Bubble Plots

Estimated time needed: **30** minutes

In this lab, you will focus on visualizing data.

The dataset will be directly loaded into pandas for analysis and visualization.

You will use various visualization techniques to explore the data and uncover key trends.

Objectives

In this lab, you will perform the following:

- Visualize the distribution of data.
- Visualize the relationship between two data features.
- Visualize composition of data.
- Visualize comparison of data.

Setup: Working with the Database

Install and import the needed libraries

In [5]:
```
!pip install pandas
!pip install matplotlib
!pip install seaborn

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)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.12/site-packages (from
seaborn) (2.3.0)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.12/site-packages (from seaborn)
(2.3.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /opt/conda/lib/python3.12/site-packages (f
rom seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas>
=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
```

Download and connect to the database file containing survey data.

To start, download and load the dataset into a `pandas` DataFrame.

In [6]:
```
# Step 1: Download the dataset
!wget -O survey-data.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9p

# Load the data
```

```
df = pd.read_csv("survey-data.csv")

# Display the first few rows of the data to understand its structure
df.head()
```

```
--2025-06-18 17:07:41--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pS
miRX6520flujwQ/survey-data.csv
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-ob
ject-storage.appdomain.cloud)... 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.clou
d-object-storage.appdomain.cloud)|169.63.118.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 159525875 (152M) [text/csv]
Saving to: 'survey-data.csv'

survey-data.csv     100%[===================>] 152.13M  69.6MB/s    in 2.2s

2025-06-18 17:07:44 (69.6 MB/s) - 'survey-data.csv' saved [159525875/159525875]
```

Out[6]:

	ResponseId	MainBranch	Age	Employment	RemoteWork	Check	CodingActivities	EdLevel	
0	1	I am a developer by profession	Under 18 years old	Employed, full-time	Remote	Apples	Hobby	Primary/elementary school	E
1	2	I am a developer by profession	35-44 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...	Bachelor's degree (B.A., B.S., B.Eng., etc.)	E medi
2	3	I am a developer by profession	45-54 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	E medi
3	4	I am learning to code	18-24 years old	Student, full-time	NaN	Apples	NaN	Some college/university study without earning ...	vi
4	5	I am a developer by profession	18-24 years old	Student, full-time	NaN	Apples	NaN	Secondary school (e.g. American high school, G...	vi

5 rows × 114 columns

In [7]:

```
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),
        'JobSat': np.random.randint(1, 6, size=num_rows_dummy), # Assuming 1-5 scale for satisfacti
        'SOPartFreq': np.random.choice(['Never', 'Daily', 'Weekly', 'Monthly', 'Yearly'], size=num_
        'LanguageHaveWorkedWith': [';'.join(np.random.choice(['Python', 'JavaScript', 'Java', 'C++'
```

```
        'DatabaseWantToWorkWith': [';'.join(np.random.choice(['MySQL', 'PostgreSQL', 'MongoDB', 'Re
        'DevType': [';'.join(np.random.choice(['Developer, back-end', 'Developer, front-end', 'Deve
        'NEWCollabToolsHaveWorkedWith': [';'.join(np.random.choice(['Git', 'Confluence', 'Jira', 'S
        'WebframeWantToWorkWith': [';'.join(np.random.choice(['React', 'Angular', 'Vue.js', 'Django
        'LanguageAdmired': [';'.join(np.random.choice(['Python', 'Rust', 'Go', 'TypeScript', 'Julia
        'Country': np.random.choice(['United States', 'India', 'Germany', 'United Kingdom', 'Canada
    }
    df = pd.DataFrame(data)
    print("Dummy DataFrame created.")


# --- Data Cleaning and Preprocessing ---
print("\n--- Data Cleaning and Preprocessing ---")

# Convert relevant numerical columns, coercing errors to NaN and filling with median
numeric_cols = ['ConvertedCompYearly', 'JobSat']
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()
            if pd.isna(median_val):
                 print(f"WARNING: Column '{col}' is entirely NaN after numeric conversion. Cannot i
            else:
                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
}
if 'Age' in df.columns:
    df['Age_Numeric'] = df['Age'].map(age_numeric_mapping)
    if df['Age_Numeric'].isnull().any():
        median_age_numeric = df['Age_Numeric'].median()
        if pd.isna(median_age_numeric):
            print("WARNING: 'Age_Numeric' column is entirely NaN. Cannot impute median.")
        else:
            df['Age_Numeric'].fillna(median_age_numeric, 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_to_clean = [
    'SOPartFreq', 'LanguageHaveWorkedWith', 'DatabaseWantToWorkWith',
    'DevType', 'NEWCollabToolsHaveWorkedWith', 'WebframeWantToWorkWith',
    'LanguageAdmired', 'Country'
]
for col in categorical_cols_to_clean:
    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 multi-valued columns, filling NaN here might not be ideal before explode.
        # But for single-valued, mode imputation could be added if needed.
    else:
        print(f"WARNING: Categorical column '{col}' not found in DataFrame for cleaning.")
```

```
--- 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 'JobSat': Imputed NaNs with median: 7.00
Created and imputed 'Age_Numeric' column.
```

```
/tmp/ipykernel_377/1828907397.py:53: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:53: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:71: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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(median_age_numeric, inplace=True)
/tmp/ipykernel_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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_377/1828907397.py:85: FutureWarning: A value is trying to be set on a copy of a DataF
rame 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 Data Distributions Using Bubble Plots

1. Bubble Plot for Age vs. Frequency of Participation

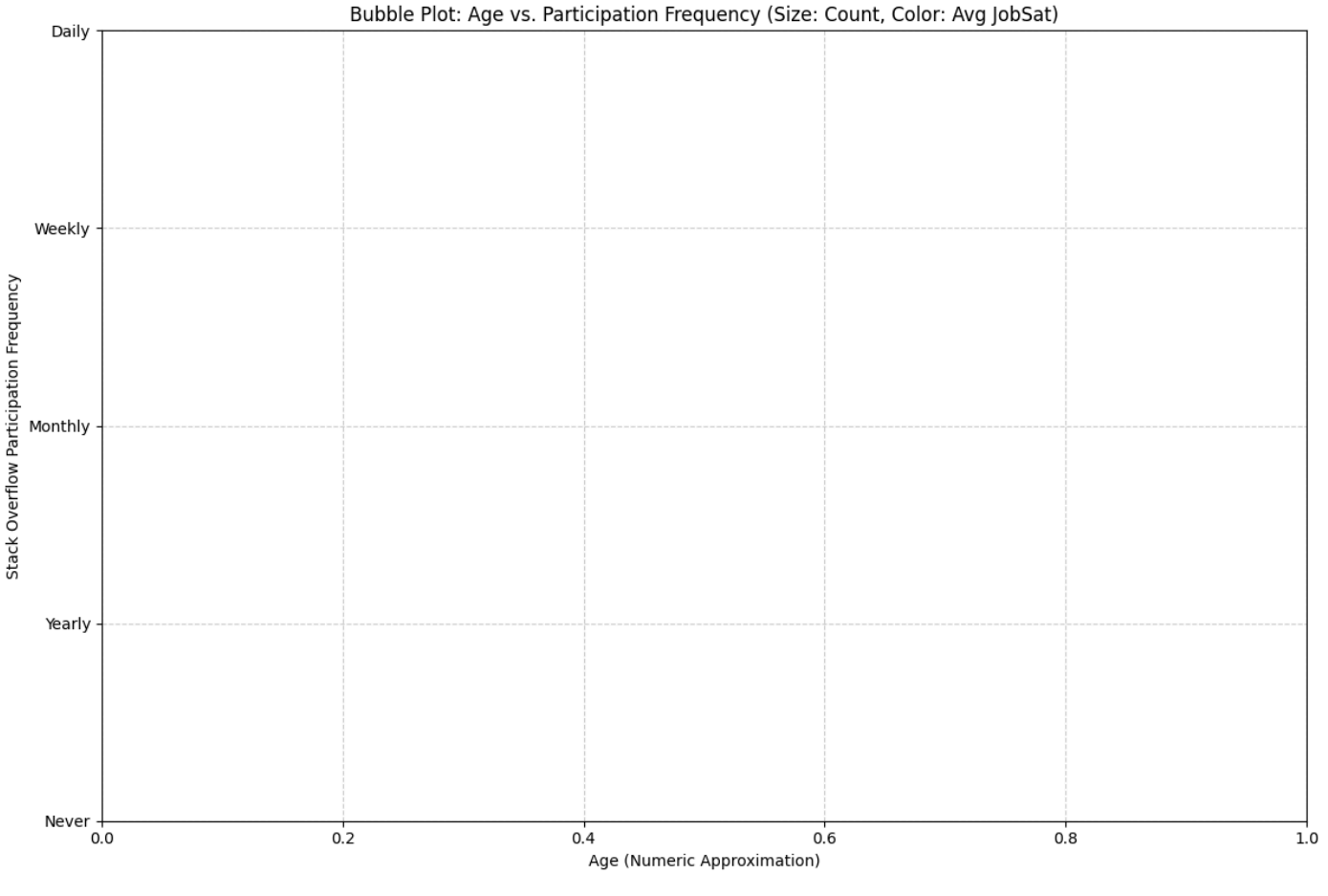
- Visualize the relationship between respondents' age and their participation frequency (`SOPartFreq`) using a bubble plot.
- Use the size of the bubbles to represent their job satisfaction (`JobSat`).

In [8]:

```python
##Write your code here
# --- Task 1: Exploring Data Distributions Using Bubble Plots ---
print("\n--- Task 1: Exploring Data Distributions Using Bubble Plots ---")

# 1. Bubble Plot for Age vs. Frequency of Participation (SOPartFreq) with JobSat as size
# For SOPartFreq, we need a numerical representation of frequency or simply count.
# Let's count occurrences of each SOPartFreq for the Y-axis.
if 'SOPartFreq' in df.columns and 'Age_Numeric' in df.columns and 'JobSat' in df.columns:
    df_task1_1 = df.dropna(subset=['SOPartFreq', 'Age_Numeric', 'JobSat']).copy()

    if not df_task1_1.empty:
        # Map frequency text to a numeric order for consistent plotting on Y-axis
        freq_order = {
            'Never': 0, 'Yearly': 1, 'Monthly': 2, 'Weekly': 3, 'Daily': 4
```

```
        }
        df_task1_1['SOPartFreq_Numeric'] = df_task1_1['SOPartFreq'].map(freq_order)

        # Aggregate data to get a manageable number of points, or use jitter for exact points
        # For a clearer bubble plot, let's group by Age_Numeric and SOPartFreq_Numeric, and
        # calculate mean JobSat and count for bubble size.
        grouped_data = df_task1_1.groupby(['Age_Numeric', 'SOPartFreq_Numeric']).agg(
            mean_jobsat=('JobSat', 'mean'),
            count=('ResponseId', 'count') # Use count for bubble size
        ).reset_index()

        plt.figure(figsize=(12, 8))
        sns.scatterplot(
            x='Age_Numeric',
            y='SOPartFreq_Numeric',
            size='count', # Bubble size represents the count of respondents in that age/frequency g
            hue='mean_jobsat', # Color based on mean job satisfaction
            data=grouped_data,
            sizes=(50, 1000), # Adjust bubble size range
            alpha=0.7,
            palette='RdYlGn', # Red-Yellow-Green for satisfaction
            legend='brief'
        )

        # Adjust y-axis ticks to show actual frequency labels
        plt.yticks(list(freq_order.values()), list(freq_order.keys()))

        plt.title('Bubble Plot: Age vs. Participation Frequency (Size: Count, Color: Avg JobSat)')
        plt.xlabel('Age (Numeric Approximation)')
        plt.ylabel('Stack Overflow Participation Frequency')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()
        print("Task 1.1: Bubble plot for Age vs. Participation Frequency with JobSat as size/hue pl
    else:
        print("Skipping Task 1.1: Not enough valid data in 'SOPartFreq', 'Age_Numeric', or 'JobSat'
else:
    print("Skipping Task 1.1: Required columns missing or not ready ('SOPartFreq', 'Age_Numeric', '
```

```
--- Task 1: Exploring Data Distributions Using Bubble Plots ---
/tmp/ipykernel_377/3982368215.py:27: UserWarning: Ignoring `palette` because no `hue` variable has b
een assigned.
  sns.scatterplot(
```



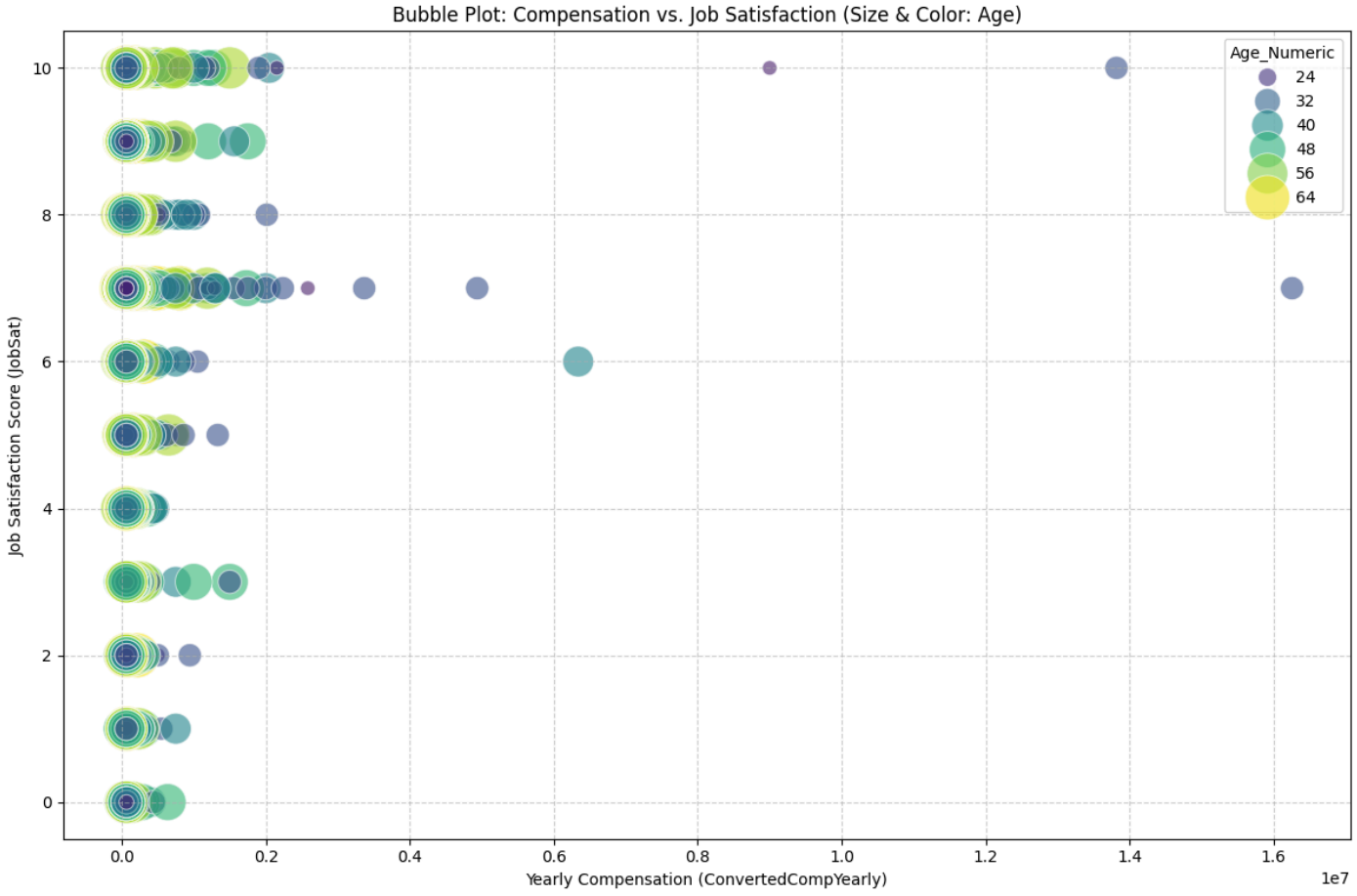
```
Task 1.1: Bubble plot for Age vs. Participation Frequency with JobSat as size/hue plotted.
```

2. Bubble Plot for Compensation vs. Job Satisfaction

-Visualize the relationship between yearly compensation (`ConvertedCompYearly`) and job satisfaction (`JobSat`).

- Use the size of the bubbles to represent respondents' age.

In [9]:
```python
##Write your code here
# 2. Bubble Plot for Compensation (ConvertedCompYearly) vs. Job Satisfaction (JobSat) with Age as s
if 'ConvertedCompYearly' in df.columns and 'JobSat' in df.columns and 'Age_Numeric' in df.columns:
    df_task1_2 = df.dropna(subset=['ConvertedCompYearly', 'JobSat', 'Age_Numeric']).copy()

    if not df_task1_2.empty:
        plt.figure(figsize=(12, 8))
        sns.scatterplot(
            x='ConvertedCompYearly',
            y='JobSat',
            size='Age_Numeric', # Bubble size represents age
            hue='Age_Numeric', # Color also by age for consistency
            data=df_task1_2,
            sizes=(20, 800), # Adjust bubble size range
            alpha=0.6,
            palette='viridis',
            legend='brief'
        )
        plt.title('Bubble Plot: Compensation vs. Job Satisfaction (Size & Color: Age)')
        plt.xlabel('Yearly Compensation (ConvertedCompYearly)')
        plt.ylabel('Job Satisfaction Score (JobSat)')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()
        print("Task 1.2: Bubble plot for Compensation vs. Job Satisfaction with Age as size plotted
    else:
        print("Skipping Task 1.2: Not enough valid data in 'ConvertedCompYearly', 'JobSat', or 'Age
else:
    print("Skipping Task 1.2: Required columns missing or not ready ('ConvertedCompYearly', 'JobSat
```




```
Task 1.2: Bubble plot for Compensation vs. Job Satisfaction with Age as size plotted.
```

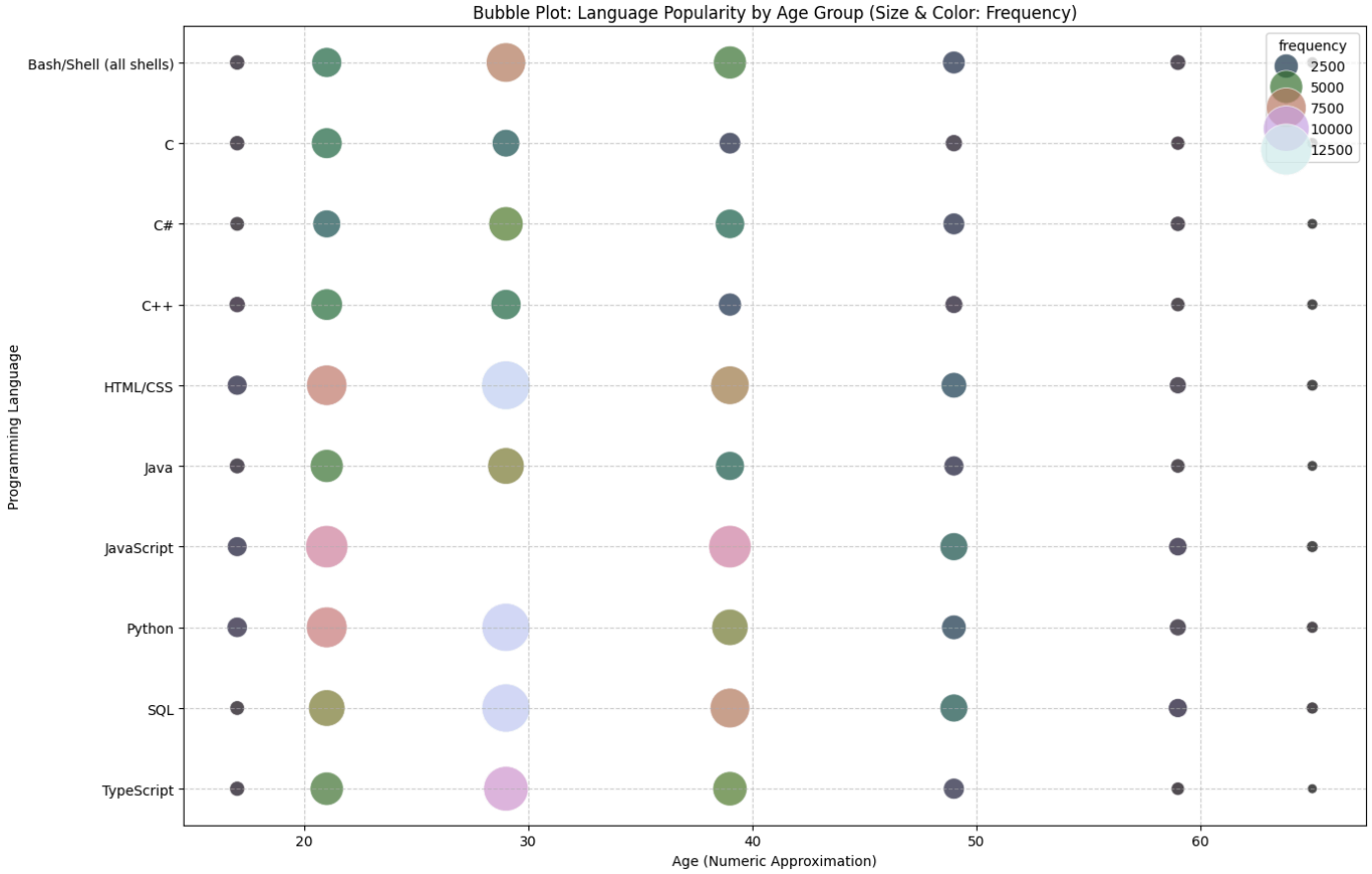
Task 2: Analyzing Relationships Using Bubble Plots

1. Bubble Plot of Technology Preferences by Age

- Visualize the popularity of programming languages respondents have worked with (`LanguageHaveWorkedWith`) across age groups.
- Use bubble size to represent the frequency of each language.

In [10]:
```
##Write your code here
# --- Task 2: Analyzing Relationships Using Bubble Plots ---
print("\n--- Task 2: Analyzing Relationships Using Bubble Plots ---")

# 1. Bubble Plot of Technology Preferences (LanguageHaveWorkedWith) by Age with Language Frequency
if 'LanguageHaveWorkedWith' in df.columns and 'Age_Numeric' in df.columns:
    df_task2_1 = df.dropna(subset=['LanguageHaveWorkedWith', 'Age_Numeric']).copy()
    if not df_task2_1.empty:
        # Explode languages and then group to count frequency per language per age group
        df_task2_1['Language'] = df_task2_1['LanguageHaveWorkedWith'].str.split(';')
        df_exploded_languages = df_task2_1.explode('Language')
        df_exploded_languages['Language'] = df_exploded_languages['Language'].str.strip()

        # Get top N languages to make the plot readable
        top_languages = df_exploded_languages['Language'].value_counts().head(10).index.tolist()
        df_top_languages = df_exploded_languages[df_exploded_languages['Language'].isin(top_languag

        # Group by language and age to get the count (frequency)
        grouped_data = df_top_languages.groupby(['Language', 'Age_Numeric']).agg(
            frequency=('ResponseId', 'count')
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(
                x='Age_Numeric',
                y='Language',
                size='frequency', # Bubble size represents frequency
                hue='frequency', # Color by frequency too
                data=grouped_data,
                sizes=(50, 1500), # Adjust bubble size range
                alpha=0.7,
                palette='cubehelix',
                legend='brief'
            )
            plt.title('Bubble Plot: Language Popularity by Age Group (Size & Color: Frequency)')
            plt.xlabel('Age (Numeric Approximation)')
            plt.ylabel('Programming Language')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 2.1: Bubble plot for Language Preferences by Age with Frequency as size plo
        else:
            print("Skipping Task 2.1: Not enough valid grouped data for Language Preferences by Age
    else:
        print("Skipping Task 2.1: Not enough valid data in 'LanguageHaveWorkedWith' or 'Age_Numeric
else:
    print("Skipping Task 2.1: Required columns missing or not ready ('LanguageHaveWorkedWith', 'Age
```

```
--- Task 2: Analyzing Relationships Using Bubble Plots ---
```



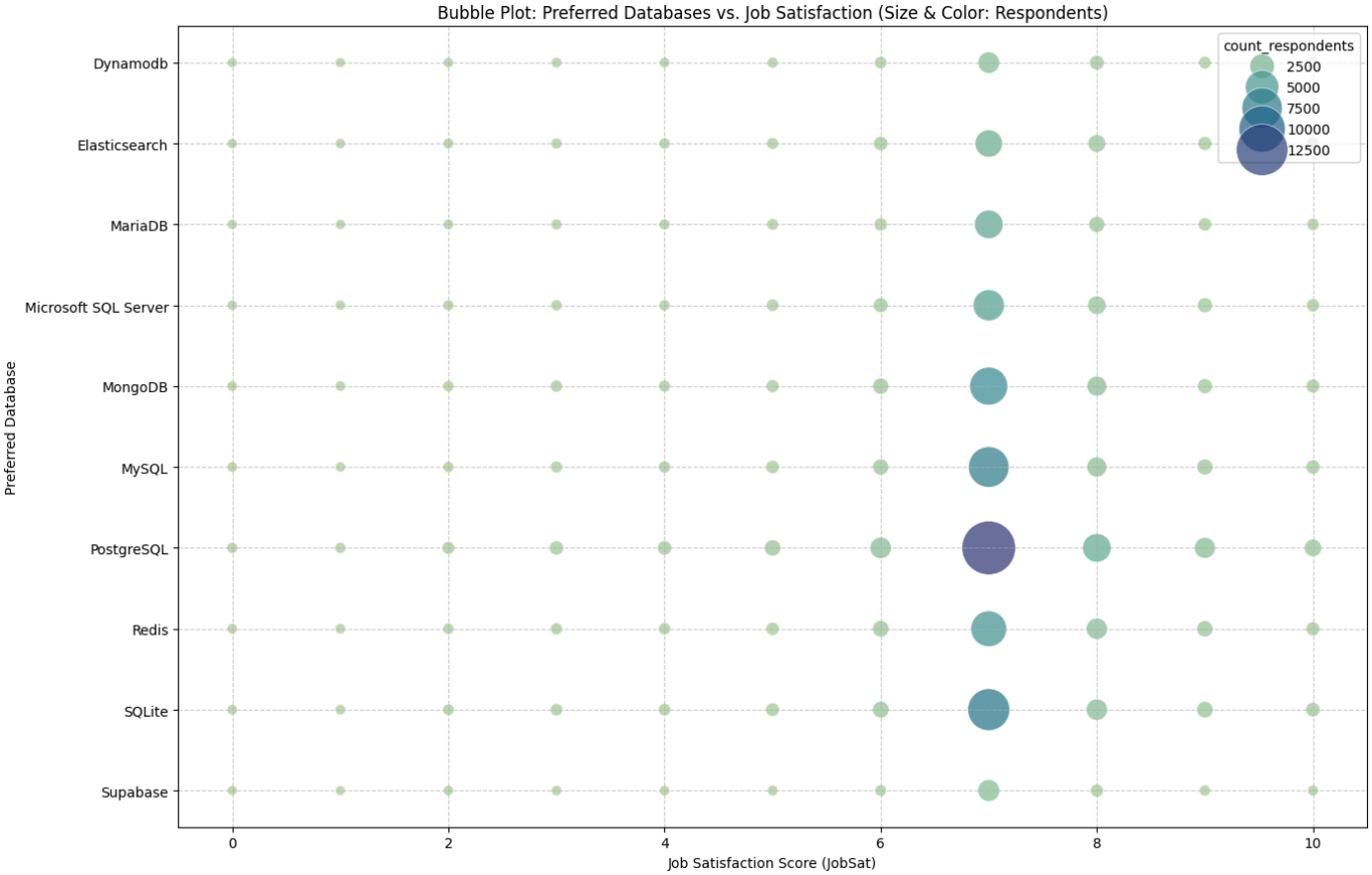
Task 2.1: Bubble plot for Language Preferences by Age with Frequency as size plotted.

2. Bubble Plot for Preferred Databases vs. Job Satisfaction

- Explore the relationship between preferred databases (`DatabaseWantToWorkWith`) and job satisfaction.
- Use bubble size to indicate the number of respondents for each database.

In [11]:
```
##Write your code here
# 2. Bubble Plot for Preferred Databases (DatabaseWantToWorkWith) vs. Job Satisfaction with number
if 'DatabaseWantToWorkWith' in df.columns and 'JobSat' in df.columns:
    df_task2_2 = df.dropna(subset=['DatabaseWantToWorkWith', 'JobSat']).copy()
    if not df_task2_2.empty:
        # Explode databases
        df_task2_2['Database'] = df_task2_2['DatabaseWantToWorkWith'].str.split(';')
        df_exploded_databases = df_task2_2.explode('Database')
        df_exploded_databases['Database'] = df_exploded_databases['Database'].str.strip()

        # Get top N databases
        top_databases = df_exploded_databases['Database'].value_counts().head(10).index.tolist()
        df_top_databases = df_exploded_databases[df_exploded_databases['Database'].isin(top_databas

        # Group by database and job satisfaction to get count of respondents
        grouped_data = df_top_databases.groupby(['Database', 'JobSat']).agg(
            count_respondents=('ResponseId', 'count')
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(
                x='JobSat',
                y='Database',
                size='count_respondents', # Bubble size is number of respondents
                hue='count_respondents', # Color by count too
                data=grouped_data,
                sizes=(50, 1500),
                alpha=0.7,
                palette='crest',
                legend='brief'
            )
            plt.title('Bubble Plot: Preferred Databases vs. Job Satisfaction (Size & Color: Respond
            plt.xlabel('Job Satisfaction Score (JobSat)')
            plt.ylabel('Preferred Database')
```

```
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 2.2: Bubble plot for Preferred Databases vs. Job Satisfaction plotted.")
        else:
            print("Skipping Task 2.2: Not enough valid grouped data for Preferred Databases vs. Job
    else:
        print("Skipping Task 2.2: Not enough valid data in 'DatabaseWantToWorkWith' or 'JobSat' for
else:
    print("Skipping Task 2.2: Required columns missing or not ready ('DatabaseWantToWorkWith', 'Job
```




```
Task 2.2: Bubble plot for Preferred Databases vs. Job Satisfaction plotted.
```

Task 3: Comparing Data Using Bubble Plots

1. Bubble Plot for Compensation Across Developer Roles

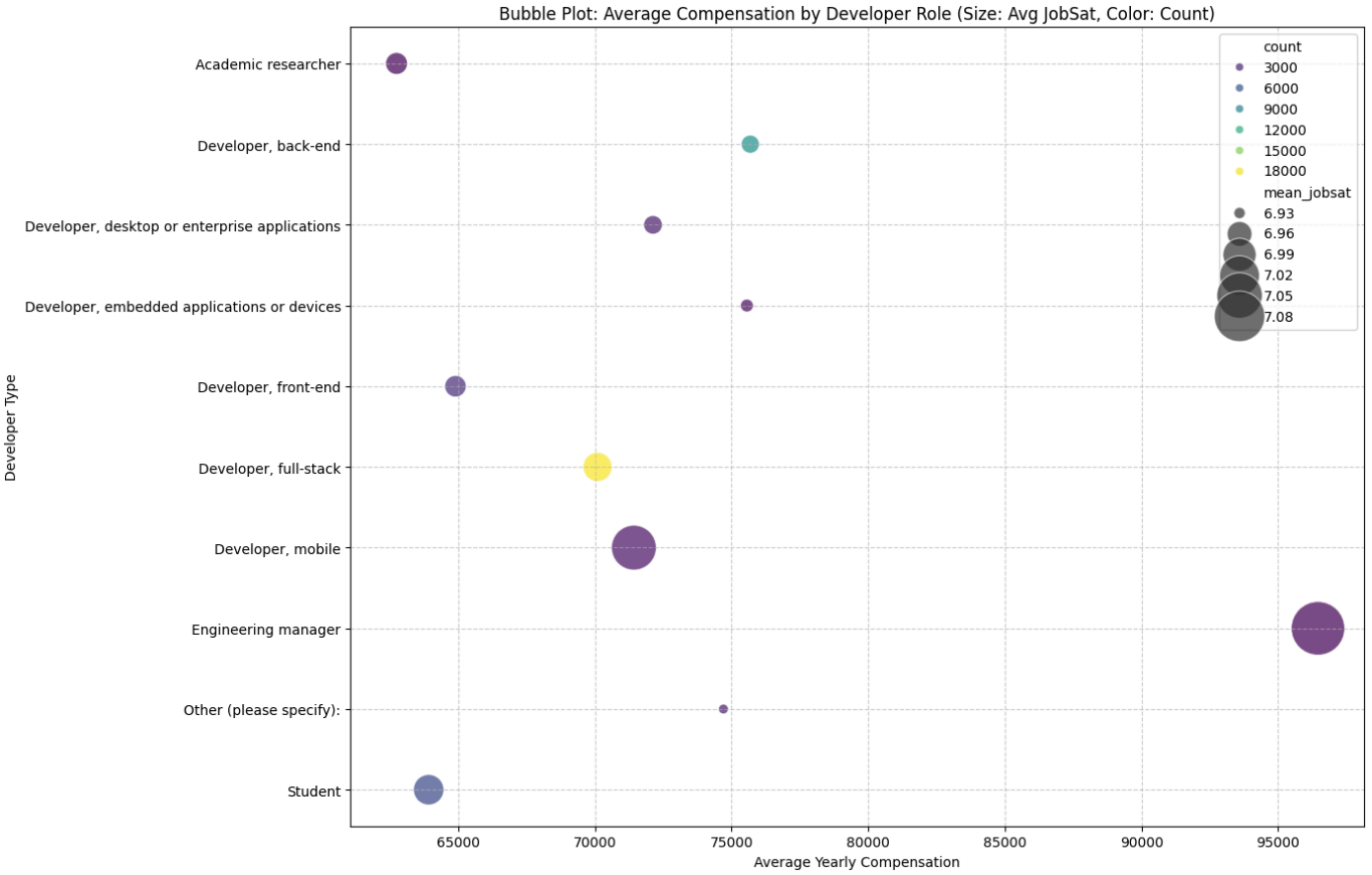
- Visualize compensation (`ConvertedCompYearly`) across different developer roles (`DevType`).
- Use bubble size to represent job satisfaction.

In [12]:

```
##Write your code here
# --- Task 3: Comparing Data Using Bubble Plots ---
print("\n--- Task 3: Comparing Data Using Bubble Plots ---")

# 1. Bubble Plot for Compensation (ConvertedCompYearly) Across Developer Roles (DevType) with Job S
if 'ConvertedCompYearly' in df.columns and 'DevType' in df.columns and 'JobSat' in df.columns:
    df_task3_1 = df.dropna(subset=['ConvertedCompYearly', 'DevType', 'JobSat']).copy()
    if not df_task3_1.empty:
        # Explode DevType
        df_task3_1['DevType_Single'] = df_task3_1['DevType'].str.split(';')
        df_exploded_devtype = df_task3_1.explode('DevType_Single')
        df_exploded_devtype['DevType_Single'] = df_exploded_devtype['DevType_Single'].str.strip()

        # Get top N developer types
        top_dev_types = df_exploded_devtype['DevType_Single'].value_counts().head(10).index.tolist(
        df_top_dev_types = df_exploded_devtype[df_exploded_devtype['DevType_Single'].isin(top_dev_t

        # Group by DevType and calculate mean compensation and mean job satisfaction
        grouped_data = df_top_dev_types.groupby('DevType_Single').agg(
            mean_compensation=('ConvertedCompYearly', 'mean'),
            mean_jobsat=('JobSat', 'mean'),
            count=('ResponseId', 'count')
```

```
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(
                x='mean_compensation',
                y='DevType_Single',
                size='mean_jobsat', # Bubble size represents mean job satisfaction
                hue='count', # Color by count of respondents in each group
                data=grouped_data,
                sizes=(50, 1500),
                alpha=0.7,
                palette='viridis',
                legend='brief'
            )
            plt.title('Bubble Plot: Average Compensation by Developer Role (Size: Avg JobSat, Color
            plt.xlabel('Average Yearly Compensation')
            plt.ylabel('Developer Type')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 3.1: Bubble plot for Compensation Across Developer Roles plotted.")
        else:
            print("Skipping Task 3.1: Not enough valid grouped data for Compensation Across Develop
    else:
        print("Skipping Task 3.1: Not enough valid data in 'ConvertedCompYearly', 'DevType', or 'Jo
else:
    print("Skipping Task 3.1: Required columns missing or not ready ('ConvertedCompYearly', 'DevTyp
```

```
--- Task 3: Comparing Data Using Bubble Plots ---
```




```
Task 3.1: Bubble plot for Compensation Across Developer Roles plotted.
```

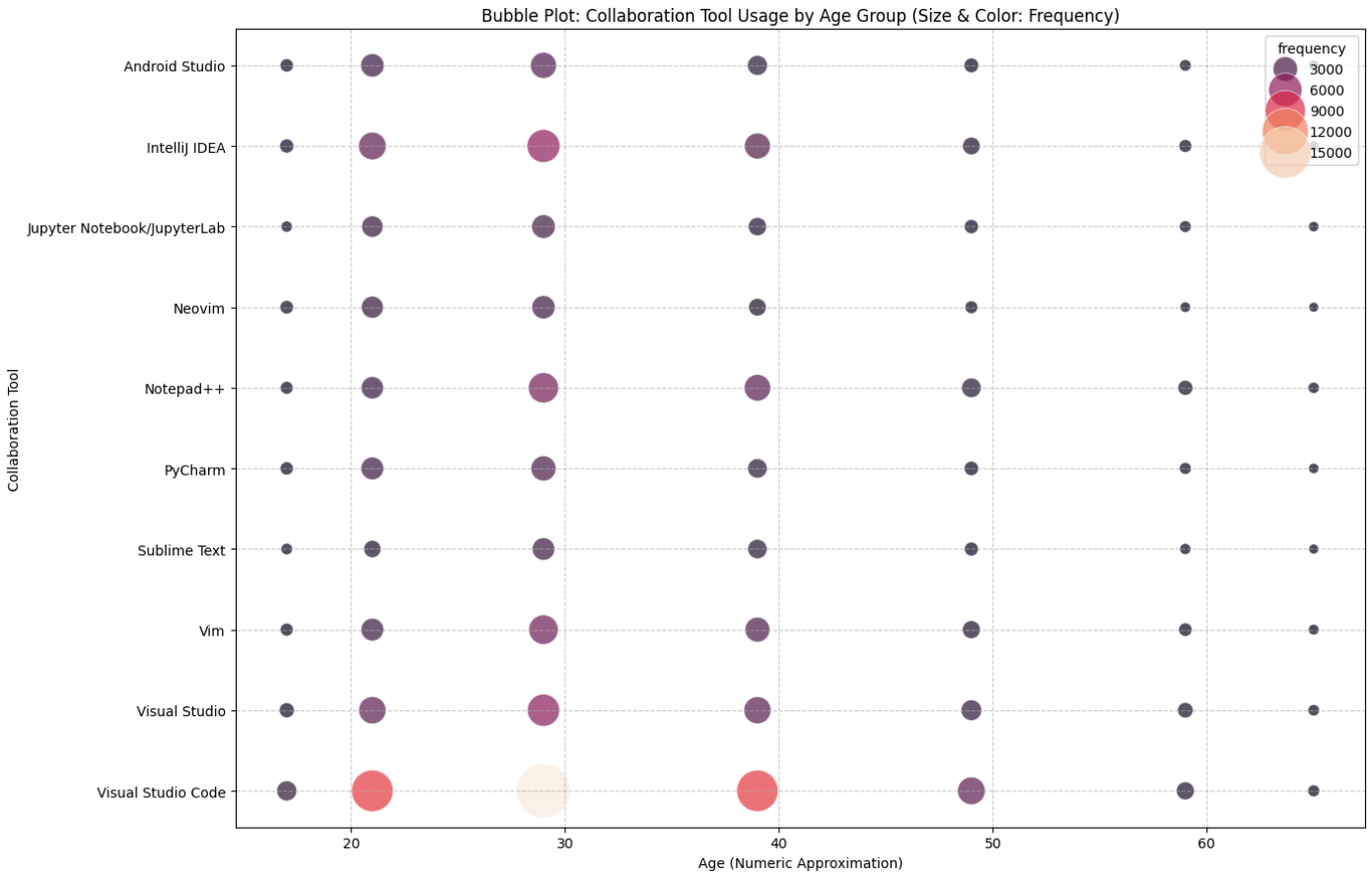
2. Bubble Plot for Collaboration Tools by Age

- Visualize the relationship between the collaboration tools used (`NEWCollabToolsHaveWorkedWith`) and age groups.
- Use bubble size to represent the frequency of tool usage.

In [13]:

```
##Write your code here
# 2. Bubble Plot for Collaboration Tools (NEWCollabToolsHaveWorkedWith) by Age with frequency as si
if 'NEWCollabToolsHaveWorkedWith' in df.columns and 'Age_Numeric' in df.columns:
    df_task3_2 = df.dropna(subset=['NEWCollabToolsHaveWorkedWith', 'Age_Numeric']).copy()
```

```
    if not df_task3_2.empty:
        # Explode collaboration tools
        df_task3_2['CollabTool'] = df_task3_2['NEWCollabToolsHaveWorkedWith'].str.split(';')
        df_exploded_tools = df_task3_2.explode('CollabTool')
        df_exploded_tools['CollabTool'] = df_exploded_tools['CollabTool'].str.strip()

        # Get top N tools
        top_tools = df_exploded_tools['CollabTool'].value_counts().head(10).index.tolist()
        df_top_tools = df_exploded_tools[df_exploded_tools['CollabTool'].isin(top_tools)]

        # Group by tool and age to get frequency
        grouped_data = df_top_tools.groupby(['CollabTool', 'Age_Numeric']).agg(
            frequency=('ResponseId', 'count')
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(
                x='Age_Numeric',
                y='CollabTool',
                size='frequency', # Bubble size is frequency
                hue='frequency', # Color by frequency
                data=grouped_data,
                sizes=(50, 1500),
                alpha=0.7,
                palette='rocket',
                legend='brief'
            )
            plt.title('Bubble Plot: Collaboration Tool Usage by Age Group (Size & Color: Frequency)
            plt.xlabel('Age (Numeric Approximation)')
            plt.ylabel('Collaboration Tool')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 3.2: Bubble plot for Collaboration Tools by Age plotted.")
        else:
            print("Skipping Task 3.2: Not enough valid grouped data for Collaboration Tools by Age.
    else:
        print("Skipping Task 3.2: Not enough valid data in 'NEWCollabToolsHaveWorkedWith' or 'Age_N
else:
    print("Skipping Task 3.2: Required columns missing or not ready ('NEWCollabToolsHaveWorkedWith'
```



Task 3.2: Bubble plot for Collaboration Tools by Age plotted.

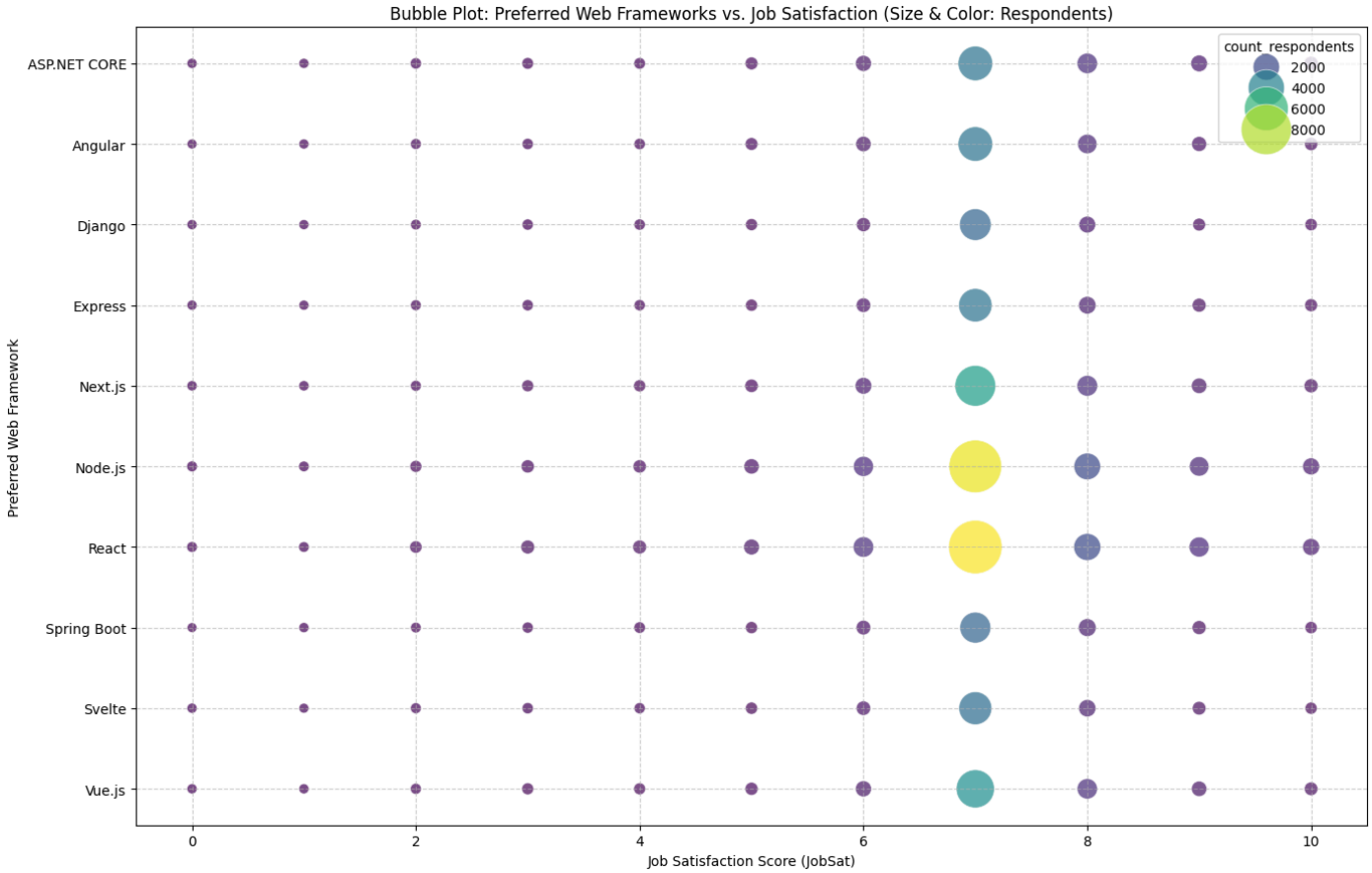
Task 4: Visualizing Technology Trends Using Bubble Plots

1. Bubble Plot for Preferred Web Frameworks vs. Job Satisfaction

- Explore the relationship between preferred web frameworks (`WebframeWantToWorkWith`) and job satisfaction.
- Use bubble size to represent the number of respondents.

In [14]:
```
# --- Task 4: Visualizing Technology Trends Using Bubble Plots ---
print("\n--- Task 4: Visualizing Technology Trends Using Bubble Plots ---")

# 1. Bubble Plot for Preferred Web Frameworks (WebframeWantToWorkWith) vs. Job Satisfaction with nu
if 'WebframeWantToWorkWith' in df.columns and 'JobSat' in df.columns:
    df_task4_1 = df.dropna(subset=['WebframeWantToWorkWith', 'JobSat']).copy()
    if not df_task4_1.empty:
        # Explode web frameworks
        df_task4_1['Webframe'] = df_task4_1['WebframeWantToWorkWith'].str.split(';')
        df_exploded_webframes = df_task4_1.explode('Webframe')
        df_exploded_webframes['Webframe'] = df_exploded_webframes['Webframe'].str.strip()

        # Get top N frameworks
        top_webframes = df_exploded_webframes['Webframe'].value_counts().head(10).index.tolist()
        df_top_webframes = df_exploded_webframes[df_exploded_webframes['Webframe'].isin(top_webfram

        # Group by framework and job satisfaction to get count of respondents
        grouped_data = df_top_webframes.groupby(['Webframe', 'JobSat']).agg(
            count_respondents=('ResponseId', 'count')
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(14, 9))
            sns.scatterplot(
                x='JobSat',
                y='Webframe',
                size='count_respondents', # Bubble size is number of respondents
                hue='count_respondents', # Color by count
                data=grouped_data,
                sizes=(50, 1500),
                alpha=0.7,
                palette='viridis',
                legend='brief'
            )
            plt.title('Bubble Plot: Preferred Web Frameworks vs. Job Satisfaction (Size & Color: Re
            plt.xlabel('Job Satisfaction Score (JobSat)')
            plt.ylabel('Preferred Web Framework')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 4.1: Bubble plot for Preferred Web Frameworks vs. Job Satisfaction plotted.
        else:
            print("Skipping Task 4.1: Not enough valid grouped data for Preferred Web Frameworks vs
    else:
        print("Skipping Task 4.1: Not enough valid data in 'WebframeWantToWorkWith' or 'JobSat' for
else:
    print("Skipping Task 4.1: Required columns missing or not ready ('WebframeWantToWorkWith', 'Job
```

```
--- Task 4: Visualizing Technology Trends Using Bubble Plots ---
```



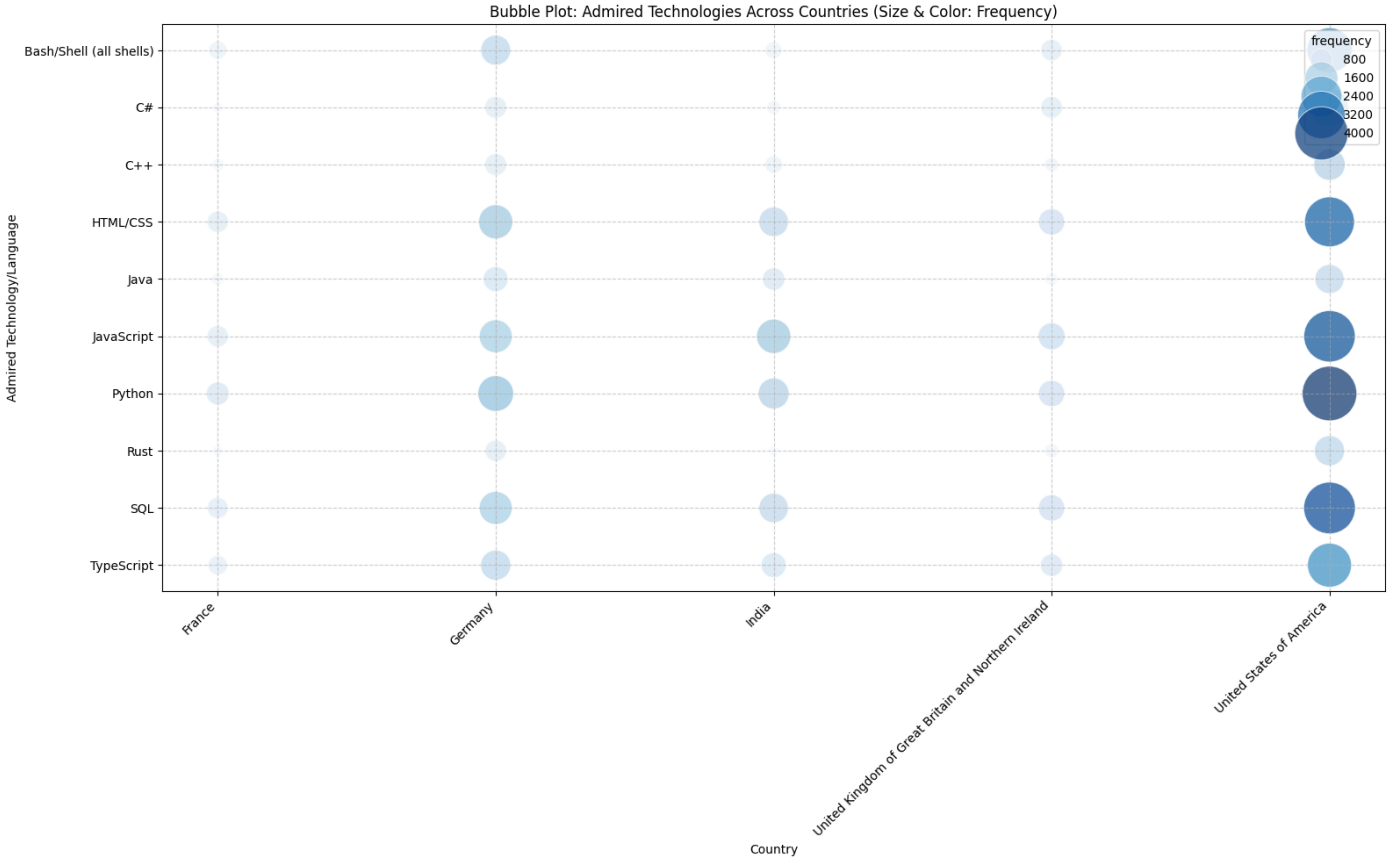
Task 4.1: Bubble plot for Preferred Web Frameworks vs. Job Satisfaction plotted.

2. Bubble Plot for Admired Technologies Across Countries

- Visualize the distribution of admired technologies (`LanguageAdmired`) across different countries (`Country`).
- Use bubble size to represent the frequency of admiration.

In [15]:
```
##Write your code here
# 2. Bubble Plot for Admired Technologies (LanguageAdmired) Across Countries (Country) with frequen
if 'LanguageAdmired' in df.columns and 'Country' in df.columns:
    df_task4_2 = df.dropna(subset=['LanguageAdmired', 'Country']).copy()
    if not df_task4_2.empty:
        # Explode admired languages
        df_task4_2['AdmiredLanguage'] = df_task4_2['LanguageAdmired'].str.split(';')
        df_exploded_admired = df_task4_2.explode('AdmiredLanguage')
        df_exploded_admired['AdmiredLanguage'] = df_exploded_admired['AdmiredLanguage'].str.strip()

        # Get top N countries and top N languages for readability
        top_countries = df_exploded_admired['Country'].value_counts().head(5).index.tolist()
        top_languages = df_exploded_admired['AdmiredLanguage'].value_counts().head(10).index.tolist

        df_filtered_data = df_exploded_admired[
            (df_exploded_admired['Country'].isin(top_countries)) &
            (df_exploded_admired['AdmiredLanguage'].isin(top_languages))
        ].copy()

        # Group by country and admired language to get frequency
        grouped_data = df_filtered_data.groupby(['Country', 'AdmiredLanguage']).agg(
            frequency=('ResponseId', 'count')
        ).reset_index()

        if not grouped_data.empty:
            plt.figure(figsize=(16, 10))
            sns.scatterplot(
                x='Country',
                y='AdmiredLanguage',
                size='frequency', # Bubble size is frequency
                hue='frequency', # Color by frequency
                data=grouped_data,
                sizes=(50, 2000),
                alpha=0.7,
                palette='Blues',
```

```
                legend='brief'
            )
            plt.title('Bubble Plot: Admired Technologies Across Countries (Size & Color: Frequency)
            plt.xlabel('Country')
            plt.ylabel('Admired Technology/Language')
            plt.xticks(rotation=45, ha='right')
            plt.grid(True, linestyle='--', alpha=0.6)
            plt.tight_layout()
            plt.show()
            print("Task 4.2: Bubble plot for Admired Technologies Across Countries plotted.")
        else:
            print("Skipping Task 4.2: Not enough valid grouped data for Admired Technologies Across
    else:
        print("Skipping Task 4.2: Not enough valid data in 'LanguageAdmired' or 'Country' for plott
else:
    print("Skipping Task 4.2: Required columns missing or not ready ('LanguageAdmired', 'Country').


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




```
Task 4.2: Bubble plot for Admired Technologies Across Countries plotted.

--- Lab Completion Summary ---
All bubble plot tasks have been attempted. Please review the plots and console output for any warnin
gs or skipped tasks.
```

Final Step: Review

After completing the lab, you will have extensively used bubble plots to gain insights into developer community preferences, demographics, compensation trends, and job satisfaction.

Summary

After completing this lab, you will be able to:

- Create and interpret bubble plots to analyze relationships and compositions within datasets.
- Use bubble plots to explore developer preferences, compensation trends, and satisfaction levels.
- Apply bubble plots to visualize complex relationships involving multiple dimensions effectively.

Authors:

Ayushi Jain

Other Contributors:

- Rav Ahuja
- Lakshmi Holla
- Malika

<!-- ## Change Log |Date (YYYY-MM-DD)|Version|Changed By|Change Description| |-|-|-|-| |2024-10-29|1.2|Madhusudhan Moole|Updated lab| |2024-10-16|1.1|Madhusudhan Moole|Updated lab| |2024-10-15|1.0|Raghul Ramesh|Created lab| --!>