Import Libraries & Load all necessary libraries for data handling and visualization

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Download and Load Alzheimer’s Dataset

```
# This section downloads the Alzheimer’s gene expression dataset from Google
# and loads it into a pandas DataFrame for analysis.
#The df.head() command displays the first five rows to verify that the datase

!pip install gdown

import gdown

file_id = "1geZIouvXrxtrIrNT-hRUInQUeoUOdLSm"
url = f"https://drive.google.com/uc?id={file_id}"

output = "AD-Data.csv"
gdown.download(url, output, quiet=False)

df = pd.read_csv(output)
df.head()
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.12/dist
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/d
```

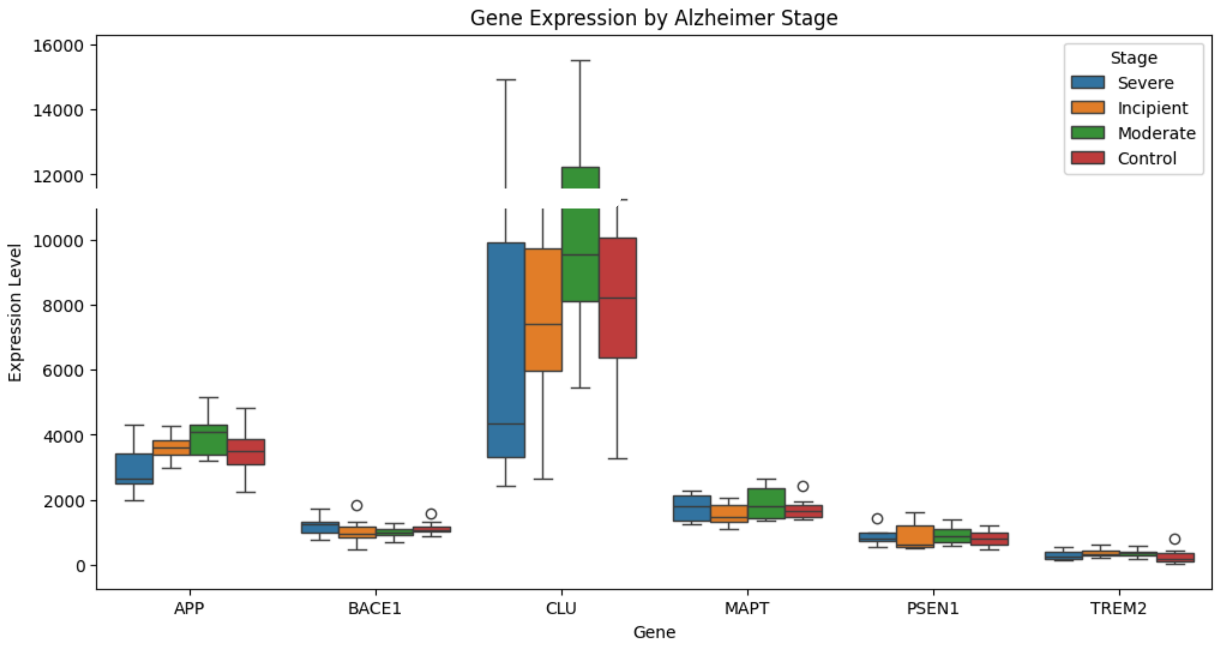
Clean and Preprocess Data: Rename columns, set index, ensure gene expression columns are numeric

```
# This code preprocesses the dataset by renaming the sample stage column to Sta
# selecting only the genes of interest, and converting their expression values
# The first few rows are displayed to verify the changes.

df = df.rename(columns={'!Sample_title': 'Stage'})

if 'ID_REF' not in df.columns:
    df = df.reset_index()

df = df.set_index('ID_REF')

gene_cols = ['APP','BACE1','CLU','MAPT','PSEN1','TREM2']
df[gene_cols] = df[gene_cols].apply(pd.to_numeric)

df.head()
```

	Patient	Stage	APP	BACE1	CLU	MAPT	PSEN1	TREM2
ID_REF								
GSM21203	yes	Severe	2357.1	901.8	3113.2	2288.5	703.9	536.3
GSM21204	yes	Incipient	2990.9	870.8	2659.0	1257.0	581.9	408.9
GSM21205	yes	Incipient	3255.8	479.6	4685.7	1807.5	526.6	221.8
GSM21206	yes	Severe	3091.1	1297.5	14932.1	2268.7	803.8	227.5
GSM21207	yes	Severe	2669.9	1073.8	2417.1	1811.9	1000.7	127.5

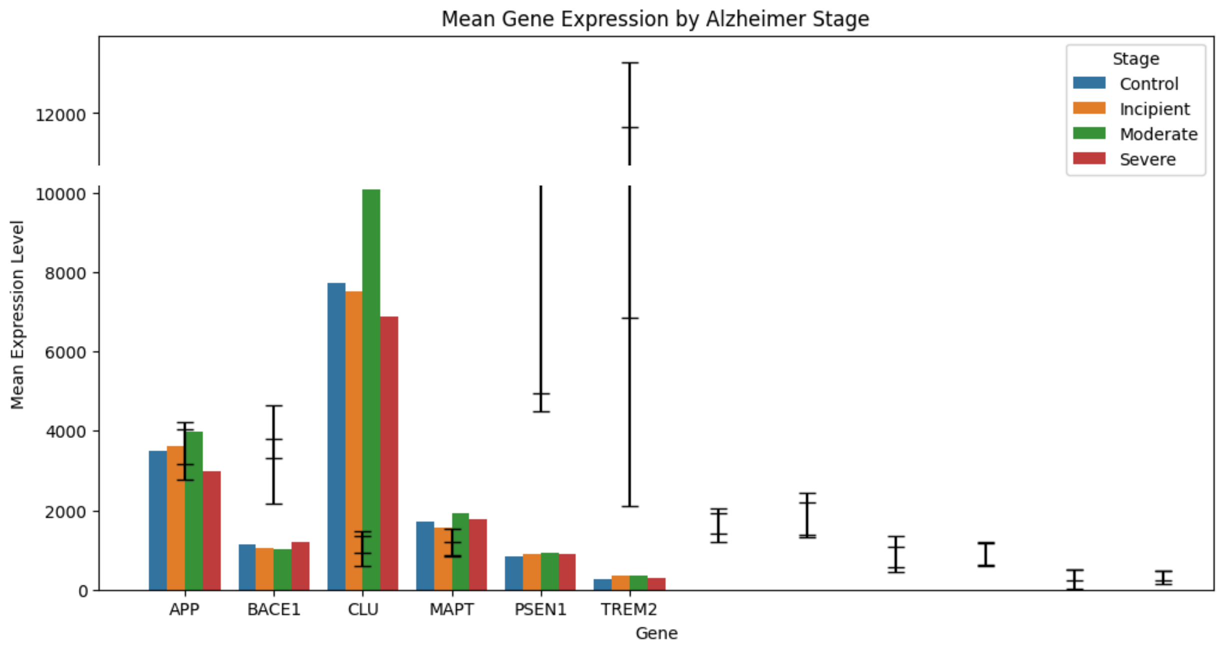
Box Plot: Gene Expression by Alzheimer Stage to compare expression levels of all key genes between Severe and Incipient stages.

```
# This code reshapes the dataset from wide to long format using melt so that ea
# It then creates a boxplot to visualize the distribution of expression levels
# and saves the figure to Google Drive.


df_long = df.reset_index().melt(id_vars=['ID_REF','Stage'],
                                value_vars=['APP','BACE1','CLU','MAPT','PSEN1',
                                var_name='Gene', value_name='Expression')

plt.figure(figsize=(12,6))
sns.boxplot(x='Gene', y='Expression', hue='Stage', data=df_long)
plt.title('Gene Expression by Alzheimer Stage')
plt.xlabel('Gene')
plt.ylabel('Expression Level')
plt.legend(title='Stage')
plt.savefig("gene_expression_boxplot.png", dpi=300, bbox_inches="tight")
plt.show()
```



Mean Gene Expression by Stage (Bar Plot with Error Bars):

```
# This code calculates the mean gene expression values for each gene across Alz
# along with the variability (standard deviation). It then creates a bar plot w
# to summarize average expression differences between Incipient and Severe stag
# the figure to Google Drive.
# Calculate mean and standard deviation for each Gene and Stage
df_summary = df_long.groupby(['Gene', 'Stage'])['Expression'].agg(['mean', 'stc

plt.figure(figsize=(12,6))
sns.barplot(x='Gene', y='mean', hue='Stage', data=df_summary, errorbar=None)

# Add error bars manually
for i, row in df_summary.iterrows():
    plt.errorbar( x=i // 2, y=row['mean'], yerr=row['std'],fmt='none',capsize=5

plt.title('Mean Gene Expression by Alzheimer Stage')
plt.xlabel('Gene')
plt.ylabel('Mean Expression Level')
plt.legend(title='Stage')
plt.savefig("mean_gene_expression_barplot.png", dpi=300, bbox_inches="tight")
plt.show()
```



Heatmap of Differentially Expressed Genes to visualize overall gene expression patterns across samples, highlighting up- and down-regulated genes.

```
# This code calculates the mean expression of each gene for every Alzheimer sta
# and sorts the genes in descending order  of fold change to identify those wit

gene_means = df.groupby('Stage')[['APP','BACE1','CLU','MAPT','PSEN1','TREM2']].


fold_change = gene_means.loc['Severe'] / gene_means.loc['Incipient']
print("Fold change (Severe / Incipient):\n", fold_change)


sorted_genes = fold_change.sort_values(ascending=False).index.tolist()
```

```
Fold change (Severe / Incipient):
 APP      0.823224
BACE1    1.149670
CLU      0.916792
MAPT     1.123004
PSEN1    0.985029
TREM2    0.823637
dtype: float64
```

```
# This part generates a heatmap of gene expression for the selected genes (sort
# The heatmap uses a color gradient to represent expression levels, with labels
# and the figure is saved to Google Drive.

plt.figure(figsize=(8,6))
sns.heatmap(df[sorted_genes], cmap='coolwarm', annot=False)
plt.title('Heatmap of Gene Expression Across Samples')
plt.ylabel('Sample ID')
plt.xlabel('Gene')
plt.savefig("heatmap.png", dpi=300, bbox_inches="tight")
plt.show()
```

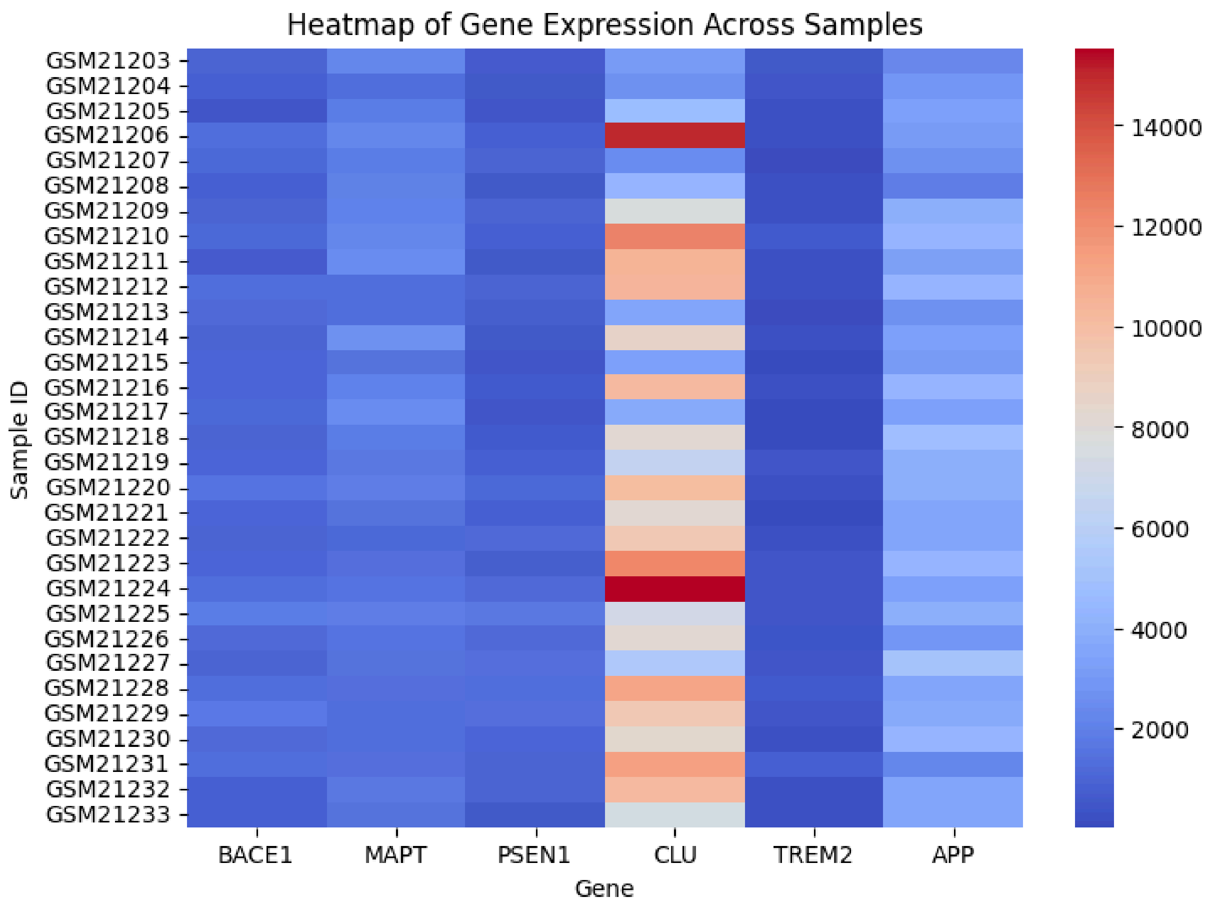


Table of Significant Genes to list genes with largest expression differences, including mean values and fold-change.

```
# This code separates the dataset into two subsets based on Alzheimer stage, cr
# Only the genes of interest are considered for further analysis.

gene_cols = ['APP','BACE1','CLU','MAPT','PSEN1','TREM2']

severe = df[df['Stage']=='Severe']
incipient = df[df['Stage']=='Incipient']
```

```
# This code iterates over the selected genes, calculates the mean expression fo
# between the two stages for each gene, and stores the results in a list of dic


results = []
```

```
for gene in gene_cols:
    mean_severe = severe[gene].mean()
    mean_incipient = incipient[gene].mean()
    fold_change = mean_severe / mean_incipient

    results.append({
        'Gene': gene,
        'Mean_Severe': mean_severe,
        'Mean_Incipient': mean_incipient,
        'Fold_Change': fold_change
    })
```

```
# This part of the code creates a visual table of the significant genes by conv
# and the figure is saved to Google Drive for presentation purposes.

results_df = pd.DataFrame(results)
sig_genes_df = results_df
plt.figure(figsize=(12, len(sig_genes_df)*0.5))

plt.axis('off')

plt.table(cellText=sig_genes_df.values,
          colLabels=sig_genes_df.columns,
          cellLoc='center',
          loc='center')
plt.title("Table 1: Significant Genes Between Alzheimer Stages", fontsize=16, f

plt.savefig("sig_genes_table.png", dpi=300, bbox_inches="tight")
plt.show()
```

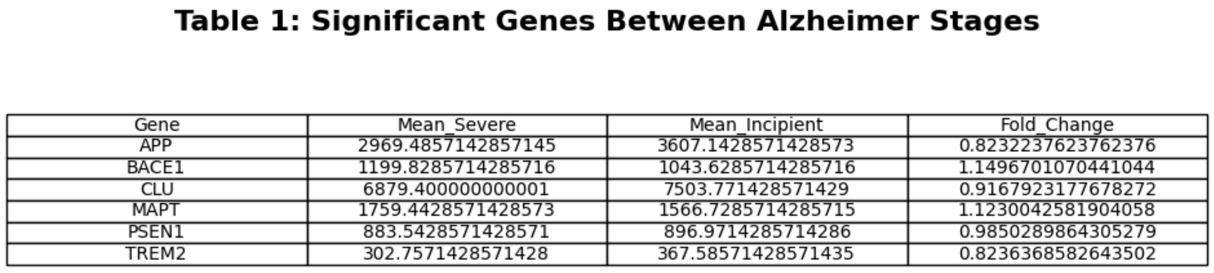
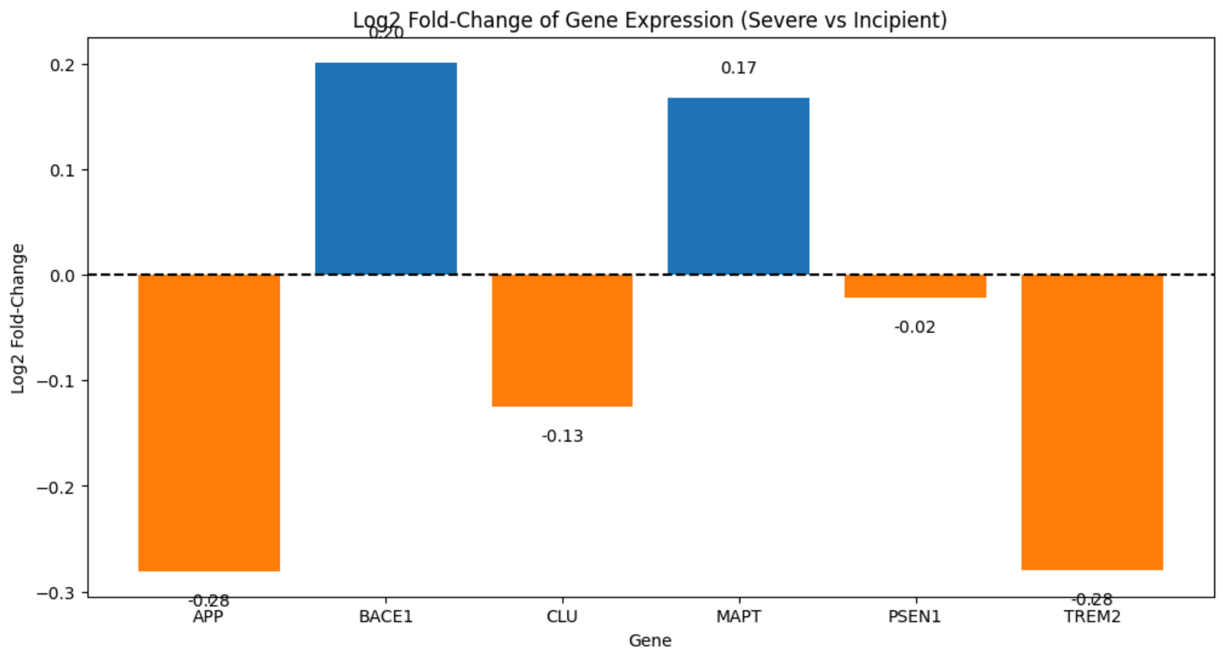


Table 1: Significant Genes Between Alzheimer Stages

Gene	Mean_Severe	Mean_Incipient	Fold_Change
APP	2969.4857142857145	3607.1428571428573	0.8232237623762376
BACE1	1199.8285714285716	1043.6285714285716	1.1496701070441044
CLU	6879.400000000001	7503.771428571429	0.9167923177678272
MAPT	1759.4428571428573	1566.7285714285715	1.1230042581904058
PSEN1	883.5428571428571	896.9714285714286	0.9850289864305279
TREM2	302.7571428571428	367.58571428571435	0.8236368582643502

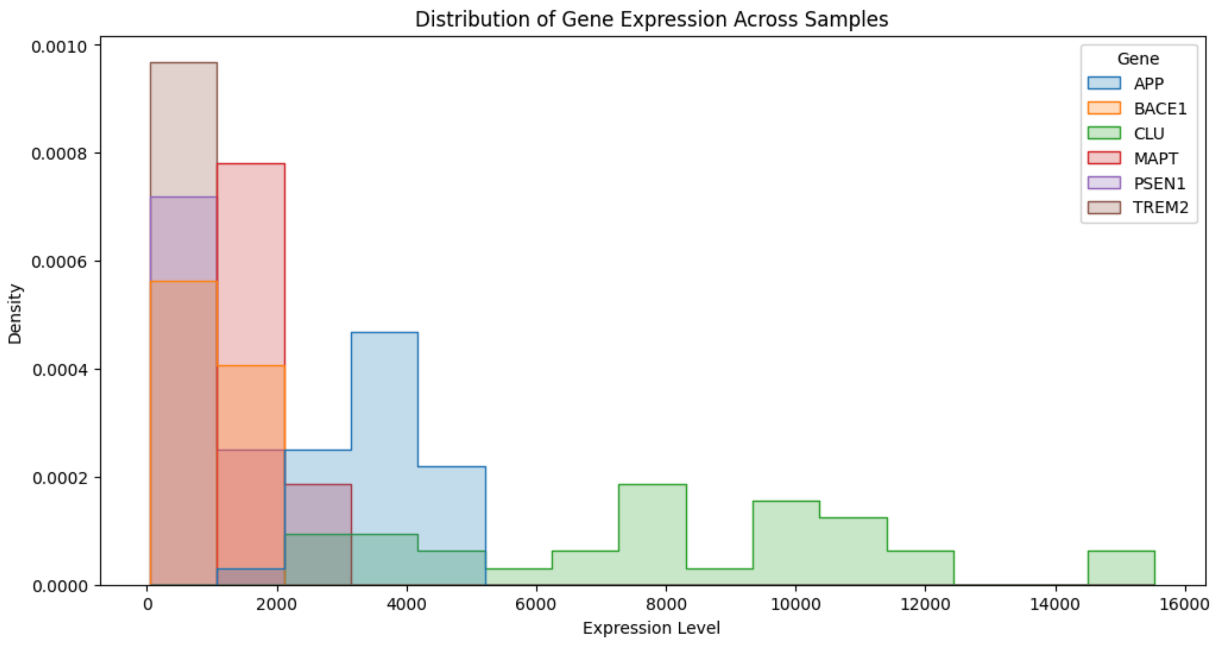
Fold-Change Bar Plot (Severe vs Incipient)

```
# This code calculates the log2 fold-change in gene expression between Severe a
# It creates a bar plot with custom colors matching the box plot, displays fold
# adds labels and a title, and saves the figure to the working directory.


# Calculate mean expression for each gene in each stage
mean_expr = df_long.groupby(['Gene', 'Stage'])['Expression'].mean().reset_index

# Pivot table to have Incipient and Severe columns
mean_expr_pivot = mean_expr.pivot(index='Gene', columns='Stage', values='Expres

# Calculate log2 fold-change: log2(Severe / Incipient)
mean_expr_pivot['log2FC'] = np.log2(mean_expr_pivot['Severe'] / mean_expr_pivot

bar_colors = ['#1f77b4' if x > 0 else '#ff7f0e' for x in mean_expr_pivot['log2F

plt.figure(figsize=(12,6))
bars = plt.bar(mean_expr_pivot['Gene'], mean_expr_pivot['log2FC'], color=bar_cc

# Add a horizontal line at y=0 for reference
plt.axhline(0, color='black', linestyle='--')

# Add fold-change values on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text( bar.get_x() + bar.get_width()/2, yval + 0.02*np.sign(yval), f'{yv

plt.title('Log2 Fold-Change of Gene Expression (Severe vs Incipient)')
plt.xlabel('Gene')
plt.ylabel('Log2 Fold-Change')
plt.savefig("fold_change_barplot.png", dpi=300, bbox_inches="tight")
plt.show()
```



Histogram / Distribution Plot of Gene Expression to show the distribution of expression levels across all samples for each gene.

```
# This code reshapes the dataset to long format and creates a histogram to visu
# The histogram is color-coded by gene, normalized to show density, and the res

df_long = df.reset_index().melt(id_vars=['ID_REF','Stage'],
                                value_vars=['APP','BACE1','CLU','MAPT','PSEN1',
                                var_name='Gene', value_name='Expression')

plt.figure(figsize=(12,6))
sns.histplot(data=df_long, x='Expression', hue='Gene', element='step', stat='de
plt.title("Distribution of Gene Expression Across Samples")
plt.xlabel("Expression Level")
plt.ylabel("Density")
plt.savefig("distribution_gene_expression_hist.png", dpi=300, bbox_inches="tigh
plt.show()
```



```
Start coding or generate with AI.
```