⇨ Part One: Data Pre-Processing and Feature Engineering

```
%%capture
!pip install numpy
```

```
%%capture
!pip install pandas
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
from zipfile import ZipFile
from urllib.request import urlretrieve
```

```
files_zipped = 'https://www.evanmarie.com/content/files/puppy_original_csvs.zip'
urlretrieve(files_zipped, 'original_csvs.zip')
with ZipFile('original_csvs.zip') as zipped_file:
    zipped_file.extractall(path='original_csvs')
```

⇨ Importing the CSV files and preprocessing each to make them combinable.

Most of the explanation of the steps in this section are noted in comments above each set off code.

```
# -------------------------CONVERT CSVs TO DATAFRAMES -----------------------------#

active_burn_df = pd.read_csv('original_csvs/active_burn.csv')
distance_walk_run_df = pd.read_csv('original_csvs/distance_walk_run.csv')
environmental_audio_df = pd.read_csv('original_csvs/environmental_audio.csv')
exercise_time_df = pd.read_csv('original_csvs/exercise_time.csv')
flights_climbed_df = pd.read_csv('original_csvs/flights_climbed.csv')
heart_rate_df = pd.read_csv('original_csvs/heart_rate.csv')
high_hr_event_df = pd.read_csv('original_csvs/high_hr_event.csv')
hr_variability_df = pd.read_csv('original_csvs/hr_variability.csv')
oxygen_saturation_df = pd.read_csv('original_csvs/oxygen_saturation.csv')
respiratory_rate_df = pd.read_csv('original_csvs/respiratory_rate.csv')
resting_burn_df = pd.read_csv('original_csvs/resting_burn.csv')
stair_ascent_df = pd.read_csv('original_csvs/stair_ascent.csv')
stair_descent_df = pd.read_csv('original_csvs/stair_descent.csv')
stand_hour_df = pd.read_csv('original_csvs/stand_hour.csv')
stand_time_df = pd.read_csv('original_csvs/stand_time.csv')
step_count_df = pd.read_csv('original_csvs/step_count.csv')
```

```
step_length_df = pd.read_csv('original_csvs/step_length.csv')
walk_speed_df = pd.read_csv('original_csvs/walk_speed.csv')
walk_steadiness_df = pd.read_csv('original_csvs/walk_steadiness.csv')
walking_double_support_df = pd.read_csv('original_csvs/walking_double_support.csv')
walking_hr_df = pd.read_csv('original_csvs/walking_hr.csv')
```

```
dataframe_list = [active_burn_df, distance_walk_run_df, environmental_audio_df, exercis
                  flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                  oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                  stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                  walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
# Function converting the times in the dataframes from UTC time to Eastern US time.

def convert_utc_to_eastern(df_list):
    for df in df_list:
        df['start'] = pd.to_datetime(df['start'], utc=True).dt.tz_convert('US/Eastern')
        df['end'] = pd.to_datetime(df['end'], utc=True).dt.tz_convert('US/Eastern')
    return df_list
```

```
dataframe_list = convert_utc_to_eastern(dataframe_list)
```

```
# Function that takes a dataframe and a date column as input
# and adds a year, month, and day column to the dataframe

def add_time_columns(df_list):
    for df in df_list:
        df['year'] = pd.DatetimeIndex(df['start']).year
        df['month'] = pd.DatetimeIndex(df['start']).month
        df['day'] = pd.DatetimeIndex(df['start']).day
    return df_list
```

```
dataframe_list = add_time_columns(dataframe_list)
```

```
# Function that takes a dataframe, its start column, and its end column as input
# and create a duration column for the dataframe

def add_duration_column(df_list):
  for df in df_list:
    df['duration_sec'] = pd.to_datetime(df['end']) - pd.to_datetime(df['start'])
  return df_list
```

```
dataframe_list = add_duration_column(dataframe_list)
```

```
# Function to convert duration column to seconds and rename the column
```

```
def convert_duration_to_seconds(df, duration_col, new_col_name):
    df[new_col_name] = df[duration_col].dt.total_seconds()
    df = df.drop(columns=duration_col)
    return df
```

```
#-------------Convert duration columns to seconds-------------------------#

active_burn_df = convert_duration_to_seconds(active_burn_df, 'duration_sec', 'active_bu
distance_walk_run_df = convert_duration_to_seconds(distance_walk_run_df, 'duration_sec'
environmental_audio_df = convert_duration_to_seconds(environmental_audio_df, 'duration_
exercise_time_df = convert_duration_to_seconds(exercise_time_df, 'duration_sec', 'exerc
flights_climbed_df = convert_duration_to_seconds(flights_climbed_df, 'duration_sec', 'f
heart_rate_df = convert_duration_to_seconds(heart_rate_df, 'duration_sec', 'heart_rate_
high_hr_event_df = convert_duration_to_seconds(high_hr_event_df, 'duration_sec', 'high_
hr_variability_df = convert_duration_to_seconds(hr_variability_df, 'duration_sec', 'hr_
oxygen_saturation_df = convert_duration_to_seconds(oxygen_saturation_df, 'duration_sec'
respiratory_rate_df = convert_duration_to_seconds(respiratory_rate_df, 'duration_sec',
resting_burn_df = convert_duration_to_seconds(resting_burn_df, 'duration_sec', 'resting
stair_ascent_df = convert_duration_to_seconds(stair_ascent_df, 'duration_sec', 'stair_a
stair_descent_df = convert_duration_to_seconds(stair_descent_df, 'duration_sec', 'stair
stand_hour_df = convert_duration_to_seconds(stand_hour_df, 'duration_sec', 'stand_hour_
stand_time_df = convert_duration_to_seconds(stand_time_df, 'duration_sec', 'stand_time_
step_count_df = convert_duration_to_seconds(step_count_df, 'duration_sec', 'step_count_
step_length_df = convert_duration_to_seconds(step_length_df, 'duration_sec', 'step_leng
walk_speed_df = convert_duration_to_seconds(walk_speed_df, 'duration_sec', 'walk_speed_
walk_steadiness_df = convert_duration_to_seconds(walk_steadiness_df, 'duration_sec', 'w
walking_double_support_df = convert_duration_to_seconds(walking_double_support_df, 'dur
walking_hr_df = convert_duration_to_seconds(walking_hr_df, 'duration_sec', 'walking_hr_
```

```
# Function that takes dataframe_list as input and returns a list of dataframes
# with a new "date" column from the year, month, and day columns

def create_date_column(df_list):
    for df in df_list:
        df['date'] = pd.to_datetime(df[['year', 'month', 'day']])
    return df_list

[active_burn_df, distance_walk_run_df, environmental_audio_df, exercise_time_df,
                  flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                  oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                  stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                  walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
os.mkdir('pre_reduction_csvs')
```

```
# ------------------------DOWNLOAD the DATAFRAMES so far as CSV files ---------------

active_burn_df.to_csv('pre_reduction_csvs/active_burn_df_before_reduction.csv')
distance_walk_run_df.to_csv('pre_reduction_csvs/distance_walk_run_df_before_reduction.c
```

```
environmental_audio_df.to_csv('pre_reduction_csvs/environmental_audio_df_before_reducti
exercise_time_df.to_csv('pre_reduction_csvs/exercise_time_df_before_reduction.csv')
flights_climbed_df.to_csv('pre_reduction_csvs/flights_climbed_df_before_reduction.csv')
heart_rate_df.to_csv('pre_reduction_csvs/heart_rate_df_before_reduction.csv')
high_hr_event_df.to_csv('pre_reduction_csvs/high_hr_event_df_before_reduction.csv')
hr_variability_df.to_csv('pre_reduction_csvs/hr_variability_df_before_reduction.csv')
oxygen_saturation_df.to_csv('pre_reduction_csvs/oxygen_saturation_df_before_reduction.c
respiratory_rate_df.to_csv('pre_reduction_csvs/respiratory_rate_df_before_reduction.csv
resting_burn_df.to_csv('pre_reduction_csvs/resting_burn_df_before_reduction.csv')
stair_ascent_df.to_csv('pre_reduction_csvs/stair_ascent_df_before_reduction.csv')
stair_descent_df.to_csv('pre_reduction_csvs/stair_descent_df_before_reduction.csv')
stand_hour_df.to_csv('pre_reduction_csvs/stand_hour_df_before_reduction.csv')
stand_time_df.to_csv('pre_reduction_csvs/stand_time_df_before_reduction.csv')
step_count_df.to_csv('pre_reduction_csvs/step_count_df_before_reduction.csv')
step_length_df.to_csv('pre_reduction_csvs/step_length_df_before_reduction.csv')
walk_speed_df.to_csv('pre_reduction_csvs/walk_speed_df_before_reduction.csv')
walk_steadiness_df.to_csv('pre_reduction_csvs/walk_steadiness_df_before_reduction.csv')
walking_double_support_df.to_csv('pre_reduction_csvs/walking_double_support_df_before_r
walking_hr_df.to_csv('pre_reduction_csvs/walking_hr_df_before_reduction.csv')
```

⇨ Averages and Sums Columns

Adding an average or sum column for all important input columns.

```
# Function to add a daytime category column that takes the dataframe, the time column
# to calculate category from, and the new column name
# 1 = middle of the night, 2 = morning, 3 = afternoon, 4 = evening

def add_daytime_category(df_list):
  for df in df_list:
      df['daytime_category'] = df['start'].apply(lambda x: 1 if x.hour < 6 else 2 if x.
                                  else 3 if x.hour < 18 else 4)
  return df_list
```

```
[active_burn_df, distance_walk_run_df, environmental_audio_df, exercise_time_df,
                  flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                  oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                  stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                  walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
#-------------------------AVERAGE DAYTIME CATEGORIES FOR DAYS-------------------------
# Function to add a new column that has the average for a column for each daytime categ

def add_average_column(df, column, new_column_name):
    df[new_column_name] = df.groupby(['date', 'daytime_category'])[column].transform('m
    return df

#-------------------------SUM DAYTIME CATEGORIES FOR DAYS-------------------------#
# Function to add a new column that has the sum for a column for each daytime category
```

```
def add_sum_column(df, column, new_column_name):
    df[new_column_name] = df.groupby(['date', 'daytime_category'])[column].transform('s
    return df
```

```
# ----------------------------SUMMING COLUMNS ----------------------------#
cols_to_sum = [
    (active_burn_df, 'active_cal_burned'),
    (active_burn_df, 'active_burn_duration_sec'),
    (distance_walk_run_df, 'distance_walk_miles'),
    (distance_walk_run_df, 'distance_walk_duration_sec'),
    (environmental_audio_df, 'environmental_audio_duration_sec'),
    (exercise_time_df, 'exercise_min'),
    (exercise_time_df, 'exercise_time_duration_sec'),
    (flights_climbed_df, 'flights_climbed_count'),
    (flights_climbed_df, 'flights_climbed_duration_sec'),
    (high_hr_event_df, 'high_hr_event_duration_sec'),
    (walking_hr_df, 'walking_hr_bpm'),
    (walking_hr_df, 'walking_hr_duration_sec'),
    (resting_burn_df, 'resting_burn_cal'),
    (resting_burn_df, 'resting_burn_duration_sec'),
    (stair_ascent_df, 'stair_ascent_ft_sec'),
    (stair_descent_df, 'stair_descent_ft_sec'),
    (stand_hour_df, 'stand_hour'),
    (stand_hour_df, 'stand_hour_duration_sec'),
    (stand_time_df, 'stand_time_min'),
    (stand_time_df, 'stand_time_duration_sec'),
    (step_count_df, 'step_count'),
    (step_count_df, 'step_count_duration_sec')
]

# ----------------------------AVERAGING COLUMNS ----------------------------#
cols_to_avg = [
    (environmental_audio_df, 'env_audio_db'),
    (heart_rate_df, 'hr_bpm'),
    (heart_rate_df, 'heart_rate_duration_sec'),
    (hr_variability_df, 'hr_variability_ms'),
    (hr_variability_df, 'hr_variability_duration_sec'),
    (oxygen_saturation_df, 'o2_sat_percentage'),
    (oxygen_saturation_df, 'oxygen_saturation_duration_sec'),
    (oxygen_saturation_df, 'o2_sat_barametric_kpa'),
    (respiratory_rate_df, 'respiratory_count_per_min'),
    (respiratory_rate_df, 'respiratory_rate_duration_sec'),
    (stair_ascent_df, 'stair_ascent_duration_sec'),
    (stair_descent_df, 'stair_descent_duration_sec'),
    (step_length_df, 'step_length_inches'),
    (step_length_df, 'step_length_duration_sec'),
    (walk_speed_df, 'walk_speed_mph'),
    (walk_speed_df, 'walk_speed_duration_sec'),
    (walk_steadiness_df, 'walk_steadiness_percentage'),
    (walk_steadiness_df, 'walk_steadiness_duration_sec'),
    (walking_double_support_df, 'walking_double_support_percentage'),
```

```
    (walking_double_support_df, 'walking_double_support_duration_sec')
]
```

```
# ----------------------------APPLY AVG and SUM FUNCTIONS ----------------------------
# Functions to apply the add_average_column and the add_sum_column functions to each co
# in the list of columns to average and sum

def apply_add_average_column(column_list):
    for col in column_list:
        add_average_column(col[0], col[1], str(col[1]) + '_avg')

def apply_add_sum_column(column_list):
    for col in column_list:
        add_sum_column(col[0], col[1], str(col[1]) + '_avg')
```

```
# ----------------------APPLY AVG & SUM FUNCTIONS to LISTS of COLS -------------------
apply_add_average_column(cols_to_avg)
apply_add_sum_column(cols_to_sum)
```

```
# -------------------------DROP FIRST and SECOND COLUMNS ----------------------------
# Function to delete the "start" and "end" columns from each dataframe

def drop_useless_columns(df_list):
    for df in df_list:
        df.drop(columns=['start', 'end', 'duration_sec'], inplace=True)
    return df_list

[active_burn_df, distance_walk_run_df, environmental_audio_df, exercise_time_df,
                    flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                    oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                    stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                    walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
# -------------------------DROP COLUMNS that have been SUMMED and AVERAGED------------

active_burn_df.drop(columns=['active_cal_burned', 'active_burn_duration_sec'], inplace=
distance_walk_run_df.drop(columns=['distance_walk_miles', 'distance_walk_duration_sec']
environmental_audio_df.drop(columns=['env_audio_db'], inplace=True)
exercise_time_df.drop(columns=['exercise_min', 'exercise_time_duration_sec'], inplace=T
flights_climbed_df.drop(columns=['flights_climbed_count', 'flights_climbed_duration_sec
heart_rate_df.drop(columns=['hr_bpm', 'heart_rate_duration_sec'], inplace=True)
high_hr_event_df.drop(columns=['high_hr_event', 'high_hr_event_duration_sec'], inplace=
hr_variability_df.drop(columns=['hr_variability_ms', 'hr_variability_duration_sec'], in
oxygen_saturation_df.drop(columns=['o2_sat_percentage', 'oxygen_saturation_duration_sec
respiratory_rate_df.drop(columns=['respiratory_count_per_min', 'respiratory_rate_durati
resting_burn_df.drop(columns=['resting_burn_cal', 'resting_burn_duration_sec'], inplace
stair_ascent_df.drop(columns=['stair_ascent_ft_sec', 'stair_ascent_duration_sec' ], inp
stair_descent_df.drop(columns=[ 'stair_descent_ft_sec',   'stair_descent_duration_sec'
stand_hour_df.drop(columns=['stand_hour', 'stand_hour_duration_sec'], inplace=True)
```

```
stand_time_df.drop(columns=['stand_time_min', 'stand_time_duration_sec'], inplace=True)
step_count_df.drop(columns=['step_count', 'step_count_duration_sec'], inplace=True)
step_length_df.drop(columns=['step_length_inches', 'step_length_duration_sec'], inplace
walk_speed_df.drop(columns=['walk_speed_mph', 'walk_speed_duration_sec'], inplace=True)
walk_steadiness_df.drop(columns=['walk_steadiness_percentage', 'walk_steadiness_duratic
walking_double_support_df.drop(columns=['walking_double_support_percentage', 'walking_d
walking_hr_df.drop(columns=['walking_hr_bpm', 'walking_hr_duration_sec'], inplace=True)
```

```
# -------------------------PRINT COLUMNS FROM ALL DATAFRAMES ------------------------
# Print dataframe info before merging to make sure they are as expected

# def print_info(df_list):
#     for df in df_list:
#         print(df.info())
#         print('')

# print_info(dataframe_list)
```

```
# And somehow this is the only one that got left in the dataframes
environmental_audio_df.drop(columns=['environmental_audio_duration_sec'], inplace=True)
```

⇨ Remove duplicate rows and let's MERGE! FINALLY! No more 21 dataframes!!!

```
active_burn_df.shape
```

```
(347622, 7)
```

```
def drop_duplicate_rows(df_list):
    for df in df_list:
        df.drop_duplicates(inplace=True)
    return df_list
```

```
[active_burn_df, distance_walk_run_df, environmental_audio_df, exercise_time_df,
                  flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                  oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                  stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                  walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
#------------------------- REMOVE YEAR, MONTH, AND DAY COLUMNS from BABY DFs ---------
# Function to remove the year, month, and day columns from each dataframe and add datet

def remove_year_month_day(df_list):
    for df in df_list:
        df.drop(columns=['year', 'month', 'day'], inplace=True)
    return df_list
```

```
baby_dfs = [distance_walk_run_df, environmental_audio_df, exercise_time_df,
                    flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                    oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                    stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                    walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
[distance_walk_run_df, environmental_audio_df, exercise_time_df,
                    flights_climbed_df, heart_rate_df, high_hr_event_df, hr_variability
                    oxygen_saturation_df, respiratory_rate_df, resting_burn_df, stair_a
                    stair_descent_df, stand_hour_df, stand_time_df, step_count_df, step
                    walk_speed_df, walk_steadiness_df, walking_double_support_df, walki
```

```
# -------------------------MERGING IS NIGH!----------------------------#
# Function to add a smaller dataframe to a larger one and not remove any rows from the
# on date and daytime_category

def merge_dataframes(df1, df2):
    df1 = df1.merge(df2, how='left', on=['date', 'daytime_category'])
    return df1
```

```
active_burn_df.shape
```

```
(2004, 7)
```

```
masterframe_df = merge_dataframes(active_burn_df, distance_walk_run_df)
masterframe_df = merge_dataframes(masterframe_df, environmental_audio_df)
masterframe_df = merge_dataframes(masterframe_df, exercise_time_df)
masterframe_df = merge_dataframes(masterframe_df, flights_climbed_df)
masterframe_df = merge_dataframes(masterframe_df, heart_rate_df)
masterframe_df = merge_dataframes(masterframe_df, high_hr_event_df)
masterframe_df = merge_dataframes(masterframe_df, hr_variability_df)
masterframe_df = merge_dataframes(masterframe_df, oxygen_saturation_df)
masterframe_df = merge_dataframes(masterframe_df, respiratory_rate_df)
masterframe_df = merge_dataframes(masterframe_df, resting_burn_df)
masterframe_df = merge_dataframes(masterframe_df, stair_ascent_df)
masterframe_df = merge_dataframes(masterframe_df, stair_descent_df)
masterframe_df = merge_dataframes(masterframe_df, stand_hour_df)
masterframe_df = merge_dataframes(masterframe_df, stand_time_df)
masterframe_df = merge_dataframes(masterframe_df, step_count_df)
masterframe_df = merge_dataframes(masterframe_df, step_length_df)
masterframe_df = merge_dataframes(masterframe_df, walk_speed_df)
masterframe_df = merge_dataframes(masterframe_df, walk_steadiness_df)
masterframe_df = merge_dataframes(masterframe_df, walking_double_support_df)
masterframe_df = merge_dataframes(masterframe_df, walking_hr_df)
```

```
# -------------------------Drop Useless Columns----------------------------#
# Pandas Profiling showed these as being completely useless.
```

```
masterframe_df.drop(columns=['stand_hour_avg', 'walk_steadiness_duration_sec_avg',
                             'walk_steadiness_percentage_avg', 'respiratory_count_per_m
                             'respiratory_rate_duration_sec_avg', 'high_hr_event_durati
                             'heart_rate_duration_sec_avg', 'active_burn_duration_sec_a
                             'distance_walk_duration_sec_avg', 'environmental_audio_dur
                             'hr_variability_duration_sec_avg',  'oxygen_saturation_dur
                              'stand_hour_duration_sec_avg', 'step_length_duration_sec_
                             'walking_hr_duration_sec_avg', 'exercise_time_duration_sec
```

⇨ Cleaning up the naming of columns to make the dataframe more easily usable.

```
# ------------------------------------RENAME COLS-------------------------------------
masterframe_df.rename(columns={'year': 'yr', 'month': 'mo', 'day': 'dy', 'date': 'dt',
                               'daytime_category': 'dy_cat',
                               'active_cal_burned_avg': 'act_cal',
                               'distance_walk_miles_avg': 'dist_mi',
                               'env_audio_db_avg': 'env_au_db',
                               'exercise_min_avg': 'ex_min',
                               'flights_climbed_count_avg': 'climb',
                               'flights_climbed_duration_sec_avg': 'climb_dur',
                               'hr_activity_level': 'hr_act', 'hr_bpm_avg': 'hr_bpm',
                               'hr_variability_ms_avg': 'hr_var_ms',
                               'o2_sat_percentage_avg': 'o2_sat',
                               'o2_sat_barametric_kpa_avg': 'o2_sat_kpa',
                               'resting_burn_cal_avg': 'rest_cal',
                               'resting_burn_duration_sec_avg': 'rest_dur',
                               'stair_ascent_duration_sec_avg': 'upst_dur',
                               'stair_ascent_ft_sec_avg': 'upst_ft/sec',
                               'stair_descent_duration_sec_avg': 'dwnst_dur',
                               'stair_descent_ft_sec_avg': 'dwnst_ft/sec',
                               'stand_hour_avg': 'stnd_hr', 'stand_time_min_avg': 'stnd
                               'step_count_avg': 'stp_cnt',
                               'step_count_duration_sec_avg': 'stp_cnt_sec',
                               'step_length_inches_avg': 'stp_ins',
                               'walk_speed_mph_avg': 'wlk_mph',
                               'walk_speed_duration_sec_avg': 'wlk_mph_dur',
                               'walking_double_support_percentage_avg': 'wlk_dbl_%',
                               'walking_double_support_duration_sec_avg': 'wlk_dbl_dur'
                               'walking_hr_bpm_avg': 'wlk_bpm',
                               }, inplace=True)
```

⇨ Because of the way the Apple watch collects data, many columns were spotty and contained a great deal of NaN values. I am replacing them with averages and with zeros and then will see which version gives the best feedback.

```
# ------------------------------NaNs to AVERAGE-----------------------------------
# Function to replace NaNs with average of column for every value in the dataframe
```

```
def replace_nans(df):
    for col in df.columns:
        df[col].fillna((df[col].mean()), inplace=True)
    return df
```

```
df_avgs = replace_nans(masterframe_df)
```

```
#-----------------------------------NaNs to Zeros-------------------------------------
# Function to replace NaNs with 0

def replace_nans_with_zero(df):
    for col in df.columns:
        df[col].fillna(0, inplace=True)
    return df

df_zeros = replace_nans_with_zero(masterframe_df)
```

```
#-----------------------------------ADDING WEEKDAY NAMES-----------------------------
# Add a column with the day of the week to the 3 merged dataframes

df_avgs['wkdy'] = df_avgs['dt'].dt.day_name()
df_zeros['wkdy'] = df_zeros['dt'].dt.day_name()
```

⇨ Although it possibly seems odd, adding a moon phase column to the data, because there has definitely been a noticed correlation over the past three years of illness relating to the phases of the moon.

```
# ---------------------------------ADDING MOON PHASES---------------------------------
"""
moonphase.py - Calculate Lunar Phase
Author: Sean B. Palmer, inamidst.com
Cf. http://en.wikipedia.org/wiki/Lunar_phase#Lunar_phase_calculation
"""

import math, decimal, datetime

dec = decimal.Decimal


def position(now=None):
    if now is None:
        now = datetime.datetime.now()

    diff = now - datetime.datetime(2001, 1, 1)
    days = dec(diff.days) + (dec(diff.seconds) / dec(86400))
    lunations = dec("0.20439731") + (days * dec("0.03386319269"))
```

```
    return lunations % dec(1)


def phase(pos):
    index = (pos * dec(8)) + dec("0.5")
    index = math.floor(index)
    return {
        0: "New Moon",
        1: "Waxing Crescent",
        2: "First Quarter",
        3: "Waxing Gibbous",
        4: "Full Moon",
        5: "Waning Gibbous",
        6: "Last Quarter",
        7: "Waning Crescent"
    }[int(index) & 7]


def main():
    pos = position()
    phasename = phase(pos)

    roundedpos = round(float(pos), 3)
    print("%s (%s)" % (phasename, roundedpos))


if __name__ == "__main__":
    main()
```

```
New Moon (0.989)
```

```
#----------------------------------APPLYING MOON PHASES-----------------------------

df_avgs['moon'] = 0
df_avgs['moon'] = df_avgs['dt'].apply(lambda x: phase(position(x)))
df_zeros['moon'] = 0
df_zeros['moon'] = df_avgs['dt'].apply(lambda x: phase(position(x)))
```

```
# masterframe_df.to_csv('masterframe_df.csv')
df_avgs.to_csv('df_avgs.csv')
df_zeros.to_csv('df_zeros.csv')
```

```
jovian.commit()
```

```
[jovian] Updating notebook "evanmarie/puppy-project-01" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/evanmarie/puppy-project-01
```

```
'https://jovian.ai/evanmarie/puppy-project-01'
```