In [1]:
```python
# importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.impute import SimpleImputer
```

In [2]:
```python
# Read datasets
df1 = pd.read_csv('mental-and-substance-use-as-share-of-disease.csv')
df2 = pd.read_csv('prevalence-by-mental-and-substance-use-disorder.csv')
```

In [3]:
```python
df1.head()
```

Out[3]:

	Entity	Code	Year	DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)
0	Afghanistan	AFG	1990	1.696670
1	Afghanistan	AFG	1991	1.734281
2	Afghanistan	AFG	1992	1.791189
3	Afghanistan	AFG	1993	1.776779
4	Afghanistan	AFG	1994	1.712986

In [4]:
```python
df2.head()
```

Out[4]:

	Entity	Code	Year	Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
0	Afghanistan	AFG	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036
1	Afghanistan	AFG	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250
2	Afghanistan	AFG	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501
3	Afghanistan	AFG	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958
4	Afghanistan	AFG	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779

In [5]:
```python
df2.describe()
```

Out[5]:

	Year	Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
count	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000
mean	2004.500000	0.281167	0.673891	0.211062	4.327525	0.746708	3.950449	1.578807
std	8.656074	0.047561	0.258594	0.152559	1.177961	0.463026	0.921021	0.934655
min	1990.000000	0.191621	0.189344	0.045425	1.974823	0.225471	1.640902	0.319900
25%	1997.000000	0.255468	0.539791	0.099857	3.567064	0.423502	3.258977	0.732826
50%	2004.500000	0.287456	0.591893	0.154143	4.094443	0.646050	3.904117	1.460045
75%	2012.000000	0.304760	0.897248	0.276891	4.797286	0.890013	4.550505	2.261262
max	2019.000000	0.506018	1.676204	1.136541	9.015948	3.699504	7.688213	4.698694

In [6]:
```python
df1.shape
df2.shape
```

Out[6]:
```
(6840, 10)
```

In [7]:
```python
# Merge datasets
data = pd.merge(df1, df2, on=['Entity', 'Code', 'Year'], how='inner')
```

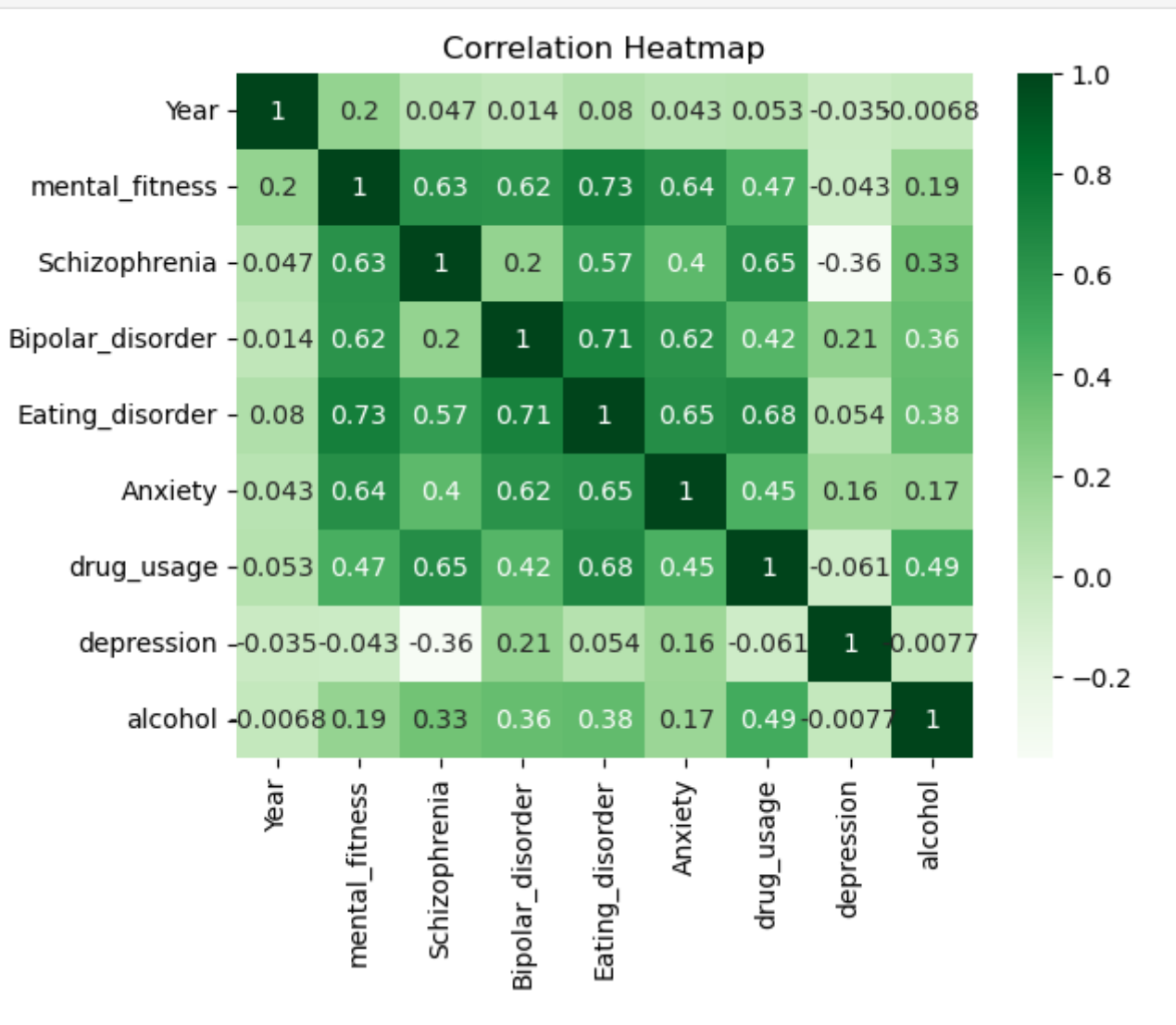
In [8]:
```python
# Data cleaning and preprocessing
data.drop(['Entity'], axis=1, inplace=True)
data.rename(columns={'DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)': 'mental_fitness',
                     'Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)': 'Schizophrenia',
                     'Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)': 'Bipolar_disorder',
                     'Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)': 'Eating_disorder',
                     'Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)': 'Anxiety',
                     'Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)': 'drug_usage',
                     'Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)': 'depression',
                     'Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)': 'alcohol'}, inplace=True)
```

In [9]:
```python
# Convert non-numeric values to NaN
data = data.apply(pd.to_numeric, errors='coerce')
```

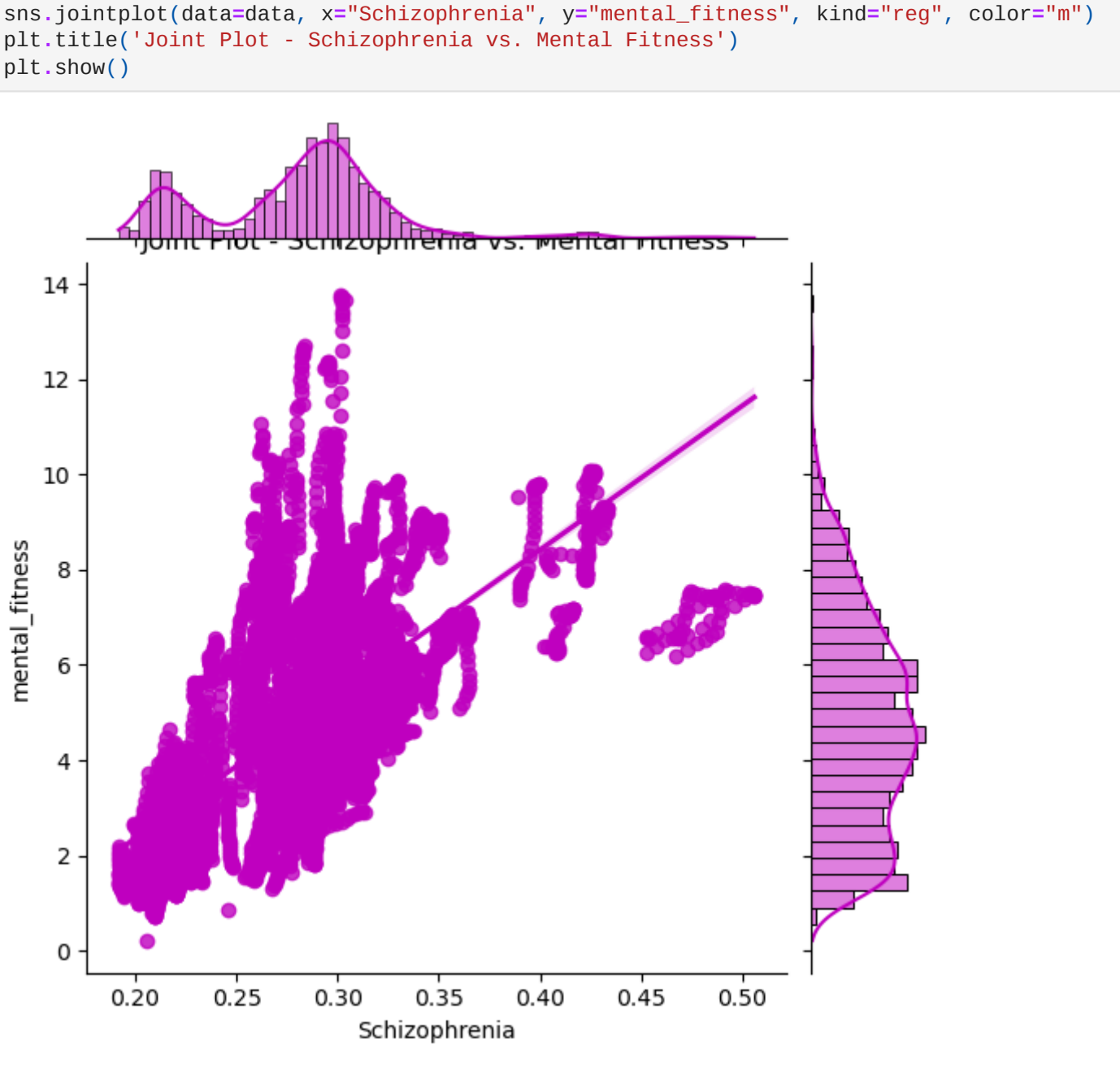
In [10]:
```python
# Drop 'Code' column
data.drop(['Code'], axis=1, inplace=True)
```

In [11]:
```python
# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean')
data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

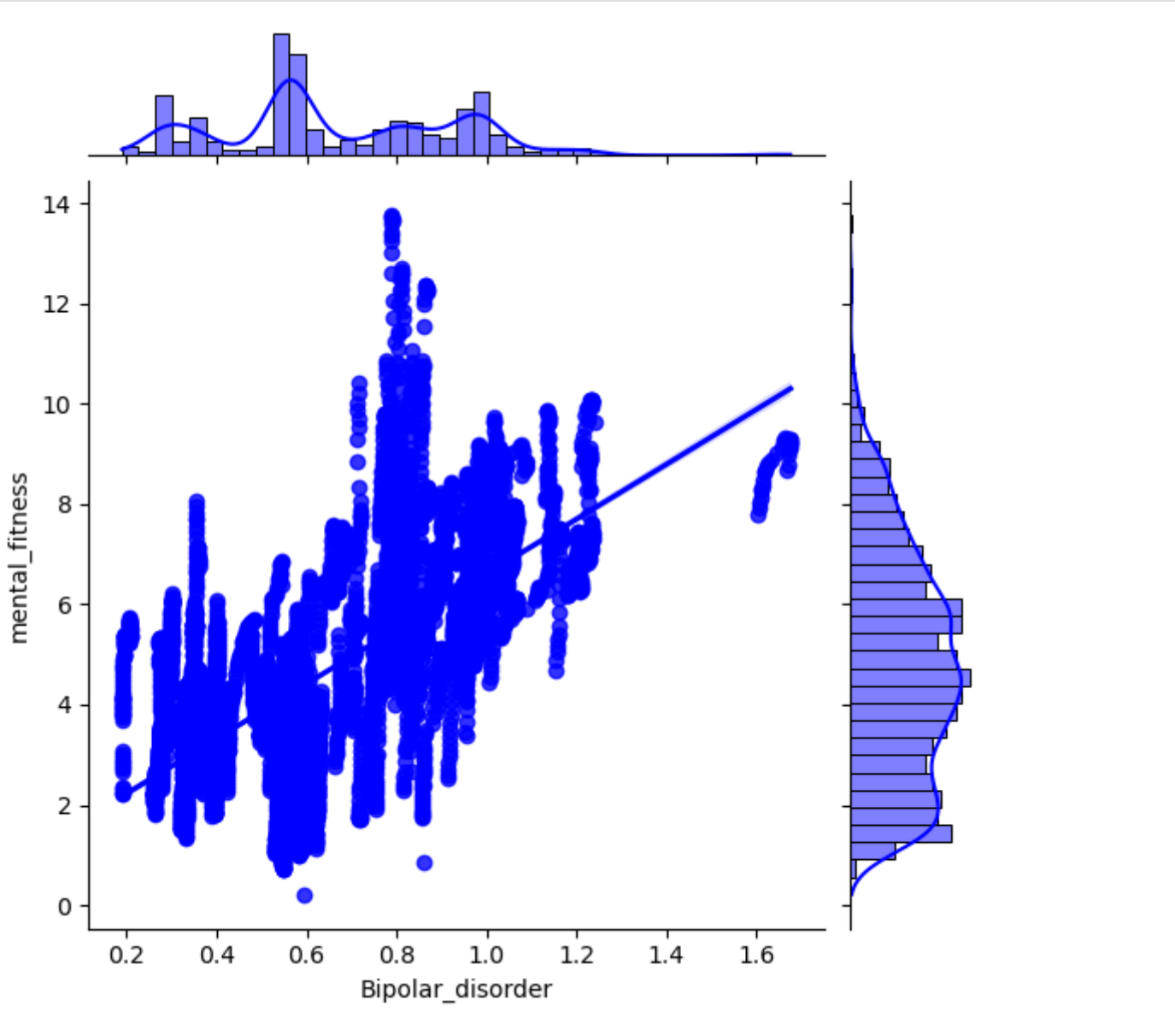
In [12]:
```python
# Exploratory analysis
# Correlation Heatmap
sns.heatmap(data.corr(), annot=True, cmap='Greens')
plt.title('Correlation Heatmap')
plt.show()
```



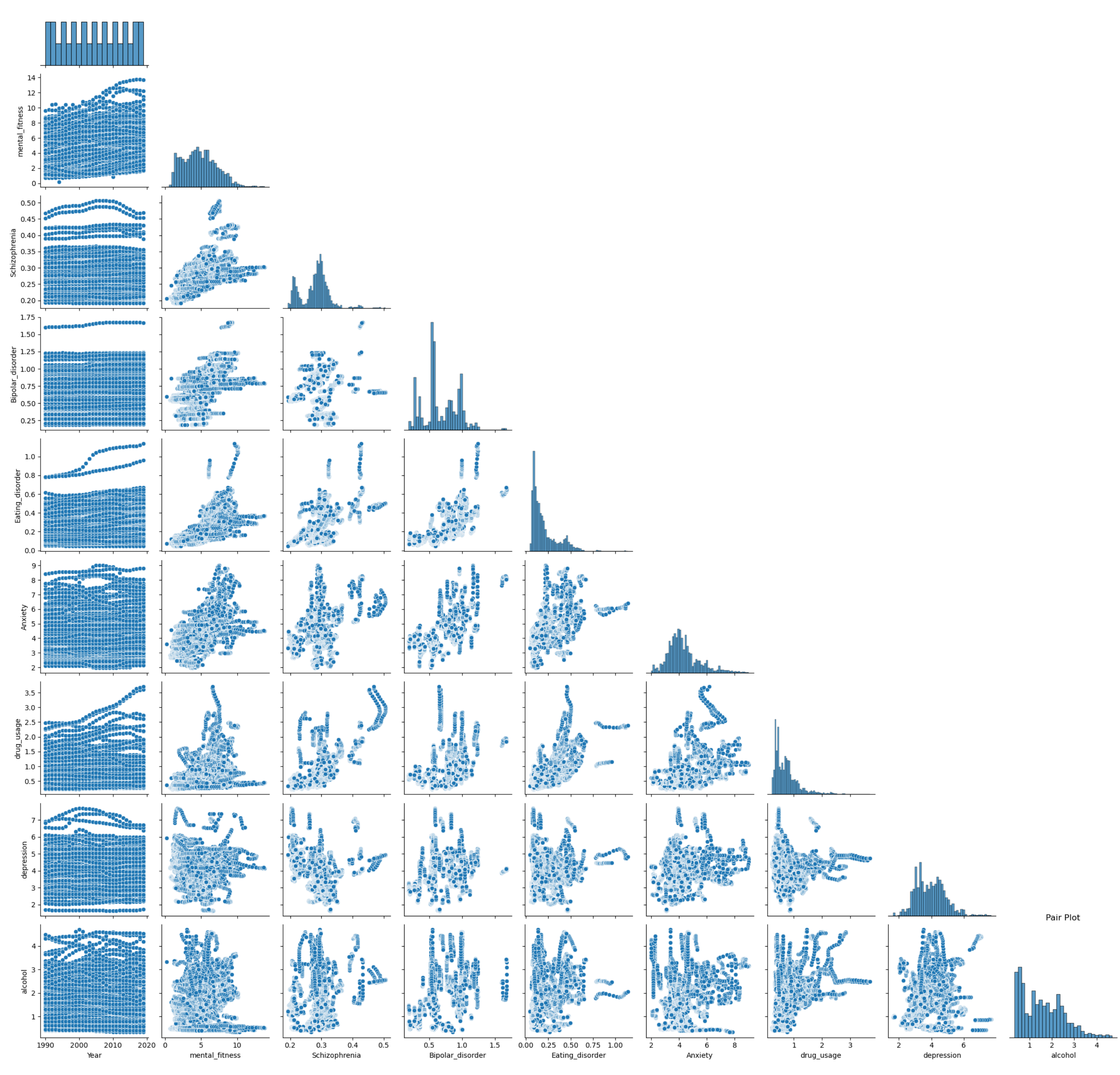
In [13]:
```python
# Joint Plot - Schizophrenia vs. Mental Fitness
sns.jointplot(data=data, x="Schizophrenia", y="mental_fitness", kind="reg", color="m")
plt.title('Joint Plot - Schizophrenia vs. Mental Fitness')
plt.show()
```



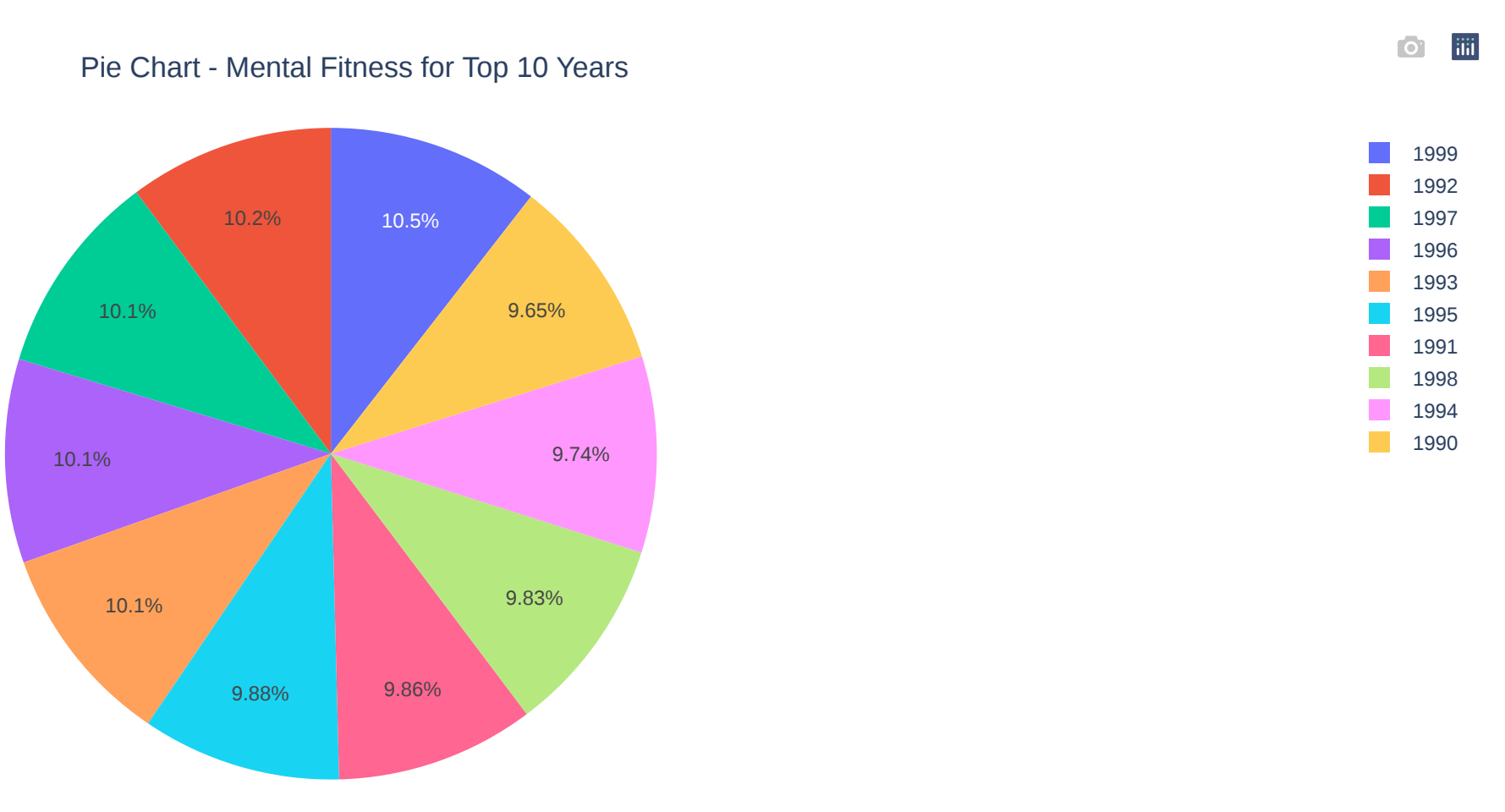
In [14]:
```python
# Joint Plot - Bipolar Disorder vs. Mental Fitness
sns.jointplot(data=data, x='Bipolar_disorder', y='mental_fitness', kind='reg', color='blue')
plt.show()
```



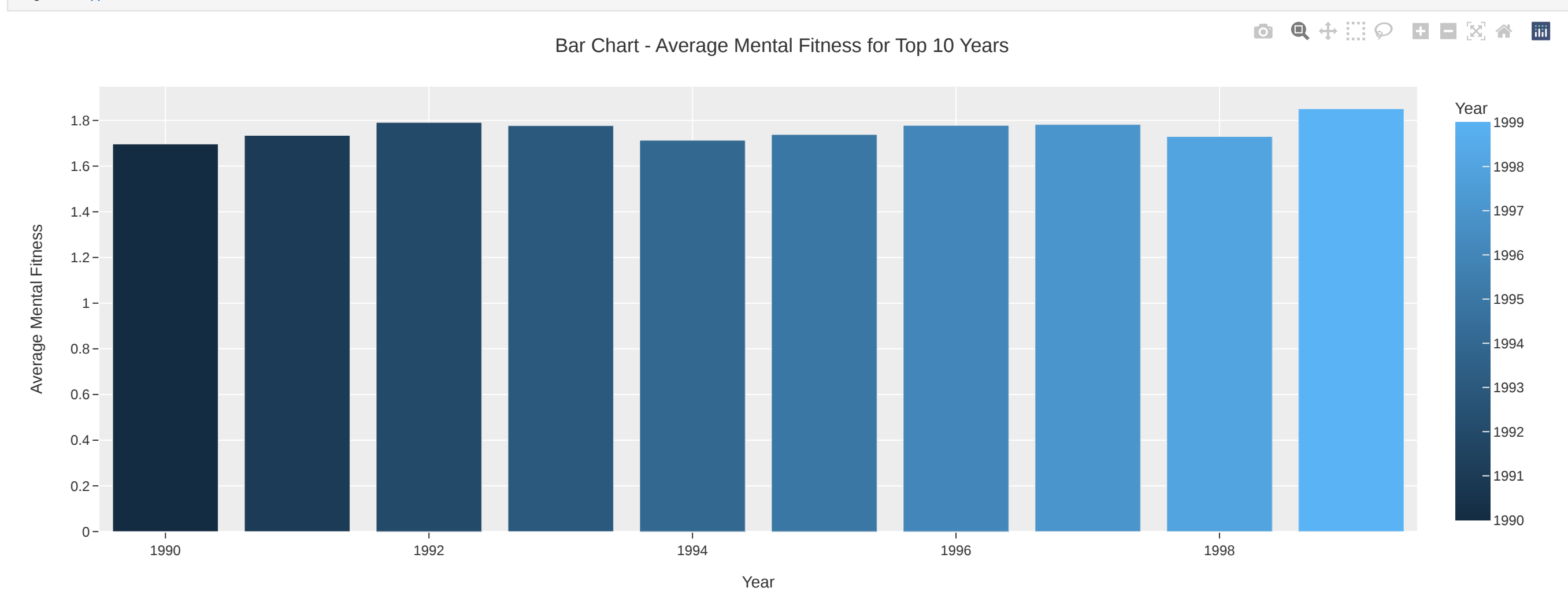
In [15]:
```python
# Pair Plot
sns.pairplot(data=data, corner=True)
plt.title('Pair Plot')
plt.show()
```



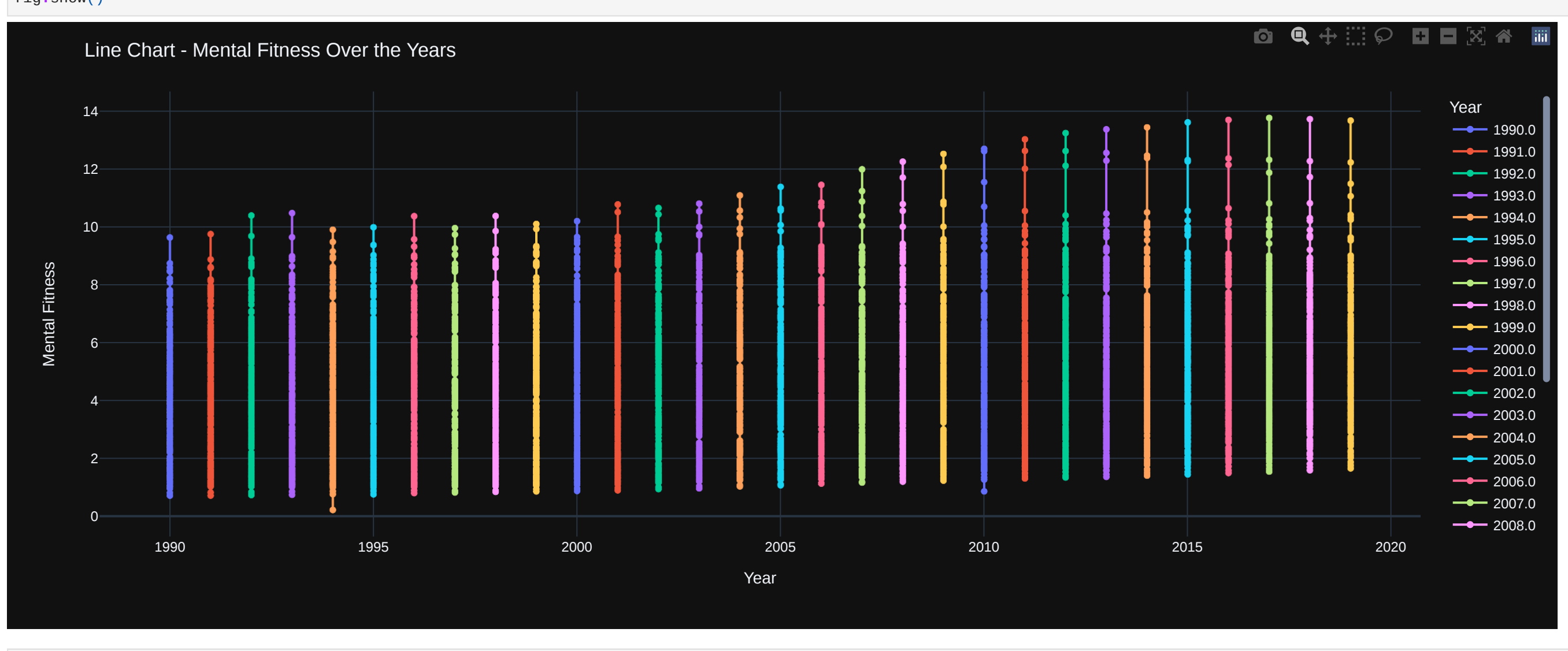
In [16]:
```python
# Pie Chart - Mental Fitness for Top 10 Years
mean_mental_fitness = data['mental_fitness'].mean()
fig = px.pie(data_frame=data.head(10), names='Year', values='mental_fitness')
fig.update_layout(title_text='Pie Chart - Mental Fitness for Top 10 Years', title_x=0.5)
fig.show()
```



In [17]:
```python
# Bar Chart - Average Mental Fitness for Top 10 Years
fig = px.bar(data.head(10), x='Year', y='mental_fitness', color='Year', template='ggplot2')
fig.update_layout(title_text='Bar Chart - Average Mental Fitness for Top 10 Years', xaxis_title='Year', yaxis_title='Average Mental Fitness')
fig.show()
```



In [18]:
```python
# Line Chart - Mental Fitness Over the Years
fig = px.line(data_frame=data, x='Year', y='mental_fitness', color='Year', markers=True, template='plotly_dark')
fig.update_layout(title_text='Line Chart - Mental Fitness Over the Years', xaxis_title='Year', yaxis_title='Mental Fitness')
fig.show()
```



In [19]:
```python
# Data preprocessing
df = data.copy()
encoder = LabelEncoder()

for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = encoder.fit_transform(df[column])

X = df.drop('mental_fitness', axis=1)
y = df['mental_fitness']
```

In [20]:
```python
# Train-test split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [21]:
```python
# Linear Regression
lr = LinearRegression()
lr.fit(x_train, y_train)

# Model evaluation for training set
y_train_pred = lr.predict(x_train)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_train_pred)

# Model evaluation for testing set
y_test_pred = lr.predict(x_test)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_test_pred)
```

In [22]:
```python
print("Linear Regression Results:")
print("Train MSE: ", mse_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MSE: ", mse_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")
```

```
Linear Regression Results:
Train MSE:  1.3924374533811699
Train RMSE:  1.1800158699700483
Train R2 Score:  0.7408634010340946


Test MSE:  1.1416819237316884
Test RMSE:  1.0684951678560313
Test R2 Score:  0.7626652123767366
```

In [23]:
```python
# Random Forest Regressor
rf = RandomForestRegressor()
rf.fit(x_train, y_train)

# Model evaluation for training set
y_train_pred = rf.predict(x_train)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_train_pred)

# Model evaluation for testing set
y_test_pred = rf.predict(x_test)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_test_pred)
```

In [24]:
```python
print("Random Forest Regressor Results:")
print("Train MSE: ", mse_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MSE: ", mse_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")
```

```
Random Forest Regressor Results:
Train MSE:  0.005279272946003874
Train RMSE:  0.07265860545044801
Train R2 Score:  0.9990175121813061


Test MSE:  0.030873005854333376
Test RMSE:  0.17570715937130665
Test R2 Score:  0.9935820668301549
```

In [25]:
```python
# Gradient Boosting Regressor
gb = GradientBoostingRegressor()
gb.fit(x_train, y_train)

# Model evaluation for training set
y_train_pred = gb.predict(x_train)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_train_pred)

# Model evaluation for testing set
y_test_pred = gb.predict(x_test)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_test_pred)
```

In [26]:
```python
print("Gradient Boosting Regressor Results:")
print("Train MSE: ", mse_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MSE: ", mse_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
```

```
Gradient Boosting Regressor Results:
Train MSE:  0.2539699668683147
Train RMSE:  0.5039543301414471
Train R2 Score:  0.9527354616224023


Test MSE:  0.2512462224779122
Test RMSE:  0.5012446732663721
Test R2 Score:  0.9477705062912459
```

In [27]:
```python
# Sample new data for prediction
new_data = pd.DataFrame({
    'Year': [2022, 2022, 2023, 2023],
    'Schizophrenia': [10.5, 9.8, 8.7, 11.2],
    'Bipolar_disorder': [6.3, 7.1, 5.9, 6.8],
    'Eating_disorder': [3.2, 2.9, 3.5, 3.0],
    'Anxiety': [8.6, 7.9, 8.2, 9.0],
    'drug_usage': [4.7, 5.1, 4.5, 4.8],
    'depression': [12.4, 11.9, 13.2, 12.8],
    'alcohol': [5.8, 5.5, 6.1, 6.3]
})
```

In [28]:
```python
# Data preprocessing for prediction
encoder = LabelEncoder()

for column in new_data.columns:
    if new_data[column].dtype == 'object':
        new_data[column] = encoder.fit_transform(new_data[column])

# Making predictions using the trained Gradient Boosting Regressor model
predicted_mental_fitness = gb.predict(new_data)

# Create a new DataFrame to store the predictions along with other features (if needed)
predictions_df = pd.DataFrame(data={'Year': new_data['Year'], 'Predicted_Mental_Fitness': predicted_mental_fitness})
```

In [29]:
```python
# Display the predictions
print(predictions_df)
```

```
   Year  Predicted_Mental_Fitness
0  2022                 10.593385
1  2022                 10.593385
2  2023                 10.593385
3  2023                 10.593385
```