In [1]:
```
import pandas as pd
import random
```

```
/Users/edmondniu/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

Enter Subject Number

In [2]:
```
#Subject Number
subjN = 99
```

Import Mapping Data and Clean It

In [3]:
```
mapping = pd.read_csv("Misophonia Mapping Sounds 2.csv")
mapping = mapping[['Name', 'Number']]
mapping.columns = ['Name', 'Sound']
```

Import Qualtrics Form Data

In [4]:
```
# Data provided by the user

# Creating the DataFrame
df = pd.read_csv("miso raw data 7.csv")

df['subject_number']
```

Out[4]:
```
0    Please enter your subject number:
1            {"ImportId":"QID13_TEXT"}
2                                    99
3                                    41
4                                     4
5                                     1
Name: subject_number, dtype: object
```

Manipulate / Wrangle the Input Qualtrics Data

In [5]:
```
#remove all rows that are not the current subject
df.rename(columns={'subject_number': 'Subject Number'}, inplace=True)
df = df[df['Subject Number'] == str(subjN)]

if df.shape[0] == 0:
    print("Error: Subject Number: " + str(subjN) + " not found!")
```

In [6]:
```
#get date
date = df['StartDate'].iloc[0]
date = date[:10]
```

```
#remove uneccessary columns
columns_list = df.columns.tolist()
#print(columns_list)
columns_to_remove = ['StartDate', 'EndDate', 'Status', 'IPAddress', 'Progres
df = df.drop(columns=columns_to_remove)

df
```

Out[6]:

	Subject Number	1_Rating_1	1_Memory	1_Trigger	2_Rating_1	2_Memory	2_Trigger	3_R
2	99	NaN	No	NaN	-2	NaN	No	

1 rows × 319 columns

In [7]:
```
# Pivoting the DataFrame to match the desired format

# Melting the dataframe to make it long
df_melted = df.melt(id_vars=["Subject Number"],
                    var_name="Sound_Trigger",
                    value_name="Value")
#df_melted
```

In [8]:
```
# Splitting the Sound_Trigger column into two separate columns: one for Sour
df_melted['Sound'] = df_melted['Sound_Trigger'].apply(lambda x: x.split('_')
df_melted['Type'] = df_melted['Sound_Trigger'].apply(lambda x: x.split('_')[
#df_melted
```

In [9]:
```
# Pivoting the table to get the correct format
df_pivoted = df_melted.pivot_table(index=['Subject Number', 'Sound'], columr
#df_pivoted
```

In [10]:
```
# Renaming columns for clarity
df_final = df_pivoted.rename(columns={"Subject Number": "Subject", "Rating":
df_final['Sound'] = df_final['Sound'].astype(int)  # Converting Sound to num

df_final = df_final.sort_values(by=['Sound']).reset_index(drop=True)
#fix qualtrics bug
df_final.loc[df_final['Sound'] > 56, 'Sound'] -= 3

#merge mapping + df_final
df_final = pd.merge(df_final, mapping, on="Sound")

df_final = df_final[['Subject', 'Sound', 'Name', 'Rating', 'Trigger', 'Memor
```

In [11]:
```
#Na Rows
nan_rows = df_final[df_final['Rating'].isna()]

nan_rows
```

Out[11]:

	Subject	Sound	Name	Rating	Trigger	Memory
0	99	1	Personalized 1	NaN	NaN	No
60	99	61	m_IADS_FemaleCough_242_s.wav	NaN	NaN	No

1. Create sound_rating_all: Sound Ratings Table for ALL Sounds

id, date, name, rating, trigger

In [12]:
```
sound_rating_all = df_final[['Subject', 'Sound', 'Name', 'Rating', 'Trigger'
sound_rating_all['Date'] = date

sound_rating_all = sound_rating_all[['Subject', 'Date', 'Sound', 'Name', 'Ra

sound_rating_all
```

Out[12]:

	Subject	Date	Sound	Name	Rating	Trigger	Memory
0	99	2024-04-08	1	Personalized 1	NaN	NaN	No
1	99	2024-04-08	2	Personalized 2	-2	No	NaN
2	99	2024-04-08	3	Personalized 3	-5	No	NaN
3	99	2024-04-08	4	Personalized 4	-5	No	NaN
4	99	2024-04-08	5	Personalized 5	-2	No	NaN
...	...	...	...	...	...	...	...
101	99	2024-04-08	102	n_RobinChirping_s.wav	4	NaN	Yes
102	99	2024-04-08	103	n_VaccuumCleaner_s.wav	0	NaN	Yes
103	99	2024-04-08	104	n_WashingMachine_s.wav	0	NaN	No
104	99	2024-04-08	105	n_WaterStream_s.wav	1	NaN	No
105	99	2024-04-08	106	n_WindChimes_s.wav	3	NaN	No

106 rows × 7 columns

2. Create df_miso_aversive: Miso/Aversive Sound Ratings Table

In [13]:
```
#get rid of all NaN values in Trigger column (get rid of positive sounds)
df_miso_aversive = df_final.dropna(subset=['Trigger'])

#Drop Memory Column (not needed)
df_miso_aversive = df_miso_aversive.drop(columns=['Memory'])

#convert str --> int
df_miso_aversive['Rating'] = df_miso_aversive['Rating'].astype(int)

#abs value of ratings
df_miso_aversive['Rating'] = df_miso_aversive['Rating'].abs()
```

3. Create df_miso: Miso Sounds Ratings Table

In [14]:
```
df_miso = df_miso_aversive[df_miso_aversive['Trigger'] == 'Yes'].reset_index
#df_miso
```

4. Create df_aversive: Aversive Sounds Ratings Table

In [15]:
```
df_aversive = df_miso_aversive[df_miso_aversive['Trigger'] == 'No'].reset_ir
#df_aversive
```

5. Create df_mri_ratings: 20 mri sounds ratings table

In [16]:
```
df_miso_10 = df_miso.sort_values(by=['Rating'], ascending = False).head(10).
df_aversive_10 = df_aversive.sort_values(by=['Rating'], ascending = False).h
```

Set Warning_MRI (Make sure df_miso_10 and df_aversive_10 HAVE 10 sounds each before merging into df_mri_ratings)

In [17]:
```
warning_MRI_miso_less10 = False
warning_MRI_aver_less10 = False
numRows_MRI_miso = df_miso_10.shape[0]
numRows_MRI_aversive = df_aversive_10.shape[0]
numRows_MRI_miso
```

Out[17]:
```
10
```

In [18]:
```
def createDuplicates(df, numRows):
    numRepeats = 10 - numRows
    items = df['Name'].tolist()
    if numRows < 5:
        #duplicates = random.choices(items, k=numRepeats)
        duplicates = df.sample(n=numRepeats, replace=True)
```

```
    else:
        #duplicates = random.sample(items, k=numRepeats)
        duplicates = df.sample(n=numRepeats)
    return duplicates

if numRows_MRI_miso < 10:
    warning_MRI_miso_less10 = True
    duplicates_miso = createDuplicates(df_miso_10, numRows_MRI_miso)
    df_miso_10 = pd.concat([df_miso_10, duplicates_miso], ignore_index=True)

if numRows_MRI_aversive < 10:
    warning_MRI_aver_less10 = True
    duplicates_aver = createDuplicates(df_aversive_10, numRows_MRI_aversive)
    df_aversive_10 = pd.concat([df_aversive_10, duplicates_aver], ignore_inc
```

In [19]:
```
df_mri_ratings = pd.concat([df_miso_10, df_aversive_10], ignore_index=True)
```

5.1. Create Rankings/Order_labels For Nimesha

In [20]:
```
#Create Ranking

# Group by 'Trigger', rank each group, and map the rankings back to the orig
df_mri_ratings['Ranking'] = df_mri_ratings.groupby('Trigger')['Rating'].rank

# Now sort by 'Trigger' and then by 'Ranking' within each group to see the r
df_mri_ratings.sort_values(by=['Trigger', 'Ranking'], inplace=True)

df_mri_ratings = df_mri_ratings.reset_index(drop=True)
```

In [21]:
```
#Create Order Label
dictionary1 = {1: "A",
               2: "B",
               3: "C",
               4: "D",
               5: "E",
               6: "F",
               7: "G",
               8: "H",
               9: "I",
               10: "J" }

dictionary2 = {"A": 1,
               "B": 5,
               "C": 2,
               "D": 6,
               "E": 3,
               "F": 7,
               "G": 9,
               "H": 4,
               "I": 8,
               "J": 10}

dictionary11 = {1: "A",
               2: "C",
               3: "E",
```

```
                4: "H",
                5: "B",
                6: "D",
                7: "F",
                8: "I",
                9: "G",
                10: "J" }

dictionary22 = {"A": 1,
                "B": 2,
                "C": 3,
                "D": 4,
                "E": 5,
                "F": 6,
                "G": 7,
                "H": 8,
                "I": 9,
                "J": 10}

df_mri_ratings['Alpha'] = df_mri_ratings['Ranking'].map(dictionary11)
df_mri_ratings['Order_Label'] = df_mri_ratings['Alpha'].map(dictionary22)

#df_mri_ratings
```

5.1. Create df_mri_ratings_nimesha: Ratings CSV file to be sent to nimesha

In [22]:
```
column_to_remove = ['Subject', 'Sound', 'Alpha']
df_mri_ratings_nimesha = df_mri_ratings.drop(columns=column_to_remove)

df_mri_ratings_nimesha
```

Out[22]:

	Name	Rating	Trigger	Ranking	Order_Label
0	a_WomanWailing_s.wav	6	No	1	1
1	a_ScreamWithEcho_s.wav	6	No	2	3
2	a_PuppyCrying_s.wav	6	No	3	5
3	a_FireTruckAlarm_s.wav	6	No	4	8
4	a_CryingMan_s.wav	6	No	5	2
5	m_iads_Whistling_270_s.wav	6	No	6	4
6	Personalized 6	6	No	7	6
7	a_Fart_s.wav	8	No	8	9
8	a_CarsHonking_s.wav	8	No	9	7
9	a_AlarmClock_s.wav	9	No	10	10
10	m_DrinkingWater_s.wav	5	Yes	1	1
11	m_AppleEating_s.wav	5	Yes	2	3
12	m_HotTeaSlurping_s.wav	5	Yes	3	5
13	m_ChewingGum_s.wav	5	Yes	4	8
14	m_ChewingPopcornManyCrunches_s.wav	5	Yes	5	2
15	m_HeavyBreathing2_s.wav	6	Yes	6	4
16	m_FemalePanting_s.wav	6	Yes	7	6
17	m_EatingSaladCutlery_s.wav	6	Yes	8	9
18	m_ChewingFoodWithMouthOpen_s.wav	7	Yes	9	7
19	m_SlowHardBreathing_s.wav	7	Yes	10	10

5.2 Create df_mri_sound_names: Just 20 MRI Sound Names

In [23]:
```
df_mri_sound_names = df_mri_ratings['Name']
df_mri_sound_names
```

Out[23]:
```
0                    a_WomanWailing_s.wav
1                  a_ScreamWithEcho_s.wav
2                     a_PuppyCrying_s.wav
3                  a_FireTruckAlarm_s.wav
4                       a_CryingMan_s.wav
5              m_iads_Whistling_270_s.wav
6                           Personalized 6
7                             a_Fart_s.wav
8                     a_CarsHonking_s.wav
9                      a_AlarmClock_s.wav
10                  m_DrinkingWater_s.wav
11                    m_AppleEating_s.wav
12                 m_HotTeaSlurping_s.wav
13                     m_ChewingGum_s.wav
14    m_ChewingPopcornManyCrunches_s.wav
15                m_HeavyBreathing2_s.wav
16                  m_FemalePanting_s.wav
17             m_EatingSaladCutlery_s.wav
18       m_ChewingFoodWithMouthOpen_s.wav
19               m_SlowHardBreathing_s.wav
Name: Name, dtype: object
```

6. Create df_tms_ratings: 24 tms sounds ratings table

Warning: If there are <12 miso sounds and <24 miso sounds...

In [24]:
```
warning_TMS_less12 = False
warning_TMS_less24 = False

numMiso = df_miso.shape[0]
num_Mprefix_needed = 0
numMiso
```

Out[24]:
```
36
```

In [25]:
```
if numMiso < 12: #if we need to add miso non-trigger sounds
    num_Mprefix_needed = 12 - numMiso
    #Create Highest Aversive Rating of Sounds 26-98 "m_...wav"

    #aversive_sounds_Mprefix = df_aversive[(df_aversive['Sound'] >= 26) & (d
    aversive_sounds_Mprefix = df_final[(df_final['Sound'] >= 26) & (df_final

    #DUPLICATES_aversive_sounds_Mprefix_needed = aversive_sounds_Mprefix.sor
    DUPLICATES_aversive_sounds_Mprefix_needed = aversive_sounds_Mprefix.sort

    df_tms_ratings = pd.concat([df_miso, DUPLICATES_aversive_sounds_Mprefix_
    df_tms_ratings_copy = df_tms_ratings.copy()
    df_tms_ratings = pd.concat([df_tms_ratings, df_tms_ratings_copy], ignore
    category = [0,0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3]
    df_tms_ratings['Category'] = category

    warning_TMS_less12 = True
```

In [26]:
```
if (numMiso >= 12) and (numMiso < 24):
    num_Mprefix_needed = 24 - numMiso
    DUPLICATES_12to24_tms = df_miso.sample(n=num_Mprefix_needed)
    df_tms_ratings = pd.concat([df_miso, DUPLICATES_12to24_tms], ignore_inde

    category = [0,0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3]
    df_tms_ratings['Category'] = category

    warning_TMS_less24 = True
```

In [27]:
```
if numMiso >= 24:
    #create 6 personalized, and then choose 6 top and 6 bottom from nonpersc
    #change this
    df_miso_personalized = df_final[df_final['Sound'] <= 6]
    #df_miso_personalized = df_miso[df_miso['Sound'] <= 6]

    df_miso_noPersonalized = df_miso[df_miso['Sound'] > 6]
    df_miso_NP_6Highest = df_miso_noPersonalized.sort_values(by=['Rating'],
    df_miso_NP_6Lowest = df_miso_noPersonalized.sort_values(by=['Rating'], a

    #Get rid of top 6 and bottom 6 sounds to pull random sounds for middle 6
    NP_Or6Highest = pd.concat([df_miso_noPersonalized, df_miso_NP_6Highest,
    NP_Or6Highest_Or6Lowest = pd.concat([NP_Or6Highest, df_miso_NP_6Lowest,
    df_miso_NP_6Middle = NP_Or6Highest_Or6Lowest.sample(n=6)

    #concat all together
    df_tms_ratings = pd.concat([df_miso_NP_6Lowest, df_miso_NP_6Middle, df_m
    category = [0,0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3]
    df_tms_ratings['Category'] = category
```

In [28]:
```
df_tms_ratings_FINAL = df_tms_ratings[['Subject', 'Category', 'Name', 'Trigg
df_tms_ratings_FINAL
```

Out[28]:

	Subject	Category	Name	Trigger	Rating
0	99	0	m_iads_Chewing_724_s.wav	Yes	1
1	99	0	n_BirdsSinging_s.wav	Yes	2
2	99	0	m_EatingSmallCrunches_s.wav	Yes	2
3	99	0	m_DrinkingWater3_s.wav	Yes	2
4	99	0	m_Sneeze_s.wav	Yes	2
5	99	0	m_EatingCandy_s.wav	Yes	3
6	99	1	m_Yawn_s.wav	Yes	4
7	99	1	m_DrinkingWater2_s.wav	Yes	4
8	99	1	m_BallBouncing_s.wav	Yes	3
9	99	1	m_GumChewing_s.wav	Yes	5
10	99	1	m_CatTunaLicking_s.wav	Yes	4
11	99	1	m_SlurpingNoodles_s.wav	Yes	4
12	99	2	m_ChewingFoodWithMouthOpen_s.wav	Yes	7
13	99	2	m_SlowHardBreathing_s.wav	Yes	7
14	99	2	m_HeavyBreathing2_s.wav	Yes	6
15	99	2	m_FemalePanting_s.wav	Yes	6
16	99	2	m_EatingSaladCutlery_s.wav	Yes	6
17	99	2	m_DrinkingWater_s.wav	Yes	5
18	99	3	Personalized 1	NaN	NaN
19	99	3	Personalized 2	No	-2
20	99	3	Personalized 3	No	-5
21	99	3	Personalized 4	No	-5
22	99	3	Personalized 5	No	-2
23	99	3	Personalized 6	No	-6

6.2 Create df_tms_sound_names: 24 TMS Sound Names

In [29]:
```
df_tms_sound_names = df_tms_ratings['Name']
df_tms_sound_names
```

Out[29]:
```
0             m_iads_Chewing_724_s.wav
1                 n_BirdsSinging_s.wav
2           m_EatingSmallCrunches_s.wav
3                m_DrinkingWater3_s.wav
4                        m_Sneeze_s.wav
5                   m_EatingCandy_s.wav
6                          m_Yawn_s.wav
7                m_DrinkingWater2_s.wav
8                  m_BallBouncing_s.wav
9                    m_GumChewing_s.wav
10               m_CatTunaLicking_s.wav
11              m_SlurpingNoodles_s.wav
12    m_ChewingFoodWithMouthOpen_s.wav
13           m_SlowHardBreathing_s.wav
14             m_HeavyBreathing2_s.wav
15               m_FemalePanting_s.wav
16          m_EatingSaladCutlery_s.wav
17               m_DrinkingWater_s.wav
18                       Personalized 1
19                       Personalized 2
20                       Personalized 3
21                       Personalized 4
22                       Personalized 5
23                       Personalized 6
Name: Name, dtype: object
```

Save Miso and Non-Miso DFs as CSVs to SubjectData/SubjectNum

Deal with Warnings

In [30]:
```
# Creating a CSV file for each subject containing only the Sound column
csv_paths = []

#if warning is true
if warning:
    #add those duplicates to the output file

    #create warning text file. Populate with numDuplicates and what are the
    csv_file_path_warning = os.path.join(subdirectory_name, f'subject_{subje
    dup_set = set(duplicates)
    with open(csv_file_path_warning, 'w') as file:
        file.write("WARNING! SUBJECT DID NOT CLASSIFY 10 UNIQUE SOUNDS AS MI
                   "Total Number of Missing Miso Sounds (i.e. # Audio Files
                   + str(numRepeats) + "\n\n" +
                   "Audio Files that were repeated: " + str(dup_set))
    print(f"File saved to {csv_file_path_warning}")
    csv_paths.append(csv_file_path_warning)
```

```
    # Add duplicates to the existing sounds to create the final sounds (with
    duplicates_df = pd.DataFrame(duplicates)
    df_subject_yes = pd.concat([df_subject_yes, duplicates_df], ignore_index
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 5
      2 csv_paths = []
      4 #if warning is true
----> 5 if warning:
      6     #add those duplicates to the output file
      7
      8     #create warning text file. Populate with numDuplicates and what
are the duplicates?
      9     csv_file_path_warning = os.path.join(subdirectory_name, f'subjec
t_{subject}_warning.txt')
     10     dup_set = set(duplicates)

NameError: name 'warning' is not defined
```

In [31]:
```
import os

# Name of the subdirectory to create within the current directory
subdirectory_name = 'SubjectData/subject_' + str(subjN)

# Create the subdirectory if it doesn't exist
if not os.path.exists(subdirectory_name):
    os.makedirs(subdirectory_name)

# Define file path
csv_file_path_20MRI_sounds = os.path.join(subdirectory_name, f'subject_{subj
csv_file_path_sound_rating_MRI = os.path.join(subdirectory_name, f'subject_{
csv_file_path_24TMS_sounds = os.path.join(subdirectory_name, f'subject_{subj
csv_file_path_sound_rating_TMS = os.path.join(subdirectory_name, f'subject_{
csv_file_path_sound_rating_ALL = os.path.join(subdirectory_name, f'subject_{


# Save the Sound column to CSV
df_mri_sound_names.to_csv(csv_file_path_20MRI_sounds, index=False, header=Fa
df_mri_ratings_nimesha.to_csv(csv_file_path_sound_rating_MRI, index=False, h
df_tms_sound_names.to_csv(csv_file_path_24TMS_sounds, index=False, header=Fa
df_tms_ratings_FINAL.to_csv(csv_file_path_sound_rating_TMS, index=False, hea
sound_rating_all.to_csv(csv_file_path_sound_rating_ALL, index=False, header=

# Collecting file paths for download links
csv_paths.append(csv_file_path_20MRI_sounds)
csv_paths.append(csv_file_path_sound_rating_MRI)
csv_paths.append(csv_file_path_24TMS_sounds)
csv_paths.append(csv_file_path_sound_rating_TMS)
csv_paths.append(csv_file_path_sound_rating_ALL)


csv_paths
```

Out[31]:
```
['SubjectData/subject_99/subject_99_20MRI_sounds_names.csv',
 'SubjectData/subject_99/subject_99_20MRI_sounds_ratings.csv',
 'SubjectData/subject_99/subject_99_24TMS_sounds_names.csv',
 'SubjectData/subject_99/subject_99_24TMS_sounds_ratings.csv',
 'SubjectData/subject_99/subject_99_ALL_sounds_ratings.csv']
```

In [ ]: