# APA - Data Preparation Template

## DATA 3300

## Name(s):

**Read and follow these assignment instructions carefully! Ordinarily, we'd jump into cleansing data as needed and we'd each do it slightly differently. This is difficult to grade, so please go in the order of this document and follow these instructions.**

**For this assignment students should submit one .ipynb file, one clean, sorted Excel file, and a PDF of this iPython notebook file. See the assignment background for attribute descriptions and the assignment requirements link on the Canvas assignment page.**

**Students should alter code in lines with "replace code..." language. Otherwise, students do not need to alter the provided code.**

## Q1

**What does the term "data quality" refer to and why is it important within the context of using data to solve business problems?**

Data quality refers to how accurate and "clean" data is. It's important because with unlcean data, misrepresantation is present. Solving business problems with poor data will lead to innacurate results.

## Q2

**The lecture and textbook discussed five characteristics of data quality: accuracy, completeness, consistency, timeliness, and uniqueness. While "accuracy" in this dataset is hard for us to gauge without further inquiry and"timeliness" kind of depends on what it is the data are to be used for, there are clear examples of problems with each of the other three characteristics.**

**For each of the other three (completeness, consistency, uniqueness), identify a specific situation within the *hbdata-orig.csv* dataset (identify record numbers when applicable). Hint: For uniqueness, look at the VisitSpan attribute. Be sure your answers are clearly labelled and described.**

**This can be done by viewing the data in Excel or importing it into Python. If using Python, begin by importing your libraries and then read in the .csv file.**

```
import pandas as pd
import numpy as np

#just run this code block after importing libraries above, no need to alter the below code!
from datetime import datetime #you'll also need this method to convert a variable to date-time later in the assign
import warnings
warnings.filterwarnings("ignore") #this can be removed but will help the code be a little cleaner by ignoring warn


#replace with code for importing the dataset
df = pd.read_excel('/content/hbdata-orig24.xlsx')
#run this code block after importing dataset above
pd.set_option("display.max_rows", None, "display.max_columns", None) #View full dataframe
df
```

- Completeness: VisitIDs 2209, 2395, missing data

- Consistency: VisitIDs 2250-2257, medium instead of M. Lots of oz instead of ml as well
- Uniqueness: 2235-2336: Time is the same, duplicate entry

## Q3

**Answer the following questions regarding the importance of each of the named data scrubbing steps.**

### 3A

**What negative outcome would likely occur during data analysis if duplicated visits were not removed (consolidated)?**

The dulicate data would be recorded multiple times, scewing the results of a particular bird or feeder

### 3B

**What negative outcome would likely occur during data analysis if Location entries were not standardized?**

Some locations would have missing entries, for example "Deck. 2" would not be counted in "Deck 2"

### 3C

**What negative outcome would likely occur during data analysis if, due to measurement issues, amount eaten values were not consistent?**

One might assume a bird has eaten much more or much less than it actually has. This can cause future predictions of how much to feed to be wrong.

## Q4

**In scrubbing the data, you should address each of the following. It's recommended to go through this assignment in the order of steps below.**

### 4A

In Python trim all text columns to ensure there are no leading or trailing spaces on any of the data values. If you still see two seemingly identical entries, remember that trimming does not trim internal spaces between two words.

```
df_obj = df.select_dtypes(['object']) #select all columns containing text (of type object)
df[df_obj.columns] = df_obj.apply(lambda x: x.str.strip()) #trim all selected columns
#replace with code to view the header of the dataframe (cheat sheet!)
df.head()
#keep below code to verify date/time of your work
from datetime import datetime
now = datetime.now()
formatted_time = now.strftime("%Y-%m-%d %H:%M:%S")
print(f"Current Date and Time: {formatted_time}")
```

```
Current Date and Time: 2024-01-18 21:31:09
```

### 4B

Ensure each bird's first and last name should be included in separate fields (columns).

```
#replace with code to split column, updating column names and delimiter type (cheat sheet)
df[['BirdFirstName','BirdLastName']] = df['BirdName'].str.split(' ', expand=True)
#replace with code to drop birdname column
df = df.drop('BirdName', axis = 1)
#replace with code to show df heading
df.head()
```

| | VisitID | BirdType | BirdSize | Gender | VisitDate | VisitSpan | Location | Hoverfeed |
|---|---|---|---|---|---|---|---|---|
| **0** | 2203 | Rufus | S | female | NaN | 23:13:05.736-23:13:21.109 | Back Patio | yes |
| **1** | 2204 | Calliope | M | male | NaN | 07:45:07.65-07:45:17.97 | NE Corner | yes |
| **2** | 2205 | Rufous | M | female | NaN | 08:09:52.10-08:10:22.74 | Deck 2 | no |
| **3** | 2206 | Rufous | M | male | NaN | 08:41:54.07-08:42:07.48 | Deck 1 | yes |
| **4** | 2207 | Anna's | M | female | NaN | 09:18:59.33-09:19:13.41 | Garage | yes |

## 4C

Ensure that data values are standardized in each column (pay particular attention to the textual columns, BirdName, BirdType, BirdSize, Gender, Location, and Hoverfeed) so that each “idea” (e.g., a species of bird) is specified (i.e., spelled) the same way.

This can be checked for each column by using the .value_counts() method! After applying the standarization, show that the correction has been made by using the .value_counts() method to display all categories for each column of interest.

```
#Replace with code to examine levels in BirdType using the value_counts() method, update column name
df['BirdType'].value_counts()
```

```
Rufous      103
Anna's       85
Calliope     61
Rufus        18
Prefect      14
Name: BirdType, dtype: int64
```

```
#Replace with code to replace redundant value in BirdType, for only 1 original and new label
df = df.replace('Rufus','Rufous')
```

```
#Replace with code to examine levels of BirdSize. Ignore null, "medium" and 20ml lines; these will be fixed later.
df['BirdSize'].value_counts()
```

```
M         162
S          48
L          33
XL         23
medium     13
20 ml       5
Name: BirdSize, dtype: int64
```

```
#Replace with code to replace redundant value in BirdSize
df = df.replace('medium','M')
```

```
#Replace with code to examine levels of Location. Ignore two lines with all numerical output; these will be fixed
df['Location'].value_counts()
```

```
Back Patio                76
Deck 2                    61
Deck 1                    46
```

```
NE Corner                  44
Garage                     29
Entry                      29
15:16:40.19-15:16:55.98     4
13:14:33.36-13:15:09.25     2
Name: Location, dtype: int64
```

```
#Replace with code to replace redundant values of Location, for 2 original and new labels
df = df.replace({'Back  Patio':'Back Patio','Deck. 2':'Deck 2'})
```

## 4D

The amount eaten is very important to our analysis. Where this value is null (blank), omit the record. Note that values of 0 are not the same as null. Retain (don't delete) the records where the value of Amount is 0. Other issues with this attribute will be addressed later.

Before removing missing columns first examine the length of the dataset currently using the len() method. Then split the *Amount* column into two columns, *Amount_Eat* and *Unit*. Finally, change *Amount_Eat* to a float object using pd.to_numeric().

Once missing values are dropped, view the length of the dataset again using the len() method to see how many observations were dropped.

```
len(df) #return length of dataframe (i.e., number of rows), just run this code block
```

```
291
```

```
#Replace with code to create new split columns for 'Amount_Eat' and 'Unit' from the 'Amount' column, updating colu
df[['Amount_Eat','Unit']] = df['Amount'].str.split(' ', expand=True)
#Replace with code to drop 'Amount' column
df = df.drop('Amount', axis = 1)
#Replace with code to show header of dataframe
df.head()
```

```
df['Amount_Eat'] = df['Amount_Eat'].apply(pd.to_numeric, errors='coerce') #convert Amount_Eat column variable dtyp
#just run this code block
df
```

```
#Replace with code to remove missing data for the 'Amount_Eat' column
df = df[df['Amount_Eat'].notna()] # drop rows with missing data based on a specified column
len(df) #check the new length of the dataset
```

```
283
```

## 4E

Ensure that each visit is unique. In other words, no visit should be recorded more than once (duplicated). Remove duplicates from the VisitSpan column using the drop_duplicates() method.

Then use the len() method to show how many observations were dropped.

```
df = df.drop_duplicates(subset=['VisitSpan']) #drop duplicate records from the 'VisitSpan' column

#Replace with code to check the new length of the dataset (see above)
len(df)
```

```
279
```

## 4F

Split the VisitSpan column into two columns: rename them StartTime and EndTime.

Then ensure that both time columns are set as datetime objects by using the pd.to_datetime() method.

```
#Replace with code to split 'VisitSpan' based on the '-' delimiter, create a StartTime and EndTime column
df[['StartTime', 'EndTime']] = df['VisitSpan'].str.split('-', expand = True) # split based on _ delimeter
df['StartTime'] = pd.to_datetime(df['StartTime'], format='%H:%M:%S.%f') #converts StartTime to hours:minutes:secon
df['EndTime'] = pd.to_datetime(df['EndTime'], format='%H:%M:%S.%f') #converts EndTime to hours:minutes:seconds for

#Replace with code to drop the 'VisitSpan' column
df = df.drop('VisitSpan', axis = 1)
#Replace with code to show header of dataframte
df.head()
```

## 4G

Some recordings of Amount appear to have been made in ounces rather than in milliliters. All of them should be in milliliters. Convert those in ounces to milliliters (1 oz. = 29.57 ml).

Start by subsetting the data to only those observations containing oz, then apply the conversion to the *Amount_Eat* column and replace the values in the *Unit* column with ml.

```
df_oz = df[df['Unit'] == 'oz'] #subsets dataset to only contain oz
df_oz['Amount_Eat'] = 29.57*df['Amount_Eat'] #applies conversion formula
df_oz['Unit'] = df_oz['Unit'].replace('oz', 'ml') #replaces oz with ml
df_oz.head() #just run this entire code block as is
```

## 4H

**Next, you'll combine your original dataframe with the new subset dataframe using the concat() method. Finally, drop rows containing 'oz' in the *Unit* column.**

**Make sure you have the correct number of observations still by using the len() method.**

```
frames = [df, df_oz]
df = pd.concat(frames) #combines the original df with the new subsetted df
df = df[df.Unit != 'oz'] #removes rows containing oz

#Replace with code to examine length of dataframe (see earlier portions of this assignment)
len(df)
```

```
279
```

## 4I

Createa new column, called “IsFemale”. This column should include a 1 if the hummingbird that made the visit is female, or 0 if the hummingbird is male.

```
#Replace with code to create new column called IsFemale using the np.where method
df['IsFemale'] = np.where(df['Gender'] == 'female',1,0)
df.head()
```

## 4J

Create a new column called Duration, which includes the duration of the visit in seconds. Creat this column from the *StartTime* and *EndTime* columns then convert to seconds using dt.total_seconds().

```
df['Duration'] = df['EndTime'] - df['StartTime'] #creates new 'Duration' column by subtracting StartTime from EndT
```

```
df['Duration'] = df['Duration'].dt.total_seconds() #converts 'Duration' to seconds
df.head()
```

## 4K

View the full dataset, to spot check your cleaned data and address any remaining issues you notice.

```
pd.set_option("display.max_rows", None, "display.max_columns", None) #views full dataframe
df
```

```
#Replace with code to remove any remaining unwanted columns
df = df.drop('Unnamed: 10', axis = 1)
df.head()
```

```
pd.set_option("display.max_rows", None, "display.max_columns", None) #views full dataframe
df
```

## Q5

When you're finished cleaning the datafile, sort the file by Visit ID (low to high), then save the cleaned file as an Excel document for submission via Canvas.

```
#Replace with code to set index to VisitID
df = df.set_index('VisitID')
df.sort_index() #sorts index
```

```
#export your cleaned Excel file for submission, along with your notebook

#For Google Colab
from google.colab import files
df.to_excel("cleaned_data_HB.xlsx")
files.download("cleaned_data_HB.xlsx")

#For JupyterNotebooks
df.to_excel("/content/hbdata-orig24.xlsx")

#keep below code to verify date/time of your work
from datetime import datetime
now = datetime.now()
formatted_time = now.strftime("%Y-%m-%d %H:%M:%S")
print(f"Current Date and Time: {formatted_time}")
```

```
Current Date and Time: 2024-01-18 22:00:32
```

For submission to Canvas: Download/Save your .ipynb file, print your .ipynb file to PDF, and export your cleaned dataset as an Excel file.