In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import pointbiserialr
from scipy.stats import spearmanr
import re
```

This file calculates all quantitative data, all of the participants data, the table and all corelations

@author: Jonathan van Oudheusden @date: 2024-06-19

Required files: - Output/Intro_codes.csv - Output/Intro_peer_codes.csv Both need: 'rand_id', 'smoker', Intro_codes needs: 'Codes' Intro_peer_codes needs: 'Peer_codes'

Required but not included in dataset as of release - 'Data/prolific_profile_bsc_anonym.csv' needs: 'Gender', 'Age', 'Highest education level completed' - 'Data/prescreening_questionnaire_bsc_anonym.csv' needs: 'Vaping_Freq', 'PIV_Smoking_Freq', 'Quit_Before_24_Hours', 'Num_Quit_24h_Last_Y' 'Smok-vap_identity_1', 'Smok-vap_identity_2', 'Smok-vap_identity_3', 'Quitter_SelfIdentity_1', 'Quitter_SelfIdentity_2', 'Quitter_SelfIdentity_4', 'Not_smok-vap_identit_1', 'Not_smok-vap_identit_2', 'Not_smok-vap_identit_3' - 'Output/AllSessionsData.csv' generated by PreProcessing.py needs: 'state_humansupport', 'dropout_response', 'state_importance', 'session_num' - 'Data/postquestionnaire_bsc_anonym.csv' needs: 'rand_id', 'P_effect_R_1'

Output files:

- themes.png
- It prints all other values the terminal/section below the code

In [ ]:
```
## Merge all files together

## add intro data
coded = pd.read_csv('Output/Intro_codes.csv')
peer_coded = pd.read_csv('Output/Intro_peer_codes.csv')

df_intro = pd.merge(coded, peer_coded[['rand_id', 'Peer_codes']], on='ran

### add prolific daa

prolificDF = pd.read_csv('Data/prolific_profile_bsc_anonym.csv')

df_merged_into_prolific = pd.merge(df_intro, prolificDF, on='rand_id', ho

## add prescreening daa

prescreening_df = pd.read_csv('Data/prescreening_questionnaire_bsc_anonym

df = pd.merge(df_merged_into_prolific, prescreening_df, on='rand_id', how
```

```
## add more data from sessions
# Columns you want to keep
columns_to_keep = [ 'state_humansupport', 'dropout_response', 'state_impo

df_sessions = pd.read_csv('Output/AllSessionsData.csv')

# Calculate the mean of 'state_humansupport' for each rand_id
mean_values = df_sessions.groupby('rand_id')[columns_to_keep].mean().rese

# Rename the columns
mean_values = mean_values.rename(columns={
    'state_humansupport': 'Mean_HumanSupport',
    'dropout_response': 'Mean_dropout_response',
    'state_importance': 'Mean_Importance'
})


# Identify the first session for each rand_id
df_sessions_first = df_sessions.groupby('rand_id').first().reset_index()
df_sessions_first_keepSome = df_sessions_first[['rand_id'] + columns_to_k

df_to_merge = pd.merge(df_sessions_first_keepSome, mean_values, on='rand_

df = pd.merge(df,df_to_merge, on='rand_id', how='left')

count_sessions = df_sessions.groupby('rand_id').last().reset_index()
count_sessions = count_sessions[['rand_id', 'session_num']]
count_sessions = count_sessions.rename(columns={'session_num': 'count_ses

df = pd.merge(df,count_sessions, on='rand_id', how='left')

## add rating of feedback during the study
df_postQuest = pd.read_csv('Data/postquestionnaire_bsc_anonym.csv')
df_postQuest2 = df_postQuest[['rand_id', 'P_effect_R_1']]

df = pd.merge(df,df_postQuest2, on='rand_id', how='left')
```

In [ ]:

```
## Add count of codes in each theme to dataframe

# Define themes and their associated codes
themes_and_codes = {
    'Motivation for quitting': ['Health Concerns', 'Financial motivations
                                'Athletic performance', 'Pre-study improv
    'Previous attempts to quit': ['History and attempts', 'Replaced smoki
    'Barriers to Quitting': ['Hardships of quitting', 'Emotional and Psyc
                             'Physical Limitations', 'Nicotine', 'Habitua
    'Desired support': ['Desire for support', 'Quitting Needs', 'Motivati
    'Usage Patterns': ['Habitual behavior', 'Location/Environment Depende
    'Identity' : ['smoker/vaper identity', 'non-smoker/vaper identity']
}

# Function to count occurrences of theme codes in each row
def count_theme_codes(row, theme_codes):
    if isinstance(row, float):
        row = ''
    row_codes = [code.strip() for code in row.split(',')]

    # Count occurrences of each item
    item_counts = Counter(row_codes)
```

```
    # Filter and print items that appear more than once
    duplicates = [item for item, count in item_counts.items() if count >
    if not (len(duplicates) == 0):
        print("Duplicates:", item_counts)

    count = sum(1 for code in row_codes if code in theme_codes)
    return count

# Add a column for each theme with counts of theme codes
for theme, codes in themes_and_codes.items():
    df[theme] = df['Codes'].apply(lambda row: count_theme_codes(row, code
```

In [ ]:

```
## Create plot of themes

Themes_list = [
    'Motivation for quitting',
    'Previous attempts to quit',
    'Barriers to Quitting',
    'Desired support',
    'Usage Patterns',
    'Identity'
]

## add value of length of intro
df['len_intro'] = df['human_coach_introduction_slot'].apply(lambda row: l

## add value of count of codes per introduction
df['count_codes'] = df[Themes_list].sum(axis=1)
```

In [ ]:

```
# Mapping dictionary
opinion_mapping = {
    'Disagree strongly': 1,
    'Disagree': 2,
    'Neither agree nor disagree': 3,
    'Agree': 4,
    'Agree strongly': 5
}

# Convert categorical values to numerical values using the mapping
df['Smok-vap_identity_1_Numeric'] = df['Smok-vap_identity_1'].map(opinion
df['Smok-vap_identity_2_Numeric'] = df['Smok-vap_identity_2'].map(opinion
df['Smok-vap_identity_3_Numeric'] = df['Smok-vap_identity_3'].map(opinion

df['Quitter_SelfIdentity_1_Numeric'] = df['Quitter_SelfIdentity_1'].map(o
df['Quitter_SelfIdentity_2_Numeric'] = df['Quitter_SelfIdentity_2'].map(o
df['Quitter_SelfIdentity_4_Numeric'] = df['Quitter_SelfIdentity_4'].map(o

df['Not_smok-vap_identit_1_Numeric'] = df['Not_smok-vap_identit_1'].map(o
df['Not_smok-vap_identit_2_Numeric'] = df['Not_smok-vap_identit_2'].map(o
df['Not_smok-vap_identit_3_Numeric'] = df['Not_smok-vap_identit_3'].map(o

df['smoke_identity'] = df[['Smok-vap_identity_1_Numeric', 'Smok-vap_ident
                           'Smok-vap_identity_3_Numeric']].sum(axis=1)

df['quit_identity'] = df[['Quitter_SelfIdentity_1_Numeric', 'Quitter_Self
                          'Quitter_SelfIdentity_4_Numeric']].sum(axis=1)

df['Not_smoke_identity'] = df[['Not_smok-vap_identit_1_Numeric', 'Not_smo
```

```
                            'Not_smok-vap_identit_3_Numeric']].sum(axis=1)
```

In [ ]:

```
# Mapping dictionary
smoking_mapping = {
    "Not applicable" :0,
    "Less than 4 times a month": 1,
    "1-6 times a week":2,
    "Once a day":3,
    "2-5 times a day":4,
    "6-10 times a day":5,
    "11-19 times a day":6,
    "More than 20 times a day":7,
}

# Map categorical values to numerical values for both columns
df['PIV_Smoking_Mapped'] = df['PIV_Smoking_Freq'].map(smoking_mapping)
df['Vaping_Mapped'] = df['Vaping_Freq'].map(smoking_mapping)


# combine smoking and vaping usage
df['Usage'] = df['PIV_Smoking_Mapped'].fillna(df['Vaping_Mapped'])

# drop original columns
df.drop(columns=['PIV_Smoking_Mapped', 'Vaping_Mapped'], inplace=True)

# Turn quit before into binary vaues
df['Quit_Before'] = df['Quit_Before_24_Hours'].apply(lambda x: 1 if x ==
```

In [ ]:

```
## count how many characters are in all the introdutions combined

charcount = 0
wordcount = 0

for index, row in df.iterrows():
    text = row['human_coach_introduction_slot']
    wordcount += len(text.split())
    charcount += len(re.sub(r'\s+', '', text))

print(f"Character count: {charcount}, Word count: {wordcount}")
```

```
Character count: 133533, Word count: 32304
```

In [ ]:

```
### Get population data
smoker_distribution = df['smoker'].value_counts()
print(smoker_distribution)

gender_distribution = df['Gender'].value_counts()
print(gender_distribution)

print( df.groupby('smoker')['Gender'].value_counts())

print(df['Age'].describe())

age_by_smoker = df.groupby('smoker')['Age'].describe()
print(age_by_smoker)


education_distribution = df['Highest education level completed'].value_co
print(education_distribution)
```

```
print(df.groupby('smoker')['Highest education level completed'].value_cou

vape_freq = df['Vaping_Freq'].value_counts()
print(vape_freq)

smoke_freq = df['PIV_Smoking_Freq'].value_counts()
print(smoke_freq)

quit_before = df['Quit_Before_24_Hours'].value_counts()
print(quit_before)

print(df.groupby('smoker')['Quit_Before_24_Hours'].value_counts())


amount_of_attemps = df['Num_Quit_24h_Last_Y'].value_counts()
print(amount_of_attemps)

print(df.groupby('smoker')['Num_Quit_24h_Last_Y'].value_counts())
```

```
smoker
0    401
1    397
Name: count, dtype: int64
Gender
Woman (including Trans Female/Trans Woman)     398
Man (including Trans Male/Trans Man)           382
Non-binary (would like to give more detail)     18
Name: count, dtype: int64
smoker  Gender
0       Man (including Trans Male/Trans Man)           200
        Woman (including Trans Female/Trans Woman)     191
        Non-binary (would like to give more detail)     10
1       Woman (including Trans Female/Trans Woman)     207
        Man (including Trans Male/Trans Man)           182
        Non-binary (would like to give more detail)      8
Name: count, dtype: int64
count    798.000000
mean      36.030075
std       11.184956
min       18.000000
25%       27.000000
50%       34.000000
75%       43.000000
max       77.000000
Name: Age, dtype: float64
        count       mean        std   min   25%   50%   75%   max
smoker
0       401.0  32.648379  10.136989  18.0  25.0  30.0  38.0  69.0
1       397.0  39.445844  11.167468  20.0  31.0  38.0  47.0  77.0
Highest education level completed
Undergraduate degree (BA/BSc/other)     308
High school diploma/A-levels            172
Graduate degree (MA/MSc/MPhil/other)    119
Technical/community college             107
Secondary education (e.g. GED/GCSE)      70
Doctorate degree (PhD/other)             11
Don't know / not applicable               6
No formal qualifications                  5
Name: count, dtype: int64
smoker  Highest education level completed
0       Undergraduate degree (BA/BSc/other)     155
        High school diploma/A-levels             80
        Graduate degree (MA/MSc/MPhil/other)     69
        Technical/community college              52
        Secondary education (e.g. GED/GCSE)      34
        Doctorate degree (PhD/other)              6
        Don't know / not applicable               4
        No formal qualifications                  1
1       Undergraduate degree (BA/BSc/other)     153
        High school diploma/A-levels             92
        Technical/community college              55
        Graduate degree (MA/MSc/MPhil/other)     50
        Secondary education (e.g. GED/GCSE)      36
        Doctorate degree (PhD/other)              5
        No formal qualifications                  4
        Don't know / not applicable               2
Name: count, dtype: int64
Vaping_Freq
More than 20 times a day    188
```

```
2-5 times a day              67
11-19 times a day            61
6-10 times a day             58
Once a day                   27
Name: count, dtype: int64
PIV_Smoking_Freq
11-19 times a day           125
6-10 times a day            109
More than 20 times a day     94
2-5 times a day              59
Once a day                   10
Name: count, dtype: int64
Quit_Before_24_Hours
Yes    600
No     197
Name: count, dtype: int64
smoker  Quit_Before_24_Hours
0       Yes                        268
        No                         132
1       Yes                        332
        No                          65
Name: count, dtype: int64
Num_Quit_24h_Last_Y
1 - 5 times                                                      348
I have NOT tried to quit ${e://Field/VERB_ING} in the last year    150
6 - 10 times                                                      61
More than 10 times                                                41
Name: count, dtype: int64
smoker  Num_Quit_24h_Last_Y
0       1 - 5 times
183
        6 - 10 times
39
        More than 10 times
28
        I have NOT tried to quit ${e://Field/VERB_ING} in the last year
18
1       1 - 5 times
165
        I have NOT tried to quit ${e://Field/VERB_ING} in the last year
132
        6 - 10 times
22
        More than 10 times
13
Name: count, dtype: int64
```

In [ ]:

```
## Total amount all codes of each theme is used
print(df['Motivation for quitting'].sum())
print(df['Previous attempts to quit'].sum())
print(df['Barriers to Quitting'].sum())
print(df['Desired support'].sum())
print(df['Usage Patterns'].sum())
print(df['Identity'].sum())
print(' ')

## Total amount atleast one code of a theme is used
print((df['Motivation for quitting'] != 0).sum())
print((df['Previous attempts to quit'] != 0).sum())
print((df['Barriers to Quitting'] != 0).sum())
```

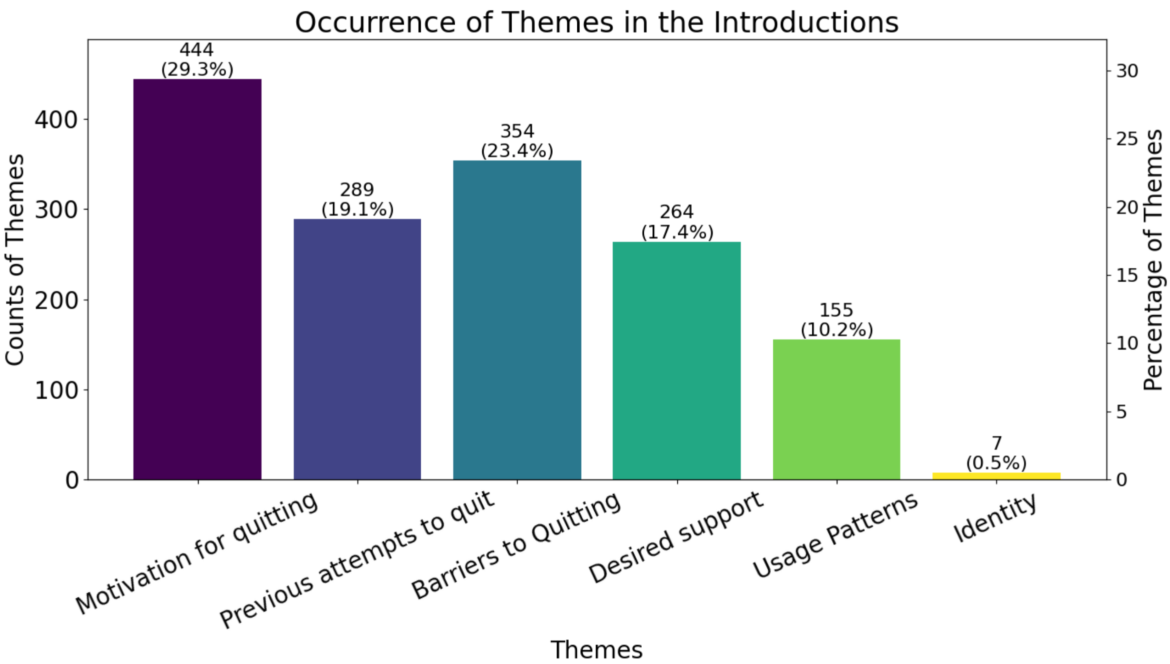
```
print((df['Desired support'] != 0).sum())
print((df['Usage Patterns'] != 0).sum())
print((df['Identity'] != 0).sum())
```

```
600
341
437
316
177
7

444
289
354
264
155
7
```

In [ ]:

```
## Create plot of themes

counts = [444, 289, 354, 264, 155, 7]

# Total for percentage calculation
total = sum(counts)

# Calculate percentages
percentages = [count / total * 100 for count in counts]
# Colors for each bar (using a colormap)
colors = plt.cm.viridis(np.linspace(0, 1, len(Themes_list)))

# Create a figure and a primary y-axis
fig, ax1 = plt.subplots(figsize=(14, 8))

# Plotting the bar graph
bars = ax1.bar(Themes_list, counts, color=colors)

# Set labels and title for the primary y-axis
ax1.set_xlabel('Themes', fontsize=20)
ax1.set_ylabel('Counts of Themes', fontsize=20)
ax1.set_title('Occurrence of Themes in the Introductions', fontsize=24)

# Set tick parameters for x-axis and y-axis
ax1.tick_params(axis='x', labelsize=20)
ax1.tick_params(axis='y', labelsize=20)

# Rotate x-axis labels
plt.xticks(rotation=25, ha='center')

# Adding the count and percentage text on the bars
for bar, count, percentage in zip(bars, counts, percentages):
    height = bar.get_height()
    ax1.text(
        bar.get_x() + bar.get_width() / 2.0,
        height ,  # Adjust the position to prevent overlap
        f'{count}\n({percentage:.1f}%)',
        ha='center',
        va='bottom',
        fontsize=16,
        color='black'
    )
```

```
# Increase the height of the y-axis to ensure text is not cut off
ax1.set_ylim(0, max(counts) * 1.1)

# Create a secondary y-axis for the percentages
ax2 = ax1.twinx()
ax2.set_ylabel('Percentage of Themes', fontsize=20)

# Set tick parameters for the secondary y-axis
ax2.tick_params(axis='y', labelsize=16)

# Plot percentages on the secondary y-axis
ax2.plot(Themes_list, percentages, color='none')  # Plot invisible line

# Set y-axis limits for percentages
ax2.set_ylim(0, max(percentages) * 1.1)  # Adjust the y-axis limits

# Tight layout for better spacing
plt.tight_layout()

plt.savefig('Plots/themes.png')

# Show the plot
plt.show()
```



In [ ]:

```
## Function for correlation with theme using point biserial correlation
def CalcCorrelationTheme(df_in, Column, Theme):
    data = df_in.copy()
    data[Theme] = data[Theme].apply(lambda x: 1 if x >= 1 else 0)
    CalcCorrelationBivarate(data, Column, Theme)

## function for calculating correlation using point biserial
def CalcCorrelationBivarate(df_in, Column, Column2):
    data  = df_in.copy()

    # Identify rows with NaN or Inf values
    mask = data[[Column, Column2]].isnull().any(axis=1)
    removed_rows = data[mask]

    # Print the randid of rows being removed
```

```
    if not removed_rows.empty:
        print(f'amount of rows remove {len(removed_rows)}')
        print("Rows being removed due to NaN or Inf values:")
        print(removed_rows['rand_id'].tolist())

    # Drop rows with NaN or Inf values
    data_cleaned = data.dropna(subset=[Column, Column2])

    # Calculate the point biserial correlation
    correlation, p_value = pointbiserialr(data_cleaned[Column2], data_cle

    print(f"Point Biserial Correlation between '{Column2}' and '{Column}'
    print(f"P-value: {p_value}")
```

In [ ]:

```
# funcion for calculating spearman correlaion
def CalcCorrelationSpearMan(df_in, Column, Column2):
    data = df_in.copy()

    # Identify rows with NaN or Inf values
    mask = data[[Column, Column2]].isnull().any(axis=1)
    removed_rows = data[mask]

    # Print the randid of rows being removed
    if not removed_rows.empty:
        print(f'amount of rows remove {len(removed_rows)}')
        print("Rows being removed due to NaN or Inf values:")
        print(removed_rows['rand_id'].tolist())

    # Drop rows with NaN or Inf values
    data_cleaned = data.dropna(subset=[Column, Column2])

    try:
        # Calculate Spearman's correlation
        spearman_corr, spearman_p = spearmanr(data_cleaned[Column], data_
        print(f"Spearman Correlation between '{Column2}' and '{Column}':


    except Exception as e:
        print(f"Error calculating Spearman correlation: {e}")
```

In [ ]:

```
## Motivation correlations

CalcCorrelationTheme(df, 'len_intro', 'Motivation for quitting')

CalcCorrelationTheme(df, 'Mean_Importance', 'Motivation for quitting')
CalcCorrelationTheme(df, 'count_sessions', 'Motivation for quitting')
```

```
Point Biserial Correlation between 'Motivation for quitting' and 'len_intr
o': 0.02713083350731219
P-value: 0.4440598672985398
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Point Biserial Correlation between 'Motivation for quitting' and 'Mean_Imp
ortance': 0.028718695118736124
P-value: 0.41813711347885807
Point Biserial Correlation between 'Motivation for quitting' and 'count_se
ssions': -0.0057436103765499185
P-value: 0.8713077121089434
```

In [ ]:
```
### Previous attmemps to quit

CalcCorrelationTheme(df, 'Quit_Before', 'Previous attempts to quit')
CalcCorrelationTheme(df, 'count_sessions', 'Previous attempts to quit')
```

```
Point Biserial Correlation between 'Previous attempts to quit' and 'Quit_B
efore': 0.06463164457137582
P-value: 0.0680283948562543
Point Biserial Correlation between 'Previous attempts to quit' and 'count_
sessions': 0.024000320677470904
P-value: 0.49839500313150803
```

In [ ]:
```
### Correlaions for 'Barriers to Quitting'


CalcCorrelationTheme(df, 'Quit_Before', 'Barriers to Quitting')
```

```
Point Biserial Correlation between 'Barriers to Quitting' and 'Quit_Before
': -0.012645686230978637
P-value: 0.721330676248413
```

In [ ]:
```
# correlations for 'Desired support'

CalcCorrelationTheme(df, 'Mean_HumanSupport', 'Desired support')
CalcCorrelationTheme(df, 'Quit_Before', 'Desired support')
```

```
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Point Biserial Correlation between 'Desired support' and 'Mean_HumanSuppor
t': 0.05705134222089879
P-value: 0.10752666065390933
Point Biserial Correlation between 'Desired support' and 'Quit_Before':
-0.03389126779209835
P-value: 0.33898900207212407
```

In [ ]:
```
## Correlaions for usage patterns

CalcCorrelationTheme(df, 'Usage', 'Usage Patterns')

CalcCorrelationSpearMan(df,'Usage', 'count_codes'  )
```

```
Point Biserial Correlation between 'Usage Patterns' and 'Usage': 0.0416151
512494038
P-value: 0.24029573880701582
Spearman Correlation between 'count_codes' and 'Usage': 0.1076178604872873
, P-value: 0.0023332713099323663
```

In [ ]:
```
# Correlations for identity


print('smoker identity')
CalcCorrelationSpearMan(df, 'smoke_identity', 'len_intro')

print('quitter identity')
CalcCorrelationSpearMan(df, 'quit_identity', 'len_intro')

print('non smoker identity')
CalcCorrelationSpearMan(df, 'Not_smoke_identity', 'len_intro')

CalcCorrelationBivarate(df, 'quit_identity', 'Quit_Before')
```

```
smoker identity
Spearman Correlation between 'len_intro' and 'smoke_identity': -0.14265246
854004174, P-value: 5.251159299613837e-05
quitter identity
Spearman Correlation between 'len_intro' and 'quit_identity': 0.0666937485
7448357, P-value: 0.059677553196735025
non smoker identity
Spearman Correlation between 'len_intro' and 'Not_smoke_identity': 0.04802
165380003377, P-value: 0.17534937113851715
Point Biserial Correlation between 'Quit_Before' and 'quit_identity': 0.09
169118511766111
P-value: 0.009553762001461347
```

In [ ]:

```
## Other correlations

CalcCorrelationSpearMan(df,'len_intro', 'count_codes'  )
CalcCorrelationSpearMan(df, 'len_intro',  'count_sessions')
CalcCorrelationSpearMan(df, 'len_intro',  'state_importance')



CalcCorrelationSpearMan(df, 'P_effect_R_1', 'len_intro'  )
CalcCorrelationSpearMan(df, 'Mean_dropout_response',  'len_intro')


CalcCorrelationBivarate(df, 'count_sessions', 'smoker')
CalcCorrelationBivarate(df, 'Mean_Importance', 'smoker')

CalcCorrelationTheme(df,'smoker', 'Usage Patterns')
CalcCorrelationSpearMan(df,'Usage', 'smoker'  )


CalcCorrelationSpearMan(df, 'Usage', 'Age')





CalcCorrelationSpearMan(df, 'len_intro', 'Mean_HumanSupport')
CalcCorrelationSpearMan(df, 'len_intro', 'Mean_Importance')
```

```
Spearman Correlation between 'count_codes' and 'len_intro': 0.460956609505
0918, P-value: 3.1447627881896135e-43
Spearman Correlation between 'count_sessions' and 'len_intro': -0.00673910
1750425654, P-value: 0.8492496464055206
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Spearman Correlation between 'state_importance' and 'len_intro': -0.015788
666998925595, P-value: 0.656275678918101
amount of rows remove 527
Rows being removed due to NaN or Inf values:
['P1', 'P10', 'P1008', 'P1009', 'P101', 'P1010', 'P1012', 'P1017', 'P102',
'P1020', 'P1023', 'P1024', 'P1026', 'P1027', 'P1029', 'P103', 'P1030', 'P1
032', 'P1036', 'P1039', 'P104', 'P1042', 'P1043', 'P1048', 'P1049', 'P1050
', 'P1051', 'P1054', 'P1055', 'P1057', 'P1058', 'P1059', 'P106', 'P1063',
'P1065', 'P1068', 'P107', 'P1074', 'P1082', 'P1085', 'P1086', 'P1089', 'P1
097', 'P1098', 'P110', 'P1104', 'P1105', 'P111', 'P116', 'P117', 'P118', '
P119', 'P12', 'P120', 'P121', 'P122', 'P124', 'P13', 'P130', 'P131', 'P133
', 'P136', 'P140', 'P141', 'P142', 'P143', 'P144', 'P147', 'P149', 'P15',
'P150', 'P151', 'P154', 'P156', 'P158', 'P159', 'P16', 'P161', 'P162', 'P1
64', 'P165', 'P166', 'P167', 'P168', 'P170', 'P172', 'P173', 'P174', 'P177
', 'P18', 'P181', 'P183', 'P186', 'P187', 'P189', 'P19', 'P196', 'P197', '
P2', 'P20', 'P200', 'P201', 'P203', 'P204', 'P206', 'P208', 'P209', 'P21',
'P210', 'P211', 'P212', 'P213', 'P214', 'P215', 'P22', 'P221', 'P223', 'P2
24', 'P227', 'P228', 'P229', 'P230', 'P231', 'P236', 'P237', 'P238', 'P24'
, 'P240', 'P242', 'P244', 'P246', 'P252', 'P254', 'P255', 'P256', 'P261',
'P262', 'P264', 'P265', 'P267', 'P268', 'P269', 'P27', 'P271', 'P273', 'P2
76', 'P277', 'P28', 'P281', 'P283', 'P286', 'P289', 'P291', 'P292', 'P293'
, 'P295', 'P297', 'P298', 'P3', 'P302', 'P303', 'P305', 'P308', 'P309', 'P
312', 'P313', 'P314', 'P315', 'P318', 'P32', 'P321', 'P322', 'P323', 'P326
', 'P327', 'P328', 'P329', 'P33', 'P331', 'P332', 'P333', 'P336', 'P338',
'P339', 'P341', 'P343', 'P344', 'P345', 'P347', 'P35', 'P350', 'P351', 'P3
56', 'P358', 'P359', 'P360', 'P362', 'P363', 'P368', 'P370', 'P371', 'P372
', 'P375', 'P376', 'P377', 'P38', 'P382', 'P383', 'P384', 'P386', 'P387',
'P39', 'P394', 'P396', 'P398', 'P4', 'P400', 'P404', 'P406', 'P412', 'P413
', 'P415', 'P416', 'P418', 'P423', 'P425', 'P429', 'P430', 'P431', 'P433',
'P436', 'P438', 'P439', 'P44', 'P440', 'P441', 'P442', 'P444', 'P445', 'P4
47', 'P454', 'P455', 'P456', 'P457', 'P459', 'P460', 'P468', 'P469', 'P473
', 'P474', 'P477', 'P479', 'P48', 'P480', 'P482', 'P484', 'P485', 'P487',
'P489', 'P490', 'P494', 'P496', 'P497', 'P498', 'P5', 'P50', 'P506', 'P507
', 'P512', 'P514', 'P518', 'P519', 'P520', 'P521', 'P522', 'P523', 'P524',
'P525', 'P526', 'P527', 'P531', 'P532', 'P538', 'P54', 'P540', 'P541', 'P5
43', 'P544', 'P545', 'P546', 'P552', 'P555', 'P556', 'P557', 'P559', 'P560
', 'P563', 'P564', 'P566', 'P570', 'P573', 'P575', 'P577', 'P578', 'P579',
'P58', 'P582', 'P584', 'P587', 'P588', 'P589', 'P59', 'P590', 'P593', 'P59
5', 'P596', 'P597', 'P599', 'P6', 'P602', 'P605', 'P606', 'P607', 'P608',
'P61', 'P612', 'P613', 'P617', 'P619', 'P62', 'P620', 'P621', 'P622', 'P62
5', 'P631', 'P632', 'P633', 'P635', 'P637', 'P638', 'P640', 'P641', 'P642'
, 'P646', 'P650', 'P651', 'P652', 'P653', 'P654', 'P656', 'P659', 'P661',
'P662', 'P664', 'P665', 'P667', 'P668', 'P669', 'P67', 'P671', 'P676', 'P6
77', 'P679', 'P682', 'P683', 'P684', 'P685', 'P686', 'P687', 'P691', 'P695
', 'P696', 'P697', 'P698', 'P7', 'P701', 'P704', 'P706', 'P707', 'P709', '
P71', 'P713', 'P717', 'P72', 'P722', 'P724', 'P725', 'P729', 'P73', 'P730'
, 'P731', 'P735', 'P738', 'P74', 'P740', 'P745', 'P746', 'P747', 'P748', '
P75', 'P751', 'P753', 'P755', 'P757', 'P758', 'P759', 'P76', 'P760', 'P761
', 'P767', 'P768', 'P770', 'P773', 'P775', 'P776', 'P778', 'P78', 'P780',
'P782', 'P784', 'P788', 'P789', 'P79', 'P793', 'P794', 'P797', 'P8', 'P80'
, 'P804', 'P805', 'P807', 'P808', 'P81', 'P810', 'P813', 'P814', 'P815', '
P816', 'P818', 'P821', 'P825', 'P828', 'P83', 'P830', 'P831', 'P835', 'P83
7', 'P839', 'P840', 'P841', 'P843', 'P847', 'P848', 'P849', 'P85', 'P851',
```

```
'P852', 'P854', 'P857', 'P86', 'P861', 'P865', 'P866', 'P869', 'P879', 'P8
81', 'P885', 'P886', 'P889', 'P891', 'P892', 'P893', 'P896', 'P897', 'P899
', 'P90', 'P901', 'P902', 'P905', 'P908', 'P91', 'P911', 'P912', 'P916', '
P918', 'P919', 'P92', 'P920', 'P921', 'P925', 'P927', 'P928', 'P93', 'P930
', 'P931', 'P932', 'P933', 'P935', 'P936', 'P937', 'P941', 'P944', 'P946',
'P948', 'P949', 'P952', 'P954', 'P957', 'P96', 'P960', 'P962', 'P965', 'P9
7', 'P970', 'P971', 'P972', 'P973', 'P979', 'P980', 'P984', 'P988', 'P989'
, 'P99', 'P990', 'P991', 'P992', 'P997']
Spearman Correlation between 'len_intro' and 'P_effect_R_1': 0.03033336666
078116, P-value: 0.6190787073792546
amount of rows remove 118
Rows being removed due to NaN or Inf values:
['P101', 'P1023', 'P1024', 'P1032', 'P1036', 'P1048', 'P1049', 'P1050', 'P
1051', 'P1059', 'P121', 'P13', 'P130', 'P140', 'P143', 'P166', 'P170', 'P1
87', 'P19', 'P200', 'P201', 'P203', 'P224', 'P230', 'P264', 'P265', 'P267'
, 'P27', 'P273', 'P28', 'P283', 'P292', 'P293', 'P295', 'P3', 'P302', 'P30
3', 'P312', 'P32', 'P322', 'P326', 'P341', 'P347', 'P356', 'P363', 'P370',
'P371', 'P38', 'P383', 'P384', 'P386', 'P394', 'P418', 'P433', 'P439', 'P4
42', 'P445', 'P456', 'P469', 'P473', 'P477', 'P484', 'P494', 'P496', 'P519
', 'P523', 'P527', 'P538', 'P54', 'P544', 'P577', 'P578', 'P584', 'P605',
'P61', 'P612', 'P613', 'P620', 'P632', 'P653', 'P659', 'P668', 'P687', 'P7
06', 'P717', 'P72', 'P725', 'P729', 'P731', 'P738', 'P753', 'P760', 'P761'
, 'P767', 'P78', 'P808', 'P815', 'P831', 'P848', 'P85', 'P852', 'P865', 'P
891', 'P893', 'P896', 'P90', 'P905', 'P911', 'P916', 'P920', 'P928', 'P941
', 'P949', 'P960', 'P97', 'P980', 'P984', 'P988']
Spearman Correlation between 'len_intro' and 'Mean_dropout_response': 0.06
702471072194416, P-value: 0.08071727252057084
Point Biserial Correlation between 'smoker' and 'count_sessions': -0.12130
405817852347
P-value: 0.0005946947351416499
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Point Biserial Correlation between 'smoker' and 'Mean_Importance': -0.0016
947883593852852
P-value: 0.9618988850810566
Point Biserial Correlation between 'Usage Patterns' and 'smoker': -0.01337
7074013538324
P-value: 0.7059417040932633
Spearman Correlation between 'smoker' and 'Usage': -0.12202935233114587,
P-value: 0.0005508862457119108
Spearman Correlation between 'Age' and 'Usage': 0.08975353958774117, P-val
ue: 0.011193960546614123
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Spearman Correlation between 'Mean_HumanSupport' and 'len_intro': 0.097416
39916475735, P-value: 0.005915782737382775
amount of rows remove 1
Rows being removed due to NaN or Inf values:
['P386']
Spearman Correlation between 'Mean_Importance' and 'len_intro': -0.0090481
44813725541, P-value: 0.7986889724459536
```