnostic routine searches for these symbols and non-numeric data types, verifying numeric integrity before conducting comparison tests. These issues represent text-format errors, not invalid measurements.

Outliers Inspection

The script workflow scanned the dataset to identify outliers. Because standard outlier detection methods only work on numeric data, the script first scanned the numeric columns (Cabral, 2025). It used the `select_dtypes` method to separate fields containing only integer and float data types. The code applied the `.describe().T` function to generate summary statistics (count, mean, standard deviation, minimum, quartiles, and maximum) for each numeric variable. The script added a column called `OutliersPresent`, evaluated whether any numeric variable had at least one `Upper` or `LowerFence` outlier, and printed `True` for counts `>0` and `False` for counts `=0`. By comparing quartile ranges (25%, 50%, 75%) against extreme minimum and maximum values, the code detected unusual values for review. This approach enabled the quick detection of anomalies without the need for complex statistical methods. The `DrivingCommuterDistance` column preview revealed absurd values, while null entries indicated missing data. The check returned 1343 invalid or missing values.

B2. DISCUSSION OF FINDINGS FOR EACH QUALITY ISSUE

Duplicate Entries Findings

The dataset contains 99 exact row duplicates. The primary key check confirms 99 repeated `EmployeeNumber` values, indicating that each repeat is a precise copy of its original record. Keeping the first occurrence and dropping exact duplicates reduces the data from 10,199 to 10,100 rows, restores the integrity of the data to one row per Employee, and minimizes the row count by around 0.97%.

Missing Values Findings

The inspection identified significant gaps in three variables: `TextMessageOptIn` (2258), `AnnualProfessionalDevHrs` (1947), and `NumCompaniesPreviouslyWorked` (663). Together, 4868 data points were incomplete. For `TextMessageOptIn,` the code inventoried the unique values and counted nulls; after normalizing case to only `‘Yes’ and ‘No.’` The code treated all other values as missing.

At the row level, the analysis revealed that a significant number of employees had at least one unrecorded field, but most columns were complete. The dataset had 4183 missing employee records at the row level. The key column (`EmployeeNumber`) did not contain any missing values, enabling the unique tracking of each Employee and supporting the integrity of the analysis of employee turnover (which is the stated business purpose of the analysis).

Inconsistent Entries Findings

The code analyzed label names and printed the multiple representations of identical categories. `Mail Check` vs. `Mailed_Check` have the same meaning. Still, this inconsistency creates the appearance of additional categories and can skew analysis, leading to potential distortions in summaries and charts. The preview identified 7 unique categories for `PaycheckMethod`: (1) `Mail Check`, (2) `MailedCheck`, and (3) `Mailed Check and Mail_Check` (4); `Direct Deposit`, (5) `DirectDeposit`, (6) `Direct_Deposit`. In reality, these are just two distinct categories.

The code also found 2122 mismatched values in the derived AnnualSalary column. The values did not meet the business rule calculation, which multiplies `HourlyRate` by 40 (`HoursWeekly`) and 52.

Although not a strict consistency problem, data storage for primary keys enforces business rule consistency; therefore, this document addresses transforming the `EmployeeNumber` column and changing the storage from `int` to `string`. It is safer to store the unique identifier as a string to prevent calculations and the loss of a primary key.

Formatting Errors Findings

Categorical fields contained underscores, extra trailing spaces, and strings that produced additional unnecessary labels.

JobRoleArea
The `JobRoleArea` column had 944 records for `Human Resources`, 51 records for `HumanResources`, 899 records for `Information Technology`, and 80 for `InformationTechnology`, creating a duplicate category caused by missing spaces.

PaycheckMethod
The `PaycheckMethod` had 5,464 records for `Mail Check`, 2,425 records for `Mailed Check`, and 49 records for `MailedCheck`, which accounted for formatting issues.

HourlyRate
The `HourlyRate` column name included a trailing space. This minute detail would interfere with numerical operations on the data in the future. Labels included underscores between words, increasing the distinct label counts. `HourlyRate` values included the dollar sign symbol (a string). The presence of a currency symbol could interfere with future numerical calculations for the `AnnualSalary` derived column.

Outliers Findings

The inspection revealed that most numeric fields fall within realistic ranges. However, `DrivingCommuterDistance` contained several outlier values. The minimum value of –275 is invalid for a distance field. The maximum of 950 is an implausible commute. All other columns (`Age`, `Tenure`, `HourlyRate`, `AnnualSalary`, etc.) did not raise suspicion since they were reasonable and within an acceptable range in the context. Notably, 1,343 records contained negative driving distance values, which are invalid and nonsensical for a field that logically exists only as a positive number. On the other side of the spectrum, 205 records (approximately 2%) contained unusually high positive distances greater than the 99th percentile. These were flagged using a new variable, `DrivingCommuterDistanceFlag`, for contextual review rather than removal. 9,895 rows remain within the acceptable range. The business rules define `DrivingCommuterDistance` as the driving distance to the office, which means that negative values are likely due to data entry errors. Very high distances may indicate remote workers or incorrect ZIP-based calculations.

C1. DATA MODIFICATIONS USING PYTHON

Duplicate Entries Modifications

The duplicate cleanup starts by creating a list of duplicates (second copies of the same row) for review. The code removes the second instance of the repeated row, reports the number of rows remaining after removal, and displays the percentage of data rows that the script removed. The code runs a primary key uniqueness test to confirm that no exact duplicates remain. The workflow saves the deduplicated CSV file for audit as df_dedup.csv. As a final checkpoint, the code verifies whether the primary key is unique, and the program prints True, confirming that the primary keys are indeed unique.

Missing Values Modifications

TextMessagesOptIn
The `TextMessagesOptIn` column had 2258 missing values, representing 22.3% of the column's data. The code normalized values by converting to title case, kept NaN, and added a flag (No-0 and Yes-1) for missing values. The script added a derived, display-only column, replacing NaN with Missing.

AnnualProfessionalDevHours
The `AnnualProfessionalDevHrs` column had 1947 missing values, representing 19.2% of all the data in that column. The cleaning method included two imputation methods:

- *Option 1*:
  The code marked each record with a flag indicating whether it had missing values (1 for a missing record, 0 for an existing string record). The flag identified employees with no recorded development hours, allowing for analysis to determine if missing training hours relate to the employee turnover rates. The code fills only the flagged values with 0.0 to represent that the Employee has completed no development training. The script also enforces the float data type to allow the column to hold decimals, so the future inputs into the fields allow partial hours. The sequence stabilizes the data, preserves information on employees who have no recorded development hours, and keeps the data ready for analysis.

- *Option 2:*
  The code calculates the statistical distribution of the `AnnualProfessionalDevHrs` column. Next, the script creates a separate imputed column, uses the median (15.0) for the values, aligning with the central tendency and keeping the same distribution for a future sensitivity analysis.

NumCompaniesPreviouslyWorked
`NumCompaniesPreviouslyWorked` had 663 missing values, representing 6.5% of all the column's data. The code adds a flag for missing values (1 for no record and 0 for recorded), fills the unrecorded values with the median (4), and stores the values in whole numbers to preserve business meaning.

Formatting Issues Modifications

Whitespaces and Underscores

The code prioritizes repairing formatting issues, handling spaces and underscores, and then addressing semantic problems in the data values. Python utilizes regular expressions and string methods to trim leading, trailing, and internal spaces, ensuring that only one space remains between words. Additionally, it replaces underscores with spaces to standardize the entries. The target columns for space modifications are `CompensationType`, `JobRoleArea`, `Gender`, `MaritalStatus`, `PaycheckMethod`, `TextMessageOptIn`, and `Turnover` (not changing column names, only the cell values within those columns).

Column Names
The code prints all column names, finding that the ``` `HourlyRate ` ``` column has a trailing space. Python removes the space with `str.strip()` from all headers with a `.columns` vectorized approach.

Numeric Field Structure Formatting
The workflow also removes commas and currency symbols from addresses using `str.replace()` and converts the values to numeric using `pd.to_numeric`, coercing errors to NaN.

Formatting Diagnostic Using Python
Python performs a diagnostic inventory post-cleaning using a for loop in category columns and prints the unique value counts after formatting, without dropping missing values. It explicitly displays NaN with its count to indicate the number of blanks remaining.

Inconsistencies Modifications

Standardization of Categories and Column Names (Canonical Mapping)
The preliminary profiling step (early consistency inspection in Part A2) printed a list of various values within each column. A later step removed underscore symbols from the cell values (formatting issue, not inconsistency, but important to explain here what happened to the count of distinct values). Since the code has cleaned up underscores, the number of distinct values in Payment Methods went down from 7 various categories to 5, removing the underscore from Direct_Deposit and Mail_Check.

Next, to further address consistency issues, the script enumerated observed category variations and established a unifying vocabulary, for example, stating that `DirectDeposi`t should be written as `Direct  Deposit` with a space between the text. The workflow used a vectorized replacement method `.replace` (mapping), consolidating synonyms and misspellings so each label appears only in one form for a total of 2 distinct values.

Changing EmeployeeNumber Data Type - Business Rule Consistency
Although casting is not an explicit inconsistency fix, it supports consistency by enforcing a fixed vocabulary. Here, the code casts categorical columns to the categorical data type using `astype("category")` to implement the schema, thereby preventing the appearance of new unintended labels.

Calculated Field
The workflow verifies that the `AnnualSalary` field follows the business rules defined in the data dictionary. Here, the expected `AnnualSalary` is the result of multiplying the hours worked weekly by the hourly rate and then multiplying that result by 52 weeks. The code checks whether the derived field includes the correct numbers and creates a flag for rows with either missing values or values that exceed the tolerance level, specifically those with a difference greater than $1

between the existing value and the accurate calculation, using `.gt(TOL)`. The script replaced miscalculated values with the corrected amounts, in keeping with the business rule.

Inappropriate Data Type Modification – Inconsistency in Business Rule
Python modified the data type of the primary key by casting `EmployeeNumber` from numeric int64 to string using the `astype('string')` method. After casting, the script runs a conditional loop to check whether the field is non-null and unique, since strings do not have a native non-null functionality. If the condition fails in the future, the code execution will stop for correction. By storing the primary key as a string, the code will enforce consistency.

Verification Code
The workflow prints the value counts, data types, and summary statistics to verify that the data is consistent. The script verifies that `HourlyRate` and `HoursWeekly` are non-negative, ensuring they conform to business rules.

Outliers Modifications

The code replaced all negative commuter distance values with missing entries (NaN) to indicate invalid or unavailable data. It then imputed the median distance of 49 miles to those values. The code created a new field, `DrivingCommuterDistanceFlag`, to identify records where the commuter distance exceeded the 99th percentile threshold. The code did not remove high-distance variations but instead flagged them for contextual review, as they may represent remote employees or anomalies caused by ZIP code-based distance calculations.

C2. JUSTIFICATION FOR THE DATA CLEANING METHODS

Duplicate Entries Justification

Exact row deduplications and primary key checks deliver speed, clarity, and carry low risk. The approach preserves uniqueness, prevents inflated counts, and ensures accurate metrics (MacDonald, 2025). With one row for each Employee, the averages no longer skew from overcounted records. Deduplication prepares the dataset for reliable analysis by preserving the grain and integrity. As a final checkpoint, the code verifies whether the primary key is unique, and the program prints 'True'.

Missing Values Justification

TextMessagesOptIn
The `TextMessagesOptIn` column contained the highest number of missing values. This column contains information on whether an employee receives text messages from the employer. Approximately half of the data for this variable contained missing values. `TextMessageOptIn`, as a categorical (nominal) column, received a Missing label (added in a display-only, derived column where data was missing to avoid dropping the entire column, and to preserve the meaning (other values include `Yes` and `No`). Adding the derived column enhances integrity in case the information has business use in a future analysis project and improves communication with stakeholders. Here, the information will not provide significant value to assess the reasons for attrition, but removing the entire variable is not necessary.

AnnualProfessionalDevHrs

`The AnnualProfessionalDevHr column` holds the data about any professional training an Employee has completed. Implementing two imputation options for the AnnualProfessionalDevHrs variable provides value to the business. Since zeros across the professional development column may signal limited access to training, retaining zeros for this data could draw attention to areas for improvement. The second imputation option, which replaces missing values with medians, helps reduce the risk of making incorrect conclusions about an employee's level of engagement or participation in development activities. Both imputation strategies have distinct purposes, but they work together to inform the business with accurate and complete insights:

- *Option 1*
  The first imputation method replaces missing values with zeros to indicate possible lack of employee engagement. If zeros truly indicate that employees do not have access to training opportunities provided by the employer, highlighting this issue can lead to higher engagement, increased retention, and skill development in the workforce. Flagging the 0.0 imputed columns maintains trust and accuracy, and points decision makers in the right direction to accomplish the stated business purpose of this analysis project.
- *Option 2*
  The second alternative imputes medians if desired by the stakeholders. Including the additional option supports flexible analysis and highlights potential insights.

NumCompaniesPreviouslyWorked
`NumCompaniesPreviouslyWorked` stores information about the number of prior companies worked for. This variable can indicate frequent job changes and be associated with turnover risk. Imputing zeros would obscure the meaning of this variable and could indicate that the Employee had worked for previous employers. Imputing the median reduces the distortion in the meaning. The missingness flag maintains the distinction between employment history being truly missing and having no prior employers, allowing tests of whether a higher rate of job switching aligns with higher attrition.

Formatting Issues/Inconsistent Entries Justification (Combined Summary)

Addressing formatting issues first structures the data cleaning process into two practical steps. First, data formatting cleanup centers on syntax and noise. String operations handle surface inconsistencies such as spacing, underscores, and blanks. Consistency centers on enforcing business rules with mapping dictionaries to fix semantic duplication. Casting data to a categorical data type restricts the recurrence of both formatting and consistency issues in the future. Each method for handling consistency and formatting issues aligns with the business problem and the data dictionary.

Formatting Justification

Whitespaces and additional characters typically stem from data entry or export, rather than from real variations in business rules. Removing the noise with .str and regex methods ensures that any subsequent analysis remains sound. Numeric cleanup ensures arithmetic calculations on numbers are processed correctly.

Inconsistency Standardization Justification

`The PaycheckMethod` and `JobRoleArea` categorical columns must align with the definitions in the data dictionary to enforce business rules. Categories under these columns (`Mailed Check`, `Mail Check`) have the same meaning, so mapping them to unified values ensures total counts remain accurate. The standardization ensures that the data insights accurately address the business problem questions and that there are no distortions in those answers caused by errors resulting from manual entry.

Replacing the `AnnualSalary` values with correctly calculated amounts follows the explicit business rules (`HourlyRate` x 40 x 52).

Converting `EmployeeNumber` to a string dtype aligned with the business rule. IDs represent the primary key and need to be unique, non-modifiable through arithmetic operations, and resistant to formatting changes. The standardization of EmployeeNumber reduces risk of errors, enables joins, and improves clarity.

Data Dictionary Alignment Justification

Aligning data cleaning efforts with the business purpose of each variable and enforcing allowed values ensures accurate and actionable data insights, while also protecting the data for future use.

Outliers Justification

Negative driving distance values are illogical. These are likely data-entry or data transformation errors. The code then imputed the median of 49 miles to the values that were NaN to ensure consistency and avoid distorting the distributions. The code did not remove high distances above the 99th percentile, because such values could correspond to remote employees whose home ZIP codes are far from their assigned office. Distances within a reasonable range are more realistic. Data flags for out-of-bound values add a corrective measure. Rather than deleting them, the flags add a layer of trustworthiness (Koneshloo, 2025). The workflow created a DrivingComuterDistance flag to identify extreme cases, distinguishing realistic outliers from illogical values. Since the data is right-skewed (Q1= 13, median= 42, Q3= 71, max= 950), imputing the median made more sense. Imputing the mean value would distort and overstate typical commutes, while the median remains stable without causing distortions in overall distributions.

Summary of Justification: Business Problem Alignment

The business goal for this data analysis project is to predict and prevent an increase in turnover. The goal aligns with reducing costs associated with training new employees rather than retaining existing ones. Inconsistencies in categorical variables can distort variable distributions (Hellerstein, 2008). Inconsistent numeric variables can inflate or deflate ratios. The key metrics for assessing the relationship between employee turnover and pay include department, role, and compensation type. Accurately storing the data for those variables is essential for addressing the business problem and necessary to understand workload differences and pay deviations.

C3. ADVANTAGES OF THE DATA CLEANING APPROACH

Preserved Data Integrity and Meaning

The data cleaning methods in this project focused less on deleting data and more on preserving it as closely as possible in its original form, focusing on solving the business problem. For example, the code imputed `NaN` values to missing data in commuter distances. Using flags to indicate missing values (e.g., `DevHoursMissingFlag`, `DrivingCommuterDistanceFlag`) preserved the transparency of what data had changed during the cleaning process (Alation, 2025). Imputing '`NaN`' instead of zeros in `AnnualProfessionalDevHrs` enabled analytical flexibility by keeping the original meaning. Missing values for professional development hours could, among other reasons, either indicate that an employee did not attend any training or that no one ever asked them to provide the information.

The true cause of missingness can have two disparate business signals. One implies a lack of employee engagement, and the other a process gap. The consequences that follow in organizations without analyzing the true reasons for missing data can be worlds apart. Incorrect assumptions about missing data can lead to operational and strategic mistakes. For example, a company may erroneously invest in skill gaps while overlooking data gathering process gaps. These errors can lead to lost business, wasted resources, flawed workforce planning, and lost business opportunities. By maintaining data flags, instead of imputing zeros, aggregates stay the same, and the Key Performance Indicators remain accurate. This approach improves the quality and reliability of insights.

Enhanced Reliability and Consistency

Data Reliability and Consistency are essential for accurate analytical results, stable modeling, and visualizations. Deduplication logic checks the derived `AnnualSalary` column (correcting miscalculated pay), replacing illogical values in commuting distance, ensuring accurate counts, and increasing the reliability of data in this project.

Standardizing labels in categorical columns and providing a single standard label per category increases consistency (e.g., `HumanResources` to `Human  Resources`). The methodology also ensures consistency through deduplication by maintaining entity integrity (one row per Employee) and utilizing a single imputation rule for commuting distance values (median).

C4. DISADVANTAGES OF THE DATA CLEANING APPROACH

Reducing Natural Variations in Data by Imputing Medians

Methods that replace a missing value in data with a central value, such as the median, pull the data towards the center, reducing the inherent variability in the data. Imputing missing values with a median can lead to misleading conclusions because it reduces the natural variations. The median imputation approach obscures the actual differences between employees who live close to the office and those who live far away. The median imputation limits precision in models that rely on the distance measure (Zhao, 2020).

Potential Oversimplification from Categorical Standardization

Standardizing categorical values by merging similar labels into one assumes that all variations are the same. While `Mail Check` and `Mailed Check` appear to have the same meaning, differences in labels may sometimes represent distinct processes (Guo et al., 2023). Data deletion, canonicalization, modifications, and imputations require analysts to apply discernment and common sense to data-driven decisions. It is always a good idea to confirm information with stakeholders, especially if a given label can significantly alter the outcome of the data analysis. (Zhao, 2020). Unifying similar and seemingly duplicate labels carries the risk of oversimplifying things. It may obscure nuanced business meanings.

D1. DATA CLEANING REPORT

This document is the Data Cleaning Report that satisfies the requirement of section D1.

D2. PYTHON CODE FILE

The submission includes the Python code file.

D3. CLEANED DATASET

I have included the cleaned dataset as an attachment in the D599 Task 1 submission.

D4. LINK TO THE VIDEO PRESENTATION

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=16ff609a-ddb9-4c18-828e-b37a00154d51

References

Zhao, Y., & Udell, M. (2020, June 16). Missing value imputation for mixed data via Gaussian copula. arXiv.org. https://arxiv.org/abs/1910.12845

Guo, M., Wang, Y., Yang, Q., Li, R., Zhao, Y., Li, C., Zhu, M., Cui, Y., Jiang, X., Sheng, S., Li, Q., & Gao, R. (2023, September 21). Normal workflow and key strategies for data cleaning toward real-world data: Viewpoint. *Interactive journal of medical research*. https://pmc.ncbi.nlm.nih.gov/articles/PMC10557005/

Alation. Data cleansing for AI SUCCESS: Best Practices and Implementation Guide. Data Cleansing for AI Success: Best Practices and Implementation Guide. (2025, April 15). https://www.alation.com/blog/data-cleansing-ai-best-practices-guide/

Hellerstein, J. Quantitative data cleaning for large databases. (n.d.). https://dsf.berkeley.edu/jmh/papers/cleaning-unece.pdf

Koneshloo, A., & Du, D. (2018). Coronary Heart Disease Diagnosis Using Kernel PCA and Adaptive Boosting. IISE Annual Conference. Proceedings, (), 1157–1162.Cabral, E. F., Vinces, B. V. S., Silva, G. D. F., Sander, J., & Cordeiro, R. L. F. (2025, March 7). *Efficient outlier detection in numerical and categorical data - data mining and knowledge discovery*. SpringerLink. https://link.springer.com/article/10.1007/s10618-024-01084-1

MacDonald, L. (2025b, August 16). *The Complete Guide to Data Uniqueness*. Monte Carlo Data. https://www.montecarlodata.com/blog-data-uniqueness/

Pennsylvania State University. (n.d.). *Identifying Outliers: IQR Method*. Identifying outliers: IQR method. https://online.stat.psu.edu/stat200/book/export/html/63