```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.table import Table

def load_data():
    # Load the dataset
    return pd.read_csv('customer_data_updated.csv')

def preprocess_data(data):
    # Convert 'Email_Engagement_Level' from categorical to numeric
    engagement_levels = {'Low': 1, 'Medium': 2, 'High': 3}
    data['Email_Engagement_Level'] = data['Email_Engagement_Level'].map(engagement_levels)
    return data

def analyze_demographics(data):
    # Analyze demographic data for average spending by age group
    try:
        age_spending = data.groupby('Age')['Average_Spend'].mean()
        plt.figure(figsize=(15, 6))
        # print(age_spending.index, age_spending.values)
        sns.barplot(x=age_spending.index, y=age_spending.values)
        plt.title('Average Spending by Age Group')
        plt.xlabel('Age')
        plt.ylabel('Average Spending')
        plt.show()
    except Exception as e:
        print(f"Failed to generate Spending by Age Group: {e}")

def identify_and_visualize_segments(data):
    try:
        # Identify high-value segments based on high income
        high_value_segments = data[data['Income'] > data['Income'].quantile(0.75)]

        # Analyzing risk of churn by low engagement (low session duration and frequency of
visits)
        at_risk_segments = data[(data['Session_Duration'] <
data['Session_Duration'].quantile(0.25)) &
                                (data['Frequency_of_Visits'] <
data['Frequency_of_Visits'].quantile(0.25))]

        # Visualizing High-Value Segments
        plt.figure(figsize=(10, 6))
        sns.histplot(high_value_segments['Income'], color='blue', kde=True, label='High Value
Segments')
        plt.title('Distribution of Income for High Value Segments')
        plt.xlabel('Income')
        plt.ylabel('Frequency')
        plt.legend()
        plt.show()
```

```
        # Visualizing At-Risk Segments
        plt.figure(figsize=(10, 6))
        sns.histplot(at_risk_segments['Session_Duration'], color='red', kde=True, label='At
Risk Segments - Session Duration')
        plt.title('Distribution of Session Duration for At Risk Segments')
        plt.xlabel('Session Duration')
        plt.ylabel('Frequency')
        plt.legend()
        plt.show()

        # Scatter plot visualization of high-value and at-risk segments
        plt.figure(figsize=(12, 8))
        # Plot all customers
        sns.scatterplot(x='Income', y='Session_Duration', data=data, color='grey', alpha=0.5,
label='All Customers')
        # Overlay high-value segments
        sns.scatterplot(x='Income', y='Session_Duration', data=high_value_segments,
color='blue', s=100, label='High Value Segments')
        # Overlay at-risk segments
        sns.scatterplot(x='Income', y='Session_Duration', data=at_risk_segments, color='red',
s=100, label='At Risk Segments - Low Session Duration')
        # Enhance the plot
        plt.title('Relationship between Income and Session Duration across Customer
Segments')
        plt.xlabel('Income')
        plt.ylabel('Session Duration')
        plt.legend()
        plt.grid(True)
        plt.show()

        # Identify customers who are both high-value and at-risk
        high_value_at_risk = pd.merge(high_value_segments, at_risk_segments, how='inner')
        columns_to_display = ['fname', 'lname', 'Location', 'Age', 'Income',
'Email_Engagement_Level', 'Interests', 'Values', 'Lifestyle']

        if not high_value_at_risk.empty:
            display_table(high_value_at_risk[columns_to_display])
        else:
            print("No customers are both high-value and at-risk based on the given
criteria.")
    except Exception as e:
        print(f"Failed to identify segments due to: {e}")

def display_table(data):
    # Create figure and axis
    fig, ax = plt.subplots(figsize=(14, 3.5))  # Adjust the figure size to fit the table
properly
    ax.axis('off')  # Hide the axis
```

```
    # Creating the table inside the plot
    the_table = ax.table(cellText=data.values, colLabels=data.columns, loc='upper center',
cellLoc='center', fontsize=14)

    # Adjusting column widths based on a fixed scale factor or predetermined widths
    # This part is a bit tricky in matplotlib as it doesn't automatically adjust based on
content
    col_widths = [max(len(str(x)) for x in data[col])*0.1 for col in data.columns]  # Example
heuristic
    for i, width in enumerate(col_widths):
        the_table.auto_set_column_width(col=i)

    # Shading the header row
    for (i, col_label), cell in the_table.get_celld().items():
        if i == 0:  # Only header row
            cell.set_facecolor('#40466e')
            cell.set_text_props(color='white')

    # Adding a title at the top of the figure
    fig.suptitle('Customer Segments: High-Value and At-Risk', fontsize=16, fontweight='bold')

    plt.show()

def visualize_engagement_levels(data):
    # Generate a heatmap for engagement levels
    try:
        pivot_table = data.pivot_table(index='Age', columns='Gender',
values='Email_Engagement_Level', aggfunc='mean')
        plt.figure(figsize=(8, 30))
        sns.heatmap(pivot_table, annot=True)
        plt.title('Engagement Levels by Age and Gender')
        plt.xlabel('Gender')
        plt.ylabel('Age')
        plt.show()
    except Exception as e:
        print(f"Failed to generate Engagement heatmap: {e}")

def main():
    data = load_data()
    data = preprocess_data(data)
    analyze_demographics(data)
    identify_and_visualize_segments(data)
    visualize_engagement_levels(data)

if __name__ == '__main__':
    main()
```