DSC530 Week 12 Final Project

Name: Madhuri Basava

Date: 06/01/2023

Import the necessary packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_palette('Set3')
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Load the data

In [52]:
```
music_df = pd.read_csv('music.csv')

music_df.head()
```

Out[52]:

	Timestamp	Age	Primary streaming service	Hours per day	While working	Instrumentalist	Composer	Fav genre	Exploratory
0	8/27/2022 19:29	18.0	Spotify	3.0	Yes	Yes	Yes	Latin	Yes
1	8/27/2022 19:57	63.0	Pandora	1.5	Yes	No	No	Rock	Yes
2	8/27/2022 21:28	18.0	Spotify	4.0	No	No	No	Video game music	No
3	8/27/2022 21:40	61.0	YouTube Music	2.5	Yes	No	Yes	Jazz	Yes
4	8/27/2022 21:54	18.0	Spotify	4.0	Yes	No	No	R&B	Yes

5 rows × 33 columns

Display all columns

In [3]:
```
music_df.columns.sort_values()
```

Out[3]:
```
Index(['Age', 'Anxiety', 'BPM', 'Composer', 'Depression', 'Exploratory',
       'Fav genre', 'Foreign languages', 'Frequency [Classical]',
       'Frequency [Country]', 'Frequency [EDM]', 'Frequency [Folk]',
       'Frequency [Gospel]', 'Frequency [Hip hop]', 'Frequency [Jazz]',
       'Frequency [K pop]', 'Frequency [Latin]', 'Frequency [Lofi]',
       'Frequency [Metal]', 'Frequency [Pop]', 'Frequency [R&B]',
       'Frequency [Rap]', 'Frequency [Rock]', 'Frequency [Video game music]',
       'Hours per day', 'Insomnia', 'Instrumentalist', 'Music effects', 'OCD',
       'Permissions', 'Primary streaming service', 'Timestamp',
       'While working'],
      dtype='object')
```

Display the types of each column

In [4]:
```
music_df.dtypes
```

Out[4]:
```
Timestamp                      object
Age                           float64
Primary streaming service      object
Hours per day                 float64
While working                  object
Instrumentalist                object
Composer                       object
Fav genre                      object
Exploratory                    object
Foreign languages              object
BPM                           float64
Frequency [Classical]          object
Frequency [Country]            object
Frequency [EDM]                object
Frequency [Folk]               object
Frequency [Gospel]             object
Frequency [Hip hop]            object
Frequency [Jazz]               object
Frequency [K pop]              object
Frequency [Latin]              object
Frequency [Lofi]               object
Frequency [Metal]              object
Frequency [Pop]                object
Frequency [R&B]                object
Frequency [Rap]                object
Frequency [Rock]               object
Frequency [Video game music]   object
Anxiety                       float64
Depression                    float64
Insomnia                      float64
OCD                           float64
Music effects                  object
Permissions                    object
dtype: object
```

Check for missing values

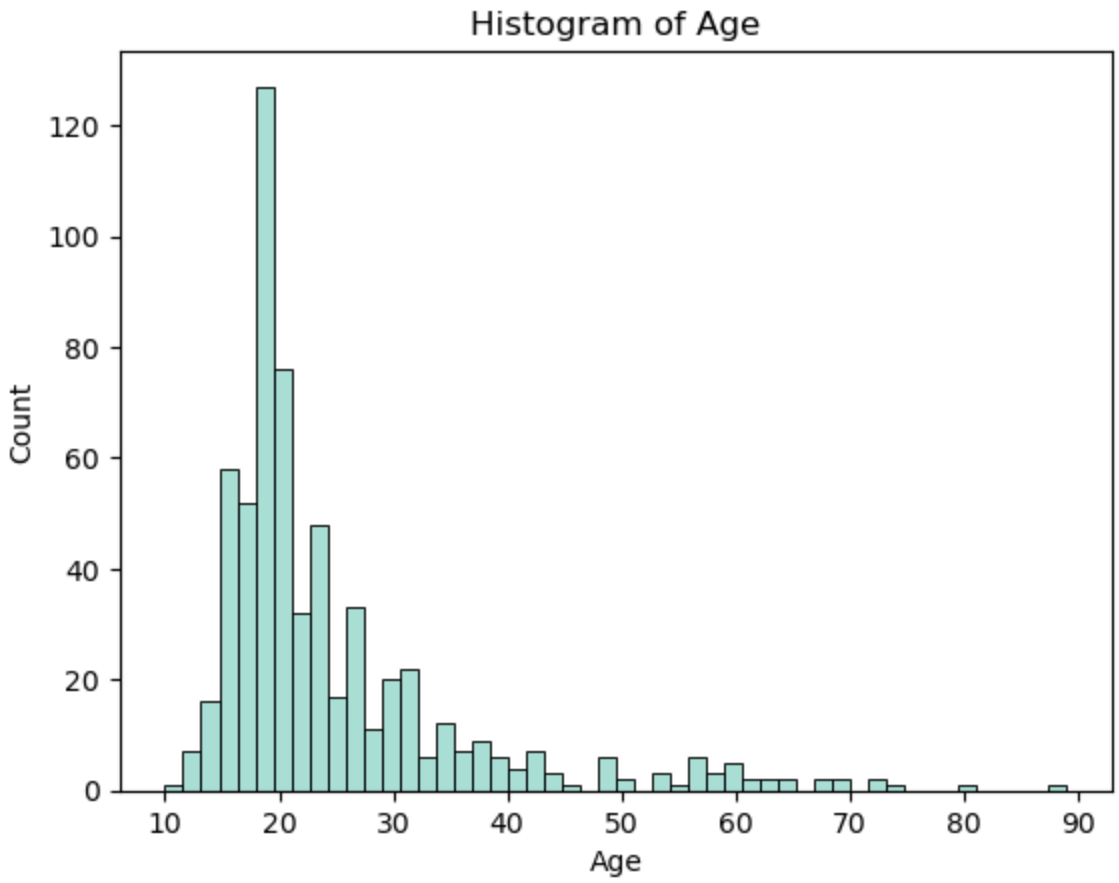
In [5]:
```
music_df.isnull().sum()
```

Out[5]:
```
Timestamp                         0
Age                               1
Primary streaming service         1
Hours per day                     0
While working                     3
Instrumentalist                   4
Composer                          1
Fav genre                         0
Exploratory                       0
Foreign languages                 4
BPM                             107
Frequency [Classical]             0
Frequency [Country]               0
Frequency [EDM]                   0
Frequency [Folk]                  0
Frequency [Gospel]                0
Frequency [Hip hop]               0
Frequency [Jazz]                  0
Frequency [K pop]                 0
Frequency [Latin]                 0
Frequency [Lofi]                  0
Frequency [Metal]                 0
Frequency [Pop]                   0
Frequency [R&B]                   0
Frequency [Rap]                   0
Frequency [Rock]                  0
Frequency [Video game music]      0
Anxiety                           0
Depression                        0
Insomnia                          0
OCD                               0
Music effects                     8
Permissions                       0
dtype: int64
```

In [6]:
```
music_df = music_df.dropna()
```

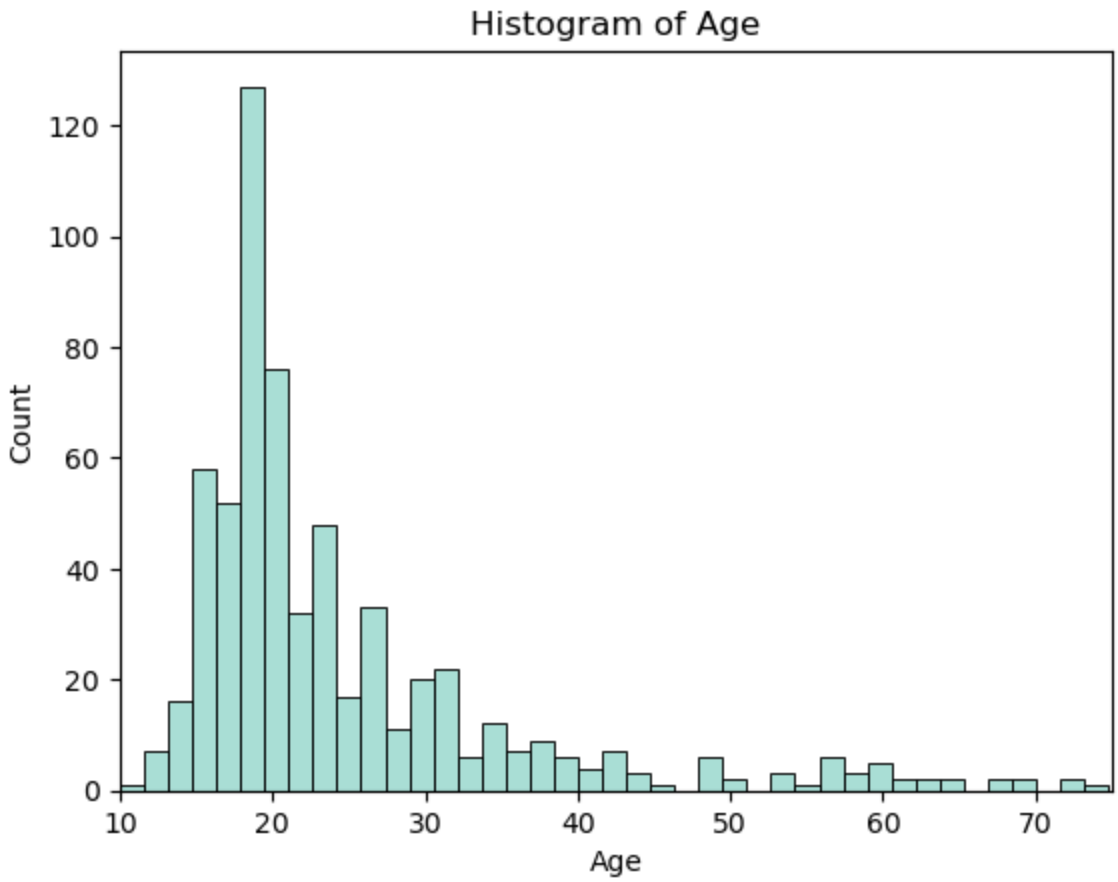
Plot the Histogram of Age before removing outlliers

In [7]:
```
g = sns.histplot(music_df['Age'], kde=False, bins=50)
g.set_title('Histogram of Age')
g.set_xlabel('Age')
g.set_ylabel('Count')
plt.show()
```



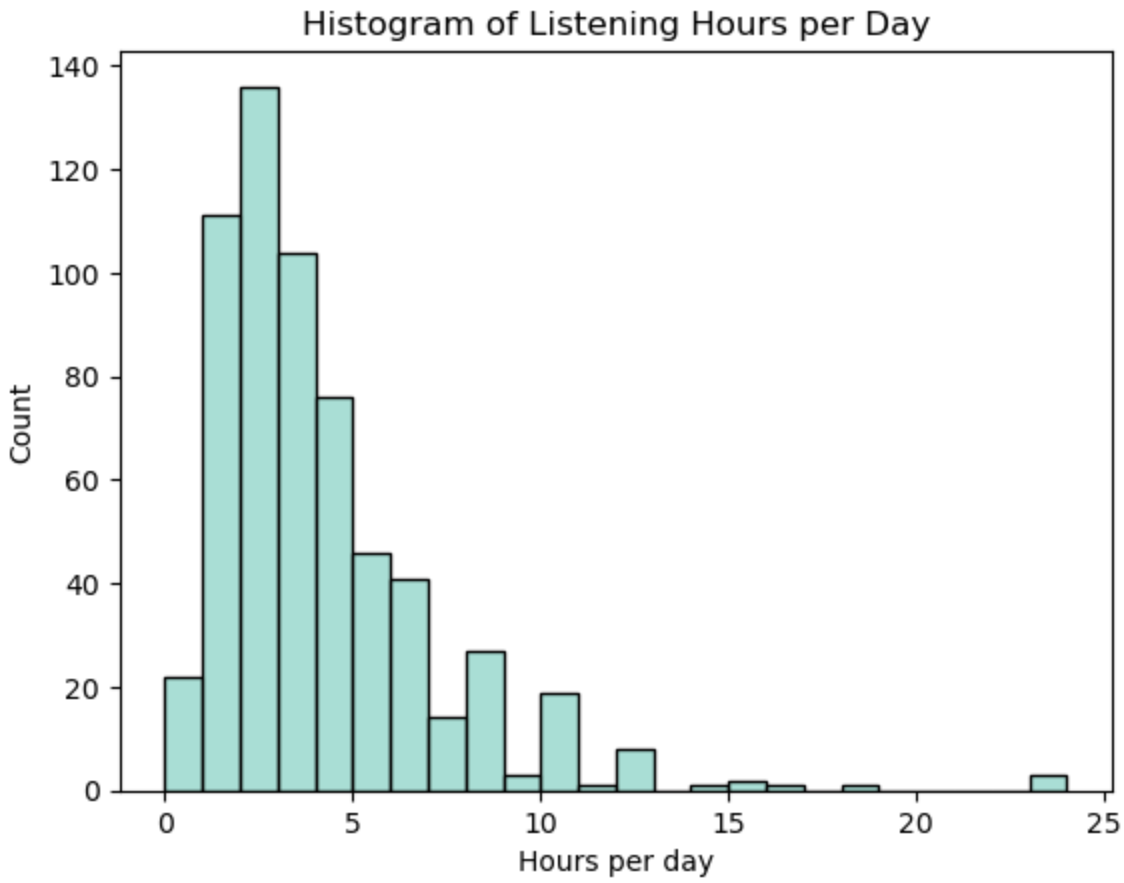
Plot the Histogram of Age after removing outliers

In [8]:
```
g = sns.histplot(music_df['Age'], kde=False, bins=50)
g.set_title('Histogram of Age')
g.set_xlabel('Age')
g.set_ylabel('Count')
plt.xlim([10,75])
plt.show()
```



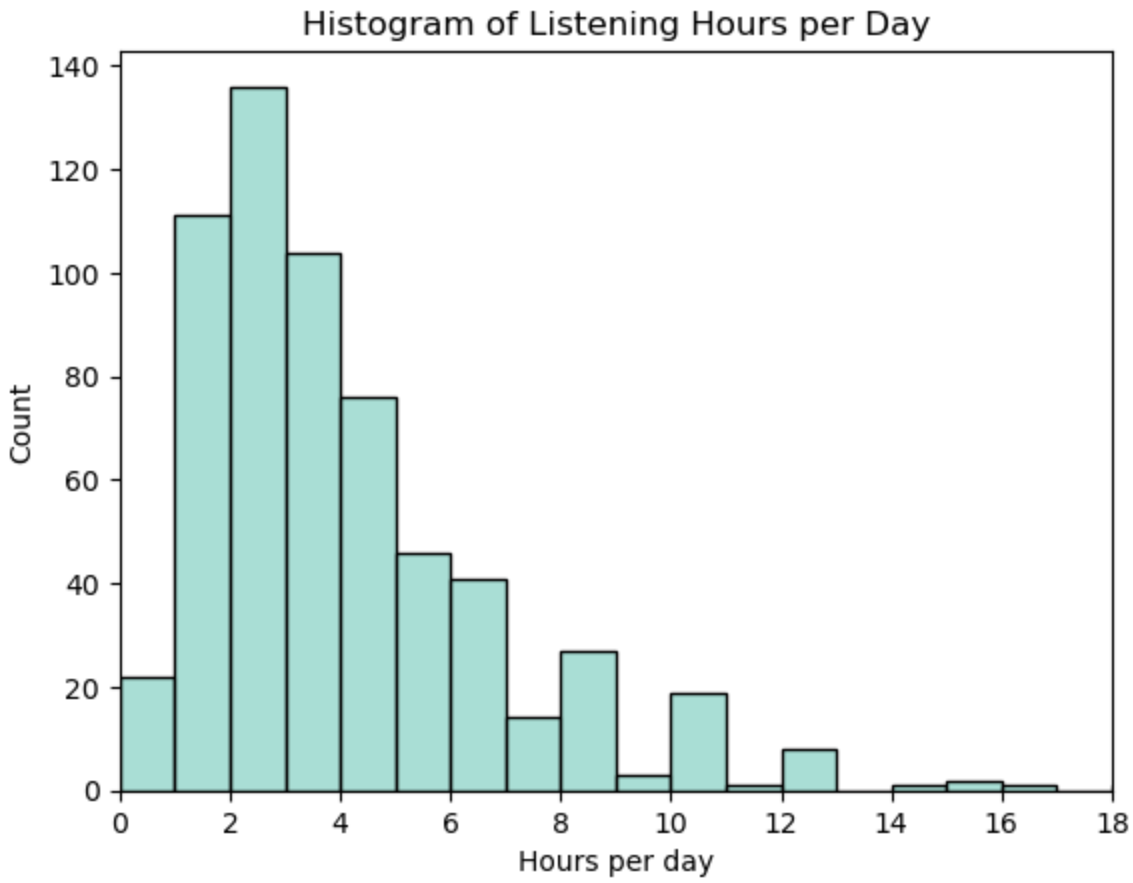
In [9]:
```
#### Histogram of listening hours per day
```

In [10]:
```
g = sns.histplot(music_df['Hours per day'], kde=False, bins=24)
g.set_title('Histogram of Listening Hours per Day')
g.set_xlabel('Hours per day')
g.set_ylabel('Count')
plt.show()
```



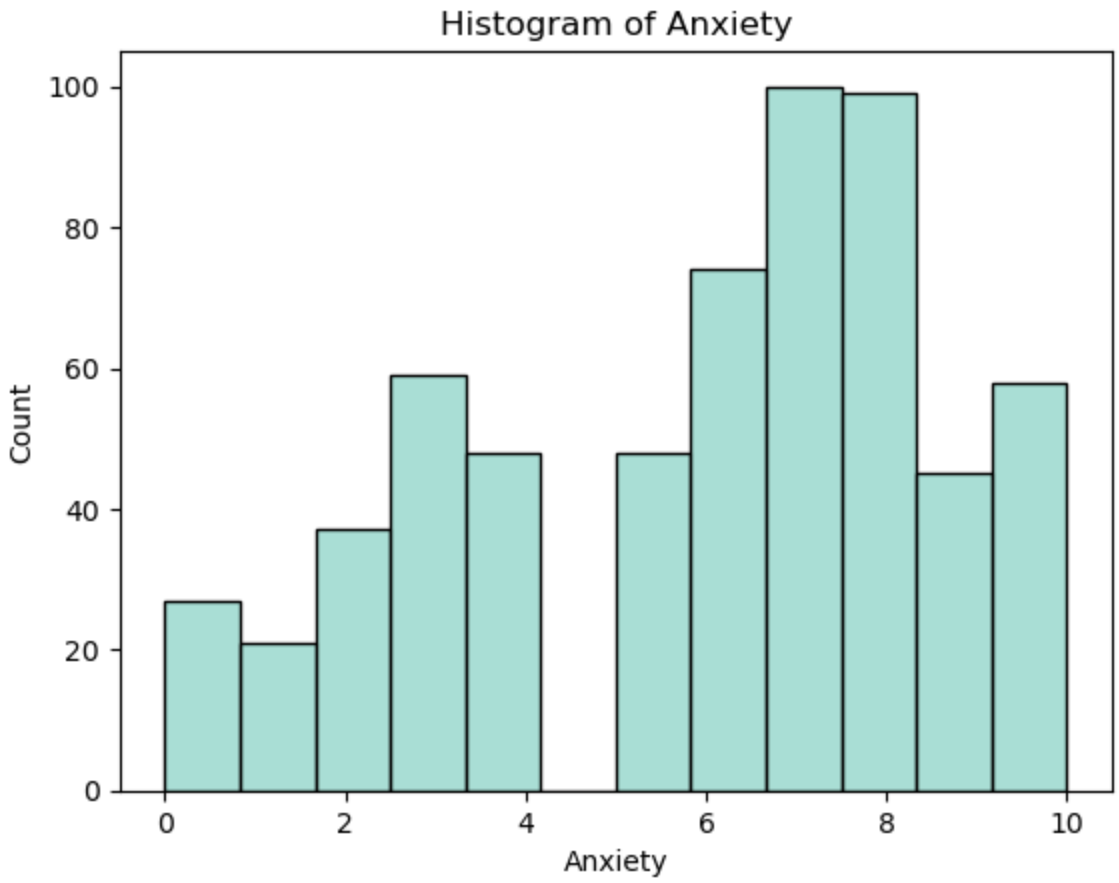
Histogram of listening hours per day after removing outlier, assuming it is not practically feasible to listen for more than 18 hours.

In [11]:
```
g = sns.histplot(music_df['Hours per day'], kde=False, bins=24)
g.set_title('Histogram of Listening Hours per Day')
g.set_xlabel('Hours per day')
g.set_ylabel('Count')
plt.xlim([0,18])
plt.show()
```



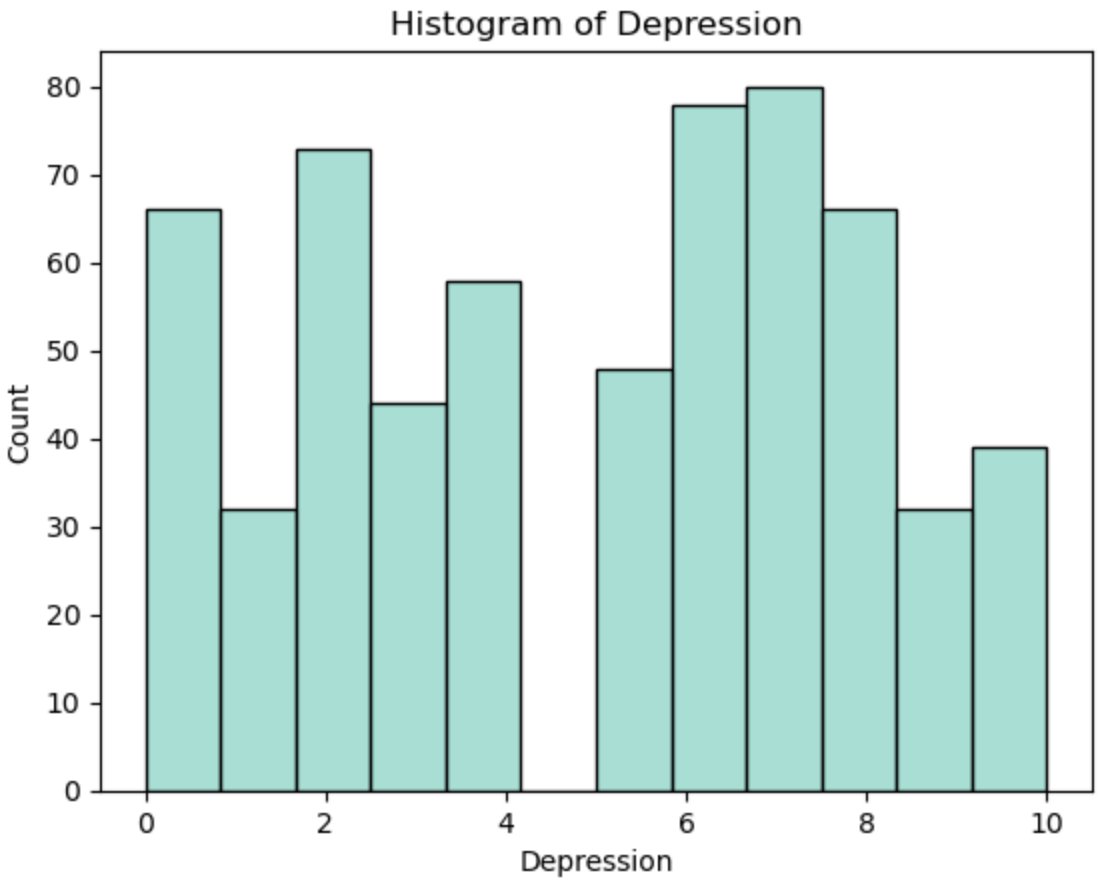
Histogram of Anxiety

In [12]:
```
g = sns.histplot(music_df['Anxiety'], kde=False, bins=12)
g.set_title('Histogram of Anxiety')
g.set_xlabel('Anxiety')
g.set_ylabel('Count')
plt.show()
```



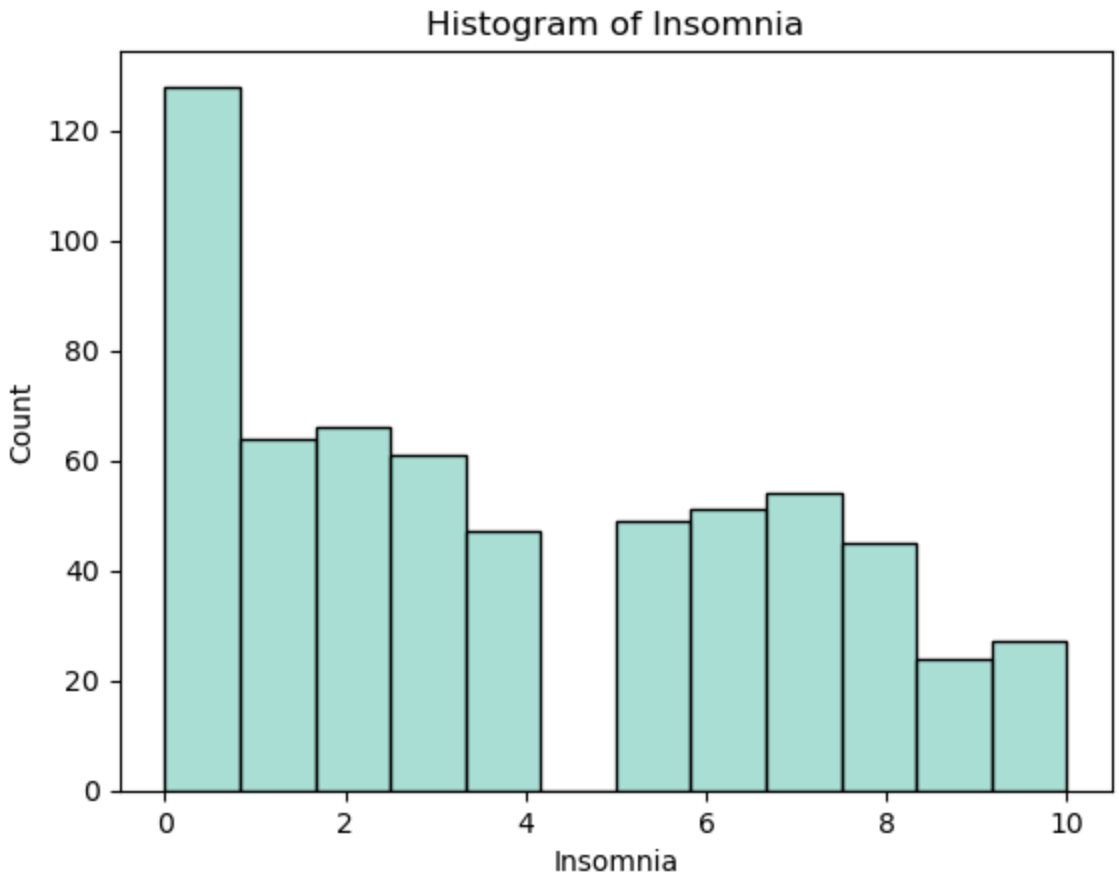
Histogram of Depression

In [13]:
```
g = sns.histplot(music_df['Depression'], kde=False, bins=12)
g.set_title('Histogram of Depression')
g.set_xlabel('Depression')
g.set_ylabel('Count')
plt.show()
```



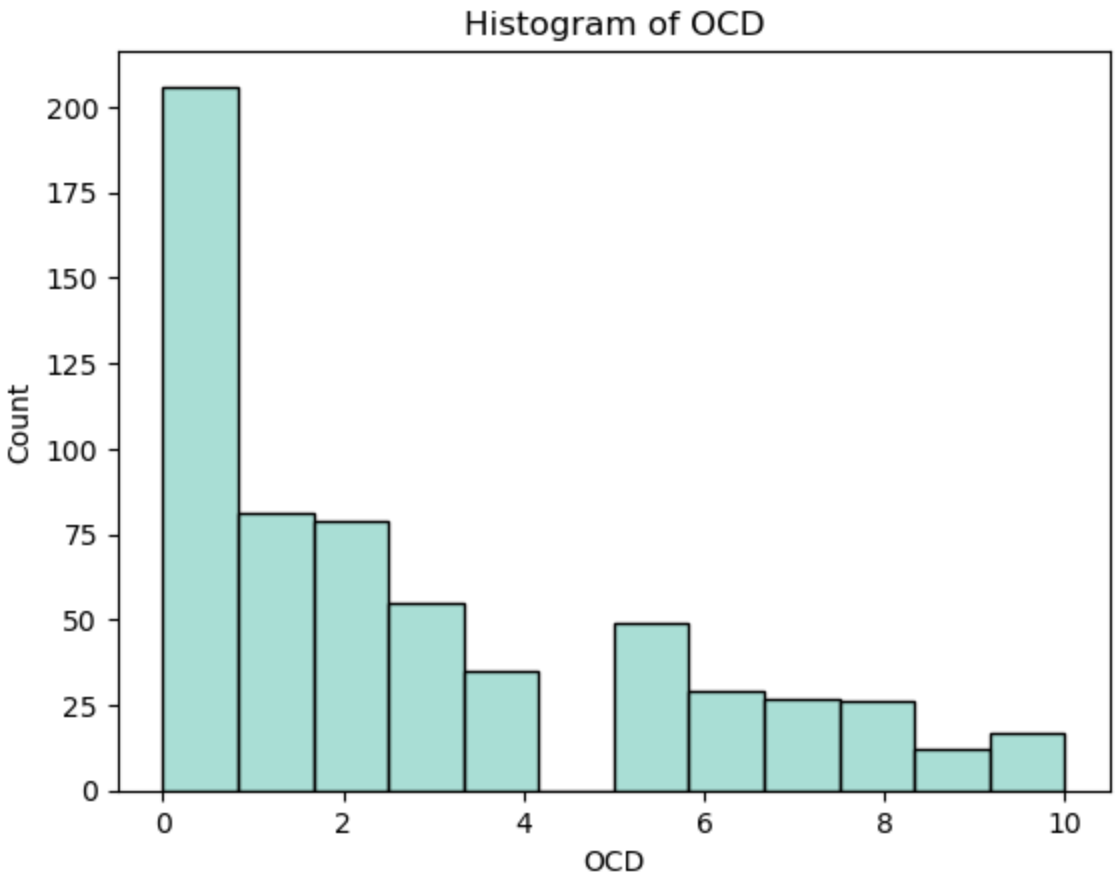
Histogram of Insomnia

In [14]:
```
g = sns.histplot(music_df['Insomnia'], kde=False, bins=12)
g.set_title('Histogram of Insomnia')
g.set_xlabel('Insomnia')
g.set_ylabel('Count')
plt.show()
```



Histogram of OCD

In [15]:
```
g = sns.histplot(music_df['OCD'], kde=False, bins=12)
g.set_title('Histogram of OCD')
g.set_xlabel('OCD')
g.set_ylabel('Count')
plt.show()
```



Include the other descriptive characteristics about the variables: Mean, Mode, Spread, and Tails

Compute the mean, mode, spread and tail for column Age

In [16]:
```
music_df['Age'].mean(), music_df['Age'].mode(), music_df['Age'].var(), music_df['Ag
```

Out[16]:
```
(24.792207792207794,
 0    18.0
 Name: Age, dtype: float64,
 135.92097983317515,
 11.658515335718144,
 731    17.0
 732    18.0
 733    19.0
 734    19.0
 735    29.0
 Name: Age, dtype: float64)
```

From the above values, we can conclude that the average age of the individuals who listen to music is 25 years. Mode is 18 years, that means many people of age 18 listens to music. Variance or the spread is around 11.66 standard deviations. Above tails shows how quickly the probability drop offs as we move away from the mode

Compute the mean, mode, spread and tail for column 'Hours per day'

In [17]:
```
music_df['Hours per day'].mean(), music_df['Hours per day'].mode(), music_df['Hours
```

Out[17]:
```
(3.702435064935065,
 0    2.0
 Name: Hours per day, dtype: float64,
 9.436945280329411,
 3.0719611456412355,
 731    2.0
 732    1.0
 733    6.0
 734    5.0
 735    2.0
 Name: Hours per day, dtype: float64)
```

From the above values, we can conclude that on an average, the individuals listen to music for 3.7 hours per day. Mode is 2 hours. Variance or the spread is around 3 standard deviations.

Comparing two scenarios in the data using a PMF by splitting the music dataframe using Age column

In [18]:
```
age_below_40 = music_df[music_df.Age < 40]
others = music_df[music_df.Age >= 40]
```

Download the necessary packages

In [19]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py")
```

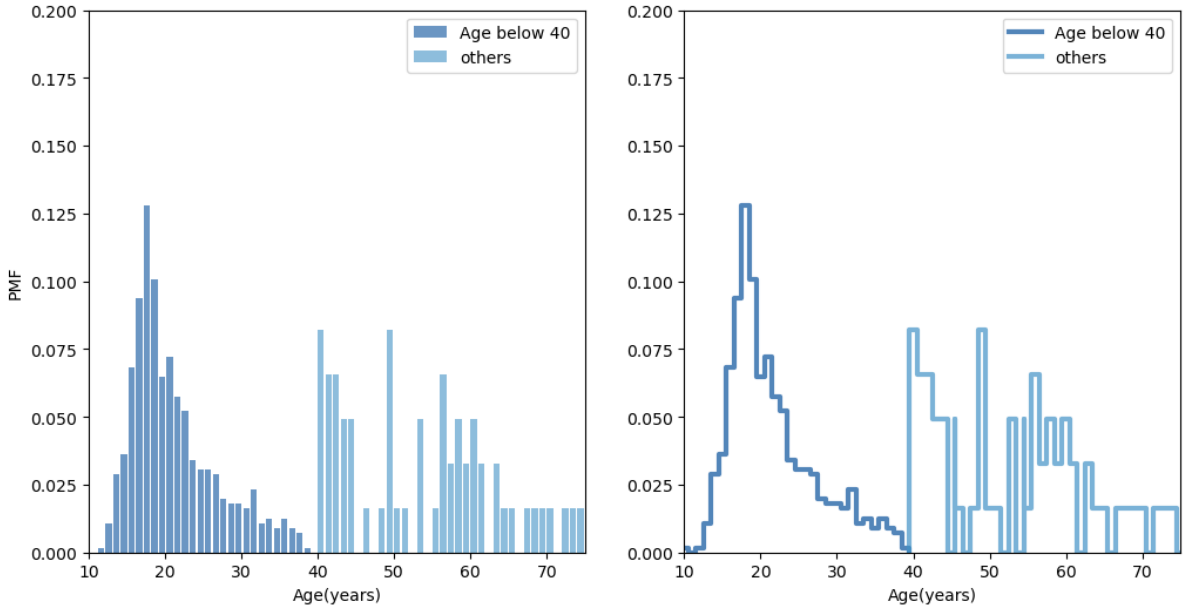
Import the necessary packages

In [20]:
```
import thinkstats2
import thinkplot
```

In [21]:
```
age_below_40_pmf = thinkstats2.Pmf(age_below_40.Age, label="Age below 40")
other_pmf = thinkstats2.Pmf(others.Age, label="others")
```

In [22]:
```
width = 0.8
axis = [10, 75, 0, 0.2]
thinkplot.PrePlot(2, cols=2)
```

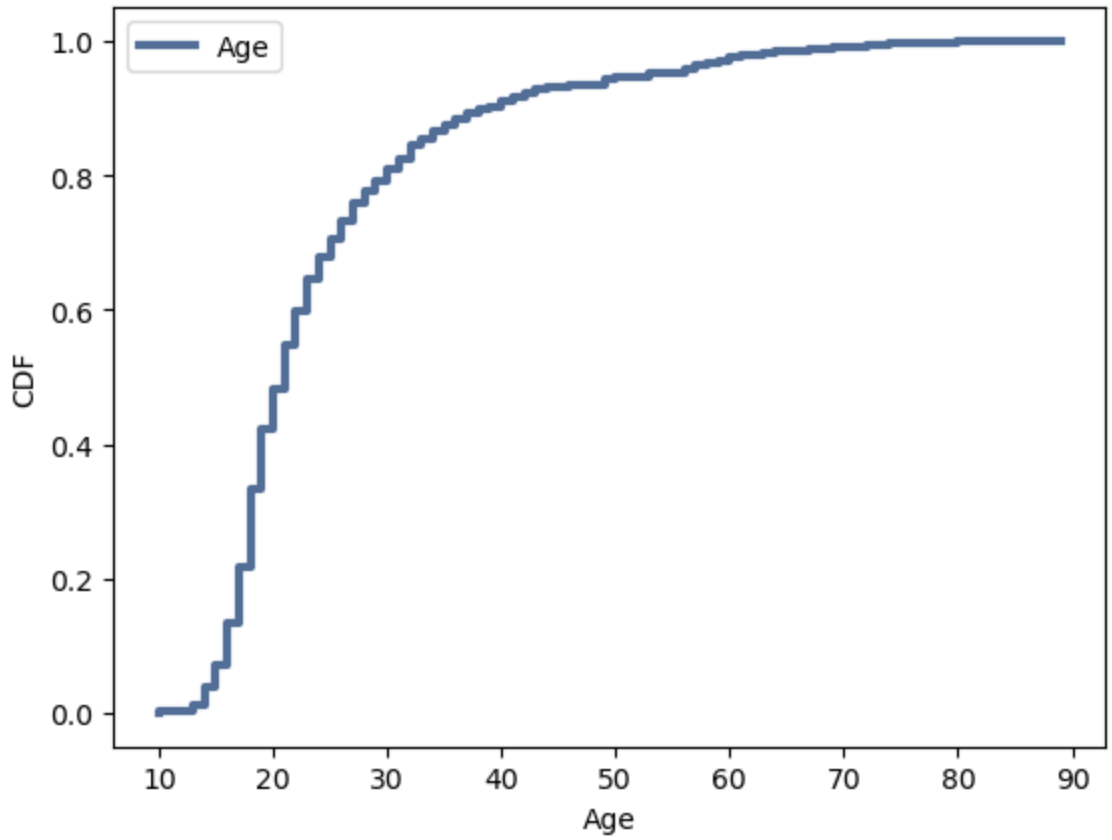
```
thinkplot.Hist(age_below_40_pmf, align="right", width=width)
thinkplot.Hist(other_pmf, align="left", width=width)
thinkplot.Config(xlabel="Age(years)", ylabel="PMF", axis=axis)

thinkplot.PrePlot(2)
thinkplot.SubPlot(2)
thinkplot.Pmfs([age_below_40_pmf, other_pmf])
thinkplot.Config(xlabel="Age(years)", axis=axis)
```



From the above plots we can conclude that the Probability mass Function among teenagers are more when compared to pmf of others.

Creating 1 CDF with Age variable

In [23]:
```
#### music_df['Music effects'] = music_df['Music effects'].map({'Improve': 1, 'No e
cdf = thinkstats2.Cdf(music_df['Age'], label='Age')
thinkplot.Cdf(cdf)
thinkplot.Show(xlabel='Age', ylabel='CDF')
```

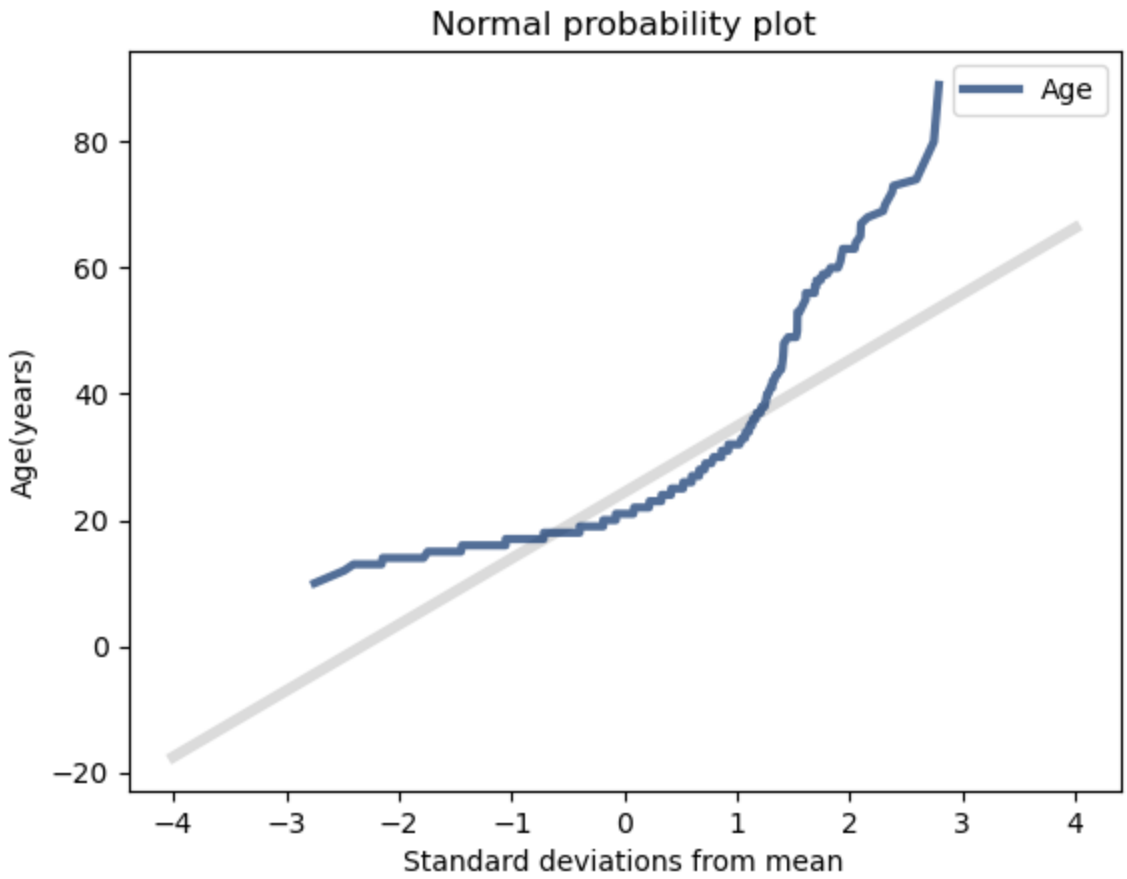



```
<Figure size 800x600 with 0 Axes>
```

As we analyze the above plot with CDF, It looks like about 80% of people are around 25years. Here the mode is 18. There are few values below 13 years. So the CDF in this range is flat. From this we can conclude that people who are around 13 o 25 are listening more music.

Plotting analytical distribution using Normal Distribution

In [24]:
```
mean, var = thinkstats2.TrimmedMeanVar(music_df['Age'], p=0.01)
std = np.sqrt(var)

xs = [-4, 4]
fxs, fys = thinkstats2.FitLine(xs, mean, std)
thinkplot.Plot(fxs, fys, linewidth=4, color="0.8")

xs, ys = thinkstats2.NormalProbability(music_df['Age'])
thinkplot.Plot(xs, ys, label="Age")

thinkplot.Config(
    title="Normal probability plot",
    xlabel="Standard deviations from mean",
    ylabel="Age(years)",
)
```

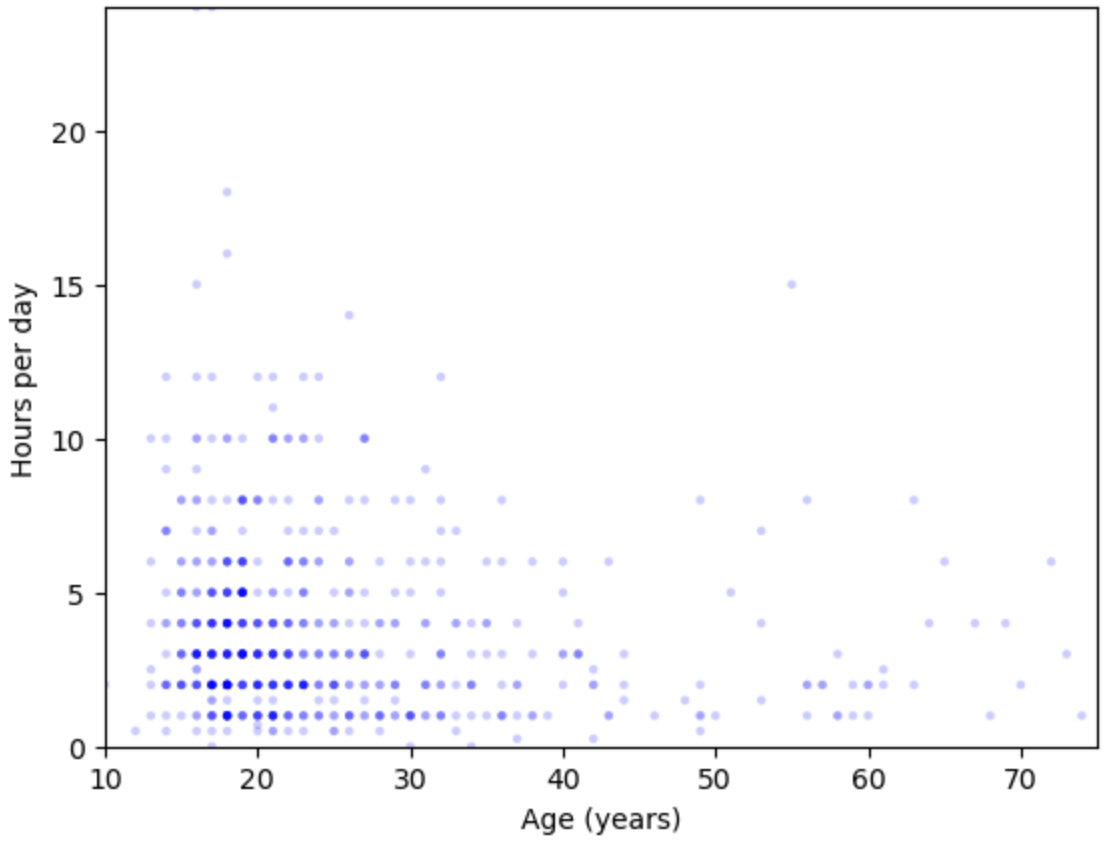


When we generate a normal probability plot with Age, we can see clearly that the data deviate from the model systematically.

Create two scatter plots comparing two variables and provide your analysis on correlation and causation. Remember, covariance, Pearson's correlation, and Non-Linear Relationships should also be considered during your analysis

In [25]:
```
#### Plot the scatter plot of ages vs 'Hours per day'
```

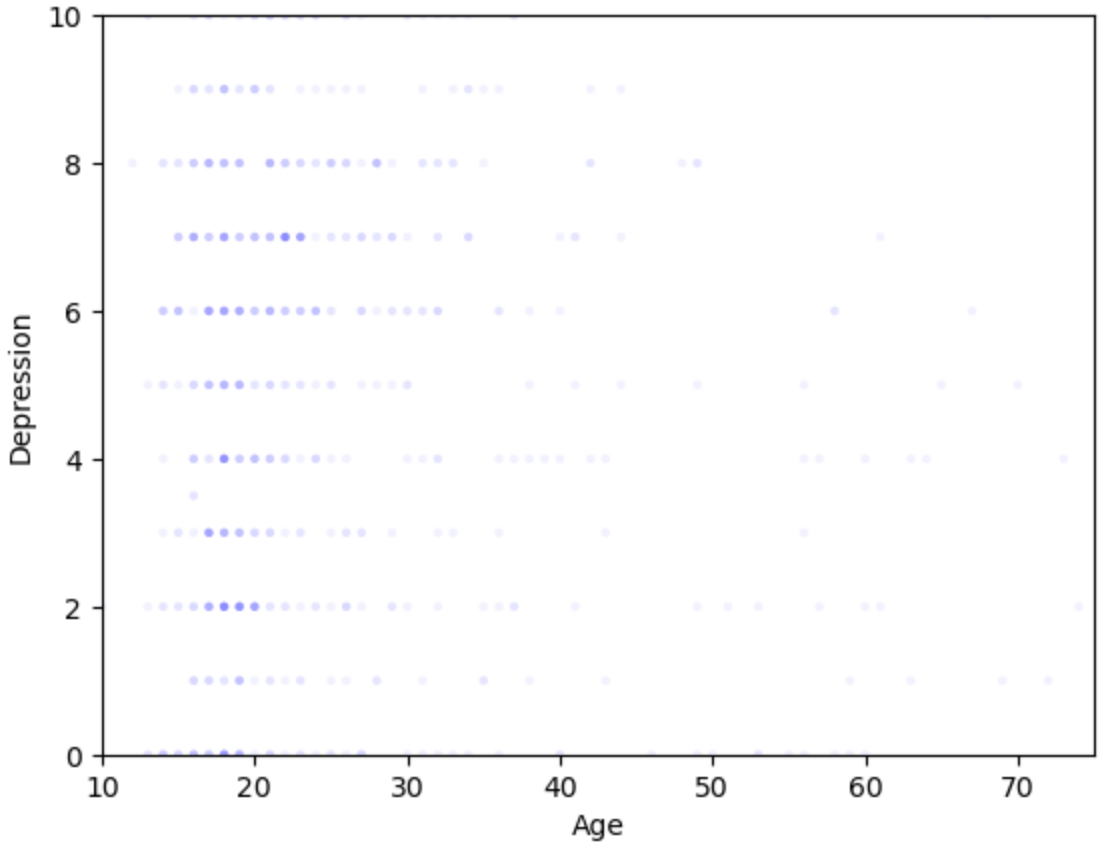
In [26]:
```
ages = music_df['Age']
hours_per_day = music_df['Hours per day']
thinkplot.Scatter(ages, hours_per_day, alpha=0.2, s=10)
thinkplot.Config(xlabel='Age (years)',
                     ylabel='Hours per day',
                     xlim=[10, 75],
                     ylim=[0, 24],
                     legend=False)
```



Most individuals from ages 13 to 25 listen to music around 2-3 hours a day

Plot the scatter plot of ages vs depression

In [27]:
```
age = music_df['Age']
depression = music_df['Depression']
thinkplot.Scatter(age, depression, alpha=0.05, s=10)
thinkplot.Config(xlabel='Age',
                     ylabel='Depression',
                     xlim=[10, 75],
                     ylim=[0, 10],
                     legend=False)
```



Mostly the ages from 13 to 25 have more depression

From the above scatter plots we can infer that the individuals from 13 to 25 years are listening to music for 2-3 hours and the depression rate among them is more.

Analysis on correlation and causation

In [28]:
```
thinkstats2.Corr(music_df.Age, music_df['Hours per day']),thinkstats2.Cov(music_df.
```

Out[28]:
```
(-0.04491700027620318, -1.606068687805702)
```

Since the correlation between age and hours per day is negative, It is inversely propertional.We cannot say that as age increases, it is the cause for decrease in 'Hours per day'. They may have different factors causing it.

In [29]:
```
thinkstats2.Corr(music_df.Anxiety, music_df['Depression']),thinkstats2.Cov(music_df
```

Out[29]:
```
(0.5279501305193643, 4.358693761595547)
```

In [30]:
```
#### Since the correlation between Anxiety and depression is positive. They are dir
```

Now we can compute how music affects mental health using correlation matrix.

In [31]:
```
features_sel = ['Age', 'Hours per day', 'While working', 'Exploratory', 'BPM',
                'Anxiety', 'Depression', 'Insomnia', 'OCD', 'Music effects']

freq2num = {'Never': 0, 'Rarely': 1, 'Sometimes': 2, 'Very frequently': 3}
df_genre = music_df[[col for col in music_df.columns if col.startswith('Frequency')]
df_genre.replace(freq2num, inplace=True)

df_sel = pd.concat([music_df[features_sel], df_genre], axis=1)

df_sel.head()
```

```
C:\Users\madhu\AppData\Local\Temp\ipykernel_35760\4017891561.py:6: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_genre.replace(freq2num, inplace=True)
```

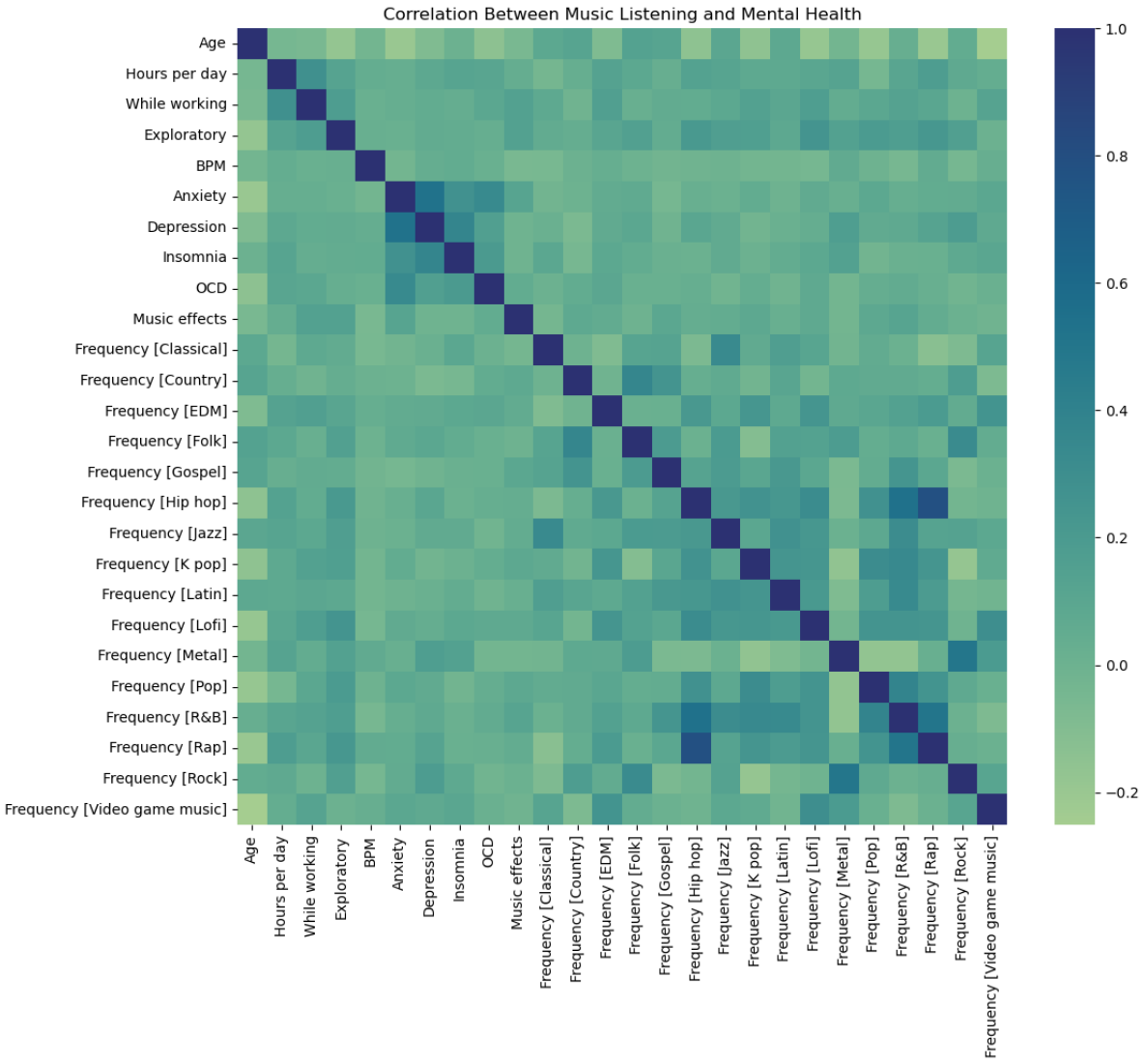
Out[31]:

	Age	Hours per day	While working	Exploratory	BPM	Anxiety	Depression	Insomnia	OCD	Music effects	...	F
2	18.0	4.0	No	No	132.0	7.0	7.0	10.0	2.0	No effect	...	
3	61.0	2.5	Yes	Yes	84.0	9.0	7.0	3.0	3.0	Improve	...	
4	18.0	4.0	Yes	Yes	107.0	7.0	2.0	5.0	9.0	Improve	...	
5	18.0	5.0	Yes	Yes	86.0	8.0	8.0	7.0	7.0	Improve	...	
6	18.0	3.0	Yes	Yes	66.0	4.0	8.0	6.0	0.0	Improve	...	

5 rows × 26 columns

In [32]:
```
# Change the categorical variables into numerical values
df_sel['While working'] = df_sel['While working'].map({'Yes': 1, 'No': 0})
df_sel['Exploratory'] = df_sel['Exploratory'].map({'Yes': 1, 'No': 0})
df_sel['Music effects'] = df_sel['Music effects'].map({'Improve': 1, 'No effect': 0
```

In [33]:
```
# plotting the correlation matrix heatmap

plt.figure(figsize=(12, 10))
sns.heatmap(df_sel.corr(), annot=False, cmap='crest')
plt.title('Correlation Between Music Listening and Mental Health')
plt.show()
```



From the above heatmap, we cannot conclude on what features contributes to improving mental health (this is evaluated by the listeners). But looks like there are interesting correlations between different music genres such as listeners who listen to hip-hop are more likely interested in listening to rap and R&B, and metal listener more likely want to listen to rock. Also, it looks like depression and anxiety are correlated it makes sense as they both are mental health issues.

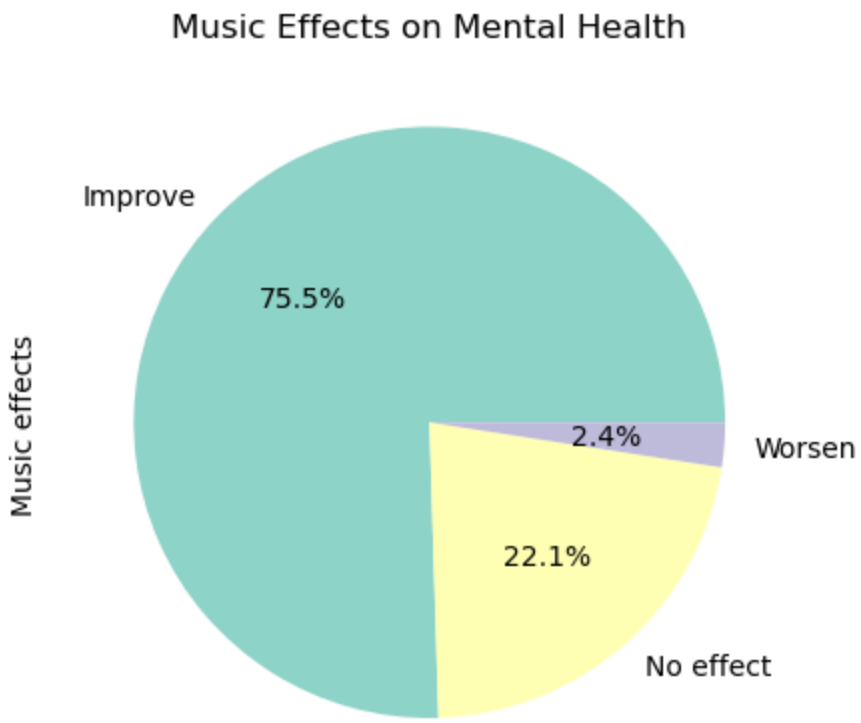
Conduct a hypothesis test on how music affects mental health

Hypothesis: Let us suppose that Mental Health can be improved by listening to Music.

We will consider various ways to prove or disprove the above hypothesis

In [34]:
```
# plot the pie chart of music effects
music_df['Music effects'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title('Music Effects on Mental Health')
```

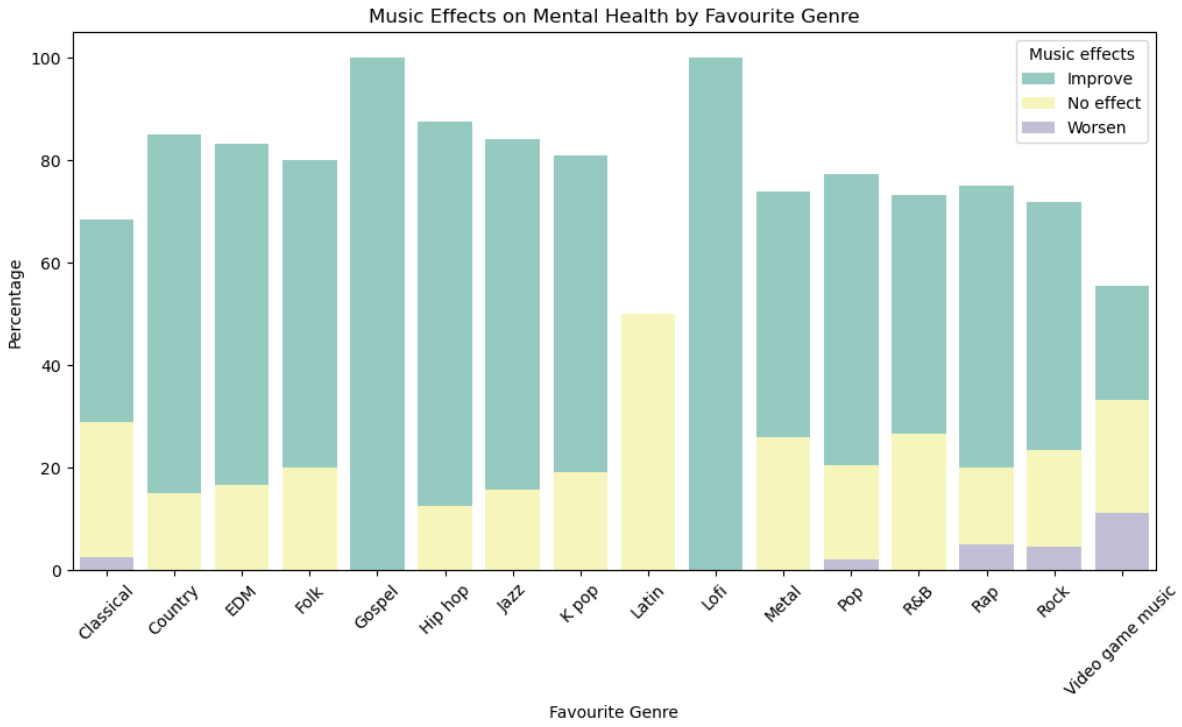
Out[34]: Text(0.5, 1.0, 'Music Effects on Mental Health')



From the above pie chart, we can infer that 75.5% of individuals that listening to music improves Mental Health. 22.1% think music has no effect on mental health and only a negligible 2.4% of people think that music worsens mental health

Which genre has the highest ratio of listeners who believe their fav genre has a positive effect on their mental health?

In [35]:
```
df_fav_genre = music_df[['Fav genre', 'Music effects']]

# normalize the counts
df_fav_genre = df_fav_genre.groupby(['Fav genre', 'Music effects']).size().groupby(

# plot the bar chart
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x='Fav genre', y='Percentage', hue='Music effects', data=df_fav_genre,
ax.set_title('Music Effects on Mental Health by Favourite Genre')
ax.set_xlabel('Favourite Genre')
# tilt x-axis labels
ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
ax.set_ylabel('Percentage')
plt.show()
```



Based on the above plots, It looks like the majority of the listeners think music improves their mental health.

Define the class CorrelationPermute

In [36]:
```
class CorrelationPermute(thinkstats2.HypothesisTest):

    def TestStatistic(self, data):
        xs, ys = data
        test_stat = abs(thinkstats2.Corr(xs, ys))
        return test_stat

    def RunModel(self):
        xs, ys = self.data
        xs = np.random.permutation(xs)
        return xs, ys
```

Define the RunTest() function

In [37]:
```
 # test correlation
def RunTest(live, iters=1000):
    music_df2 = music_df.dropna(subset=['Hours per day', 'Age'])
    data = music_df2['Hours per day'].values, music_df2.Age.values
    ht = CorrelationPermute(data)
    pvalue = ht.PValue(iters=iters)
    print('%d\t%0.2f'% (n, pvalue))
```

Run test

In [38]:
```
n = len(music_df)
for _ in range(7):
    sample = thinkstats2.SampleRows(music_df, n)
    RunTest(sample)
    n //= 2
```

```
616     0.26
308     0.25
154     0.26
77      0.27
38      0.26
19      0.28
9       0.27
```

From the above test, since the p-value is greater than 0.05, It is said to "failed to reject the null hypothesis".That means the null hypothesis is true i.e., mental health can be improved by listening to more music.

Linear Regression

Import warnings filter and ignore all the future warning

In [39]:
```
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
```

In [40]:
```
# Import the necessary libraries
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [41]:
```
#Load the data

features_selection = ['Age', 'Hours per day', 'Anxiety', 'Depression', 'Insomnia',
X =  music_df[features_selection]
music_df['Music effects'] = music_df['Music effects'].map({'Improve': 1, 'No effect
y = music_df['Music effects']
```

In [42]:
```
# Fit a linaer regression model
model = LinearRegression()
model.fit(X, y)
```

Out[42]:
```
LinearRegression()
```

In [43]:
```
# Print the coefficients
print(model.coef_) # slope of the line
print(model.intercept_) # y-intercept of the line
```

```
[-0.0013565   0.00717211  0.03040737 -0.01600524 -0.00400204  0.00101048]
0.6495436852732815
```

In [44]:
```
#Make predictions
X_new = [[22,10,5,5,5,5]] # Age:22, listen to music: 10 hours per day,Anxiety: 5, D
```

In [45]:
```
y_pred = model.predict(X_new) # predicted Music effects
print(y_pred)
```

```
[0.74847469]
C:\Users\madhu\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

From the above analysis, we can conclude that based on the linear regression, for the individual age 22, listen to music: 10 hours per day,Anxiety: 5, Depression: 5, Insomnia:5, OCD: 5, if the music listening is increased from an average of 2 hours per day to 10 hours per day, mental health improves from 0.65 to 0.75

Another way to perform linear regression model by considering only 2 variables 'hours per day' and age between 18 to 75 years.

Import formula.api from statsmodels package

In [46]:
```
import statsmodels.formula.api as smf
```

In [47]:
```
music_df['Music effects'].info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 616 entries, 2 to 735
Series name: Music effects
Non-Null Count  Dtype
--------------  -----
616 non-null    int64
dtypes: int64(1)
memory usage: 25.8 KB
```

In [48]:
```
music_effects = music_df['Music effects']
hours_per_day = music_df['Hours per day']
age = music_df['Age']
model = smf.ols('music_effects ~ (hours_per_day <= 18) + (age <= 75)', data = music
results = model.fit()
results.summary()
```

Out[48]:

OLS Regression Results

Dep. Variable:	music_effects	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.003
Method:	Least Squares	**F-statistic:**	0.2173
Date:	Tue, 30 May 2023	**Prob (F-statistic):**	0.805
Time:	01:25:03	**Log-Likelihood:**	-441.35
No. Observations:	616	**AIC:**	888.7
Df Residuals:	613	**BIC:**	902.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5075	0.385	1.319	0.188	-0.248	1.263
hours_per_day <= 18[T.True]	-0.0150	0.314	-0.048	0.962	-0.633	0.603
age <= 75[T.True]	0.2387	0.385	0.620	0.535	-0.517	0.994

Omnibus:	174.382	**Durbin-Watson:**	2.046
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	348.667
Skew:	-1.624	**Prob(JB):**	1.94e-76
Kurtosis:	4.742	**Cond. No.**	42.9

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [49]:
```
columns = ['hours_per_day', 'age']
new = pd.DataFrame([[10, 22]], columns=columns)  # 'Hours per day: 10, age: 22'
results.predict(new)
```

Out[49]:
```
0    0.731221
dtype: float64
```

The mental health improved from 0.65 to 0.73 when considering only two variables 'Hours per day' less than or equal to 18 and 'age' less than or equal to 75 years