In [1]:
```
!pip -q install scanpy pandas numpy matplotlib seaborn scipy
```

```
[notice] A new release of pip available: 22.3 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
```

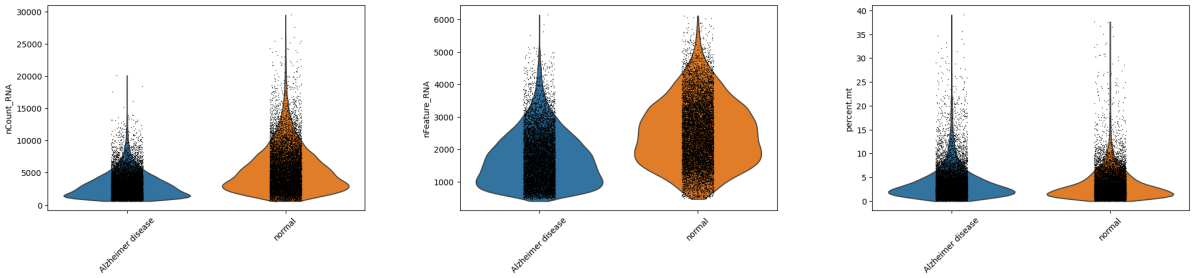
In [4]:
```
import scanpy as sc
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.sparse
import os

def comprehensive_alzheimer_analysis(adata, output_dir="analysis_results"):
    """
    Comprehensive analysis pipeline combining detailed visualization with cl
    """
    print("\n=== STEP 1: Dataset Overview and Quality Control ===")
    print("This dataset contains:")
    print(f"• {adata.n_obs} cells")
    print(f"• {adata.n_vars} genes")

    # Store raw counts
    adata.raw = adata

    # Cell type distribution
    print("\nCell Types in the dataset:")
    for cell_type in adata.obs['Cell.Types'].unique():
        count = sum(adata.obs['Cell.Types'] == cell_type)
        percentage = (count/len(adata.obs))*100
        print(f"• {cell_type}: {count} cells ({percentage:.1f}%)")

    # Create separate figures for better control
    # Figure 1: Quality metrics violin plots
    plt.figure(figsize=(12, 6))
    sc.pl.violin(adata, ['nCount_RNA', 'nFeature_RNA', 'percent.mt'],
                 groupby='disease', rotation=45)
    plt.tight_layout()
    plt.show()

    # Figure 2: Cell type composition pie chart
    plt.figure(figsize=(10, 8))
    cell_counts = adata.obs['Cell.Types'].value_counts()
    plt.pie(cell_counts, labels=cell_counts.index, autopct='%1.1f%%')
    plt.title('Cell Type Distribution')
    plt.show()

    # Figure 3: Disease state distribution
    plt.figure(figsize=(8, 6))
    disease_counts = adata.obs['disease'].value_counts()
    sns.barplot(x=disease_counts.index, y=disease_counts.values)
    plt.title('Disease State Distribution')
    plt.ylabel('Number of Cells')
    plt.xticks(rotation=45)
    plt.tight_layout()
```

```
plt.show()

# Figure 4: Braak stage distribution
plt.figure(figsize=(8, 6))
braak_counts = adata.obs['Braak'].value_counts()
sns.barplot(x=braak_counts.index, y=braak_counts.values)
plt.title('Braak Stage Distribution')
plt.ylabel('Number of Cells')
plt.xlabel('Braak Stage')

plt.tight_layout()
plt.show()



print("\n=== STEP 2: Disease and Cell Type Analysis ===")
# Cross-tabulation of cell types and disease status
cross_tab = pd.crosstab(adata.obs['Cell.Types'], adata.obs['disease'])
print("\nNumber of cells for each cell type in AD vs Normal:")
print(cross_tab)

# Stacked bar plot
plt.figure(figsize=(12, 6))
cross_tab.plot(kind='bar', stacked=True)
plt.title("Cell Type Distribution by Disease Status")
plt.xlabel("Cell Type")
plt.ylabel("Number of Cells")
plt.legend(title="Disease Status")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

print("\n=== STEP 3: Braak Stage Analysis ===")
print("\nDistribution of Braak stages:")
for stage in braak_counts.index:
    percentage = (braak_counts[stage]/len(adata.obs))*100
    print(f"• Stage {stage}: {braak_counts[stage]} cells ({percentage:.1

# Analyze gene expression changes across Braak stages
sc.tl.rank_genes_groups(adata, 'Braak', method='wilcoxon')
print("\nTop differentially expressed genes across Braak stages:")
sc.pl.rank_genes_groups(adata, n_genes=10, sharey=False)

print("\n=== STEP 4: Cell Type-Specific Analysis ===")
cell_types = adata.obs['Cell.Types'].unique()

for cell_type in cell_types:
    print(f"\nAnalyzing {cell_type}")
    cell_type_data = adata[adata.obs['Cell.Types'] == cell_type]

    # Compare gene expression between AD and normal
    sc.tl.rank_genes_groups(cell_type_data, 'disease',
                            groups=['Alzheimer disease'],
                            reference='normal',
                            method='wilcoxon')
```

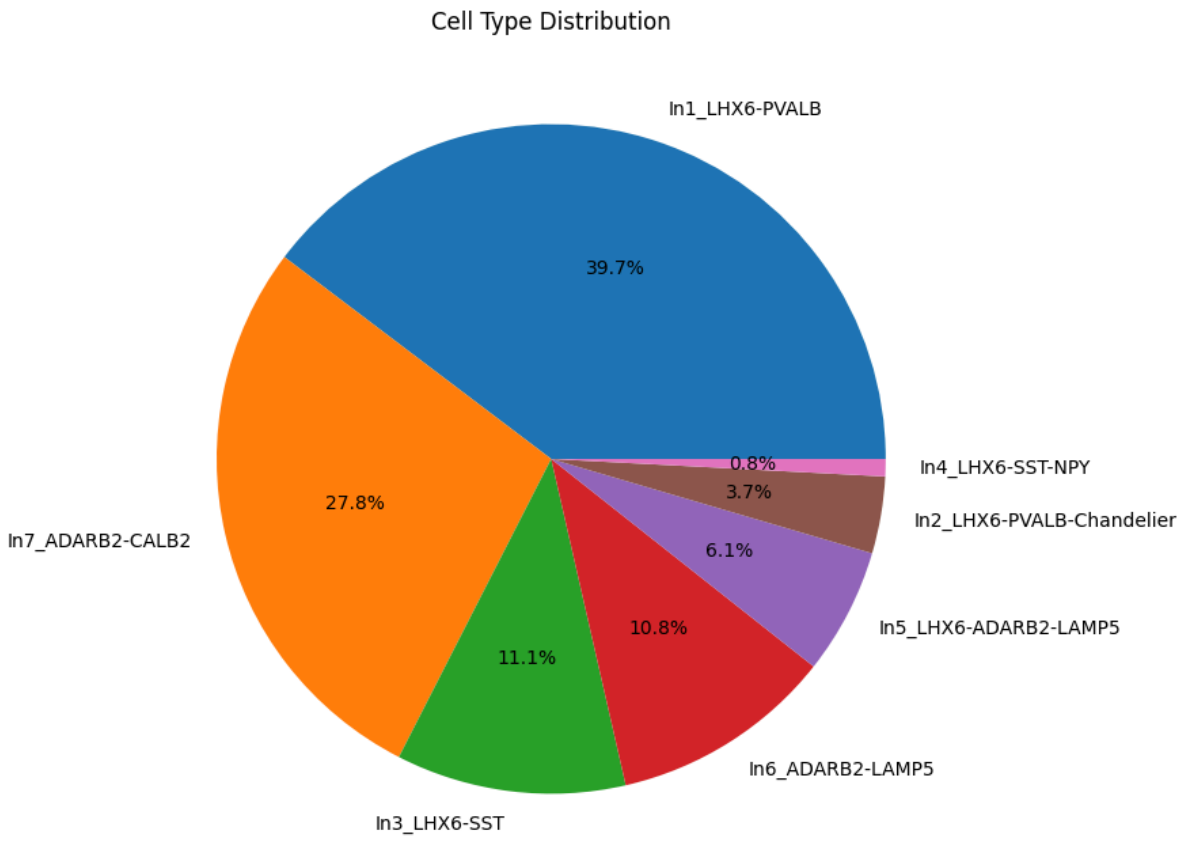
```
        print(f"\nTop differentially expressed genes in {cell_type}:")
        sc.pl.rank_genes_groups(cell_type_data, n_genes=10, sharey=False)

print("\n=== STEP 5: Demographic Analysis ===")
# Age distribution
plt.figure(figsize=(10, 6))
sns.boxplot(data=adata.obs, x='disease', y='Age')
plt.title('Age Distribution by Disease Status')
plt.show()

# Sex distribution
sex_disease_cross = pd.crosstab(adata.obs['sex'], adata.obs['disease'])
plt.figure(figsize=(8, 6))
sex_disease_cross.plot(kind='bar')
plt.title('Sex Distribution by Disease Status')
plt.xlabel('Sex')
plt.ylabel('Number of Cells')
plt.legend(title='Disease Status')
plt.tight_layout()
plt.show()

print("\n=== STEP 6: Dimensional Reduction Visualization ===")
# Compute UMAP if not already computed
if 'X_umap' not in adata.obsm_keys():
    sc.pp.normalize_total(adata, target_sum=1e4)
    sc.pp.log1p(adata)
    sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_
    sc.pp.pca(adata, svd_solver='arpack')
    sc.pp.neighbors(adata)
    sc.tl.umap(adata)

# Create UMAP visualizations
fig, axes = plt.subplots(2, 2, figsize=(15, 15))
axes = axes.flatten()

sc.pl.umap(adata, color='disease', ax=axes[0], show=False, title='Diseas
sc.pl.umap(adata, color='Cell.Types', ax=axes[1], show=False, title='Cel
sc.pl.umap(adata, color='Braak', ax=axes[2], show=False, title='Braak St
sc.pl.umap(adata, color='sex', ax=axes[3], show=False, title='Sex')

plt.tight_layout()
plt.show()

# Save results if output directory is specified
if output_dir:
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    # Save the processed AnnData object
    adata.write(f"{output_dir}/processed_data.h5ad")

    # Save cell type markers
    if 'rank_genes_groups' in adata.uns:
        pd.DataFrame(adata.uns['rank_genes_groups']['names']).to_csv(
            f"{output_dir}/cell_type_markers.csv")
```

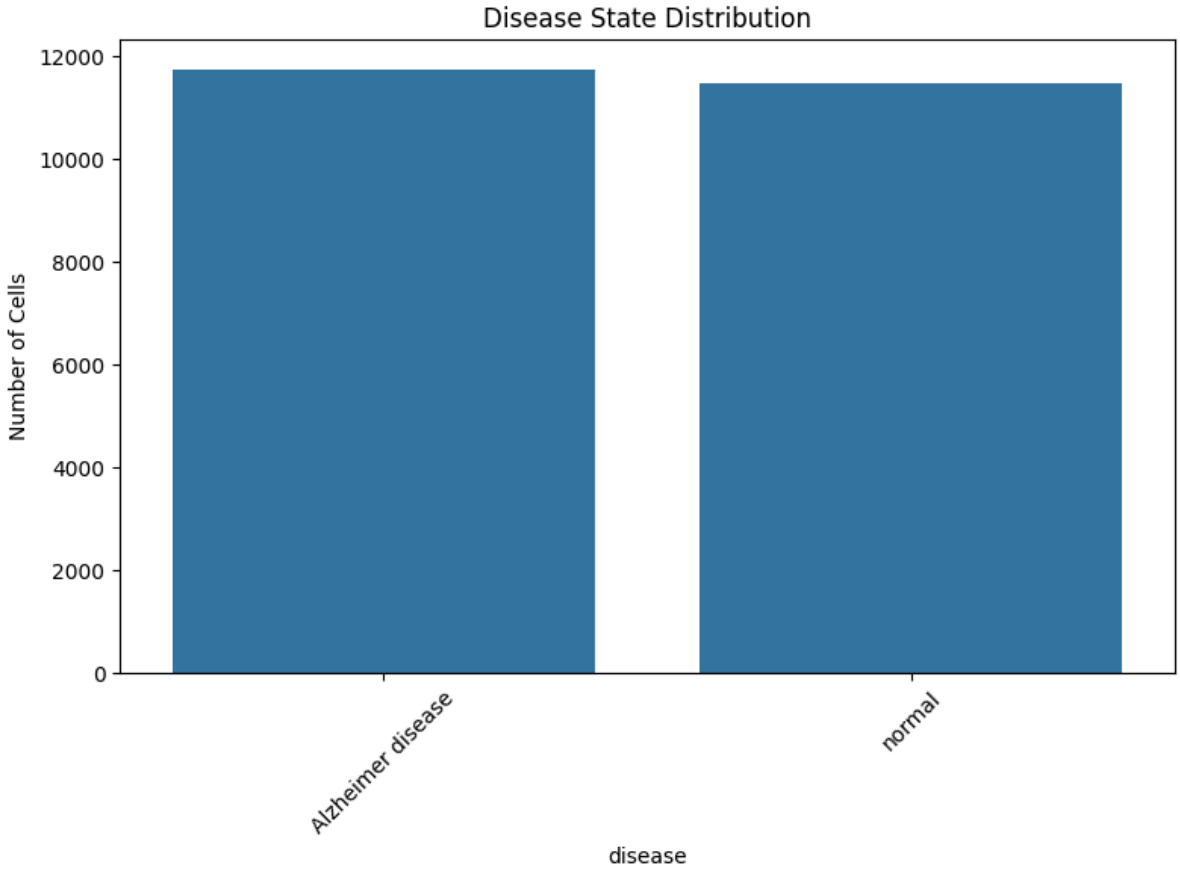
```
        print(f"\nResults saved to {output_dir}/")

    return adata

# Example usage:
if __name__ == "__main__":
    print("Loading data...")
    adata = sc.read_h5ad("alzheimer_single_cell_data.h5ad")
    print("Running analysis...")
    adata = comprehensive_alzheimer_analysis(adata)
```

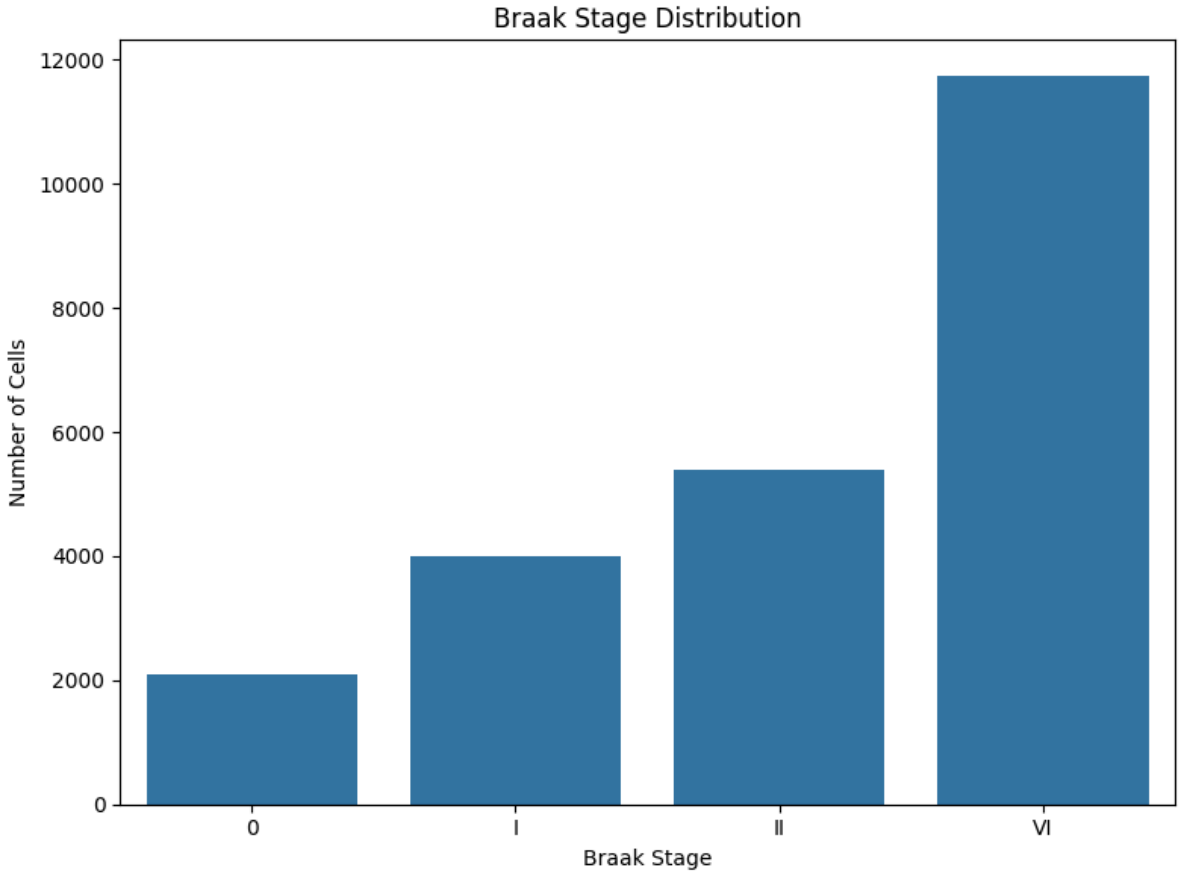
```
Loading data...
Running analysis...

=== STEP 1: Dataset Overview and Quality Control ===
This dataset contains:
• 23197 cells
• 33091 genes

Cell Types in the dataset:
• In7_ADARB2-CALB2: 6445 cells (27.8%)
• In6_ADARB2-LAMP5: 2500 cells (10.8%)
• In1_LHX6-PVALB: 9208 cells (39.7%)
• In5_LHX6-ADARB2-LAMP5: 1414 cells (6.1%)
• In3_LHX6-SST: 2575 cells (11.1%)
• In2_LHX6-PVALB-Chandelier: 861 cells (3.7%)
• In4_LHX6-SST-NPY: 194 cells (0.8%)
<Figure size 1200x600 with 0 Axes>
```



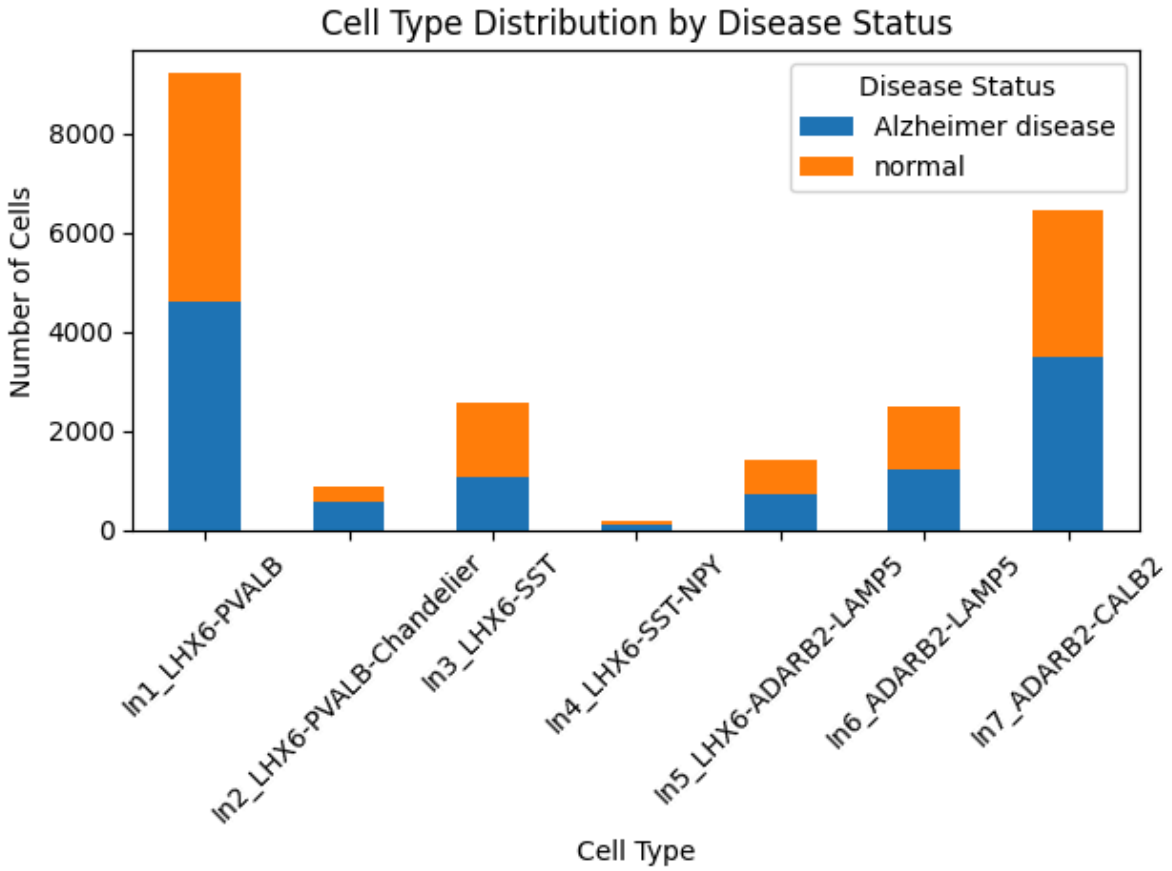

```
<Figure size 640x480 with 0 Axes>
```


Cell Type Distribution
In1_LHX6-PVALB
39.7%
In4_LHX6-SST-NPY
0.8%
In2_LHX6-PVALB-Chandelier
3.7%
In5_LHX6-ADARB2-LAMP5
6.1%
In6_ADARB2-LAMP5
10.8%
In3_LHX6-SST
11.1%
In7_ADARB2-CALB2
27.8%

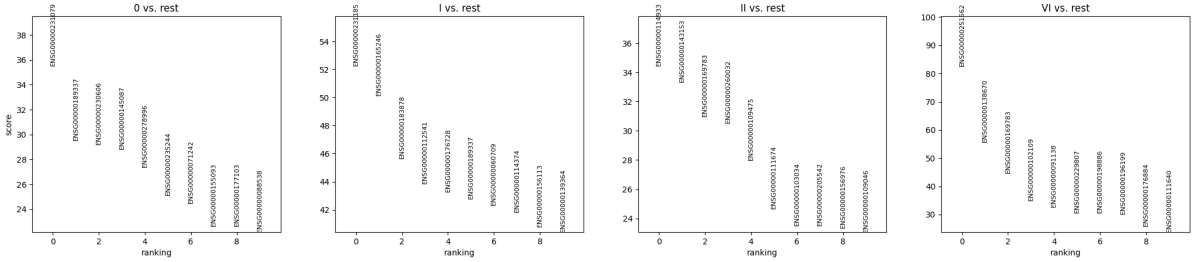

Disease State Distribution
Number of Cells
0
2000
4000
6000
8000
10000
12000
Alzheimer disease
normal
disease



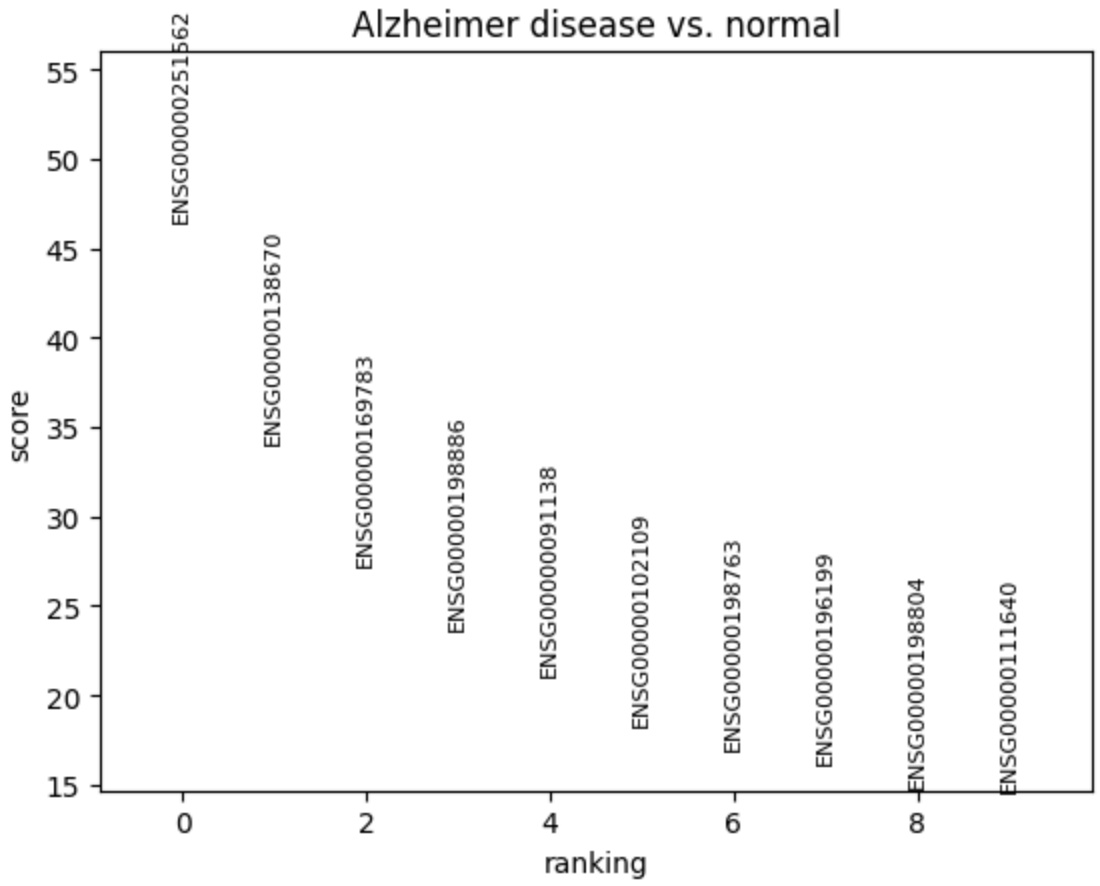

```
=== STEP 2: Disease and Cell Type Analysis ===

Number of cells for each cell type in AD vs Normal:
disease                     Alzheimer disease  normal
Cell.Types
In1_LHX6-PVALB                           4604    4604
In2_LHX6-PVALB-Chandelier                 545     316
In3_LHX6-SST                             1072    1503
In4_LHX6-SST-NPY                           83     111
In5_LHX6-ADARB2-LAMP5                     728     686
In6_ADARB2-LAMP5                         1212    1288
In7_ADARB2-CALB2                         3489    2956
<Figure size 1200x600 with 0 Axes>
```




```
=== STEP 3: Braak Stage Analysis ===

Distribution of Braak stages:
• Stage VI: 11733 cells (50.6%)
• Stage II: 5379 cells (23.2%)
• Stage I: 3996 cells (17.2%)
• Stage 0: 2089 cells (9.0%)

Top differentially expressed genes across Braak stages:
```



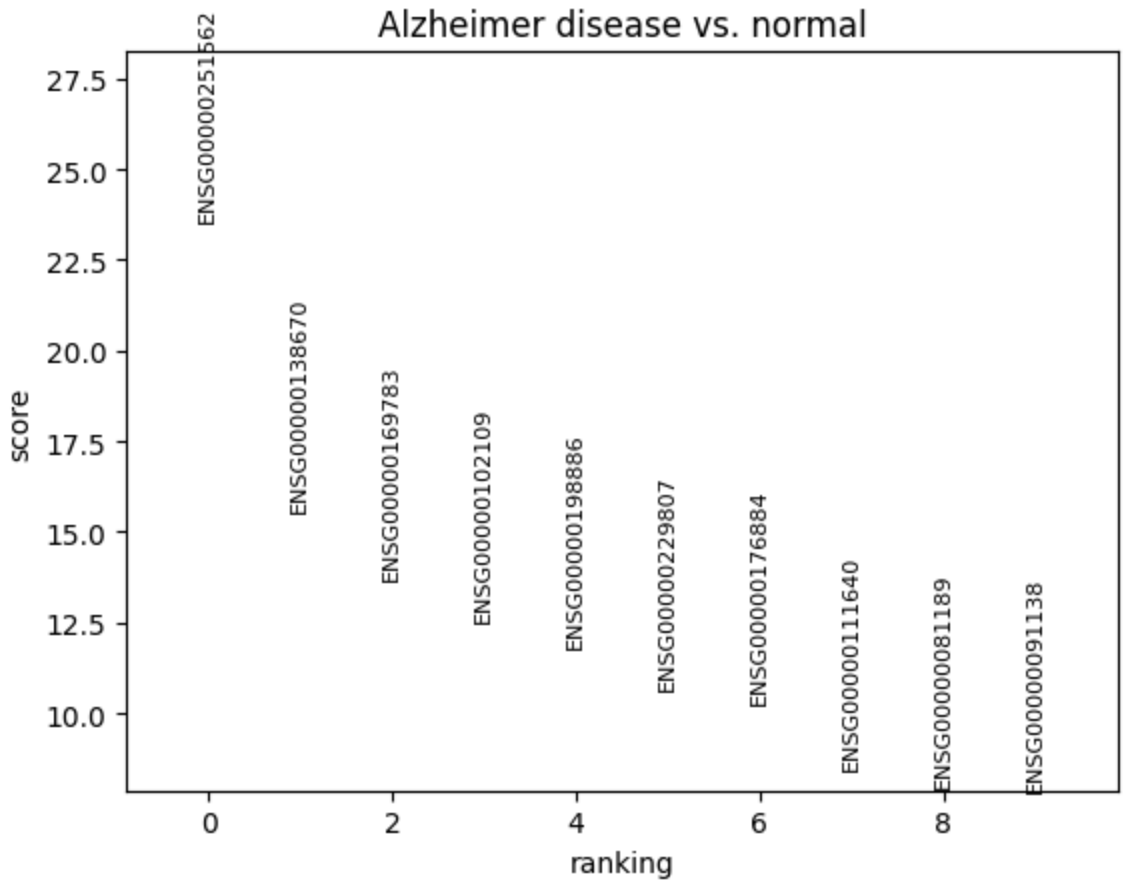

```
=== STEP 4: Cell Type-Specific Analysis ===

Analyzing In7_ADARB2-CALB2
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanpy/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to modify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
Top differentially expressed genes in In7_ADARB2-CALB2:
```




```
Analyzing In6_ADARB2-LAMP5
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

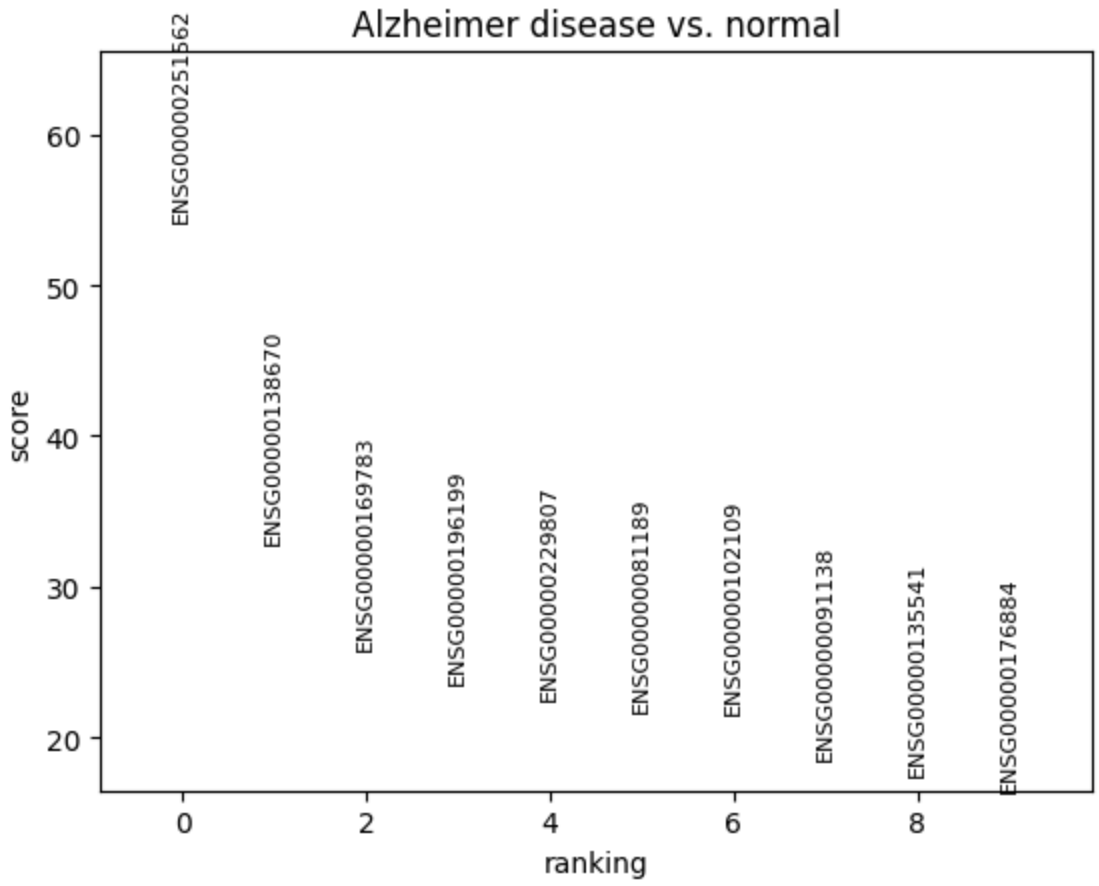
```
Top differentially expressed genes in In6_ADARB2-LAMP5:
```




```
Analyzing In1_LHX6-PVALB
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

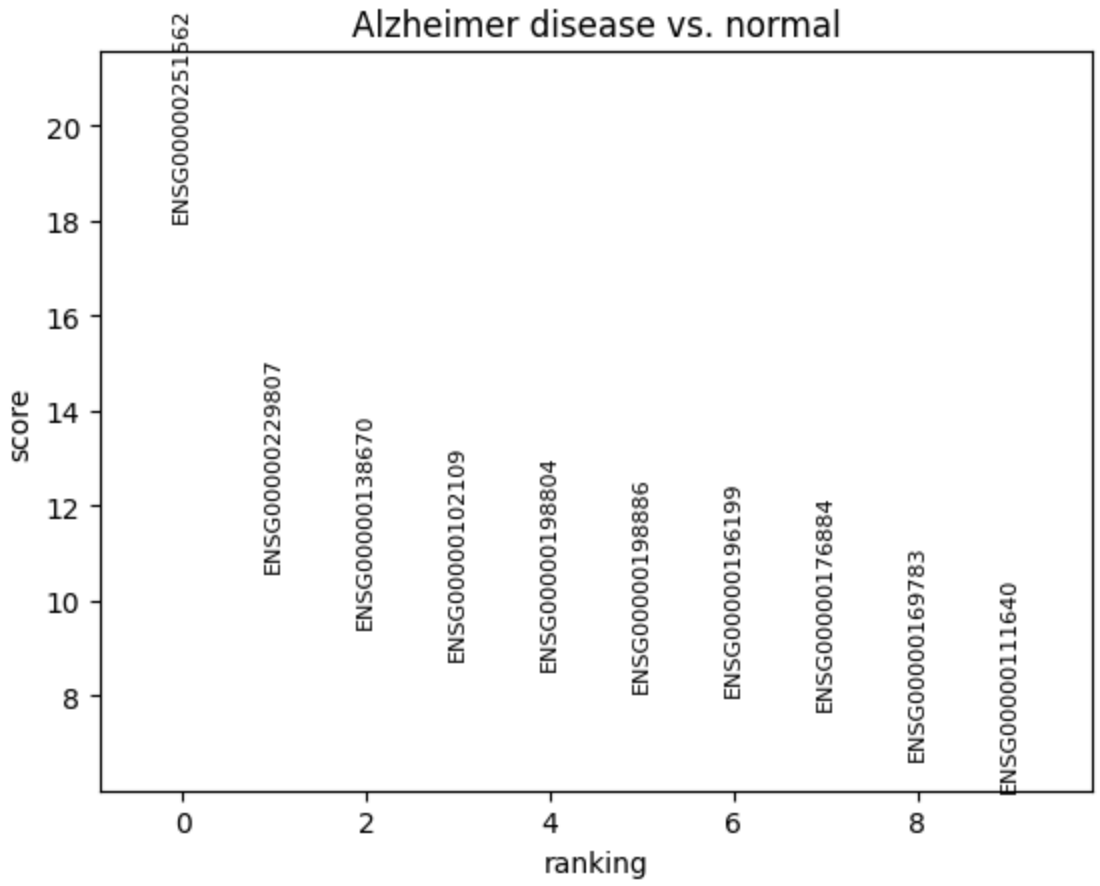
```
Top differentially expressed genes in In1_LHX6-PVALB:
```




```
Analyzing In5_LHX6-ADARB2-LAMP5
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

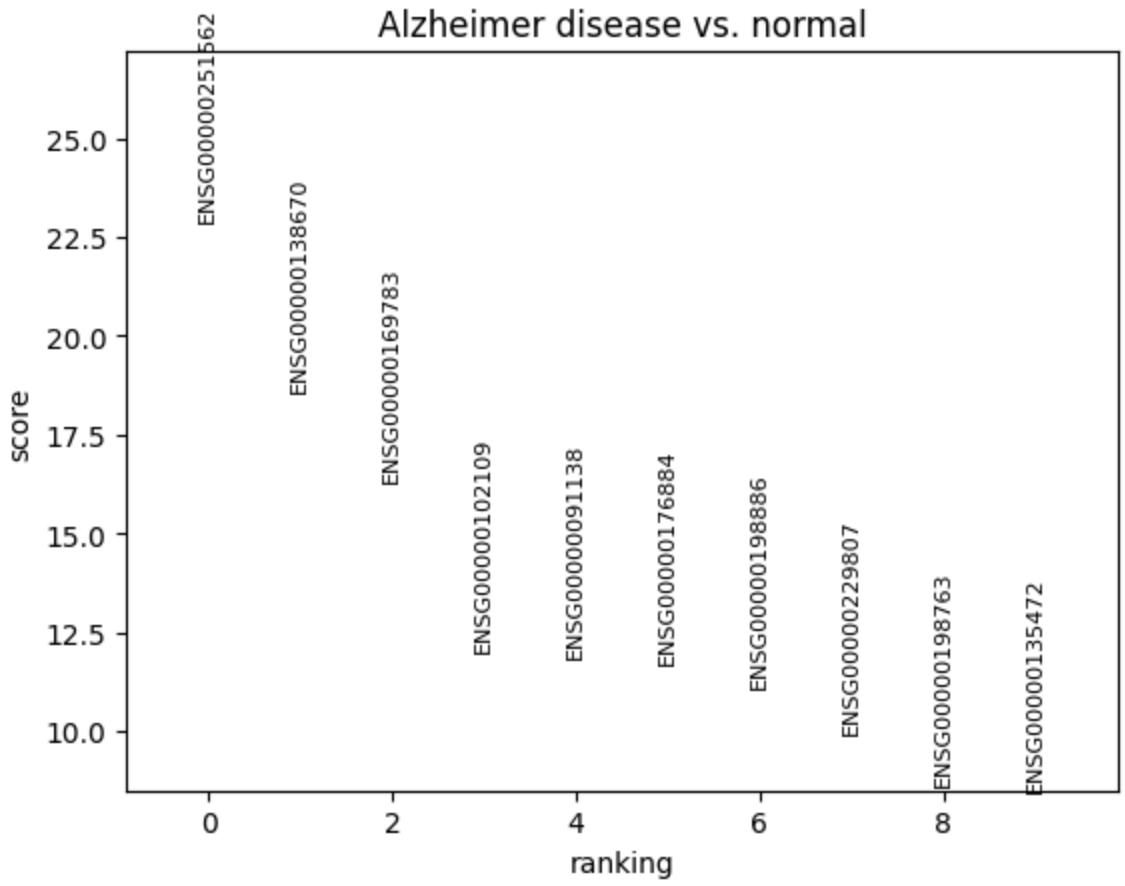
```
Top differentially expressed genes in In5_LHX6-ADARB2-LAMP5:
```




```
Analyzing In3_LHX6-SST
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

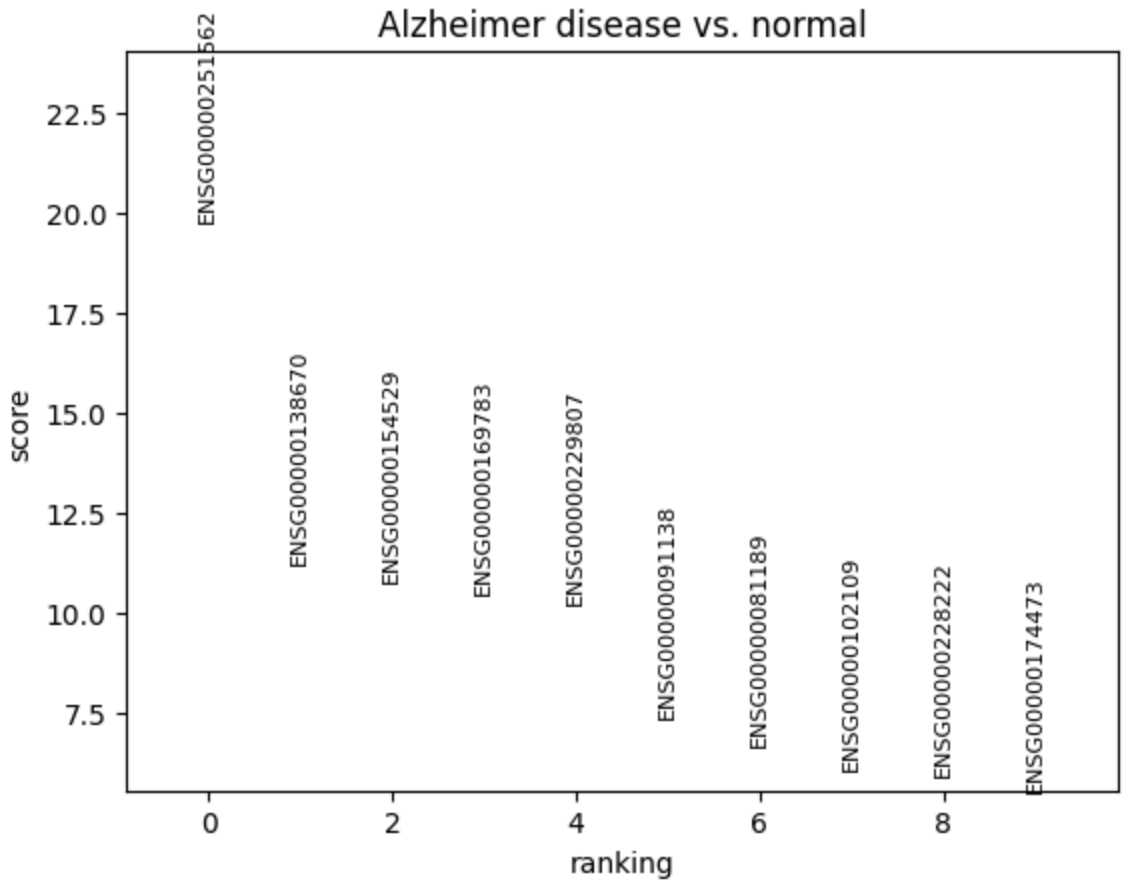
```
Top differentially expressed genes in In3_LHX6-SST:
```




```
Analyzing In2_LHX6-PVALB-Chandelier
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

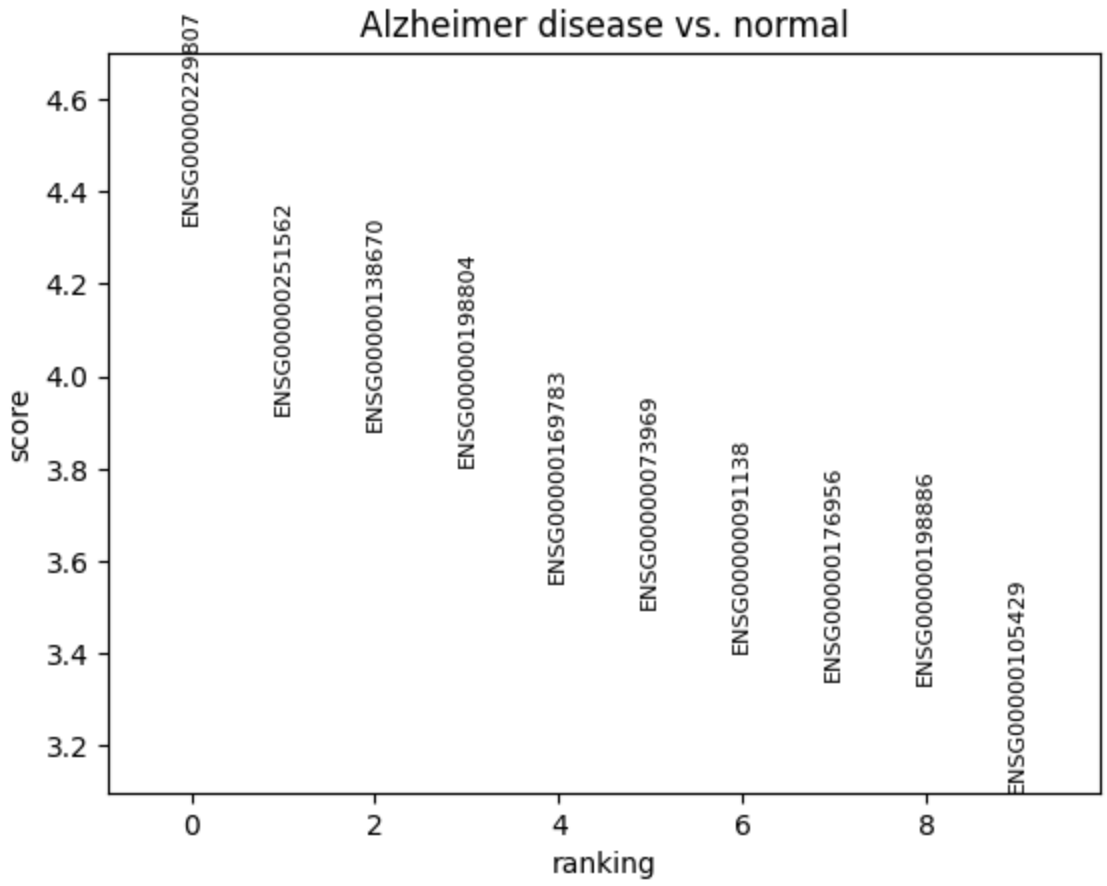
```
Top differentially expressed genes in In2_LHX6-PVALB-Chandelier:
```



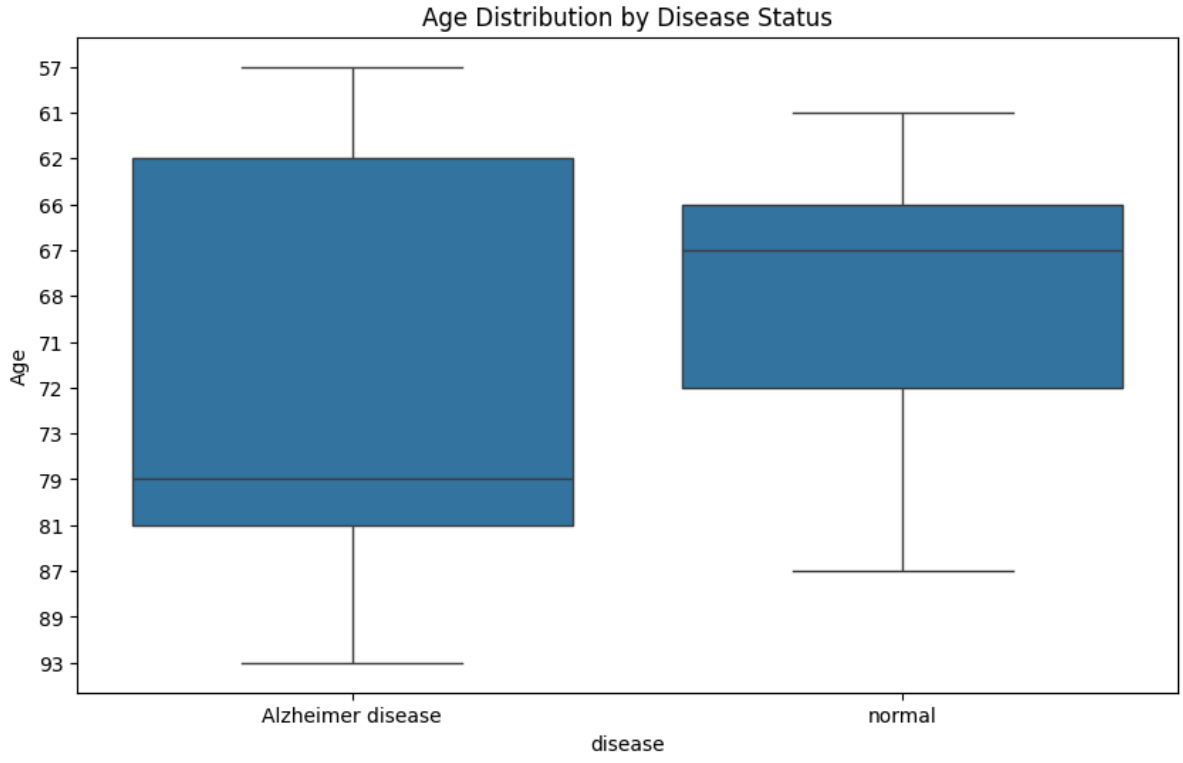

```
Analyzing In4_LHX6-SST-NPY
```

```
/home/studio-lab-user/.conda/envs/default/lib/python3.9/site-packages/scanp
y/tools/_rank_genes_groups.py:645: ImplicitModificationWarning: Trying to m
odify attribute `._uns` of view, initializing view as actual.
  adata.uns[key_added] = {}
```

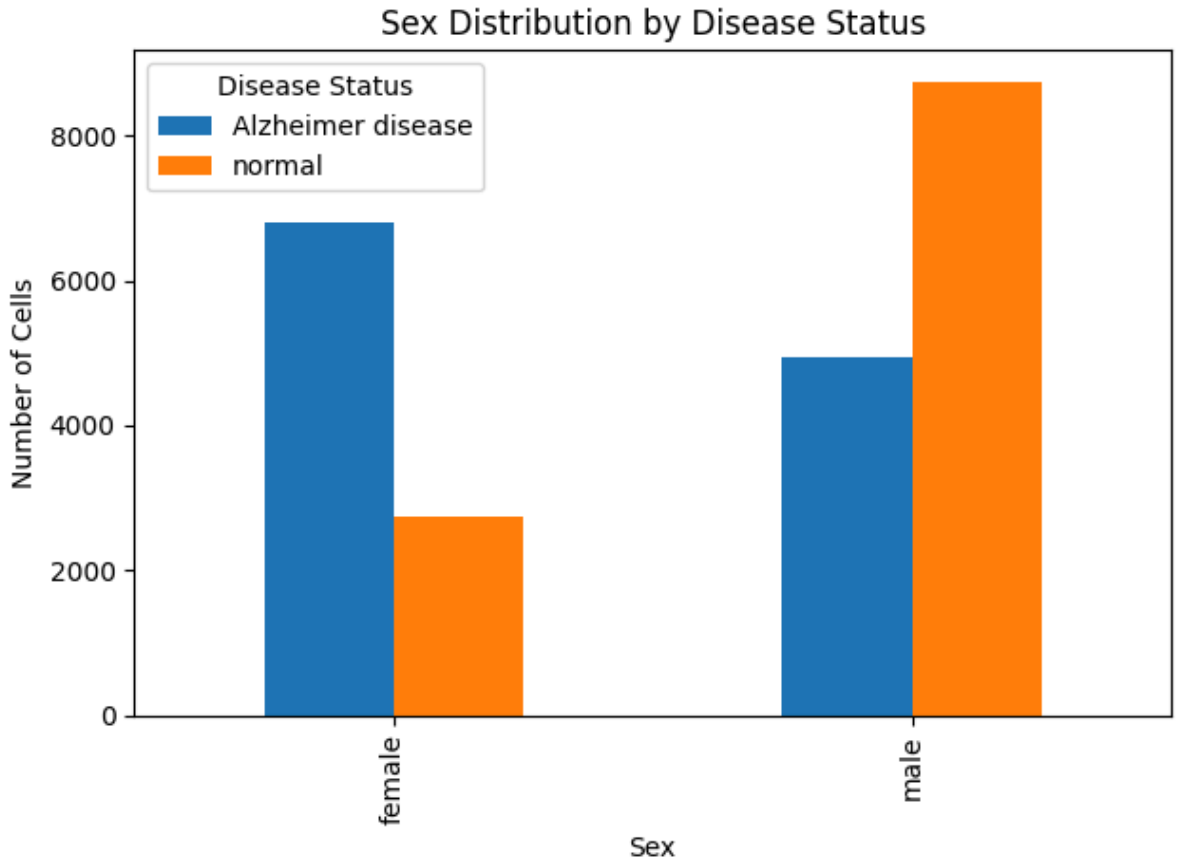
```
Top differentially expressed genes in In4_LHX6-SST-NPY:
```



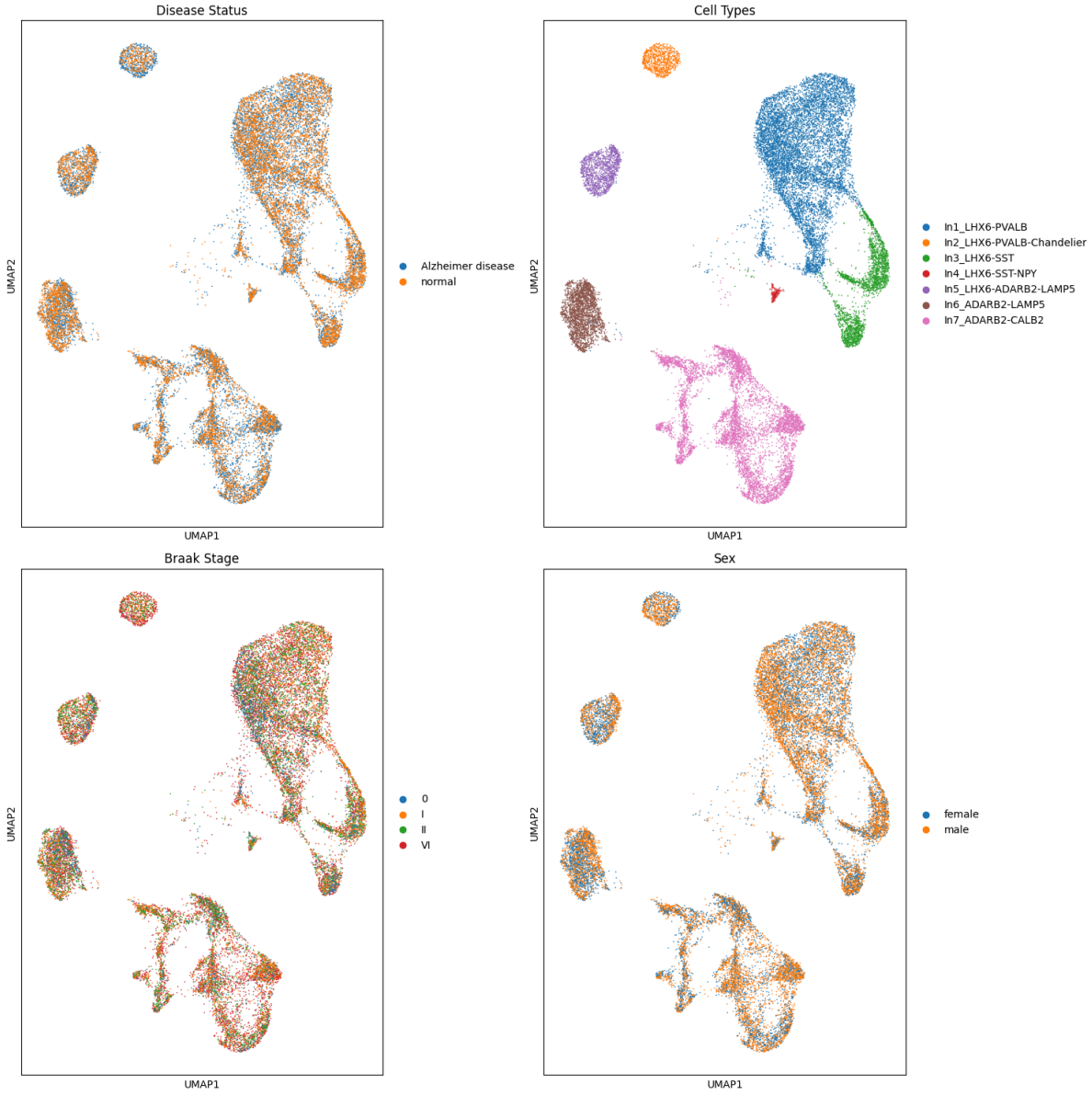

```
=== STEP 5: Demographic Analysis ===
```




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




```
=== STEP 6: Dimensional Reduction Visualization ===
```




```
Results saved to analysis_results/
```

In [ ]: