Scatter Plot

Estimated time needed: **45** minutes

Overview

In this lab, you will focus on creating and interpreting scatter plots to visualize relationships between variables and trends in the dataset. The provided dataset will be directly loaded into a pandas DataFrame, and various scatter plot-related visualizations will be created to explore developer trends, compensation, and preferences.

Objectives

In this lab, you will:

- Create and analyze scatter plots to examine relationships between variables.
- Use scatter plots to identify trends and patterns in the dataset.
- Focus on visualizations centered on scatter plots for better data-driven insights.

Setup: Working with the Database

Install and import the required libraries

In [1]:
```
!pip install pandas
!pip install matplotlib

import pandas as pd
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.12/site-packages (2.3.0)
Requirement already satisfied: numpy>=1.26.0 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.12/site-packages (fr
om pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas)
(2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.12/site-packages (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-packages (from matplotl
ib) (2.3.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib) (1.17.0)
```

Step 1: Load the dataset

In [2]:
```python
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520flu

df = pd.read_csv(file_path)
```

In [3]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np # For NaN handling and dummy data

# --- Setup: Load the dataset ---
print("--- Setup: Loading the dataset ---")

# The PDF specifies loading from a URL.
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520flu

try:
    df = pd.read_csv(file_path, na_values=['NA', 'N/A', 'nan', 'NaN', 'null', 'Null', '', ' ', '-']
    print(f"Dataset loaded successfully from: {file_path}")
    print(f"Initial DataFrame shape: {df.shape}")
    print(f"Initial DataFrame columns: {df.columns.tolist()}")
except Exception as e:
    print(f"ERROR: Could not load dataset from URL: {e}")
    print("Creating a dummy DataFrame for demonstration purposes...")
    # Create a dummy DataFrame if URL loading fails
    num_rows_dummy = 200
    data = {
        'ResponseId': range(1, num_rows_dummy + 1),
        'Age': np.random.choice(['Under 18 years old', '18-24 years old', '25-34 years old', '35-44
        'ConvertedCompYearly': np.random.normal(loc=90000, scale=40000, size=num_rows_dummy),
        'JobSatPoints_6': np.random.randint(1, 6, size=num_rows_dummy), # Assuming 1-5 scale for sa
        'YearsCodePro': np.random.randint(0, 30, size=num_rows_dummy),
        'LanguageHaveWorkedWith': [';'.join(np.random.choice(['Python', 'JavaScript', 'Java', 'C++'
        'Employment': np.random.choice(['Employed, full-time', 'Employed, part-time', 'Student', 'I
        'Country': np.random.choice(['United States', 'India', 'Germany', 'United Kingdom', 'Canada
    }
    df = pd.DataFrame(data)
    print("Dummy DataFrame created.")

# --- Data Cleaning and Preprocessing ---
```

```python
print("\n--- Data Cleaning and Preprocessing ---")

# Convert relevant columns to numeric, coercing errors to NaN and filling with median
numeric_cols = ['ConvertedCompYearly', 'JobSatPoints_6', 'YearsCodePro']
for col in numeric_cols:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')
        if df[col].isnull().any():
            median_val = df[col].median()
            df[col].fillna(median_val, inplace=True)
            print(f"Cleaned '{col}': Imputed NaNs with median: {median_val:.2f}")
    else:
        print(f"WARNING: Numerical column '{col}' not found in DataFrame.")

# Map 'Age' to a numeric approximation for scatter plots and fill NaNs
age_numeric_mapping = {
    'Under 18 years old': 17, '18-24 years old': 21, '25-34 years old': 29,
    '35-44 years old': 39, '45-54 years old': 49, '55-64 years old': 59,
    '65 years or older': 65, 'Prefer not to say': np.nan # Treat "Prefer not to say" as NaN
}
if 'Age' in df.columns:
    df['Age_Numeric'] = df['Age'].map(age_numeric_mapping)
    if df['Age_Numeric'].isnull().any():
        df['Age_Numeric'].fillna(df['Age_Numeric'].median(), inplace=True)
        print("Created and imputed 'Age_Numeric' column.")
else:
    print("WARNING: 'Age' column not found, 'Age_Numeric' will not be created.")

# Clean categorical columns (e.g., strip whitespace, replace common NaN strings)
categorical_cols = ['LanguageHaveWorkedWith', 'Employment', 'Country']
for col in categorical_cols:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip()
        df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
        # For plotting, we often drop rows with NaN in key categorical columns
        # For simplicity here, we'll let plotting functions handle it, or we could use mode imputat
    else:
        print(f"WARNING: Categorical column '{col}' not found in DataFrame.")
```

```
--- Setup: Loading the dataset ---
Dataset loaded successfully from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
d/n01PQ9pSmiRX6520flujwQ/survey-data.csv
Initial DataFrame shape: (65437, 114)
Initial DataFrame columns: ['ResponseId', 'MainBranch', 'Age', 'Employment', 'RemoteWork', 'Check',
'CodingActivities', 'EdLevel', 'LearnCode', 'LearnCodeOnline', 'TechDoc', 'YearsCode', 'YearsCodePr
o', 'DevType', 'OrgSize', 'PurchaseInfluence', 'BuyNewTool', 'BuildvsBuy', 'TechEndorse', 'Country',
'Currency', 'CompTotal', 'LanguageHaveWorkedWith', 'LanguageWantToWorkWith', 'LanguageAdmired', 'Dat
abaseHaveWorkedWith', 'DatabaseWantToWorkWith', 'DatabaseAdmired', 'PlatformHaveWorkedWith', 'Platfo
rmWantToWorkWith', 'PlatformAdmired', 'WebframeHaveWorkedWith', 'WebframeWantToWorkWith', 'WebframeA
dmired', 'EmbeddedHaveWorkedWith', 'EmbeddedWantToWorkWith', 'EmbeddedAdmired', 'MiscTechHaveWorkedW
ith', 'MiscTechWantToWorkWith', 'MiscTechAdmired', 'ToolsTechHaveWorkedWith', 'ToolsTechWantToWorkWi
th', 'ToolsTechAdmired', 'NEWCollabToolsHaveWorkedWith', 'NEWCollabToolsWantToWorkWith', 'NEWCollabT
oolsAdmired', 'OpSysPersonal use', 'OpSysProfessional use', 'OfficeStackAsyncHaveWorkedWith', 'Offic
eStackAsyncWantToWorkWith', 'OfficeStackAsyncAdmired', 'OfficeStackSyncHaveWorkedWith', 'OfficeStack
SyncWantToWorkWith', 'OfficeStackSyncAdmired', 'AISearchDevHaveWorkedWith', 'AISearchDevWantToWorkWi
th', 'AISearchDevAdmired', 'NEWSOSites', 'SOVisitFreq', 'SOAccount', 'SOPartFreq', 'SOHow', 'SOCom
m', 'AISelect', 'AISent', 'AIBen', 'AIAcc', 'AIComplex', 'AIToolCurrently Using', 'AIToolInterested
in Using', 'AIToolNot interested in Using', 'AINextMuch more integrated', 'AINextNo change', 'AINext
More integrated', 'AINextLess integrated', 'AINextMuch less integrated', 'AIThreat', 'AIEthics', 'AI
Challenges', 'TBranch', 'ICorPM', 'WorkExp', 'Knowledge_1', 'Knowledge_2', 'Knowledge_3', 'Knowledge
_4', 'Knowledge_5', 'Knowledge_6', 'Knowledge_7', 'Knowledge_8', 'Knowledge_9', 'Frequency_1', 'Freq
uency_2', 'Frequency_3', 'TimeSearching', 'TimeAnswering', 'Frustration', 'ProfessionalTech', 'Profe
ssionalCloud', 'ProfessionalQuestion', 'Industry', 'JobSatPoints_1', 'JobSatPoints_4', 'JobSatPoints
_5', 'JobSatPoints_6', 'JobSatPoints_7', 'JobSatPoints_8', 'JobSatPoints_9', 'JobSatPoints_10', 'Job
SatPoints_11', 'SurveyLength', 'SurveyEase', 'ConvertedCompYearly', 'JobSat']

--- Data Cleaning and Preprocessing ---
Cleaned 'ConvertedCompYearly': Imputed NaNs with median: 65000.00
Cleaned 'JobSatPoints_6': Imputed NaNs with median: 20.00
Cleaned 'YearsCodePro': Imputed NaNs with median: 8.00
Created and imputed 'Age_Numeric' column.
```

```
/tmp/ipykernel_3192/1115976739.py:45: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_3192/1115976739.py:45: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_3192/1115976739.py:45: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_3192/1115976739.py:59: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df['Age_Numeric'].fillna(df['Age_Numeric'].median(), inplace=True)
/tmp/ipykernel_3192/1115976739.py:69: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
/tmp/ipykernel_3192/1115976739.py:69: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
/tmp/ipykernel_3192/1115976739.py:69: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.
```

```
df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
```

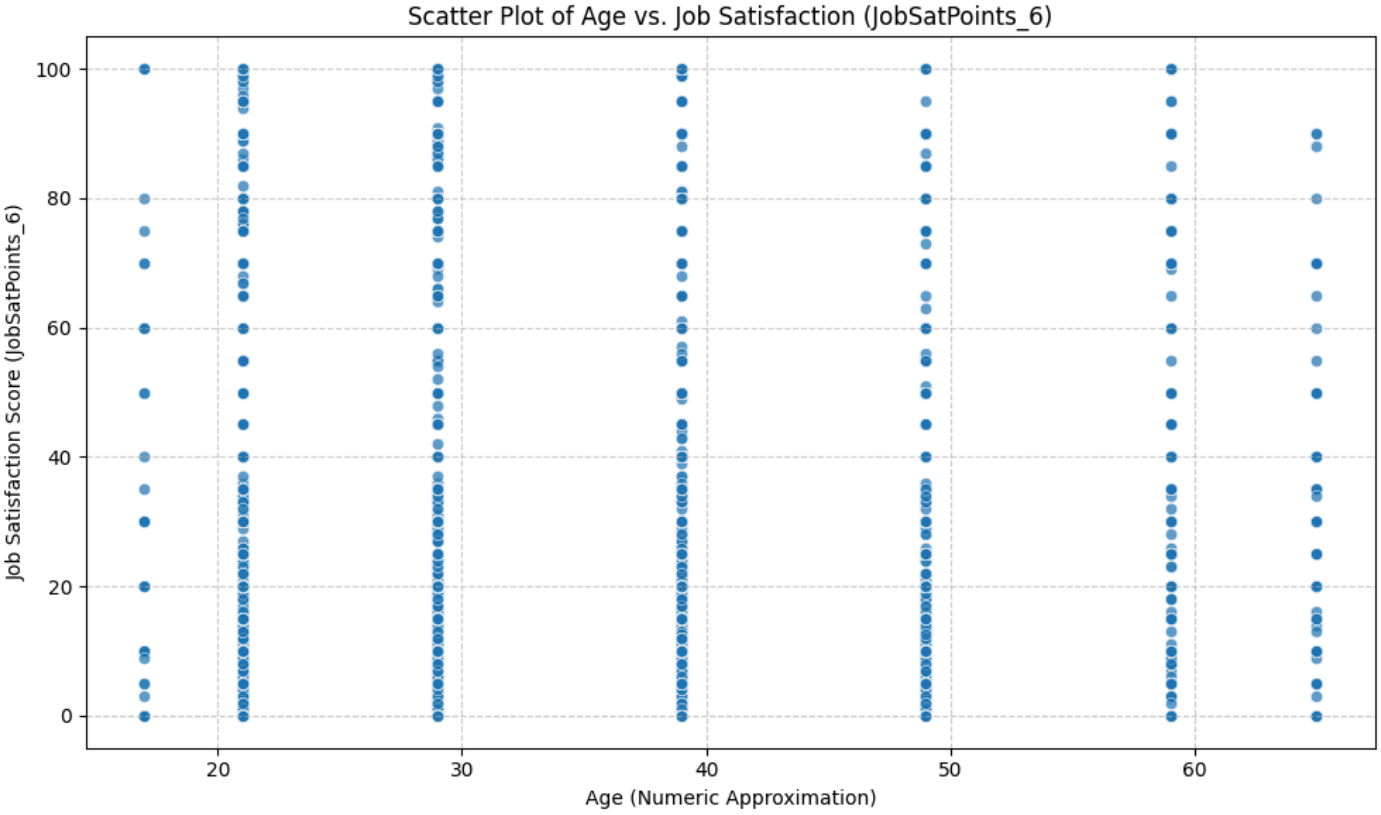
Task 1: Exploring Relationships with Scatter Plots

1. Scatter Plot for Age vs. Job Satisfaction

Visualize the relationship between respondents' age (`Age`) and job satisfaction (`JobSatPoints_6`). Use this plot to identify any patterns or trends.

In [4]:
```
## Write your code here
# --- Task 1: Exploring Relationships with Scatter Plots ---
print("\n--- Task 1: Exploring Relationships with Scatter Plots ---")

# 1. Scatter Plot for Age vs. Job Satisfaction
if 'Age_Numeric' in df.columns and 'JobSatPoints_6' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Age_Numeric', y='JobSatPoints_6', data=df, alpha=0.7)
    plt.title('Scatter Plot of Age vs. Job Satisfaction (JobSatPoints_6)')
    plt.xlabel('Age (Numeric Approximation)')
    plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
    print("Task 1.1: Scatter plot for Age vs. Job Satisfaction plotted.")
else:
    print("Skipping Task 1.1: Required columns 'Age_Numeric' or 'JobSatPoints_6' not found or not r
```

```
--- Task 1: Exploring Relationships with Scatter Plots ---
```



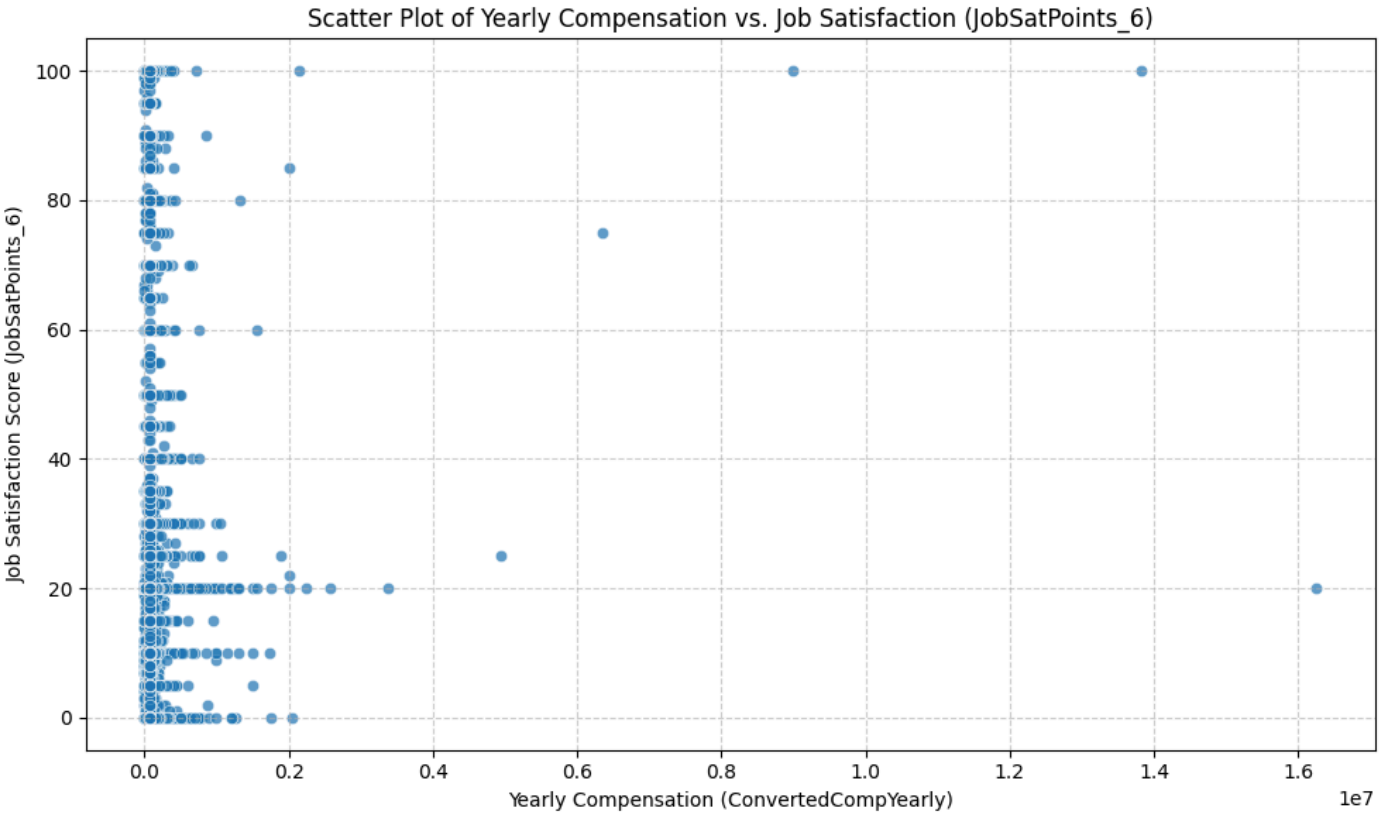

```
Task 1.1: Scatter plot for Age vs. Job Satisfaction plotted.
```

2. Scatter Plot for Compensation vs. Job Satisfaction

Explore the relationship between yearly compensation (`ConvertedCompYearly`) and job satisfaction (`JobSatPoints_6`) using a scatter plot.

In [5]:
```
## Write your code here
# 2. Scatter Plot for Compensation vs. Job Satisfaction
if 'ConvertedCompYearly' in df.columns and 'JobSatPoints_6' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='ConvertedCompYearly', y='JobSatPoints_6', data=df, alpha=0.7)
    plt.title('Scatter Plot of Yearly Compensation vs. Job Satisfaction (JobSatPoints_6)')
    plt.xlabel('Yearly Compensation (ConvertedCompYearly)')
```

```
    plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
    print("Task 1.2: Scatter plot for Compensation vs. Job Satisfaction plotted.")
else:
    print("Skipping Task 1.2: Required columns 'ConvertedCompYearly' or 'JobSatPoints_6' not found
```




```
Task 1.2: Scatter plot for Compensation vs. Job Satisfaction plotted.
```

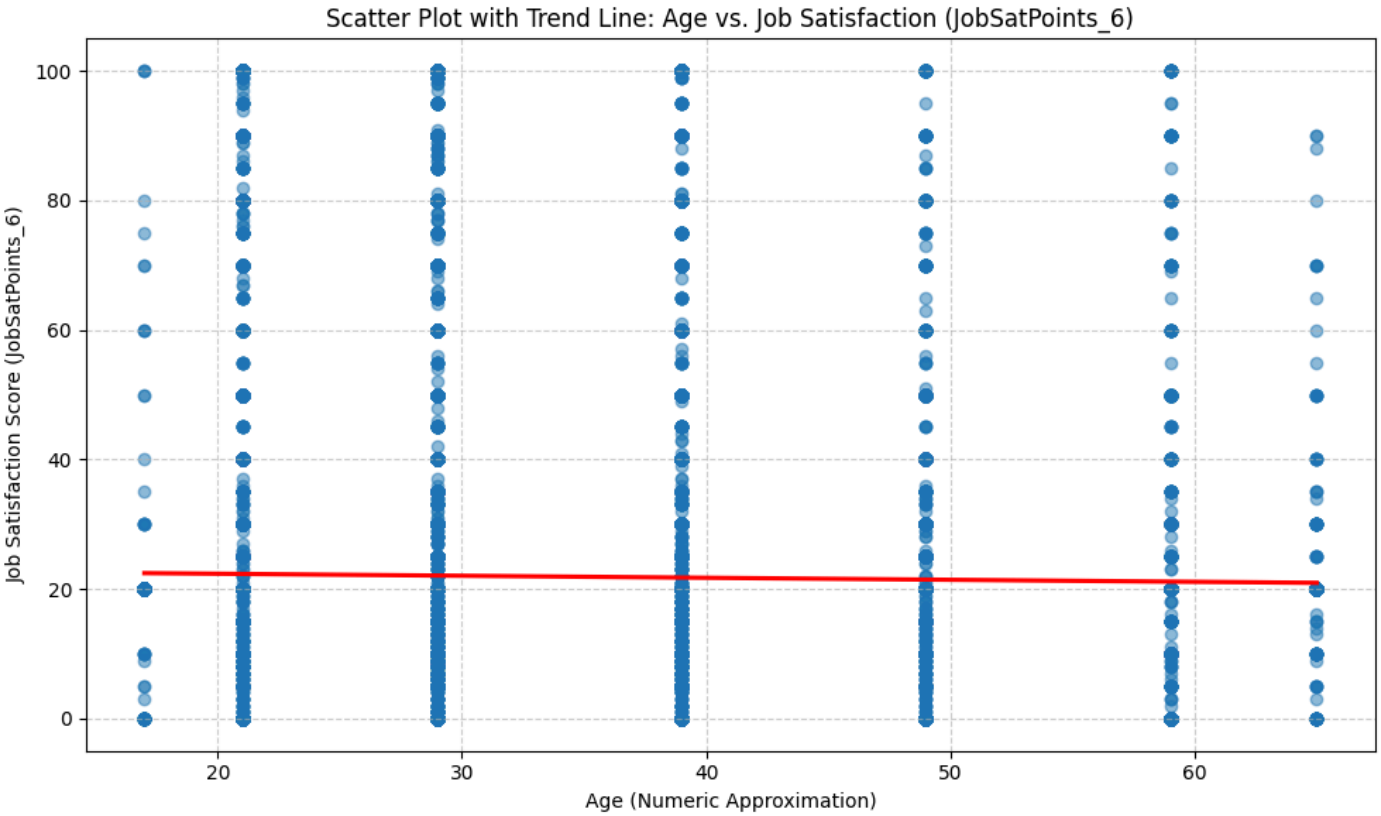
Task 2: Enhancing Scatter Plots

1. Scatter Plot with Trend Line for Age vs. Job Satisfaction

Add a regression line to the scatter plot of Age vs. JobSatPoints_6 to highlight trends in the data.

In [6]:
```
## Write your code here
# --- Task 2: Enhancing Scatter Plots ---
print("\n--- Task 2: Enhancing Scatter Plots ---")

# 1. Scatter Plot with Trend Line for Age vs. Job Satisfaction
if 'Age_Numeric' in df.columns and 'JobSatPoints_6' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.regplot(x='Age_Numeric', y='JobSatPoints_6', data=df, scatter_kws={'alpha':0.5}, line_kws={
    plt.title('Scatter Plot with Trend Line: Age vs. Job Satisfaction (JobSatPoints_6)')
    plt.xlabel('Age (Numeric Approximation)')
    plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
    print("Task 2.1: Scatter plot with trend line for Age vs. Job Satisfaction plotted.")
else:
    print("Skipping Task 2.1: Required columns 'Age_Numeric' or 'JobSatPoints_6' not found or not r
```

```
--- Task 2: Enhancing Scatter Plots ---
```

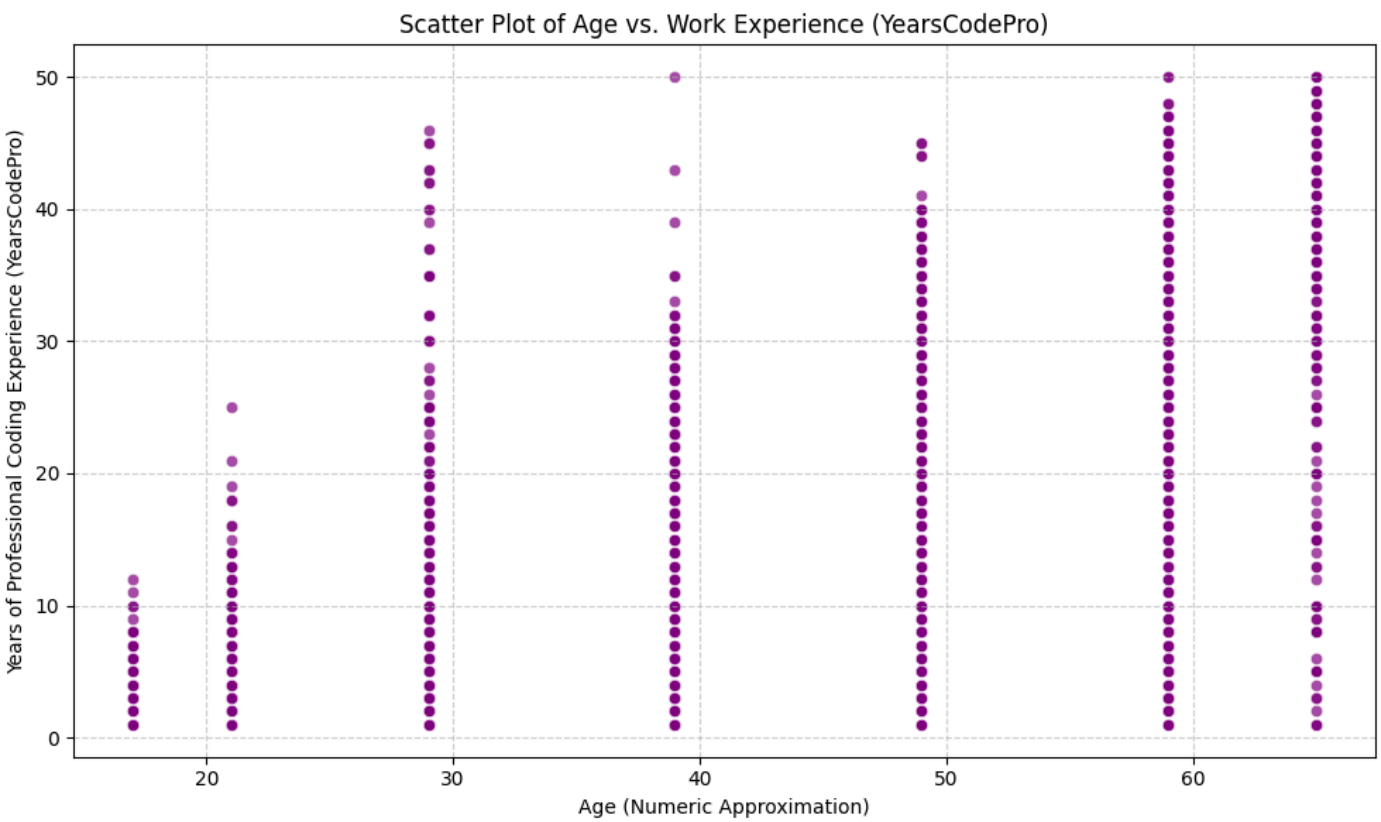



```
Task 2.1: Scatter plot with trend line for Age vs. Job Satisfaction plotted.
```

2. Scatter Plot for Age vs. Work Experience

Visualize the relationship between Age (`Age`) and Work Experience (`YearsCodePro`) using a scatter plot.

In [7]:
```
## Write your code here
# 2. Scatter Plot for Age vs. Work Experience
if 'Age_Numeric' in df.columns and 'YearsCodePro' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Age_Numeric', y='YearsCodePro', data=df, alpha=0.7, color='purple')
    plt.title('Scatter Plot of Age vs. Work Experience (YearsCodePro)')
    plt.xlabel('Age (Numeric Approximation)')
    plt.ylabel('Years of Professional Coding Experience (YearsCodePro)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
    print("Task 2.2: Scatter plot for Age vs. Work Experience plotted.")
else:
    print("Skipping Task 2.2: Required columns 'Age_Numeric' or 'YearsCodePro' not found or not rea
```




```
Task 2.2: Scatter plot for Age vs. Work Experience plotted.
```

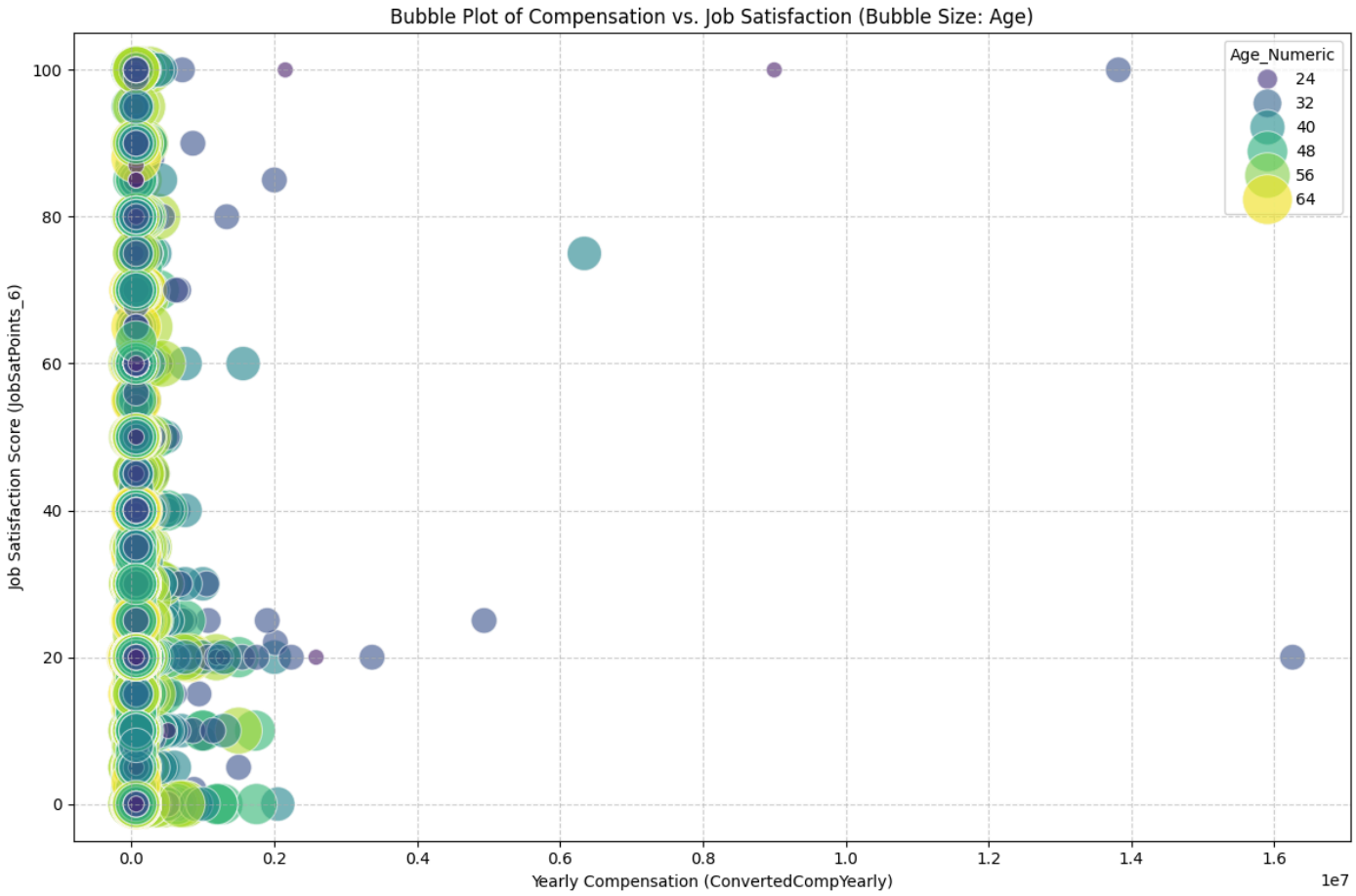
Task 3: Combining Scatter Plots with Additional Features

1. Bubble Plot of Compensation vs. Job Satisfaction with Age as Bubble Size

Create a bubble plot to explore the relationship between yearly compensation (`ConvertedCompYearly`) and job satisfaction (`JobSatPoints_6`), with bubble size representing age.

In [8]:
```
## Write your code here
# --- Task 3: Combining Scatter Plots with Additional Features ---
print("\n--- Task 3: Combining Scatter Plots with Additional Features ---")

# 1. Bubble Plot of Compensation vs. Job Satisfaction with Age as Bubble Size
if 'ConvertedCompYearly' in df.columns and 'JobSatPoints_6' in df.columns and 'Age_Numeric' in df.c
    # Drop rows with NaN in these critical columns for the plot
    df_bubble_plot = df.dropna(subset=['ConvertedCompYearly', 'JobSatPoints_6', 'Age_Numeric']).cop

    if not df_bubble_plot.empty:
        plt.figure(figsize=(12, 8))
        # 'sizes' argument controls the range of bubble sizes.
        # 'hue' adds color based on Age_Numeric for better visualization.
        sns.scatterplot(x='ConvertedCompYearly', y='JobSatPoints_6', size='Age_Numeric', data=df_bu
                        sizes=(20, 1000), alpha=0.6, hue='Age_Numeric', palette='viridis', legend='
        plt.title('Bubble Plot of Compensation vs. Job Satisfaction (Bubble Size: Age)')
        plt.xlabel('Yearly Compensation (ConvertedCompYearly)')
        plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()
        print("Task 3.1: Bubble plot of Compensation vs. Job Satisfaction with Age as bubble size p
    else:
        print("Skipping Task 3.1: Not enough valid data in 'ConvertedCompYearly', 'JobSatPoints_6',
else:
    print("Skipping Task 3.1: Required columns 'ConvertedCompYearly', 'JobSatPoints_6', or 'Age_Num
```

```
--- Task 3: Combining Scatter Plots with Additional Features ---
```



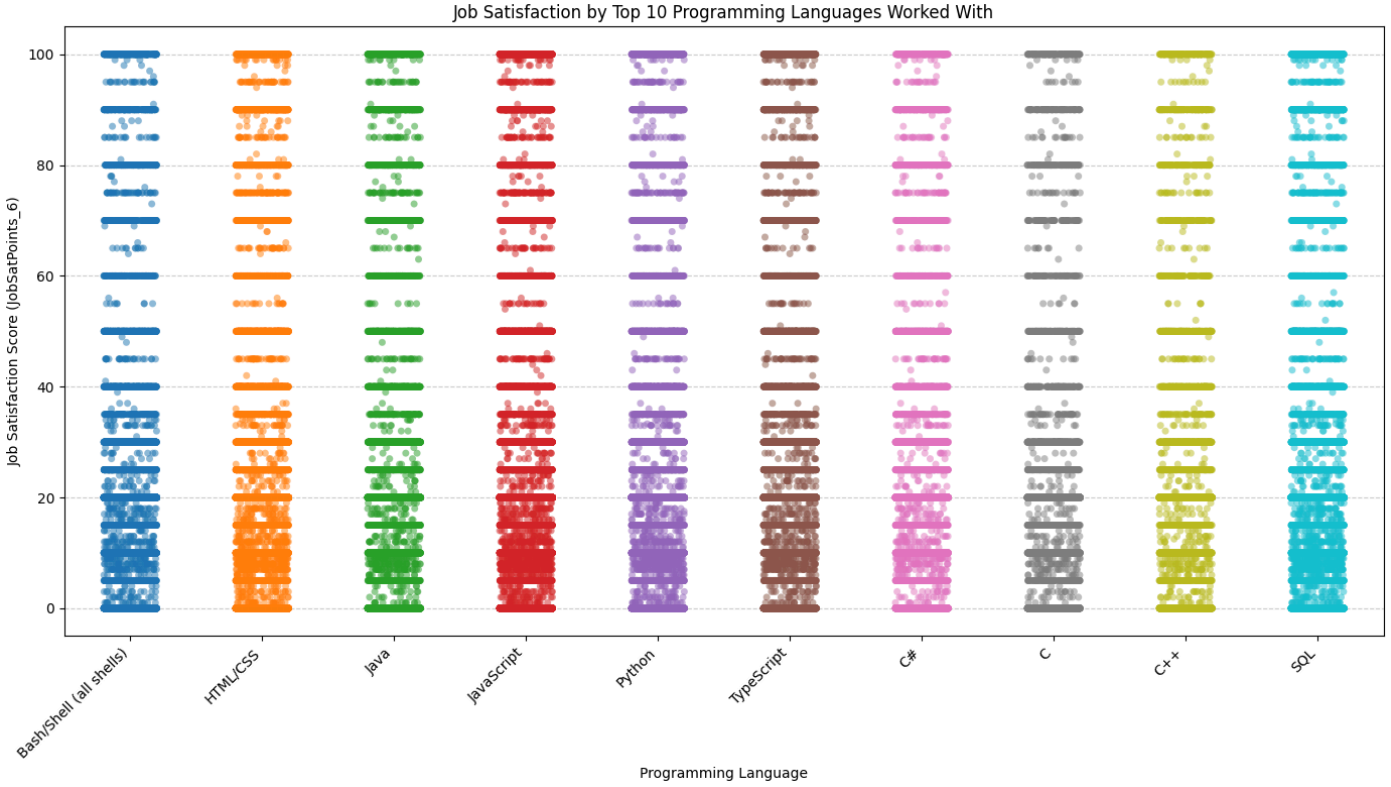
Task 3.1: Bubble plot of Compensation vs. Job Satisfaction with Age as bubble size plotted.

2. Scatter Plot for Popular Programming Languages by Job Satisfaction

Visualize the popularity of programming languages (`LanguageHaveWorkedWith`) against job satisfaction using a scatter plot. Use points to represent satisfaction levels for each language.

In [9]:
```
## Write your code here
# 2. Scatter Plot for Popular Programming Languages by Job Satisfaction
if 'LanguageHaveWorkedWith' in df.columns and 'JobSatPoints_6' in df.columns:
    # To plot this, we need to handle multi-valued 'LanguageHaveWorkedWith'
    # First, explode the column to have one language per row
    df_languages = df.dropna(subset=['LanguageHaveWorkedWith', 'JobSatPoints_6']).copy()
    if not df_languages.empty:
        df_languages['Language'] = df_languages['LanguageHaveWorkedWith'].str.split(';')
        df_exploded_languages = df_languages.explode('Language')
        df_exploded_languages['Language'] = df_exploded_languages['Language'].str.strip()

        # Get the top N most frequent languages for better visualization
        top_n_languages = df_exploded_languages['Language'].value_counts().head(10).index.tolist()
        df_top_languages = df_exploded_languages[df_exploded_languages['Language'].isin(top_n_langu

        if not df_top_languages.empty:
            plt.figure(figsize=(14, 8))
            # Use stripplot for categorical X with numerical Y, it's like a scatter plot for catego
            # or use scatterplot with jitter for better spread
            sns.stripplot(x='Language', y='JobSatPoints_6', data=df_top_languages, jitter=0.2, alph
            plt.title('Job Satisfaction by Top 10 Programming Languages Worked With')
            plt.xlabel('Programming Language')
            plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
            plt.xticks(rotation=45, ha='right')
            plt.grid(axis='y', linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 3.2: Scatter plot for Popular Programming Languages by Job Satisfaction plo
        else:
            print("Skipping Task 3.2: Not enough valid data for top languages or Job Satisfaction f
    else:
        print("Skipping Task 3.2: Not enough valid data in 'LanguageHaveWorkedWith' or 'JobSatPoint
else:
    print("Skipping Task 3.2: Required columns 'LanguageHaveWorkedWith' or 'JobSatPoints_6' not fou
```

```
/tmp/ipykernel_3192/2933618128.py:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `
x` variable to `hue` and set `legend=False` for the same effect.

  sns.stripplot(x='Language', y='JobSatPoints_6', data=df_top_languages, jitter=0.2, alpha=0.5, pale
tte='tab10')
```



Task 3.2: Scatter plot for Popular Programming Languages by Job Satisfaction plotted.

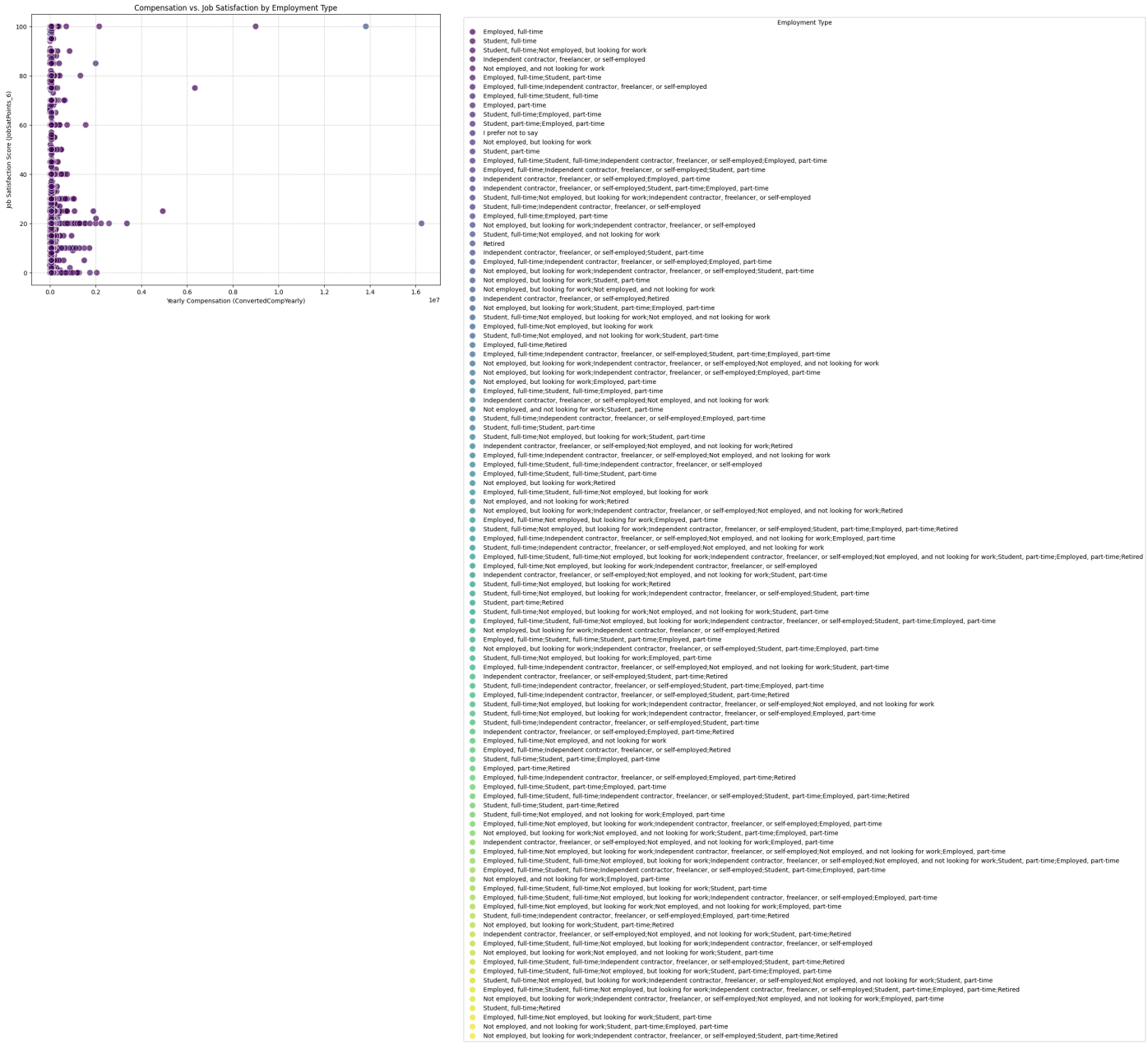
Task 4: Scatter Plot Comparisons Across Groups

1. Scatter Plot for Compensation vs. Job Satisfaction by Employment Type

Visualize the relationship between yearly compensation (`ConvertedCompYearly`) and job satisfaction (`JobSatPoints_6`), categorized by employment type (`Employment`). Use color coding or markers to differentiate between employment types.

In [10]:
```
## Write your code here
# --- Task 4: Scatter Plot Comparisons Across Groups ---
print("\n--- Task 4: Scatter Plot Comparisons Across Groups ---")

# 1. Scatter Plot for Compensation vs. Job Satisfaction by Employment Type
if 'ConvertedCompYearly' in df.columns and 'JobSatPoints_6' in df.columns and 'Employment' in df.co
    df_employment_scatter = df.dropna(subset=['ConvertedCompYearly', 'JobSatPoints_6', 'Employment'
    if not df_employment_scatter.empty:
        plt.figure(figsize=(12, 8))
        sns.scatterplot(x='ConvertedCompYearly', y='JobSatPoints_6', hue='Employment', data=df_empl
                        palette='viridis', alpha=0.7, s=100) # s for size of points
        plt.title('Compensation vs. Job Satisfaction by Employment Type')
        plt.xlabel('Yearly Compensation (ConvertedCompYearly)')
        plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.legend(title='Employment Type', bbox_to_anchor=(1.05, 1), loc='upper left')
        plt.tight_layout()
        plt.show()
        print("Task 4.1: Scatter plot for Compensation vs. Job Satisfaction by Employment Type plot
    else:
        print("Skipping Task 4.1: Not enough valid data in 'ConvertedCompYearly', 'JobSatPoints_6',
else:
    print("Skipping Task 4.1: Required columns not found or not ready.")
```

```
--- Task 4: Scatter Plot Comparisons Across Groups ---
/tmp/ipykernel_3192/544945165.py:17: UserWarning: Tight layout not applied. The left and right margi
ns cannot be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```



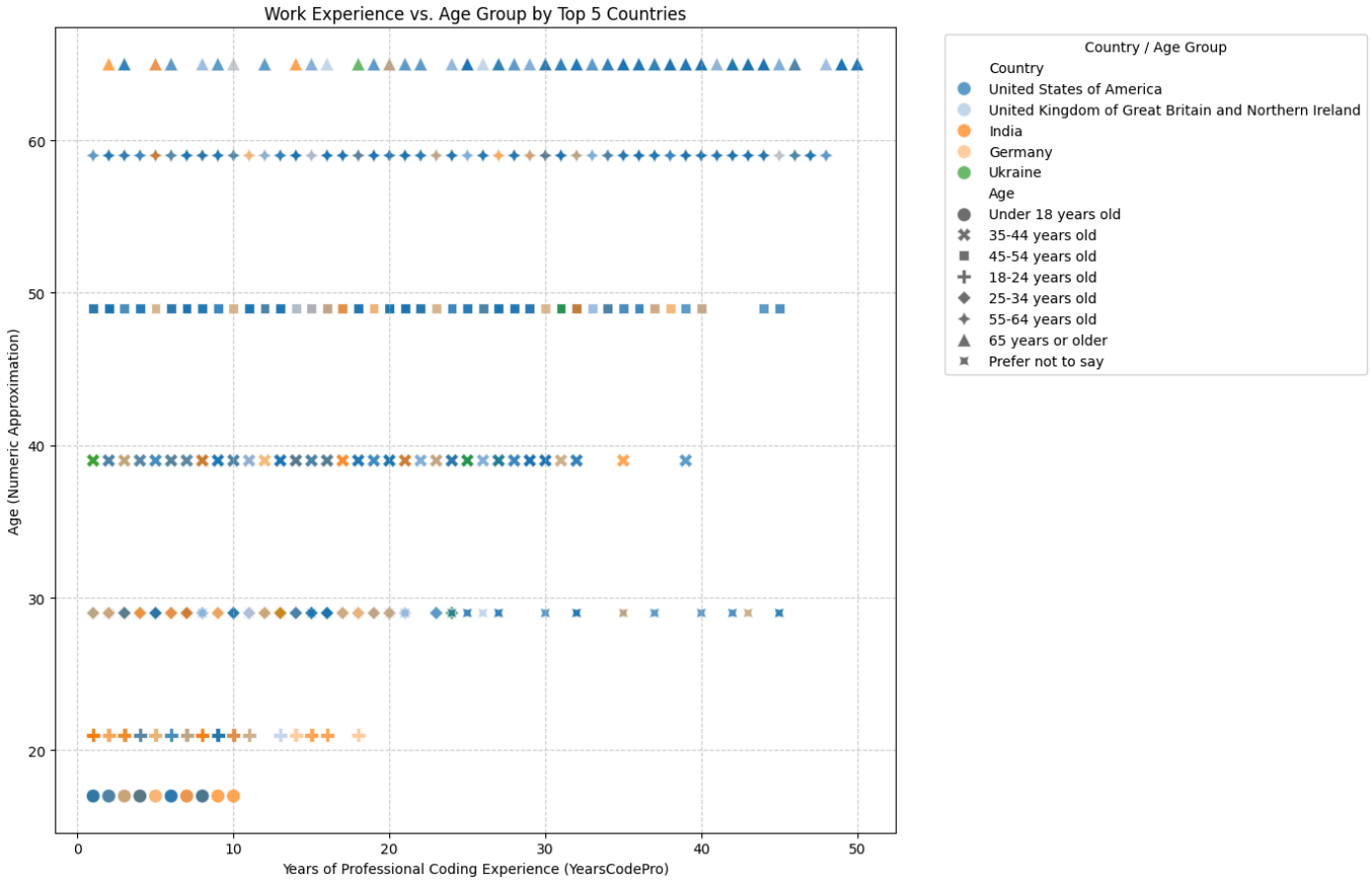
Task 4.1: Scatter plot for Compensation vs. Job Satisfaction by Employment Type plotted.

2. Scatter Plot for Work Experience vs. Age Group by Country

Compare work experience (`YearsCodePro`) across different age groups (`Age`) and countries (`Country`). Use colors to represent different countries and markers for age groups.

In [11]:
```
## Write your code here
# 2. Scatter Plot for Work Experience vs. Age Group by Country
if 'YearsCodePro' in df.columns and 'Age_Numeric' in df.columns and 'Country' in df.columns:
    df_country_scatter = df.dropna(subset=['YearsCodePro', 'Age_Numeric', 'Country']).copy()
    if not df_country_scatter.empty:
        # Get top N countries to avoid too many hues, or group less common ones
        top_n_countries = df_country_scatter['Country'].value_counts().head(5).index.tolist()
        df_top_countries_scatter = df_country_scatter[df_country_scatter['Country'].isin(top_n_coun

        if not df_top_countries_scatter.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(x='YearsCodePro', y='Age_Numeric', hue='Country', style='Age', data=df_
                            palette='tab20', s=100, alpha=0.7)
            plt.title('Work Experience vs. Age Group by Top 5 Countries')
            plt.xlabel('Years of Professional Coding Experience (YearsCodePro)')
            plt.ylabel('Age (Numeric Approximation)')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.legend(title='Country / Age Group', bbox_to_anchor=(1.05, 1), loc='upper left')
            plt.tight_layout()
            plt.show()
            print("Task 4.2: Scatter plot for Work Experience vs. Age Group by Country plotted.")
        else:
            print("Skipping Task 4.2: Not enough valid data for top countries for plotting.")
    else:
```

```
        print("Skipping Task 4.2: Not enough valid data in 'YearsCodePro', 'Age_Numeric', or 'Count
else:
    print("Skipping Task 4.2: Required columns not found or not ready.")

print("\n--- Lab Completion Summary ---")
print("All scatter plot tasks have been attempted. Please review the plots and console output for a
```




```
Task 4.2: Scatter plot for Work Experience vs. Age Group by Country plotted.

--- Lab Completion Summary ---
All scatter plot tasks have been attempted. Please review the plots and console output for any warni
ngs or skipped tasks.
```

Final Step: Review

With these scatter plots, you will have analyzed data relationships across multiple dimensions, including compensation, job satisfaction, employment types, and demographics, to uncover meaningful trends in the developer community.

Summary

After completing this lab, you will be able to:

- Analyze how numerical variables relate across specific groups, such as employment types and countries.
- Use scatter plots effectively to represent multiple variables with color, size, and markers.
- Gain insights into compensation, satisfaction, and demographic trends using advanced scatter plot techniques.

Authors:

Ayushi Jain

Other Contributors:

- Rav Ahuja
- Lakshmi Holla
- Malika