Introduction

Stack Overflow, the go-to platform for developers, conducted a global survey to capture insights into the developer community.
This survey includes various questions related to professional experience, coding activities, tools, technologies, and preferences.
The dataset offers important information about the current state of software development worldwide.

For this analysis work with a subset of the original dataset to explore, analyze, and visualize various trends.
The survey data consists of responses from developers with different professional backgrounds, educational levels and geographic locations.

Project Scenario

The primary task is to analyze the data and obtain valuable insights into current and future technological trends using the latest **Stack Overflow Developer Survey** dataset.

Roles and responsibilities:

- Collecting data from diverse sources and identifying trends for this year's report on in-demand skills.
  One key source for analysis will be the latest Developer Survey, a comprehensive dataset offering insights into the global developer community.

- Initial task is to gather data on the most sought-after programming skills from various sources, including:

  - Job postings
  - Training portals
  - Developer surveys, such as the latest Stack Overflow Developer Survey
- After collecting sufficient data, analyze it to identify key insights and trends.
  Some of the trends to explore include:

  - Which programming languages are most in demand?
  - Which database technologies are currently most sought after?
  - Which Integrated Development Environments (IDEs) are the most popular?
- Begin by scraping internet websites, accessing APIs, and working with datasets like the latest Stack Overflow Developer Survey in various formats such as .csv files, Excel sheets, and databases.

- After gathering the data, apply data-wrangling techniques to prepare the data for analysis.
- Once the data is prepared, employ statistical techniques to analyze the data, identifying key trends and insights. Then synthesize findings using IBM Cognos Analytics (or other tool) to create a dashboard that visualizes the results.
- Finally, share insights through a presentation, showcasing storytelling and data analysis skills.

Dataset

Source:

The dataset is available as part of the Stack Overflow Developer Survey under an Open Database License (ODbL).
The full dataset can be accessed at this link.

Collection

- Conducted via online survey
- Includes responses on tools, platforms, roles, preferences, and demographics

The Stack Overflow Developer Survey dataset is publicly available, structured, and self-reported by developers worldwide.

Use a subset of the full data set, which contains several thousand responses.
Please note that the conclusions drawn from this subset may not fully reflect the global developer community.

Column name	Question text
ResponseId	Randomized respondent ID number.
MainBranch	Which of the following options best describes you today?
Age	What is your age?
Employment	What is your current employment status?
RemoteWork	How often do you work remotely?
Check	Check Various verification or check questions related to survey consistency.
CodingActivities	What coding activities do you engage in (hobby, professional, and open-source contributions)?
EdLevel	What is the highest level of formal education you have completed?
LearnCode	How did you learn to code?

Column name	Question text
LearnCodeOnline	Have you used online resources to learn coding?
TechDoc	How do you use technical documentation?
YearsCode	How many years have you been coding?
YearsCodePro	How many years have you coded professionally?
DevType	What is your role or type of development work you do?
OrgSize	What is the size of the organization you work for?
PurchaseInfluence	How much influence do you have on purchasing technology at your company?
BuyNewTool	How does your company decide whether to buy new tools or technology?
BuildvsBuy	Does your company prefer to build or buy software?
TechEndorse	Do you endorse any specific technologies at your company?
Country	In which country do you reside?
Currency	Which currency do you use day-to-day?
CompTotal	What is your current total compensation (salary, bonuses, and so on)?
LanguageHaveWorkedWith	Which programming languages have you worked with in the past year?
LanguageWantToWorkWith	Which programming languages do you want to work with in the future?
LanguageAdmired	Which programming languages do you admire most?
DatabaseHaveWorkedWith	Which database technologies have you worked with in the past year?
DatabaseWantToWorkWith	Which database technologies do you want to work with in the future?
DatabaseAdmired	Which database technologies do you admire most?
PlatformHaveWorkedWith	Which platforms have you worked with in the past year?
PlatformWantToWorkWith	Which platforms do you want to work with in the future?
PlatformAdmired	Which platforms do you admire most?

|WebframeHaveWorkedWith | Which web frameworks have you worked with in the past year? |WebframeWantToWorkWith | Which web frameworks do you want to work with in the future?| |WebframeAdmired | Which web frameworks do you admire most?| |EmbeddedHaveWorkedWith | Which embedded systems have you worked with in the past year?| |EmbeddedWantToWorkWith | Which embedded systems do you want to work with in the future?| |EmbeddedAdmired | Which embedded systems do you admire most?|

|MiscTechHaveWorkedWith | Which miscellaneous technologies have you worked with in the past year?| |MiscTechWantToWorkWith | Which miscellaneous technologies do you want to work with in the future?| |MiscTechAdmired | Which miscellaneous technologies do you admire most?| |OpSysPersonal | What operating systems do you use for personal tasks?| |OpSysProfessional | What operating systems do you use for professional tasks?| |SOVisitFreq | How frequently do you visit Stack Overflow?| |SOAccount | Do you have a Stack Overflow account?| |SOPartFreq | How often do you participate in Q&A on Stack Overflow?| |AISelect | How do you feel about artificial intelligence tools for development?| |AIBen | What benefits have you experienced from using AI tools?| |AIChallenges | What challenges have you faced while using AI tools?| |JobSat | How satisfied are you with your current job?|

Import Libraries

Import all the necessary libraries; install first if not done already.

In [1]:
```
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Collection

- Analyze HTTP Requests
- Extract information from a given link
- Write the scraped data into a CSV file

Download the Dataset

In [2]:
```
# Function to download

def download(url, filename):
    response = requests.get(url)
    if response.status_code == 200:
        with open(filename, "wb") as f:
            f.write(response.content)
```

In [3]:
```
file_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01P
```

In [4]:
```
file_path = r"./so_survey_data.csv"
#download(file_url, file_path)    # FIXME: Taking too long
print(f'File downloaded at {file_path}')
```

```
File downloaded at ./so_survey_data.csv
```

Load Data into a Dataframe

In [5]:
```
dfo = pd.read_csv(file_path)
print("Data saved into dataframe!")
```

```
Data saved into dataframe!
```

In [6]:
```
# Copy data
df = dfo.copy()
```

Data Exploration

Data exploration is the initial phase of data analysis where we aim to understand the data's characteristics, identify patterns, and uncover potential insights.
It is a crucial step that helps us make informed decisions about subsequent analysis.

- Summarize the key characteristics of a dataset.
- Identify different data types commonly used in data analysis.

In [7]:
```
# Display the top 5 rows and columns

df.head()
```

Out[7]:

	ResponseId	MainBranch	Age	Employment	RemoteWork	Check	CodingActivities
0	1	I am a developer by profession	Under 18 years old	Employed, full-time	Remote	Apples	Hobby
1	2	I am a developer by profession	35-44 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...
2	3	I am a developer by profession	45-54 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...
3	4	I am learning to code	18-24 years old	Student, full-time	NaN	Apples	NaN
4	5	I am a developer by profession	18-24 years old	Student, full-time	NaN	Apples	NaN

5 rows × 114 columns

In [8]:
```
# Display the Columns
```

```python
df.columns
```

Out[8]:
```
Index(['ResponseId', 'MainBranch', 'Age', 'Employment', 'RemoteWork', 'Check',
       'CodingActivities', 'EdLevel', 'LearnCode', 'LearnCodeOnline',
       ...
       'JobSatPoints_6', 'JobSatPoints_7', 'JobSatPoints_8', 'JobSatPoints_9',
       'JobSatPoints_10', 'JobSatPoints_11', 'SurveyLength', 'SurveyEase',
       'ConvertedCompYearly', 'JobSat'],
      dtype='object', length=114)
```

In [9]:
```python
# Print the number of rows and columns in the dataset

df.shape
```

Out[9]:
```
(65437, 114)
```

In [10]:
```python
# Identify the data types of each column

df.dtypes
```

Out[10]:
```
ResponseId               int64
MainBranch              object
Age                     object
Employment              object
RemoteWork              object
                        ...
JobSatPoints_11        float64
SurveyLength            object
SurveyEase              object
ConvertedCompYearly    float64
JobSat                 float64
Length: 114, dtype: object
```

In [11]:
```python
# Basic Statistics for numerical columns

df.describe()
```

Out[11]:

	ResponseId	CompTotal	WorkExp	JobSatPoints_1	JobSatPoints_4	JobSatF
count	65437.000000	3.374000e+04	29658.000000	29324.000000	29393.000000	29411
mean	32719.000000	2.963841e+145	11.466957	18.581094	7.522140	10
std	18890.179119	5.444117e+147	9.168709	25.966221	18.422661	21
min	1.000000	0.000000e+00	0.000000	0.000000	0.000000	0
25%	16360.000000	6.000000e+04	4.000000	0.000000	0.000000	0
50%	32719.000000	1.100000e+05	9.000000	10.000000	0.000000	0
75%	49078.000000	2.500000e+05	16.000000	22.000000	5.000000	10
max	65437.000000	1.000000e+150	50.000000	100.000000	100.000000	100

In [12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65437 entries, 0 to 65436
Columns: 114 entries, ResponseId to JobSat
dtypes: float64(13), int64(1), object(100)
memory usage: 56.9+ MB
```

The dataset is the result of a world wide survey. Print how many unique countries are there in the Country column.

In [13]:
```python
df['Country'].nunique()
```

Out[13]:
```
185
```

Data Wrangling

Data wrangling, also known as *data munging*, involves cleaning and preparing datasets to make them ready for analysis.

Perform a series of data wrangling tasks using the Stack Overflow survey data:

A. Identify and remove duplicate rows

- Locate any duplicate rows within the dataset.
- Remove these duplicates to maintain data integrity.

B. Analyze missing values

- Identify missing values in the dataset
- Calculate the number of missing values in each column to assess data completeness.

C. Impute data

- Handle missing values using suitable methods
- Apply techniques to impute missing values in the dataset

D. Normalize data for comparative analysis

- Use suitable techniques to normalize data in the dataset

Handle Inconsistencies

Identifying and Removing Inconsistencies

Identify inconsistent or irrelevant entries in specific columns (e.g., `Country`).

`Country`

In [14]:
```python
df['Country'].unique()
```

Out[14]:
```
array(['United States of America',
       'United Kingdom of Great Britain and Northern Ireland', 'Canada',
       'Norway', 'Uzbekistan', 'Serbia', 'Poland', 'Philippines',
       'Bulgaria', 'Switzerland', 'India', 'Germany', 'Ireland', 'Italy',
       'Ukraine', 'Australia', 'Brazil', 'Japan', 'Austria',
       'Iran, Islamic Republic of...', 'France', 'Saudi Arabia',
       'Romania', 'Turkey', 'Nepal', 'Algeria', 'Sweden', 'Netherlands',
       'Croatia', 'Pakistan', 'Czech Republic',
       'Republic of North Macedonia', 'Finland', 'Slovakia',
       'Russian Federation', 'Greece', 'Israel', 'Belgium', 'Mexico',
       'United Republic of Tanzania', 'Hungary', 'Argentina', 'Portugal',
       'Sri Lanka', 'Latvia', 'China', 'Singapore', 'Lebanon', 'Spain',
       'South Africa', 'Lithuania', 'Viet Nam', 'Dominican Republic',
       'Indonesia', 'Kosovo', 'Morocco', 'Taiwan', 'Georgia',
       'San Marino', 'Tunisia', 'Bangladesh', 'Nigeria', 'Liechtenstein',
       'Denmark', 'Ecuador', 'Malaysia', 'Albania', 'Azerbaijan', 'Chile',
       'Ghana', 'Peru', 'Bolivia', 'Egypt', 'Luxembourg', 'Montenegro',
       'Cyprus', 'Paraguay', 'Kazakhstan', 'Slovenia', 'Jordan',
       'Venezuela, Bolivarian Republic of...', 'Costa Rica', 'Jamaica',
       'Thailand', 'Nicaragua', 'Myanmar', 'Republic of Korea', 'Rwanda',
       'Bosnia and Herzegovina', 'Benin', 'El Salvador', 'Zimbabwe',
       'Afghanistan', 'Estonia', 'Malta', 'Uruguay', 'Belarus',
       'Colombia', 'Republic of Moldova', 'Isle of Man', 'Nomadic',
       'New Zealand', 'Palestine', 'Armenia', 'United Arab Emirates',
       'Maldives', 'Ethiopia', 'Fiji', 'Guatemala', 'Uganda',
       'Turkmenistan', 'Mauritius', 'Kenya', 'Cuba', 'Gabon', 'Bahamas',
       'South Korea', 'Iceland', 'Honduras', 'Hong Kong (S.A.R.)',
       "Lao People's Democratic Republic", 'Mongolia', 'Cambodia',
       'Madagascar', 'Angola', 'Democratic Republic of the Congo',
       'Syrian Arab Republic', 'Iraq', 'Namibia', 'Senegal', 'Kyrgyzstan',
       'Zambia', 'Swaziland', "Côte d'Ivoire", 'Kuwait', 'Tajikistan',
       'Burundi', 'Trinidad and Tobago', 'Mauritania', 'Sierra Leone',
       'Panama', 'Somalia', 'North Korea', 'Dominica', 'Guyana', 'Togo',
       'Oman', 'Barbados', 'Andorra',
       "Democratic People's Republic of Korea", 'Qatar', 'Sudan',
       'Cameroon', 'Papua New Guinea', 'Bahrain', 'Yemen', 'Malawi',
       'Burkina Faso', 'Congo, Republic of the...', 'Botswana',
       'Guinea-Bissau', 'Mozambique', 'Central African Republic',
       'Equatorial Guinea', 'Suriname', 'Belize',
       'Libyan Arab Jamahiriya', 'Cape Verde', 'Brunei Darussalam',
       'Bhutan', 'Guinea', 'Niger', 'Antigua and Barbuda', 'Mali',
       'Samoa', 'Lesotho', 'Saint Kitts and Nevis', 'Monaco',
       'Micronesia, Federated States of...', 'Haiti', nan, 'Nauru',
       'Liberia', 'Chad', 'Djibouti', 'Solomon Islands'], dtype=object)
```

In [15]:
```
df['Country'].value_counts()
```

Out[15]:
```
Country
United States of America                                  11095
Germany                                                    4947
India                                                      4231
United Kingdom of Great Britain and Northern Ireland       3224
Ukraine                                                    2672
                                                          ...
Central African Republic                                      1
Equatorial Guinea                                             1
Niger                                                         1
Guinea                                                        1
Solomon Islands                                               1
Name: count, Length: 185, dtype: int64
```

In [16]:
```python
def clean_name(name):
    if pd.isna(name):
        return np.nan
    # Remove trailing ellipses and anything after commas
    if '...' in name:
        name = name.split('...')[0].strip()
    if ',' in name:
        name = name.split(',')[0].strip()
    return name
```

In [17]:
```python
df['Country'] = df['Country'].apply(clean_name)

print('Unique Countries: ', df['Country'].unique())
```

```
Unique Countries:  ['United States of America'
 'United Kingdom of Great Britain and Northern Ireland' 'Canada' 'Norway'
 'Uzbekistan' 'Serbia' 'Poland' 'Philippines' 'Bulgaria' 'Switzerland'
 'India' 'Germany' 'Ireland' 'Italy' 'Ukraine' 'Australia' 'Brazil'
 'Japan' 'Austria' 'Iran' 'France' 'Saudi Arabia' 'Romania' 'Turkey'
 'Nepal' 'Algeria' 'Sweden' 'Netherlands' 'Croatia' 'Pakistan'
 'Czech Republic' 'Republic of North Macedonia' 'Finland' 'Slovakia'
 'Russian Federation' 'Greece' 'Israel' 'Belgium' 'Mexico'
 'United Republic of Tanzania' 'Hungary' 'Argentina' 'Portugal'
 'Sri Lanka' 'Latvia' 'China' 'Singapore' 'Lebanon' 'Spain' 'South Africa'
 'Lithuania' 'Viet Nam' 'Dominican Republic' 'Indonesia' 'Kosovo'
 'Morocco' 'Taiwan' 'Georgia' 'San Marino' 'Tunisia' 'Bangladesh'
 'Nigeria' 'Liechtenstein' 'Denmark' 'Ecuador' 'Malaysia' 'Albania'
 'Azerbaijan' 'Chile' 'Ghana' 'Peru' 'Bolivia' 'Egypt' 'Luxembourg'
 'Montenegro' 'Cyprus' 'Paraguay' 'Kazakhstan' 'Slovenia' 'Jordan'
 'Venezuela' 'Costa Rica' 'Jamaica' 'Thailand' 'Nicaragua' 'Myanmar'
 'Republic of Korea' 'Rwanda' 'Bosnia and Herzegovina' 'Benin'
 'El Salvador' 'Zimbabwe' 'Afghanistan' 'Estonia' 'Malta' 'Uruguay'
 'Belarus' 'Colombia' 'Republic of Moldova' 'Isle of Man' 'Nomadic'
 'New Zealand' 'Palestine' 'Armenia' 'United Arab Emirates' 'Maldives'
 'Ethiopia' 'Fiji' 'Guatemala' 'Uganda' 'Turkmenistan' 'Mauritius' 'Kenya'
 'Cuba' 'Gabon' 'Bahamas' 'South Korea' 'Iceland' 'Honduras'
 'Hong Kong (S.A.R.)' "Lao People's Democratic Republic" 'Mongolia'
 'Cambodia' 'Madagascar' 'Angola' 'Democratic Republic of the Congo'
 'Syrian Arab Republic' 'Iraq' 'Namibia' 'Senegal' 'Kyrgyzstan' 'Zambia'
 'Swaziland' "Côte d'Ivoire" 'Kuwait' 'Tajikistan' 'Burundi'
 'Trinidad and Tobago' 'Mauritania' 'Sierra Leone' 'Panama' 'Somalia'
 'North Korea' 'Dominica' 'Guyana' 'Togo' 'Oman' 'Barbados' 'Andorra'
 "Democratic People's Republic of Korea" 'Qatar' 'Sudan' 'Cameroon'
 'Papua New Guinea' 'Bahrain' 'Yemen' 'Malawi' 'Burkina Faso' 'Congo'
 'Botswana' 'Guinea-Bissau' 'Mozambique' 'Central African Republic'
 'Equatorial Guinea' 'Suriname' 'Belize' 'Libyan Arab Jamahiriya'
 'Cape Verde' 'Brunei Darussalam' 'Bhutan' 'Guinea' 'Niger'
 'Antigua and Barbuda' 'Mali' 'Samoa' 'Lesotho' 'Saint Kitts and Nevis'
 'Monaco' 'Micronesia' 'Haiti' nan 'Nauru' 'Liberia' 'Chad' 'Djibouti'
 'Solomon Islands']
```

Standardize entries in columns like `Country` or `EdLevel` by mapping inconsistent values to a consistent format.

In [18]:
```python
country_mapping = {
    'United States of America': 'USA',
    'United Kingdom of Great Britain and Northern Ireland': 'United Kingdom',
    'Russian Federation': 'Russia',
    'Republic of Korea': 'South Korea',
    'South Korea': 'South Korea',
    'North Korea': 'North Korea',
    "Democratic People's Republic of Korea": 'North Korea',
    "Côte d'Ivoire": 'Ivory Coast',
    'Libyan Arab Jamahiriya': 'Libya',
    'Venezuela': 'Venezuela',
    'Hong Kong (S.A.R.)': 'Hong Kong',
    'Lao People\'s Democratic Republic': 'Laos',
    'Micronesia': 'Micronesia',
    'Republic of Moldova': 'Moldova',
```

```python
    'Democratic Republic of the Congo': 'DR Congo',
    'Congo': 'Republic of Congo',
    'Iran': 'Iran',
    'Nomadic': np.nan,  # To treat it as missing
    # Add any more mappings as needed
}

df['Country'] = df['Country'].replace(country_mapping)

print('Unique Countries after Standardization: ', df['Country'].unique())
```

```
Unique Countries after Standardization:  ['USA' 'United Kingdom' 'Canada' 'Norway'
'Uzbekistan' 'Serbia' 'Poland'
 'Philippines' 'Bulgaria' 'Switzerland' 'India' 'Germany' 'Ireland'
 'Italy' 'Ukraine' 'Australia' 'Brazil' 'Japan' 'Austria' 'Iran' 'France'
 'Saudi Arabia' 'Romania' 'Turkey' 'Nepal' 'Algeria' 'Sweden'
 'Netherlands' 'Croatia' 'Pakistan' 'Czech Republic'
 'Republic of North Macedonia' 'Finland' 'Slovakia' 'Russia' 'Greece'
 'Israel' 'Belgium' 'Mexico' 'United Republic of Tanzania' 'Hungary'
 'Argentina' 'Portugal' 'Sri Lanka' 'Latvia' 'China' 'Singapore' 'Lebanon'
 'Spain' 'South Africa' 'Lithuania' 'Viet Nam' 'Dominican Republic'
 'Indonesia' 'Kosovo' 'Morocco' 'Taiwan' 'Georgia' 'San Marino' 'Tunisia'
 'Bangladesh' 'Nigeria' 'Liechtenstein' 'Denmark' 'Ecuador' 'Malaysia'
 'Albania' 'Azerbaijan' 'Chile' 'Ghana' 'Peru' 'Bolivia' 'Egypt'
 'Luxembourg' 'Montenegro' 'Cyprus' 'Paraguay' 'Kazakhstan' 'Slovenia'
 'Jordan' 'Venezuela' 'Costa Rica' 'Jamaica' 'Thailand' 'Nicaragua'
 'Myanmar' 'South Korea' 'Rwanda' 'Bosnia and Herzegovina' 'Benin'
 'El Salvador' 'Zimbabwe' 'Afghanistan' 'Estonia' 'Malta' 'Uruguay'
 'Belarus' 'Colombia' 'Moldova' 'Isle of Man' nan 'New Zealand'
 'Palestine' 'Armenia' 'United Arab Emirates' 'Maldives' 'Ethiopia' 'Fiji'
 'Guatemala' 'Uganda' 'Turkmenistan' 'Mauritius' 'Kenya' 'Cuba' 'Gabon'
 'Bahamas' 'Iceland' 'Honduras' 'Hong Kong' 'Laos' 'Mongolia' 'Cambodia'
 'Madagascar' 'Angola' 'DR Congo' 'Syrian Arab Republic' 'Iraq' 'Namibia'
 'Senegal' 'Kyrgyzstan' 'Zambia' 'Swaziland' 'Ivory Coast' 'Kuwait'
 'Tajikistan' 'Burundi' 'Trinidad and Tobago' 'Mauritania' 'Sierra Leone'
 'Panama' 'Somalia' 'North Korea' 'Dominica' 'Guyana' 'Togo' 'Oman'
 'Barbados' 'Andorra' 'Qatar' 'Sudan' 'Cameroon' 'Papua New Guinea'
 'Bahrain' 'Yemen' 'Malawi' 'Burkina Faso' 'Republic of Congo' 'Botswana'
 'Guinea-Bissau' 'Mozambique' 'Central African Republic'
 'Equatorial Guinea' 'Suriname' 'Belize' 'Libya' 'Cape Verde'
 'Brunei Darussalam' 'Bhutan' 'Guinea' 'Niger' 'Antigua and Barbuda'
 'Mali' 'Samoa' 'Lesotho' 'Saint Kitts and Nevis' 'Monaco' 'Micronesia'
 'Haiti' 'Nauru' 'Liberia' 'Chad' 'Djibouti' 'Solomon Islands']
```

`EdLevel`

In [19]:
```python
df['EdLevel'].unique()
```

Out[19]:
```
array(['Primary/elementary school',
       'Bachelor’s degree (B.A., B.S., B.Eng., etc.)',
       'Master’s degree (M.A., M.S., M.Eng., MBA, etc.)',
       'Some college/university study without earning a degree',
       'Secondary school (e.g. American high school, German Realschule or Gymnasiu
m, etc.)',
       'Professional degree (JD, MD, Ph.D, Ed.D, etc.)',
       'Associate degree (A.A., A.S., etc.)', 'Something else', nan],
      dtype=object)
```

In [20]:
```
df['EdLevel'].value_counts(dropna=False)
```

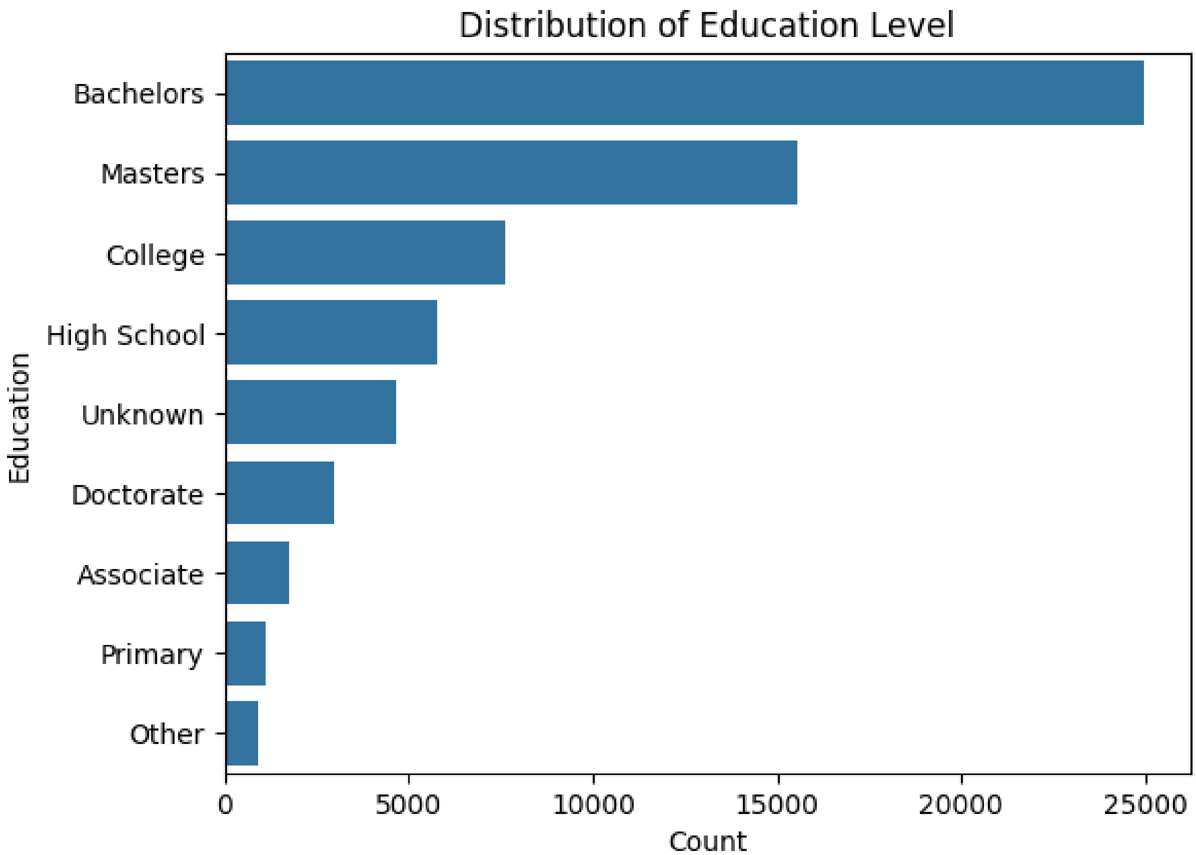
Out[20]:
```
EdLevel
Bachelor’s degree (B.A., B.S., B.Eng., etc.)
24942
Master’s degree (M.A., M.S., M.Eng., MBA, etc.)
15557
Some college/university study without earning a degree
7651
Secondary school (e.g. American high school, German Realschule or Gymnasium, etc.)
5793
NaN
4653
Professional degree (JD, MD, Ph.D, Ed.D, etc.)
2970
Associate degree (A.A., A.S., etc.)
1793
Primary/elementary school
1146
Something else
932
Name: count, dtype: int64
```

In [21]:
```
education_mapping = {
    'Primary/elementary school': 'Primary',
    'Secondary school (e.g. American high school, German Realschule or Gymnasium, e
    'Some college/university study without earning a degree': 'College',
    'Associate degree (A.A., A.S., etc.)': 'Associate',
    'Bachelor’s degree (B.A., B.S., B.Eng., etc.)': 'Bachelors',
    'Master’s degree (M.A., M.S., M.Eng., MBA, etc.)': 'Masters',
    'Professional degree (JD, MD, Ph.D, Ed.D, etc.)': 'Doctorate',
    'Something else': 'Other',
    np.nan: 'Unknown'  # or leave as nan if preferred
}

df['EdLevel'] = df['EdLevel'].map(education_mapping)

print('Unique Education levels: ', df['EdLevel'].unique())
```

```
Unique Education levels:  ['Primary' 'Bachelors' 'Masters' 'College' 'High School'
'Doctorate'
 'Associate' 'Other' 'Unknown']
```

In [22]:
```
# Plot Countplot

sns.countplot(data=df, y='EdLevel', order=df['EdLevel'].value_counts().index)
```

```
plt.title("Distribution of Education Level")
plt.xlabel("Count")
plt.ylabel("Education")
plt.show()
```



`RemoteWork`

In [23]:
```
print("Work Type Preference Value Counts: ")
df['RemoteWork'].value_counts(dropna=False)
```

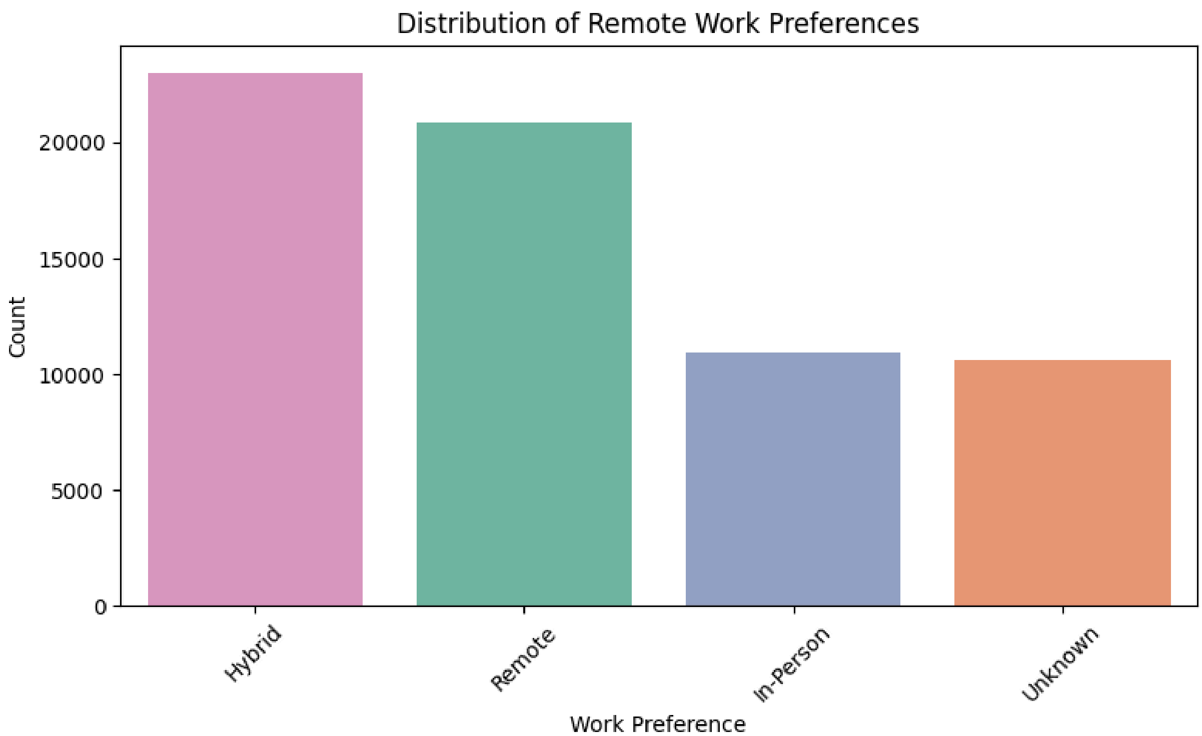
```
Work Type Preference Value Counts:
```

Out[23]:
```
RemoteWork
Hybrid (some remote, some in-person)    23015
Remote                                  20831
In-person                               10960
NaN                                     10631
Name: count, dtype: int64
```

In [24]:
```
remotework_mapping = {
    "Hybrid (some remote, some in-person)": "Hybrid",
    "Remote": "Remote",
    "In-person": "In-Person"
}

df['RemoteWork'] = df['RemoteWork'].map(remotework_mapping).fillna('Unknown')

print('Unique Work Types: ', df['RemoteWork'].unique())
```

```
Unique Work Types:  ['Remote' 'Unknown' 'In-Person' 'Hybrid']
```

In [25]:
```python
# Count Plot for Remote Work Preferences

plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='RemoteWork', order=df['RemoteWork'].value_counts().index,
plt.title("Distribution of Remote Work Preferences")
plt.xlabel("Work Preference")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Obs:

`Hybrid` work is the most common (23K), followed by `Remote` (20.8K) and `In-person` (10.9K).

- Majority of tech professionals prefer or are engaged in **flexible** (hybrid/remote) work setups.
- Employers should support **remote-friendly** policies and infrastructure.

`Age`

In [26]:
```python
print('Age Groups ad their Value Counts: ')
df['Age'].value_counts()
```

```
Age Groups ad their Value Counts:
```

Out[26]:
```
Age
25-34 years old       23911
35-44 years old       14942
18-24 years old       14098
45-54 years old        6249
55-64 years old        2575
Under 18 years old     2568
65 years or older       772
Prefer not to say       322
Name: count, dtype: int64
```

In [27]:
```python
age_mapping = {
    '25-34 years old': 29.5,
    '18-24 years old': 21.0,
    '35-44 years old': 39.5,
    '45-54 years old': 49.5,
    'Under 18 years old': 17.0, # Assuming max 17 for this category
    '55-64 years old': 59.5,
    '65 years or older': 70.0,   # Assuming midpoint of 65-75 for this category
    'Prefer not to say': 'Not Specified'
}

df['AgeNumeric'] = df['Age'].map(age_mapping)

df['AgeNumeric'].value_counts(dropna=False)
```

Out[27]:
```
AgeNumeric
29.5             23911
39.5             14942
21.0             14098
49.5              6249
59.5              2575
17.0              2568
70.0               772
Not Specified      322
Name: count, dtype: int64
```

Employment

In [28]:
```python
df['Employment'].value_counts(dropna=False)
```

Out[28]:
```
Employment
Employed, full-time
39041
Independent contractor, freelancer, or self-employed
4846
Student, full-time
4709
Employed, full-time;Independent contractor, freelancer, or self-employed
3557
Not employed, but looking for work
2341

...
Employed, full-time;Student, full-time;Independent contractor, freelancer, or self
-employed;Student, part-time;Employed, part-time;Retired                1
Employed, full-time;Independent contractor, freelancer, or self-employed;Student,
part-time;Retired                                                       1
Employed, full-time;Independent contractor, freelancer, or self-employed;Employed,
part-time;Retired                                                      1
Student, full-time;Not employed, but looking for work;Independent contractor, free
lancer, or self-employed;Not employed, and not looking for work        1
Not employed, but looking for work;Independent contractor, freelancer, or self-emp
loyed;Student, part-time;Retired                                        1
Name: count, Length: 110, dtype: int64
```

In [29]:
```
df['Employment'].unique()
```

```
Out[29]: array(['Employed, full-time', 'Student, full-time',
        'Student, full-time;Not employed, but looking for work',
        'Independent contractor, freelancer, or self-employed',
        'Not employed, and not looking for work',
        'Employed, full-time;Student, part-time',
        'Employed, full-time;Independent contractor, freelancer, or self-employed',
        'Employed, full-time;Student, full-time', 'Employed, part-time',
        'Student, full-time;Employed, part-time',
        'Student, part-time;Employed, part-time', 'I prefer not to say',
        'Not employed, but looking for work', 'Student, part-time',
        'Employed, full-time;Student, full-time;Independent contractor, freelancer,
or self-employed;Employed, part-time',
        'Employed, full-time;Independent contractor, freelancer, or self-employed;S
tudent, part-time',
        'Independent contractor, freelancer, or self-employed;Employed, part-time',
        'Independent contractor, freelancer, or self-employed;Student, part-time;Em
ployed, part-time',
        'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed',
        'Student, full-time;Independent contractor, freelancer, or self-employed',
        'Employed, full-time;Employed, part-time',
        'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed',
        'Student, full-time;Not employed, and not looking for work',
        'Retired',
        'Independent contractor, freelancer, or self-employed;Student, part-time',
        'Employed, full-time;Independent contractor, freelancer, or self-employed;E
mployed, part-time',
        'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Student, part-time',
        'Not employed, but looking for work;Student, part-time',
        'Not employed, but looking for work;Not employed, and not looking for wor
k',
        'Independent contractor, freelancer, or self-employed;Retired',
        'Not employed, but looking for work;Student, part-time;Employed, part-tim
e',
        'Student, full-time;Not employed, but looking for work;Not employed, and no
t looking for work',
        'Employed, full-time;Not employed, but looking for work',
        'Student, full-time;Not employed, and not looking for work;Student, part-ti
me',
        'Employed, full-time;Retired',
        'Employed, full-time;Independent contractor, freelancer, or self-employed;S
tudent, part-time;Employed, part-time',
        'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Not employed, and not looking for work',
        'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Employed, part-time',
        'Not employed, but looking for work;Employed, part-time',
        'Employed, full-time;Student, full-time;Employed, part-time',
        'Independent contractor, freelancer, or self-employed;Not employed, and not
looking for work',
        'Not employed, and not looking for work;Student, part-time',
        'Student, full-time;Independent contractor, freelancer, or self-employed;Em
ployed, part-time',
        'Student, full-time;Student, part-time',
```

```
       'Student, full-time;Not employed, but looking for work;Student, part-time',
       'Independent contractor, freelancer, or self-employed;Not employed, and not
looking for work;Retired',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;N
ot employed, and not looking for work',
       'Employed, full-time;Student, full-time;Independent contractor, freelancer,
or self-employed',
       'Employed, full-time;Student, full-time;Student, part-time',
       'Not employed, but looking for work;Retired',
       'Employed, full-time;Student, full-time;Not employed, but looking for wor
k',
       'Not employed, and not looking for work;Retired',
       'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Not employed, and not looking for work;Retired',
       'Employed, full-time;Not employed, but looking for work;Employed, part-tim
e',
       'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed;Student, part-time;Employed, part-time;Retired',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;N
ot employed, and not looking for work;Employed, part-time',
       'Student, full-time;Independent contractor, freelancer, or self-employed;No
t employed, and not looking for work',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed;Not employed, and not looking
for work;Student, part-time;Employed, part-time;Retired',
       'Employed, full-time;Not employed, but looking for work;Independent contrac
tor, freelancer, or self-employed',
       'Independent contractor, freelancer, or self-employed;Not employed, and not
looking for work;Student, part-time',
       'Student, full-time;Not employed, but looking for work;Retired',
       'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed;Student, part-time',
       'Student, part-time;Retired',
       'Student, full-time;Not employed, but looking for work;Not employed, and no
t looking for work;Student, part-time',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed;Student, part-time;Employed,
part-time',
       'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Retired',
       'Employed, full-time;Student, full-time;Student, part-time;Employed, part-t
ime',
       'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Student, part-time;Employed, part-time',
       'Student, full-time;Not employed, but looking for work;Employed, part-tim
e',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;N
ot employed, and not looking for work;Student, part-time',
       'Independent contractor, freelancer, or self-employed;Student, part-time;Re
tired',
       'Student, full-time;Independent contractor, freelancer, or self-employed;St
udent, part-time;Employed, part-time',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;S
tudent, part-time;Retired',
       'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed;Not employed, and not looking for work',
```

```
       'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed;Employed, part-time',
       'Student, full-time;Independent contractor, freelancer, or self-employed;St
udent, part-time',
       'Independent contractor, freelancer, or self-employed;Employed, part-time;R
etired',
       'Employed, full-time;Not employed, and not looking for work',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;R
etired',
       'Student, full-time;Student, part-time;Employed, part-time',
       'Employed, part-time;Retired',
       'Employed, full-time;Independent contractor, freelancer, or self-employed;E
mployed, part-time;Retired',
       'Employed, full-time;Student, part-time;Employed, part-time',
       'Employed, full-time;Student, full-time;Independent contractor, freelancer,
or self-employed;Student, part-time;Employed, part-time;Retired',
       'Student, full-time;Student, part-time;Retired',
       'Student, full-time;Not employed, and not looking for work;Employed, part-t
ime',
       'Employed, full-time;Not employed, but looking for work;Independent contrac
tor, freelancer, or self-employed;Employed, part-time',
       'Not employed, but looking for work;Not employed, and not looking for work;
Student, part-time;Employed, part-time',
       'Independent contractor, freelancer, or self-employed;Not employed, and not
looking for work;Employed, part-time',
       'Employed, full-time;Not employed, but looking for work;Independent contrac
tor, freelancer, or self-employed;Not employed, and not looking for work;Employed,
part-time',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed;Not employed, and not looking
for work;Student, part-time;Employed, part-time',
       'Employed, full-time;Student, full-time;Independent contractor, freelancer,
or self-employed;Student, part-time;Employed, part-time',
       'Not employed, and not looking for work;Employed, part-time',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Student, part-time',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed;Employed, part-time',
       'Employed, full-time;Not employed, but looking for work;Not employed, and n
ot looking for work;Employed, part-time',
       'Student, full-time;Independent contractor, freelancer, or self-employed;Em
ployed, part-time;Retired',
       'Not employed, but looking for work;Student, part-time;Retired',
       'Independent contractor, freelancer, or self-employed;Not employed, and not
looking for work;Student, part-time;Retired',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed',
       'Not employed, but looking for work;Not employed, and not looking for work;
Student, part-time',
       'Employed, full-time;Student, full-time;Independent contractor, freelancer,
or self-employed;Student, part-time;Retired',
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Student, part-time;Employed, part-time',
       'Student, full-time;Not employed, but looking for work;Independent contract
or, freelancer, or self-employed;Not employed, and not looking for work;Student, p
art-time',
```

```
       'Employed, full-time;Student, full-time;Not employed, but looking for work;
Independent contractor, freelancer, or self-employed;Student, part-time;Employed,
part-time;Retired',
       'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Not employed, and not looking for work;Employed, part-time',
       'Student, full-time;Retired',
       'Employed, full-time;Not employed, but looking for work;Student, part-tim
e',
       'Not employed, and not looking for work;Student, part-time;Employed, part-t
ime',
       'Not employed, but looking for work;Independent contractor, freelancer, or
self-employed;Student, part-time;Retired'],
      dtype=object)
```

In [30]:
```python
def categorize_employment_final(employment_string):
    if pd.isna(employment_string) or str(employment_string).strip() == '':
        return 'Unknown'

    # Handle "I prefer not to say" directly, as it's a unique response
    if 'I prefer not to say' == employment_string.strip():
        return 'Prefer not to say'

    # Map raw roles to a standardized set of core types for easier comparison
    # We use a set to automatically handle duplicates (like "Student, full-time" an
    raw_roles = [role.strip() for role in employment_string.split(';')]

    standardized_roles = set()
    has_ft_employed = False
    has_pt_employed = False
    has_self_employed = False
    has_student = False
    has_looking_work = False
    has_not_looking_work = False
    has_retired = False

    for role in raw_roles:
        if role == 'Employed, full-time':
            has_ft_employed = True
            standardized_roles.add('Employed (Full-time)')
        elif role == 'Employed, part-time':
            has_pt_employed = True
            standardized_roles.add('Employed (Part-time)')
        elif role == 'Independent contractor, freelancer, or self-employed':
            has_self_employed = True
            standardized_roles.add('Self-employed')
        elif 'Student' in role: # Catches both full-time and part-time student
            has_student = True
            standardized_roles.add('Student')
        elif role == 'Not employed, but looking for work':
            has_looking_work = True
            standardized_roles.add('Unemployed (Looking for work)')
        elif role == 'Not employed, and not looking for work':
            has_not_looking_work = True
            standardized_roles.add('Unemployed (Not looking for work)')
        elif role == 'Retired':
            has_retired = True
```

```
            standardized_roles.add('Retired')
        # Any other unknown individual roles will not be added to standardized_role
        # and will contribute to the 'Mixed' count if they result in >3 roles.

    num_roles = len(standardized_roles)

    # Rule: Mixed for entries that have more than three labels (standardized)
    if num_roles > 3:
        return 'Mixed'

    # --- Constructing the final label based on standardized roles ---

    # Priority 1: Full-time employed
    if has_ft_employed:
        if has_student and not (has_pt_employed or has_self_employed or has_looking
            return 'Employed (Full-time) + Student'
        if has_self_employed and not (has_pt_employed or has_student or has_looking
            return 'Employed (Full-time) + Self-employed'
        if has_looking_work and not (has_pt_employed or has_student or has_self_emp
            return 'Employed (Full-time) + Unemployed (Looking for work)'
        if has_not_looking_work and not (has_pt_employed or has_student or has_self
            return 'Employed (Full-time) + Unemployed (Not looking for work)'
        if has_retired and not (has_pt_employed or has_student or has_self_employed
            return 'Employed (Full-time) + Retired'

        # If FT employed is present, but it's not a simple FT + one other, and not
        # it probably involves more complex combos or is just FT
        if num_roles == 1:
            return 'Employed (Full-time)'
        else: # For FT + 2 other types, or specific unlisted FT combos with 2 roles
            return 'Mixed' # Or you can add specific 3-role FT combos here if commo

    # Priority 2: Part-time employed
    if has_pt_employed:
        if has_student and not (has_self_employed or has_looking_work or has_not_lo
            return 'Employed (Part-time) + Student'
        if has_self_employed and not (has_student or has_looking_work or has_not_lo
            return 'Employed (Part-time) + Self-employed'
        if has_looking_work and not (has_student or has_self_employed or has_not_lo
            return 'Employed (Part-time) + Unemployed (Looking for work)'
        if has_not_looking_work and not (has_student or has_self_employed or has_lo
            return 'Employed (Part-time) + Unemployed (Not looking for work)'
        if has_retired and not (has_student or has_self_employed or has_looking_wor
            return 'Employed (Part-time) + Retired'

        if num_roles == 1:
            return 'Employed (Part-time)'
        else: # For PT + 2 other types, or specific unlisted PT combos with 2 roles
            return 'Mixed'


    # Priority 3: Self-employed
    if has_self_employed:
        if has_student and not (has_looking_work or has_not_looking_work or has_ret
            return 'Self-employed + Student'
        if has_looking_work and not (has_student or has_not_looking_work or has_ret
```

```python
            return 'Self-employed + Unemployed (Looking for work)'
        if has_not_looking_work and not (has_student or has_looking_work or has_ret
            return 'Self-employed + Unemployed (Not looking for work)'
        if has_retired and not (has_student or has_looking_work or has_not_looking_
            return 'Self-employed + Retired'

        # 3-role combinations involving self-employed
        if has_student and has_looking_work:
             return 'Student + Unemployed (Looking for work) + Self-employed'
        if has_student and has_not_looking_work:
             return 'Student + Unemployed (Not looking for work) + Self-employed'

        if num_roles == 1:
            return 'Self-employed'
        else: # For SE + 2 other types (not specifically defined above)
            return 'Mixed' # Or specific 3-role SE combos if common

    # Priority 4: Student
    if has_student:
        if has_looking_work and not has_not_looking_work and not has_retired:
            return 'Student + Unemployed (Looking for work)'
        if has_not_looking_work and not has_looking_work and not has_retired:
            return 'Student + Unemployed (Not looking for work)'
        if has_retired and not (has_looking_work or has_not_looking_work):
            return 'Student + Retired'

        if num_roles == 1:
            return 'Student' # Will cover "Student, full-time" and "Student, part-t
        else: # For Student + 2 other types (not specifically defined above)
            return 'Mixed'

    # Priority 5: Unemployed (Looking for work)
    if has_looking_work:
        if has_not_looking_work and not has_retired:
            return 'Unemployed (Looking for work) + Unemployed (Not looking for wor
        if has_retired and not has_not_looking_work:
            return 'Unemployed (Looking for work) + Retired'

        if num_roles == 1:
            return 'Unemployed (Looking for work)'
        else: # For Unemployed (Looking) + 2 others (not specified)
            return 'Mixed'

    # Priority 6: Unemployed (Not looking for work)
    if has_not_looking_work:
        if has_retired and not has_looking_work:
            return 'Unemployed (Not looking for work) + Retired'
        if num_roles == 1:
            return 'Unemployed (Not looking for work)'
        else: # For Unemployed (Not Looking) + 2 others (not specified)
            return 'Mixed'

    # Priority 7: Retired (if not part of any prior combo)
    if has_retired:
        return 'Retired' # Should only be hit if retired is the only status
```

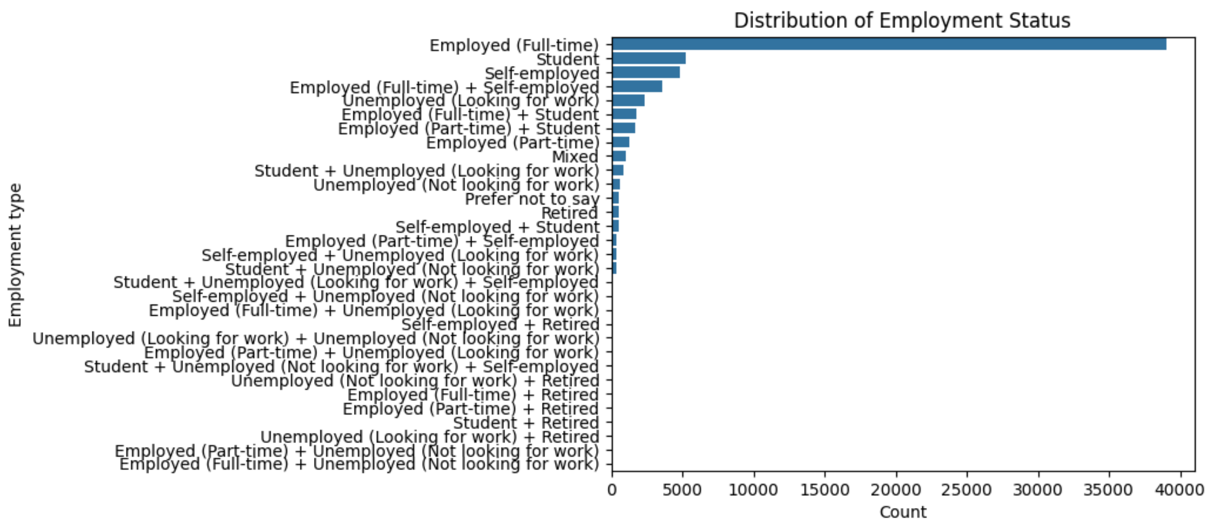
```
    # Fallback for any unhandled edge cases
    return 'Other/Undefined'
```

In [31]:
```
# Apply the categorization function

df['Employment_Simplified'] = df['Employment'].apply(categorize_employment_final)

print("\nFinal Simplified 'Employment' distribution : ")
print(df['Employment_Simplified'].value_counts())

print("\nUnique simplified 'Employment' categories : ")
print(df['Employment_Simplified'].unique())
```

```
Final Simplified 'Employment' distribution :
Employment_Simplified
Employed (Full-time)                                                 39041
Student                                                               5254
Self-employed                                                         4846
Employed (Full-time) + Self-employed                                  3557
Unemployed (Looking for work)                                         2341
Employed (Full-time) + Student                                        1738
Employed (Part-time) + Student                                        1680
Employed (Part-time)                                                  1266
Mixed                                                                 1059
Student + Unemployed (Looking for work)                                839
Unemployed (Not looking for work)                                      633
Prefer not to say                                                      546
Retired                                                                525
Self-employed + Student                                                517
Employed (Part-time) + Self-employed                                   401
Self-employed + Unemployed (Looking for work)                          383
Student + Unemployed (Not looking for work)                            370
Student + Unemployed (Looking for work) + Self-employed                137
Self-employed + Unemployed (Not looking for work)                       58
Employed (Full-time) + Unemployed (Looking for work)                    52
Self-employed + Retired                                                 51
Unemployed (Looking for work) + Unemployed (Not looking for work)       40
Employed (Part-time) + Unemployed (Looking for work)                    27
Student + Unemployed (Not looking for work) + Self-employed             21
Unemployed (Not looking for work) + Retired                             17
Employed (Full-time) + Retired                                          13
Employed (Part-time) + Retired                                           8
Student + Retired                                                        7
Unemployed (Looking for work) + Retired                                  5
Employed (Part-time) + Unemployed (Not looking for work)                 3
Employed (Full-time) + Unemployed (Not looking for work)                 2
Name: count, dtype: int64

Unique simplified 'Employment' categories :
['Employed (Full-time)' 'Student'
 'Student + Unemployed (Looking for work)' 'Self-employed'
 'Unemployed (Not looking for work)' 'Employed (Full-time) + Student'
 'Employed (Full-time) + Self-employed' 'Employed (Part-time)'
 'Employed (Part-time) + Student' 'Prefer not to say'
 'Unemployed (Looking for work)' 'Mixed'
 'Employed (Part-time) + Self-employed'
 'Student + Unemployed (Looking for work) + Self-employed'
 'Self-employed + Student' 'Self-employed + Unemployed (Looking for work)'
 'Student + Unemployed (Not looking for work)' 'Retired'
 'Unemployed (Looking for work) + Unemployed (Not looking for work)'
 'Self-employed + Retired'
 'Employed (Full-time) + Unemployed (Looking for work)'
 'Employed (Full-time) + Retired'
 'Employed (Part-time) + Unemployed (Looking for work)'
 'Self-employed + Unemployed (Not looking for work)'
 'Unemployed (Looking for work) + Retired'
 'Unemployed (Not looking for work) + Retired'
 'Student + Unemployed (Not looking for work) + Self-employed'
 'Student + Retired'
```

```
'Employed (Full-time) + Unemployed (Not looking for work)'
'Employed (Part-time) + Retired'
'Employed (Part-time) + Unemployed (Not looking for work)']
```

In [32]:
```python
# Plot Count Plot

sns.countplot(data=df, y='Employment_Simplified', order=df['Employment_Simplified']
plt.title("Distribution of Employment Status")
plt.xlabel("Count")
plt.ylabel("Employment type")
plt.show()
```



Handle Duplicates

Identify and Analyze Duplicates

Identify Duplicate Rows

- Count the number of duplicate rows in the dataset.
- Display the first few duplicate rows to understand their structure.

In [33]:
```python
df.duplicated()
```

Out[33]:
```
0        False
1        False
2        False
3        False
4        False
         ...
65432    False
65433    False
65434    False
65435    False
65436    False
Length: 65437, dtype: bool
```

In [34]:
```python
df.duplicated().sum()
```

Out[34]: 0

Obs: There are no duplicate rows.

Analyze Characteristics of Duplicates

- Identify duplicate rows based on selected columns such as `MainBranch`, `Employment`, and `RemoteWork`. Analyse which columns frequently contain identical values within these duplicate rows.
- Analyse the characteristics of rows that are duplicates based on a subset of columns, such as `MainBranch`, `Employment`, and `RemoteWork`. Determine which columns frequently have identical values across these rows.

In [35]:
```
# TODO: Do this later on when required!
```

Removing Duplicates

Remove Duplicates

- Remove duplicate rows from the dataset using the `drop_duplicates()` function.
- Verify the removal by counting the number of duplicate rows after removal .

Strategic Removal of Duplicates

- Decide which columns are critical for defining uniqueness in the dataset.
- Remove duplicates based on a subset of columns if complete row duplication is not a good criterion.

Handle Missing Values

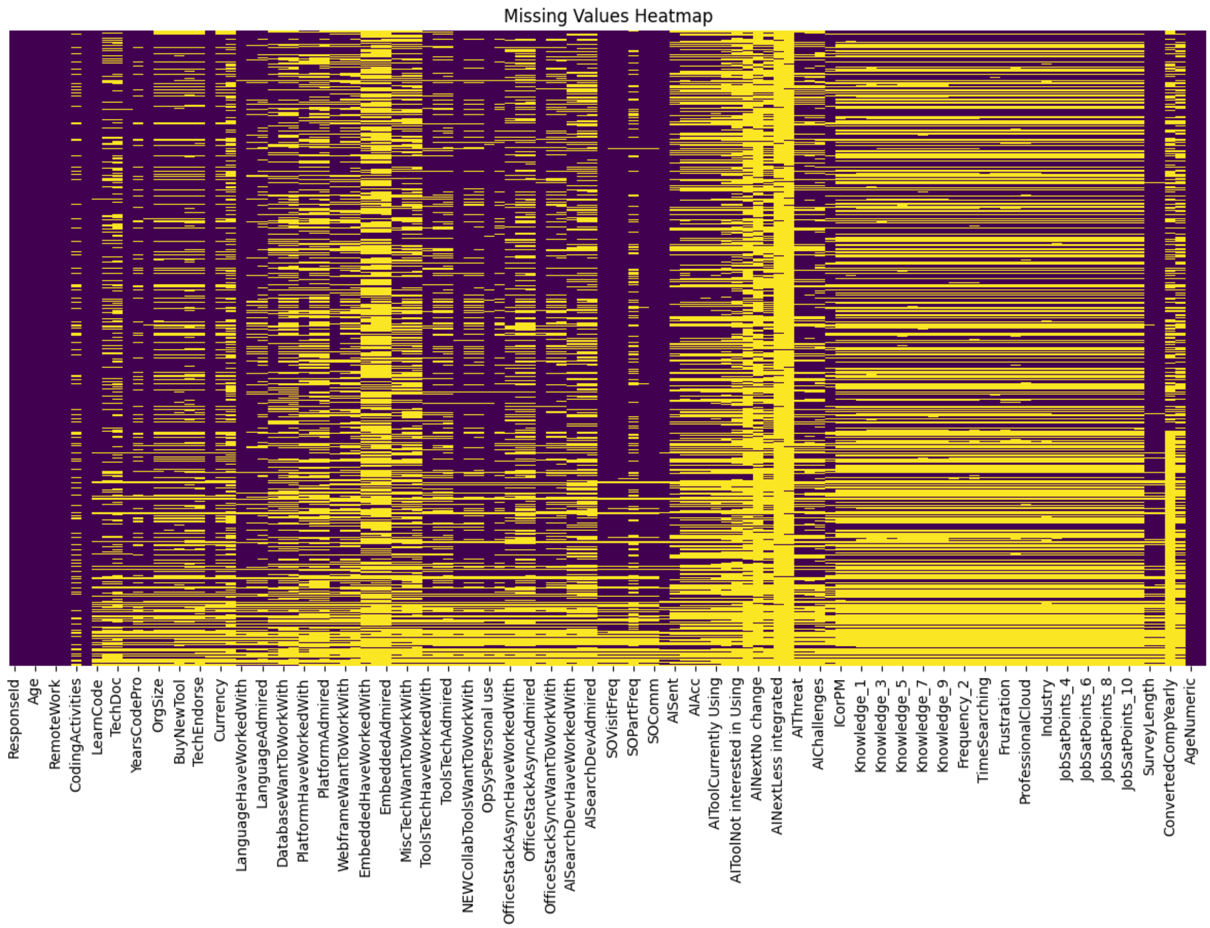
Identify Missing Values for all columns

In [36]:
```
df.isnull().sum()
```

Out[36]:
```
ResponseId                  0
MainBranch                  0
Age                         0
Employment                  0
RemoteWork                  0
                         ...
SurveyEase               9199
ConvertedCompYearly     42002
JobSat                  36311
AgeNumeric                  0
Employment_Simplified       0
Length: 116, dtype: int64
```

Visualize missing values using a heatmap

In [37]:
```python
plt.figure(figsize=(15, 8))
sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='viridis')
plt.title("Missing Values Heatmap")
plt.show()
```



Count the number of missing rows for a specific column (e.g., `Employment`).

In [38]:
```python
columns = df.columns

for column in columns:
    print(f'Null value for {column}: {df[column].isnull().sum()}')
```

```
Null value for ResponseId: 0
Null value for MainBranch: 0
Null value for Age: 0
Null value for Employment: 0
Null value for RemoteWork: 0
Null value for Check: 0
Null value for CodingActivities: 10971
Null value for EdLevel: 0
Null value for LearnCode: 4949
Null value for LearnCodeOnline: 16200
Null value for TechDoc: 24540
Null value for YearsCode: 5568
Null value for YearsCodePro: 13827
Null value for DevType: 5992
Null value for OrgSize: 17957
Null value for PurchaseInfluence: 18031
Null value for BuyNewTool: 20256
Null value for BuildvsBuy: 22079
Null value for TechEndorse: 21769
Null value for Country: 6550
Null value for Currency: 18753
Null value for CompTotal: 31697
Null value for LanguageHaveWorkedWith: 5692
Null value for LanguageWantToWorkWith: 9685
Null value for LanguageAdmired: 14565
Null value for DatabaseHaveWorkedWith: 15183
Null value for DatabaseWantToWorkWith: 22879
Null value for DatabaseAdmired: 26880
Null value for PlatformHaveWorkedWith: 23071
Null value for PlatformWantToWorkWith: 30905
Null value for PlatformAdmired: 34060
Null value for WebframeHaveWorkedWith: 20276
Null value for WebframeWantToWorkWith: 26902
Null value for WebframeAdmired: 30494
Null value for EmbeddedHaveWorkedWith: 43223
Null value for EmbeddedWantToWorkWith: 47837
Null value for EmbeddedAdmired: 48704
Null value for MiscTechHaveWorkedWith: 25994
Null value for MiscTechWantToWorkWith: 32473
Null value for MiscTechAdmired: 35841
Null value for ToolsTechHaveWorkedWith: 12955
Null value for ToolsTechWantToWorkWith: 19353
Null value for ToolsTechAdmired: 21440
Null value for NEWCollabToolsHaveWorkedWith: 7845
Null value for NEWCollabToolsWantToWorkWith: 13350
Null value for NEWCollabToolsAdmired: 14726
Null value for OpSysPersonal use: 7263
Null value for OpSysProfessional use: 12464
Null value for OfficeStackAsyncHaveWorkedWith: 17344
Null value for OfficeStackAsyncWantToWorkWith: 26471
Null value for OfficeStackAsyncAdmired: 28233
Null value for OfficeStackSyncHaveWorkedWith: 9892
Null value for OfficeStackSyncWantToWorkWith: 18726
Null value for OfficeStackSyncAdmired: 20725
Null value for AISearchDevHaveWorkedWith: 20984
Null value for AISearchDevWantToWorkWith: 28736
```

```
Null value for AISearchDevAdmired: 29894
Null value for NEWSOSites: 5151
Null value for SOVisitFreq: 5901
Null value for SOAccount: 5877
Null value for SOPartFreq: 20200
Null value for SOHow: 6475
Null value for SOComm: 6274
Null value for AISelect: 4530
Null value for AISent: 19564
Null value for AIBen: 28543
Null value for AIAcc: 28135
Null value for AIComplex: 28416
Null value for AIToolCurrently Using: 30365
Null value for AIToolInterested in Using: 34746
Null value for AIToolNot interested in Using: 41023
Null value for AINextMuch more integrated: 51999
Null value for AINextNo change: 52939
Null value for AINextMore integrated: 41009
Null value for AINextLess integrated: 63082
Null value for AINextMuch less integrated: 64289
Null value for AIThreat: 20748
Null value for AIEthics: 23889
Null value for AIChallenges: 27906
Null value for TBranch: 20960
Null value for ICorPM: 35636
Null value for WorkExp: 35779
Null value for Knowledge_1: 36773
Null value for Knowledge_2: 37416
Null value for Knowledge_3: 37342
Null value for Knowledge_4: 37407
Null value for Knowledge_5: 37557
Null value for Knowledge_6: 37573
Null value for Knowledge_7: 37659
Null value for Knowledge_8: 37679
Null value for Knowledge_9: 37802
Null value for Frequency_1: 37068
Null value for Frequency_2: 37073
Null value for Frequency_3: 37727
Null value for TimeSearching: 36526
Null value for TimeAnswering: 36593
Null value for Frustration: 37186
Null value for ProfessionalTech: 37673
Null value for ProfessionalCloud: 36946
Null value for ProfessionalQuestion: 36630
Null value for Industry: 36579
Null value for JobSatPoints_1: 36113
Null value for JobSatPoints_4: 36044
Null value for JobSatPoints_5: 36026
Null value for JobSatPoints_6: 35987
Null value for JobSatPoints_7: 35989
Null value for JobSatPoints_8: 35981
Null value for JobSatPoints_9: 35981
Null value for JobSatPoints_10: 35987
Null value for JobSatPoints_11: 35992
Null value for SurveyLength: 9255
Null value for SurveyEase: 9199
```

```
Null value for ConvertedCompYearly: 42002
Null value for JobSat: 36311
Null value for AgeNumeric: 0
Null value for Employment_Simplified: 0
```

In [39]:
```python
df['Country'].isnull().sum()
```

Out[39]:
```
6550
```

Imputing Missing Values

Impute missing values using suitable technique:

- Drop the missing rows
- Replace with Median (for Numeric columns)
- Replace with Most frequent occurence (Mode) (for Categorical Columns)
- Replace with other value (Other/Unknown/Not Specified) (for Categorical Columns)

In [40]:
```python
df['Country'] = df['Country'].fillna('Not Specified')
```

In [41]:
```python
# Verify
df['Country'].isnull().sum()
```

Out[41]:
```
0
```

In [42]:
```python
columns = ['LanguageHaveWorkedWith', 'LanguageWantToWorkWith', 'DatabaseHaveWorkedW

for column in columns:
    print(f'\nNull Values in {column} before: ', df[column].isnull().sum())
    df[column] = df[column].fillna('Not specified')
    print(f'Null Values in {column} after : ', df[column].isnull().sum())
```

```
Null Values in LanguageHaveWorkedWith before:  5692
Null Values in LanguageHaveWorkedWith after :  0

Null Values in LanguageWantToWorkWith before:  9685
Null Values in LanguageWantToWorkWith after :  0

Null Values in DatabaseHaveWorkedWith before:  15183
Null Values in DatabaseHaveWorkedWith after :  0

Null Values in DatabaseWantToWorkWith before:  22879
Null Values in DatabaseWantToWorkWith after :  0

Null Values in PlatformHaveWorkedWith before:  23071
Null Values in PlatformHaveWorkedWith after :  0

Null Values in PlatformWantToWorkWith before:  30905
Null Values in PlatformWantToWorkWith after :  0

Null Values in WebframeHaveWorkedWith before:  20276
Null Values in WebframeHaveWorkedWith after :  0

Null Values in WebframeWantToWorkWith before:  26902
Null Values in WebframeWantToWorkWith after :  0
```

In [43]:
```python
df['JobSat'].isnull().sum()
```

Out[43]:
```
36311
```

In [44]:
```python
missing_ratio = df['JobSat'].isnull().mean()
print(f"JobSat missing: {missing_ratio:.2%}")

print(df['JobSat'].describe())
```

```
JobSat missing: 55.49%
count    29126.000000
mean         6.935041
std          2.088259
min          0.000000
25%          6.000000
50%          7.000000
75%          8.000000
max         10.000000
Name: JobSat, dtype: float64
```

With `55.49%` missing values in JobSat, it only reflects the `44.5%` of respondents who answered the question.

If we impute all nulls with the median, it flattens the distribution, so our grouped medians by experience all become the median value — masking any real trends.

Normalize Data

Normalization is commonly applied to compensation data to bring values within a comparable range.

In [45]:
```
# TODO: When required
```

Data Analysis

Current Technology Usage

Display the top 10 programming languages that respondents have worked with.

`LanguageHaveWorkedWith`

In [46]:
```
df['LanguageHaveWorkedWith'].value_counts()
```

Out[46]:
```
LanguageHaveWorkedWith
Not specified                                                    5692
HTML/CSS;JavaScript;TypeScript                                   1002
Python                                                            832
HTML/CSS;JavaScript;PHP;SQL                                       503
C#                                                                452
                                                                  ...
Bash/Shell (all shells);Java;JavaScript;Python;Ruby;Scala;SQL       1
Bash/Shell (all shells);Go;Groovy;Haskell;Java;Python               1
Bash/Shell (all shells);C#;C++;HTML/CSS;JavaScript;MATLAB;Python;SQL   1
Bash/Shell (all shells);JavaScript;Perl;Python;Ruby;TypeScript      1
C;HTML/CSS;Java;JavaScript;PHP;Python;TypeScript                    1
Name: count, Length: 23865, dtype: int64
```

In [47]:
```
df['LanguageHaveWorkedWith'].value_counts().head(200)
```

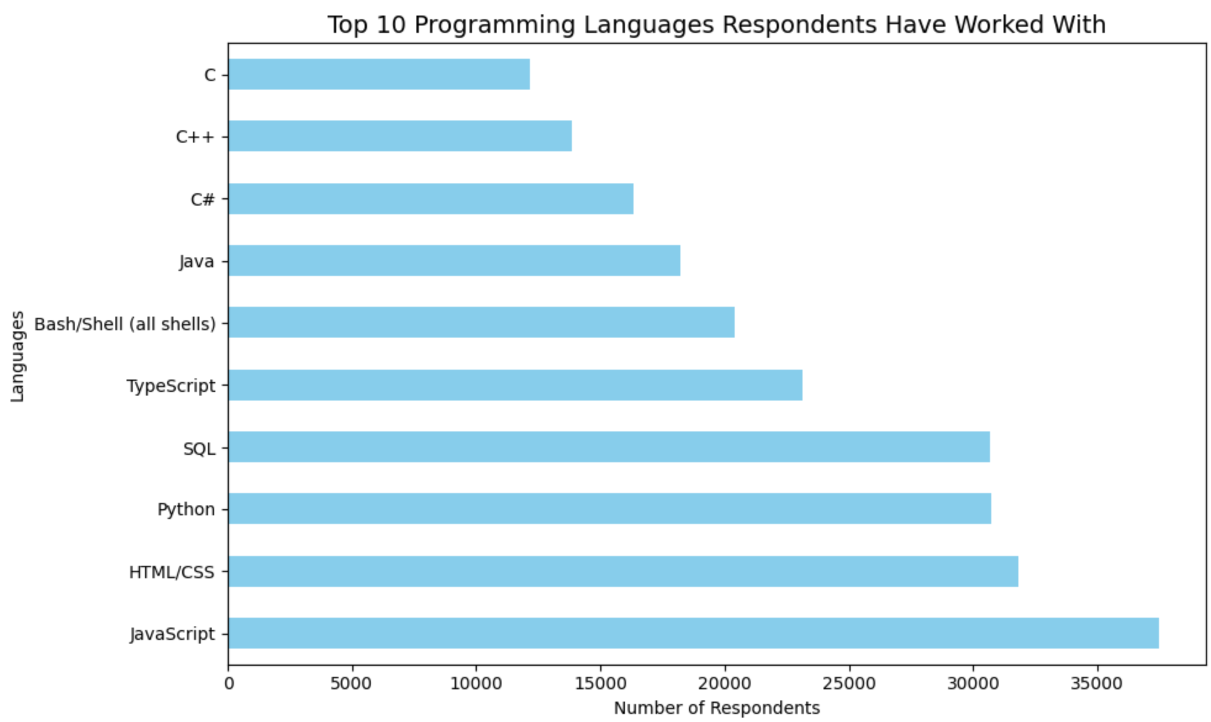
Out[47]:
```
LanguageHaveWorkedWith
Not specified                              5692
HTML/CSS;JavaScript;TypeScript             1002
Python                                      832
HTML/CSS;JavaScript;PHP;SQL                 503
C#                                          452
                                           ...
Dart;Python                                  30
Java;Kotlin;Python                           30
Bash/Shell (all shells);C#;PowerShell;SQL    29
Dart;HTML/CSS;JavaScript;TypeScript          29
C#;JavaScript;Python;SQL;TypeScript          29
Name: count, Length: 200, dtype: int64
```

In [48]:
```
# This creates a Series where each individual language gets its own entry
# Split and explode the LanguageHaveWorkedWith column
langworkedwith = df['LanguageHaveWorkedWith'].str.split(';', expand=False)
known_lang = langworkedwith.explode()

# op 10 Languages
top_known_langs = known_lang.value_counts().head(10)

print("Top 10 Programming Languages Respondents Have Worked With: ")
print(top_known_langs)
```

```
Top 10 Programming Languages Respondents Have Worked With:
LanguageHaveWorkedWith
JavaScript              37492
HTML/CSS                31816
Python                  30719
SQL                     30682
TypeScript              23150
Bash/Shell (all shells)    20412
Java                    18239
C#                      16318
C++                     13827
C                       12184
Name: count, dtype: int64
```

In [49]:

```python
# Plot Horizontal Bar Chart

plt.figure(figsize=(10, 6))
top_known_langs.plot(kind='barh', color='skyblue')
plt.title('Top 10 Programming Languages Respondents Have Worked With', fontsize=14)
plt.xlabel('Number of Respondents')
plt.ylabel('Languages')
plt.tight_layout()
plt.show()
```



Obs:

`JavaScript`, `HTML/CSS`, `Python`, and `SQL` dominate usage.

- Core web development and scripting skills are in high demand.
- These remain essential languages for entry and mid-level roles.

Show the top 10 databases that respondents have used.

`DatabaseHaveWorkedWith`

In [50]:
```
df['DatabaseHaveWorkedWith'].value_counts()
```

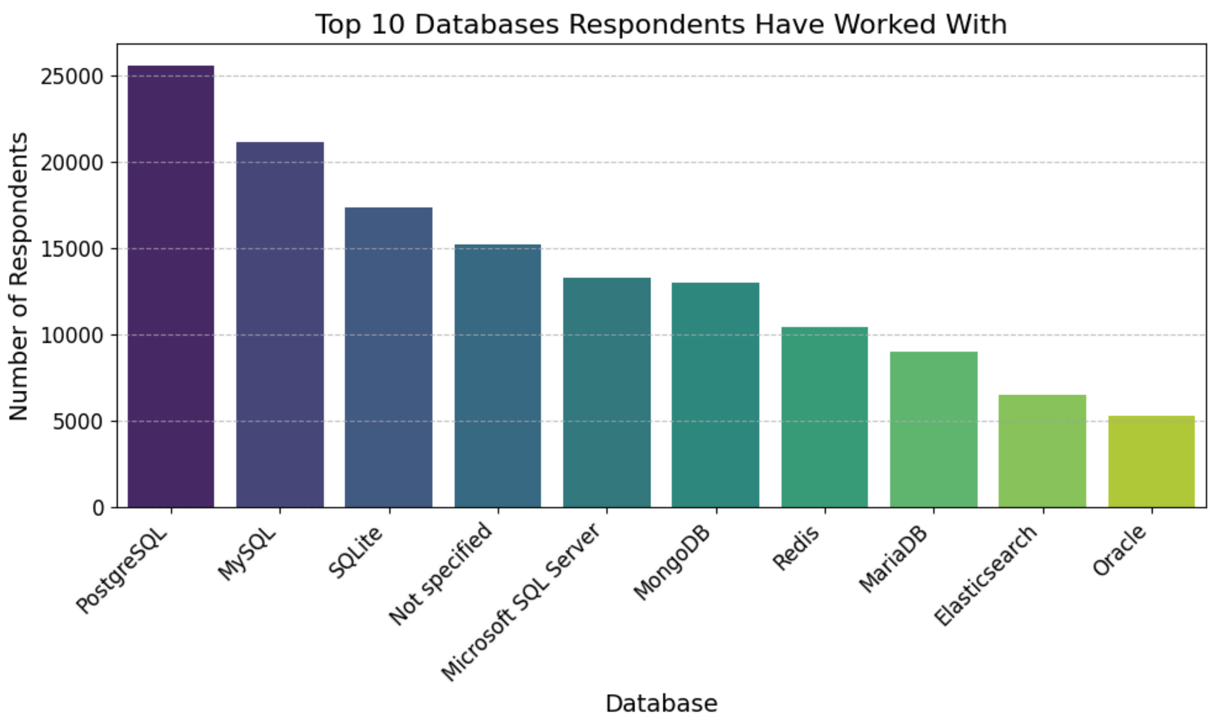
Out[50]:
```
DatabaseHaveWorkedWith
Not specified
15183
PostgreSQL
3216
Microsoft SQL Server
2239
MySQL
2099
SQLite
1762

...
Cosmos DB;Firebird;Microsoft Access;Microsoft SQL Server;MongoDB;MySQL;PostgreSQL;
SQLite        1
Cosmos DB;Elasticsearch;Microsoft SQL Server;MongoDB;MySQL;PostgreSQL;SQLite
1
Elasticsearch;InfluxDB;MySQL;PostgreSQL;SQLite
1
Elasticsearch;InfluxDB;MySQL;PostgreSQL
1
Couch DB;H2;Microsoft SQL Server;MySQL;Oracle;PostgreSQL;SQLite
1
Name: count, Length: 9051, dtype: int64
```

In [51]:
```
# Split and Explode
dbsworkedwith = df['DatabaseHaveWorkedWith'].str.split(';', expand=False)
known_dbs = dbsworkedwith.explode()

# top 10 Datanbases
top_known_dbs = known_dbs.value_counts().head(10)

print('Top 10 Databases Respondents Have Worked With')
print(top_known_dbs)
```

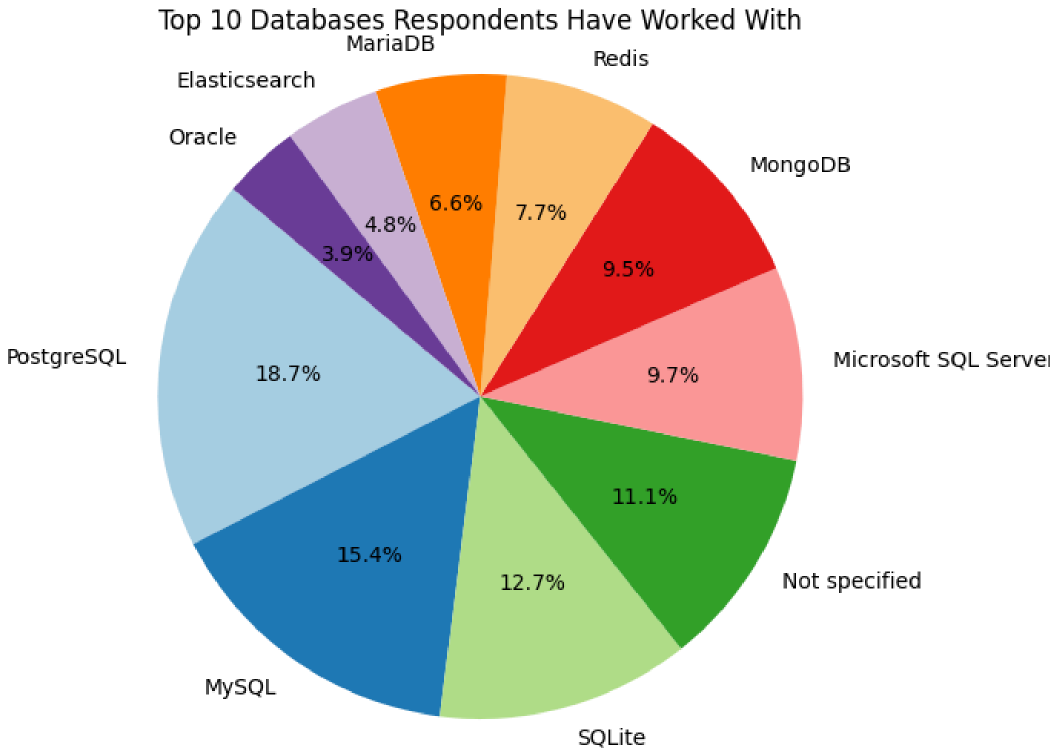
```
Top 10 Databases Respondents Have Worked With
DatabaseHaveWorkedWith
PostgreSQL              25536
MySQL                   21099
SQLite                  17365
Not specified           15183
Microsoft SQL Server    13275
MongoDB                 13007
Redis                   10463
MariaDB                  8991
Elasticsearch            6533
Oracle                   5273
Name: count, dtype: int64
```

In [52]:
```
# Plot Column Chart
```

```python
plt.figure(figsize=(10, 6))
sns.barplot(x=top_known_dbs.index, y=top_known_dbs.values, palette='viridis', hue=t

plt.title('Top 10 Databases Respondents Have Worked With', fontsize=16)
plt.xlabel('Database', fontsize=14)
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)   # Rotate x-axis labels for reada
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)   # Add a grid for easier reading of
plt.tight_layout()    # Adjusts plot to prevent labels from overlapping
plt.show()
```



In [53]:
```python
# Plot Pie Chart

plt.figure(figsize=(10, 6))
plt.pie(top_known_dbs, labels=top_known_dbs.index, autopct='%1.1f%%', startangle=14
plt.title('Top 10 Databases Respondents Have Worked With')
plt.axis('equal')  # Equal aspect ratio ensures pie is drawn as a circle.
plt.show()
```



Obs:

`PostgreSQL` is most used, followed by `MySQL` and `SQLite`.

- Open-source databases are preferred in practice.
- `PostgreSQL`'s dominance suggests strong community and enterprise trust.

Visualize the different platforms that respondents have worked with.

`PlatformHaveWorkedWith`

In [54]:
```
df['PlatformHaveWorkedWith'].value_counts()
```

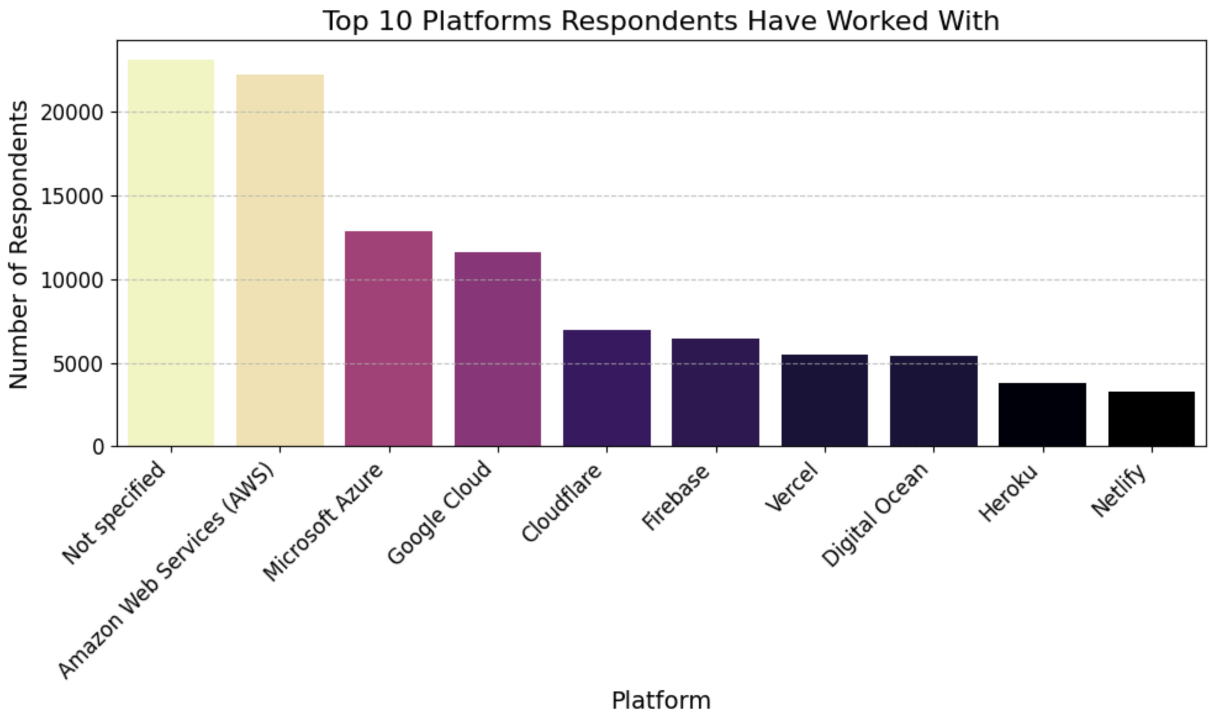
Out[54]:
```
PlatformHaveWorkedWith
Not specified
23071
Amazon Web Services (AWS)
6606
Microsoft Azure
4084
Google Cloud
1812
Amazon Web Services (AWS);Microsoft Azure
1521

...
Alibaba Cloud;Amazon Web Services (AWS);Google Cloud;Microsoft Azure;OpenStack;VMw
are        1
Amazon Web Services (AWS);Cloudflare;Hetzner;Microsoft Azure;OpenShift;OVH;Vercel
1
Cloudflare;Digital Ocean;IBM Cloud Or Watson;Oracle Cloud Infrastructure (OCI)
1
Google Cloud;Heroku;IBM Cloud Or Watson
1
Amazon Web Services (AWS);Cloudflare;Firebase;Linode, now Akamai;Vercel
1
Name: count, Length: 5468, dtype: int64
```

In [55]:
```python
# split and explode
platformworkedwith = df['PlatformHaveWorkedWith'].str.split(';', expand=False)
known_platforms = platformworkedwith.explode()

# top 10 platorms
top_known_platforms = known_platforms.value_counts().head(10)

print('Top 10 Platforms Respondents Have Worked With : ')
print(top_known_platforms)
```

```
Top 10 Platforms Respondents Have Worked With :
PlatformHaveWorkedWith
Not specified                23071
Amazon Web Services (AWS)    22191
Microsoft Azure              12850
Google Cloud                 11605
Cloudflare                    6974
Firebase                      6443
Vercel                        5491
Digital Ocean                 5409
Heroku                        3798
Netlify                       3238
Name: count, dtype: int64
```

In [56]:
```python
# Plot Bar Chart

plt.figure(figsize=(10, 6))
sns.barplot(x=top_known_platforms.index, y=top_known_platforms.values, palette='mag

plt.title('Top 10 Platforms Respondents Have Worked With', fontsize=16)
plt.xlabel('Platform', fontsize=14)
```

```
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12) # Rotate x-axis labels for readabi
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7) # Add a grid for easier reading of va
plt.tight_layout() # Adjusts plot to prevent labels from overlapping
plt.show()
```



Obs:

`AWS` leads, followed by `Azure` and `Google Cloud`.

- Cloud platforms are standard in modern development.
- Cloud certification and skills in AWS/Azure are career boosters.

Display the top 10 web frameworks respondents have used.

`WebframeHaveWorkedWith`

In [57]:
```
df['WebframeHaveWorkedWith'].value_counts()
```

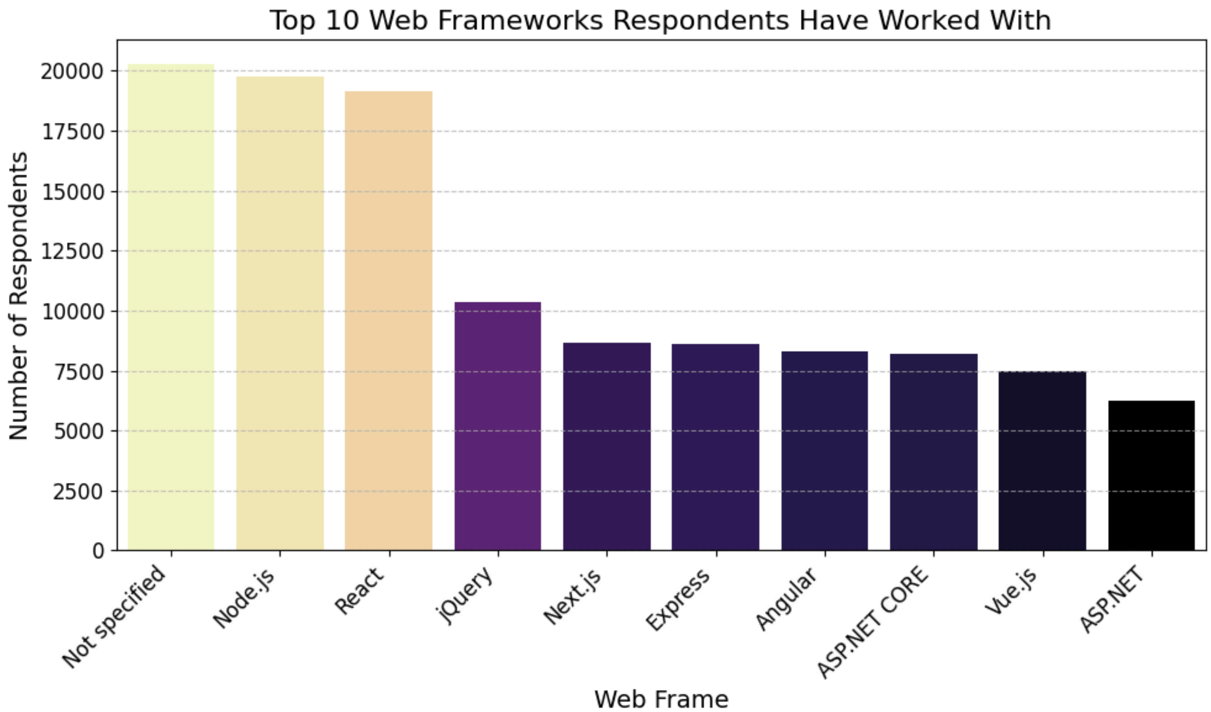
Out[57]:
```
WebframeHaveWorkedWith
Not specified
20276
React
1284
Spring Boot
1083
Node.js
907
Node.js;React
752

...
Angular;AngularJS;Htmx;Spring Boot
1
Astro;Fastify;React;Spring Boot
1
Angular;Django;jQuery;Ruby on Rails;Vue.js;WordPress
1
ASP.NET;ASP.NET CORE;Astro;Blazor;CodeIgniter;Deno;Django;Drupal;Elm;Express;FastA
PI;Fastify;Flask;Gatsby;Htmx;jQuery;Laravel;NestJS;Next.js;Node.js;Nuxt.js;Phoeni
x;Play Framework;React;Remix;Ruby on Rails;Solid.js;Spring Boot;Strapi;Svelte;Symf
ony;Vue.js;WordPress;Yii 2        1
Angular;AngularJS;ASP.NET;ASP.NET CORE;Drupal;jQuery;Next.js;Node.js;React;Spring
Boot
1
Name: count, Length: 12236, dtype: int64
```

In [58]:
```
# split and explode
WebframeHaveWorkedWith = df['WebframeHaveWorkedWith'].str.split(';', expand=False)
known_wfs = WebframeHaveWorkedWith.explode()

# Top 10 Web Frameworks
top_known_wfs = known_wfs.value_counts().head(10)

print('Top 10 Web Frameworks Respondents Have Worked With')
print(top_known_wfs)
```

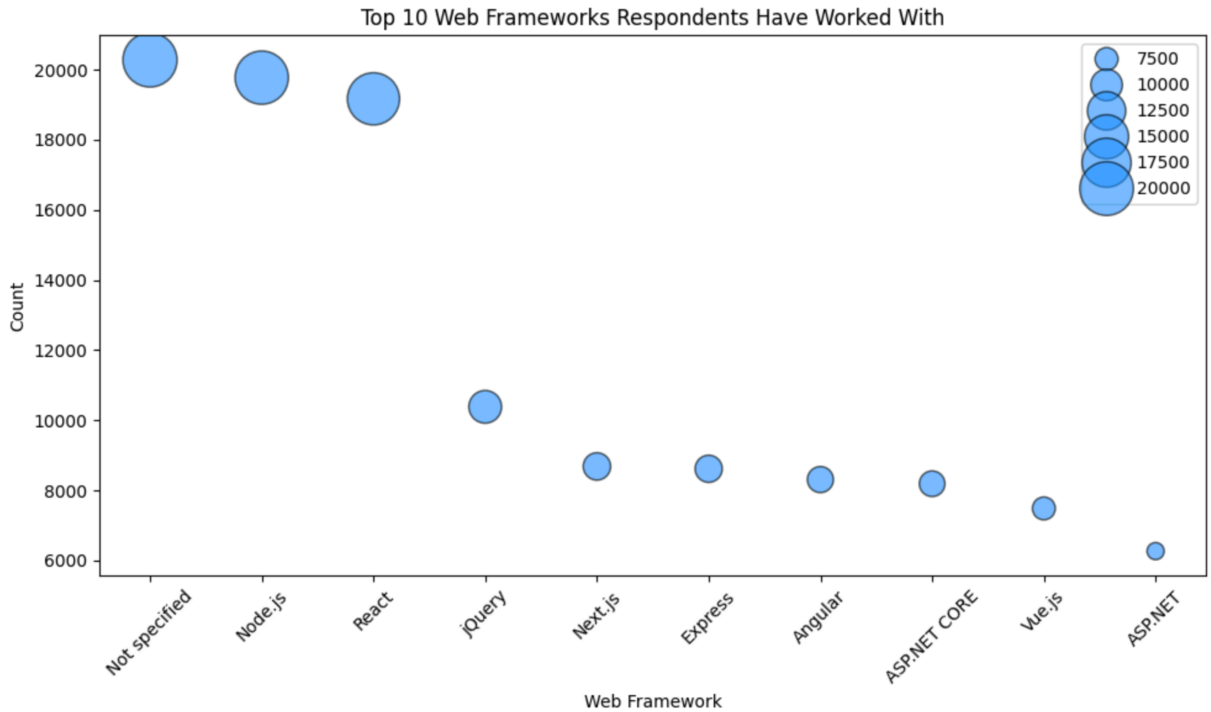
```
Top 10 Web Frameworks Respondents Have Worked With
WebframeHaveWorkedWith
Not specified    20276
Node.js          19772
React            19167
jQuery           10381
Next.js           8681
Express           8614
Angular           8306
ASP.NET CORE      8187
Vue.js            7483
ASP.NET           6265
Name: count, dtype: int64
```

In [59]:
```
# Plot Bar Chart

plt.figure(figsize=(10, 6))
sns.barplot(x=top_known_wfs.index, y=top_known_wfs.values, palette='magma', hue=top
```

```python
plt.title('Top 10 Web Frameworks Respondents Have Worked With', fontsize=16)
plt.xlabel('Web Frame', fontsize=14)
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12) # Rotate x-axis labels for readabi
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7) # Add a grid for easier reading of va
plt.tight_layout() # Adjusts plot to prevent labels from overlapping
plt.show()
```



In [60]:
```python
# Plot Scatter Bubble Plot

plt.figure(figsize=(10, 6))
sns.scatterplot(x=top_known_wfs.index, y=top_known_wfs.values, size=top_known_wfs.v

plt.title("Top 10 Web Frameworks Respondents Have Worked With")
plt.xlabel("Web Framework")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Obs:

`Node.js`, `React`, `jQuery`, `Next.js` are top choices.

- Full-stack `JavaScript` frameworks are in high practical use.
- `Node.js` and `React` continue to dominate hiring needs.

Future Technology Trends

Display the top 10 programming languages respondents want to learn next year.

`LanguageWantToWorkWith`

In [61]:
```
df['LanguageWantToWorkWith'].value_counts()
```

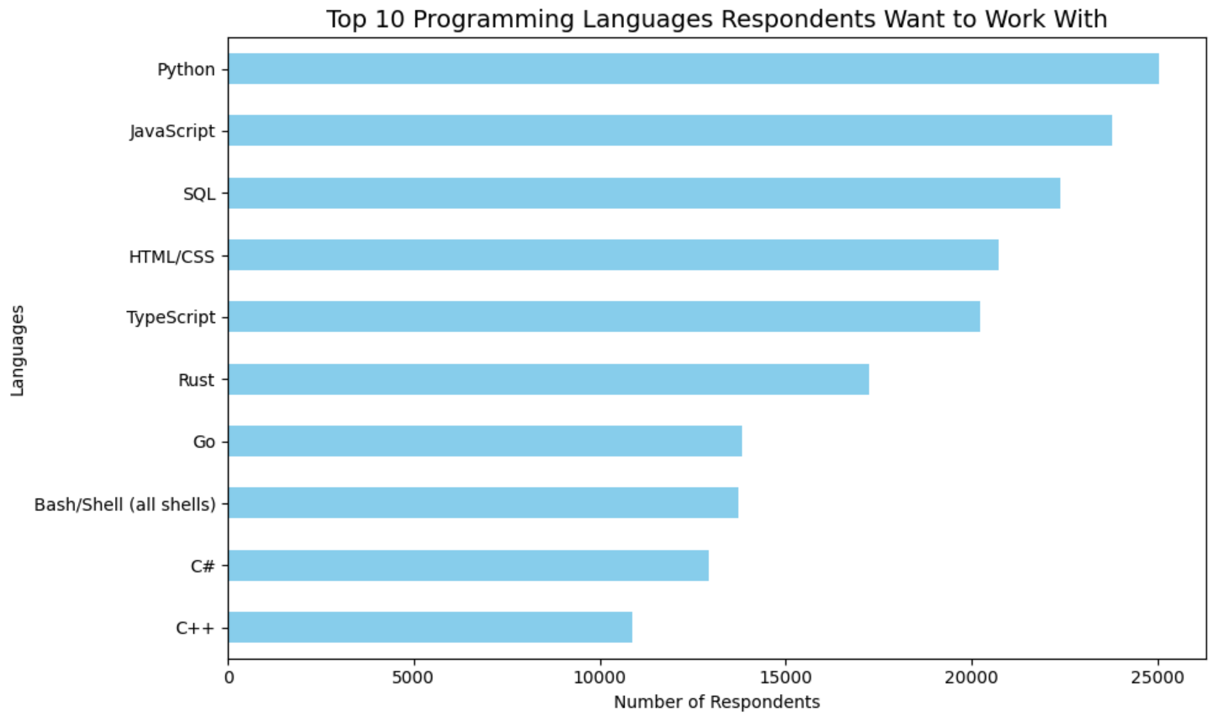
Out[61]:
```
LanguageWantToWorkWith
Not specified
9685
Python
922
Rust
737
HTML/CSS;JavaScript;TypeScript
632
C#
538

...
Bash/Shell (all shells);Rust;Scala;SQL
1
Dart;JavaScript;Kotlin;Python;Scala;SQL
1
Bash/Shell (all shells);Go;Groovy;Haskell;Python
1
Bash/Shell (all shells);C#;C++;Lua;PowerShell;Python;R;Rust;SQL;TypeScript
1
Bash/Shell (all shells);C#;Go;HTML/CSS;Java;JavaScript;Kotlin;Objective-C;Python;R
ust;SQL;Swift;TypeScript       1
Name: count, Length: 22770, dtype: int64
```

In [62]:
```python
# Split and explode
LanguageWantToWorkWith = df['LanguageWantToWorkWith'].str.split(';', expand=False)
wanted_langs = LanguageWantToWorkWith.explode()

# Top 10 Languages
top_wanted_langs = wanted_langs.value_counts().head(10)

print('Top 10 Programming Languages Respondents Want to Work With')
print(top_wanted_langs)
```

```
Top 10 Programming Languages Respondents Want to Work With
LanguageWantToWorkWith
Python                      25047
JavaScript                  23774
SQL                         22400
HTML/CSS                    20721
TypeScript                  20239
Rust                        17232
Go                          13837
Bash/Shell (all shells)     13744
C#                          12921
C++                         10873
Name: count, dtype: int64
```

In [63]:
```python
# Plot Horizontal Bar Chart

plt.figure(figsize=(10, 6))
top_wanted_langs.sort_values().plot(kind='barh', color='skyblue')
plt.title('Top 10 Programming Languages Respondents Want to Work With', fontsize=14
plt.xlabel('Number of Respondents')
plt.ylabel('Languages')
```

```python
plt.tight_layout()
plt.show()
```



Obs:

`Python`, `JavaScript`, `SQL`, `Rust`, and `Go` are most desired.

- Developers aim to deepen backend, data, and systems skills.
- Learning Rust and Go can be a differentiator in system-level roles.

Show the top 10 databases respondents want to learn next year

`DatabaseWantToWorkWith`

In [64]:
```python
df['DatabaseWantToWorkWith'].value_counts()
```

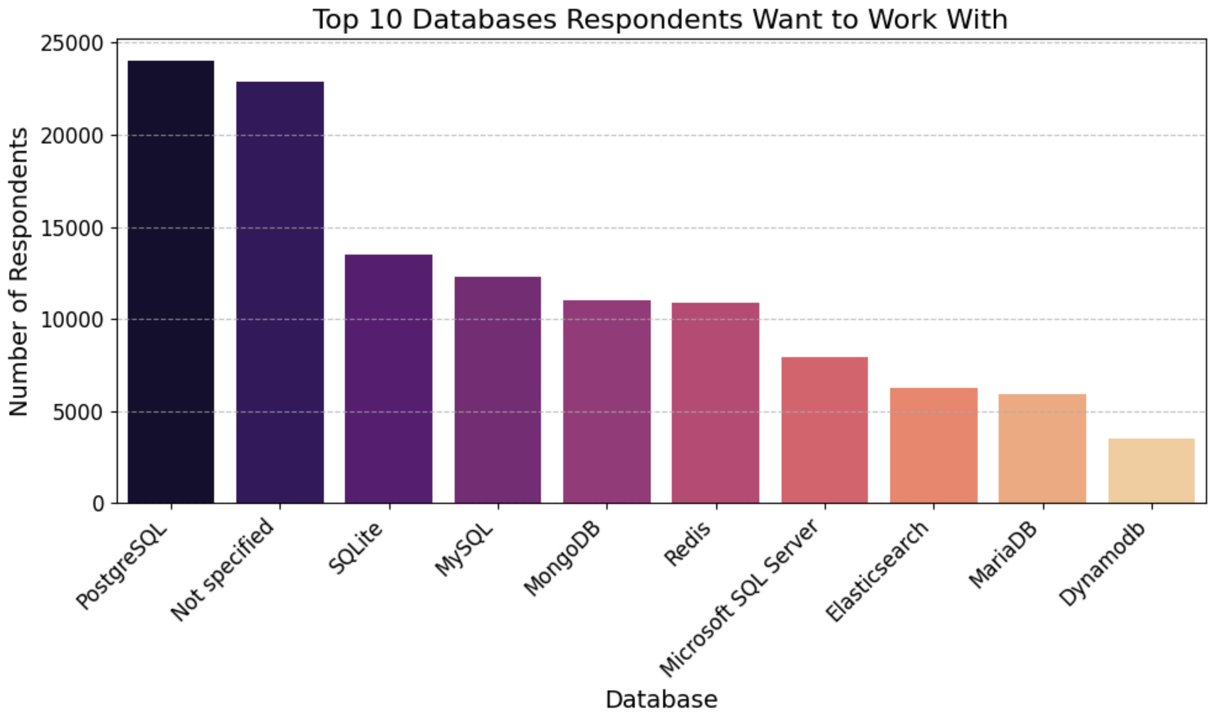
Out[64]:
```
DatabaseWantToWorkWith
Not specified
22879
PostgreSQL
3738
PostgreSQL;SQLite
1533
SQLite
1476
Microsoft SQL Server
1431

...
IBM DB2;Microsoft SQL Server;MongoDB;MySQL;Oracle;PostgreSQL
1
Cassandra;Cockroachdb;Elasticsearch;H2;InfluxDB;Neo4J;Snowflake
1
Clickhouse;DuckDB;Elasticsearch;InfluxDB;Neo4J;PostgreSQL
1
Clickhouse;Elasticsearch;MariaDB;MySQL;PostgreSQL;Redis;SQLite;TiDB
1
BigQuery;Cassandra;Databricks SQL;DuckDB;Elasticsearch;Firebase Realtime Database;
Microsoft SQL Server;MongoDB;MySQL;PostgreSQL;Redis;Snowflake;SQLite;Supabase
1
Name: count, Length: 8479, dtype: int64
```

In [65]:
```python
# split and explode
DatabaseWantToWorkWith = df['DatabaseWantToWorkWith'].str.split(';', expand=False)
wanted_dbs = DatabaseWantToWorkWith.explode()

# Top 10 Databases
top_wanted_dbs = wanted_dbs.value_counts().head(10)

print('Top 10 Databases Respondents Want to Work With')
print(top_wanted_dbs)
```

```
Top 10 Databases Respondents Want to Work With
DatabaseWantToWorkWith
PostgreSQL              24005
Not specified           22879
SQLite                  13489
MySQL                   12269
MongoDB                 10982
Redis                   10847
Microsoft SQL Server     7905
Elasticsearch            6246
MariaDB                  5947
Dynamodb                 3503
Name: count, dtype: int64
```

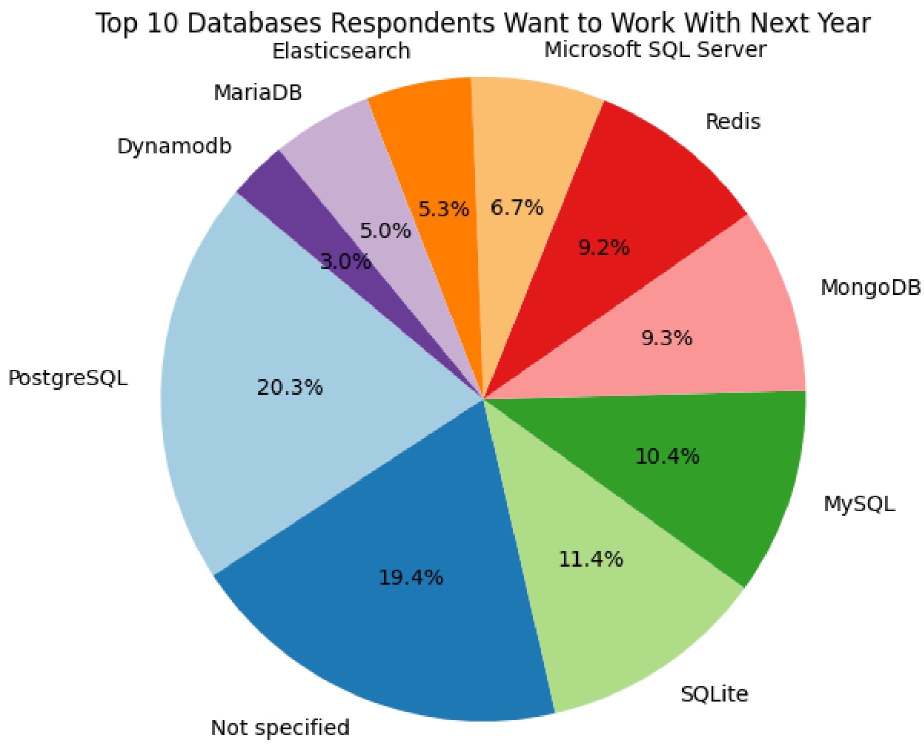
In [66]:
```python
# Plot Column Chart

plt.figure(figsize=(10, 6))
sns.barplot(x=top_wanted_dbs.index, y=top_wanted_dbs.values, palette='magma', hue=t

plt.title('Top 10 Databases Respondents Want to Work With', fontsize=16)
```

```
plt.xlabel('Database', fontsize=14)
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)   # Rotate x-axis labels for reada
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)   # Add a grid for easier reading of
plt.tight_layout()    # Adjusts plot to prevent labels from overlapping
plt.show()
```



In [67]:

```
# Plot Pie Chart

plt.figure(figsize=(10, 6))
plt.pie(top_wanted_dbs, labels=top_wanted_dbs.index, autopct='%1.1f%%', startangle=
plt.title('Top 10 Databases Respondents Want to Work With Next Year')
plt.axis('equal')  # Equal aspect ratio ensures pie is drawn as a circle.
plt.show()
```



Obs:

`PostgreSQL` , `SQLite` , `MongoDB` , `Redis` are highly sought.

- Preference for versatile and scalable databases.
- Growth in `NoSQL` and real-time systems expected.

Visualize the platforms respondents want to work with next year

`PlatformWantToWorkWith`

In [68]:
```
df['PlatformWantToWorkWith'].value_counts()
```

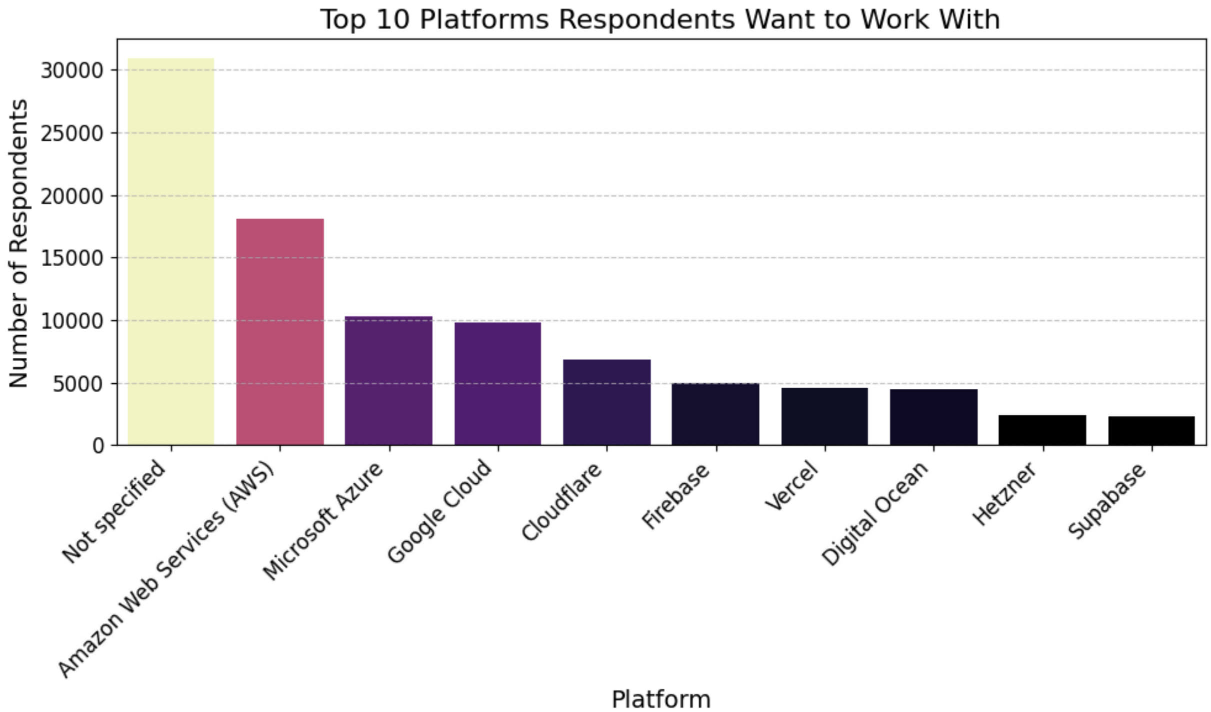
Out[68]:
```
PlatformWantToWorkWith
Not specified
30905
Amazon Web Services (AWS)
4859
Microsoft Azure
2782
Google Cloud
1377
Amazon Web Services (AWS);Microsoft Azure
1320

...
Amazon Web Services (AWS);Cloudflare;Firebase;Google Cloud;Linode, now Akamai;Micr
osoft Azure;Netlify;Vercel        1
Amazon Web Services (AWS);Cloudflare;Firebase;Google Cloud;Hetzner;OpenStack;OVH;V
ercel                             1
Amazon Web Services (AWS);Cloudflare;Databricks;Hetzner;Microsoft Azure;Vercel
1
Amazon Web Services (AWS);Digital Ocean;Google Cloud;Microsoft Azure;Netlify;Supab
ase;Vercel                        1
PythonAnywhere;Render;Vercel
1
Name: count, Length: 4785, dtype: int64
```

In [69]:
```python
# split and explode
PlatformWantToWorkWith = df['PlatformWantToWorkWith'].str.split(';', expand=False)
wanted_platforms = PlatformWantToWorkWith.explode()

# Top 10 Web Frameworks
top_wanted_platforms = wanted_platforms.value_counts().head(10)

print('Top 10 Platforms Respondents Want to Work With')
print(top_known_platforms)
```

```
Top 10 Platforms Respondents Want to Work With
PlatformHaveWorkedWith
Not specified                23071
Amazon Web Services (AWS)    22191
Microsoft Azure              12850
Google Cloud                 11605
Cloudflare                    6974
Firebase                      6443
Vercel                        5491
Digital Ocean                 5409
Heroku                        3798
Netlify                       3238
Name: count, dtype: int64
```

In [70]:
```python
# Plot Bar Chart

plt.figure(figsize=(10, 6))
sns.barplot(x=top_wanted_platforms.index, y=top_wanted_platforms.values, palette='m

plt.title('Top 10 Platforms Respondents Want to Work With', fontsize=16)
plt.xlabel('Platform', fontsize=14)
```

```python
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12) # Rotate x-axis labels for readabi
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7) # Add a grid for easier reading of va
plt.tight_layout() # Adjusts plot to prevent labels from overlapping
plt.show()
```



Obs:

`AWS`, `Azure`, and `GCP` top choices again.

- Developer demand aligns with enterprise cloud use.
- `Cloud` fluency is a long-term requirement.

Display the top 10 web frameworks respondents want to learn next year

`WebframeWantToWorkWith`

In [71]:
```python
df['WebframeWantToWorkWith'].value_counts()
```

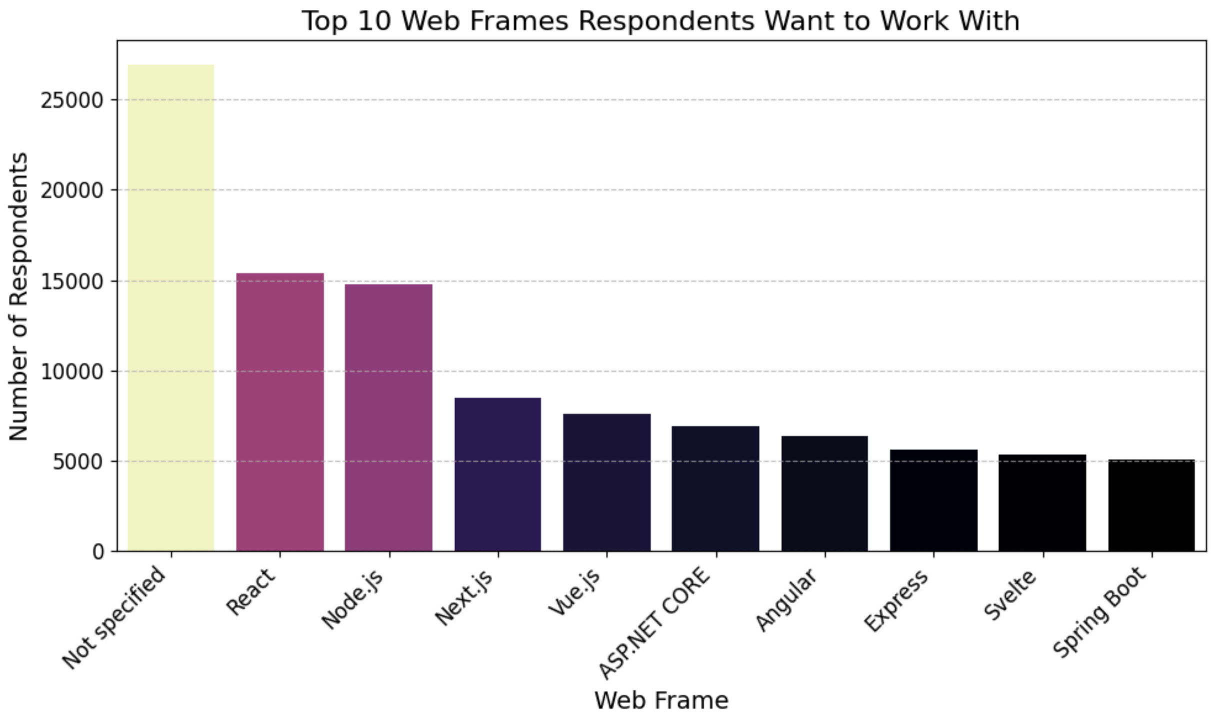
Out[71]:
```
WebframeWantToWorkWith
Not specified
26902
React
997
Spring Boot
950
Node.js
619
ASP.NET CORE
607

...
Angular;AngularJS;ASP.NET;Blazor;Node.js;React;Svelte
1
Express;React;Spring Boot;Strapi
1
Django;Express;Laravel;React
1
Deno;Fastify;Svelte
1
Django;Express;Laravel;NestJS;Next.js;Node.js;Nuxt.js;React;Spring Boot;Symfony;Vu
e.js;WordPress        1
Name: count, Length: 11655, dtype: int64
```

In [72]:
```python
# split and explode
WebframeWantToWorkWith = df['WebframeWantToWorkWith'].str.split(';', expand=False)
wanted_wfs = WebframeWantToWorkWith.explode()

# Top 10 Web Frameworks
top_wanted_wfs = wanted_wfs.value_counts().head(10)

print('Top 10 Web Frames Respondents Want to Work With')
print(top_wanted_wfs)
```

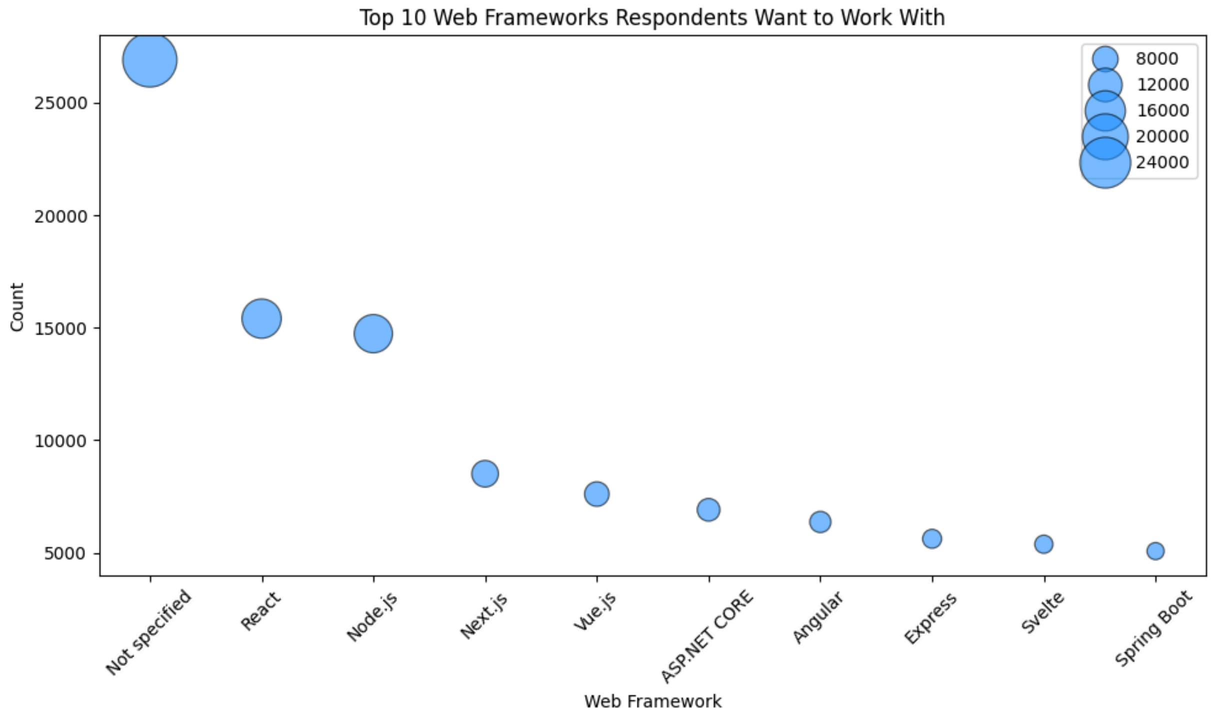
```
Top 10 Web Frames Respondents Want to Work With
WebframeWantToWorkWith
Not specified    26902
React            15404
Node.js          14735
Next.js           8507
Vue.js            7604
ASP.NET CORE      6905
Angular           6364
Express           5616
Svelte            5374
Spring Boot       5068
Name: count, dtype: int64
```

In [73]:
```python
# Plot Bar Chart

plt.figure(figsize=(10, 6))
sns.barplot(x=top_wanted_wfs.index, y=top_wanted_wfs.values, palette='magma', hue=t

plt.title('Top 10 Web Frames Respondents Want to Work With', fontsize=16)
plt.xlabel('Web Frame', fontsize=14)
```

```
plt.ylabel('Number of Respondents', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12) # Rotate x-axis labels for readabi
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7) # Add a grid for easier reading of va
plt.tight_layout() # Adjusts plot to prevent labels from overlapping
plt.show()
```



In [74]:
```
# Plot Scatter Bubble Plot

plt.figure(figsize=(10, 6))
sns.scatterplot(x=top_wanted_wfs.index, y=top_wanted_wfs.values, size=top_wanted_wf

plt.title("Top 10 Web Frameworks Respondents Want to Work With")
plt.xlabel("Web Framework")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Obs:

`React`, `Node.js`, `Next.js`, `Vue.js`, `ASP.NET Core` are in demand.

- Strong interest in modern `JavaScript` ecosystems.
- Frameworks like `React` and `Next.js` should be prioritized in upskilling.

Demographics

Display respondents by `Age` group.

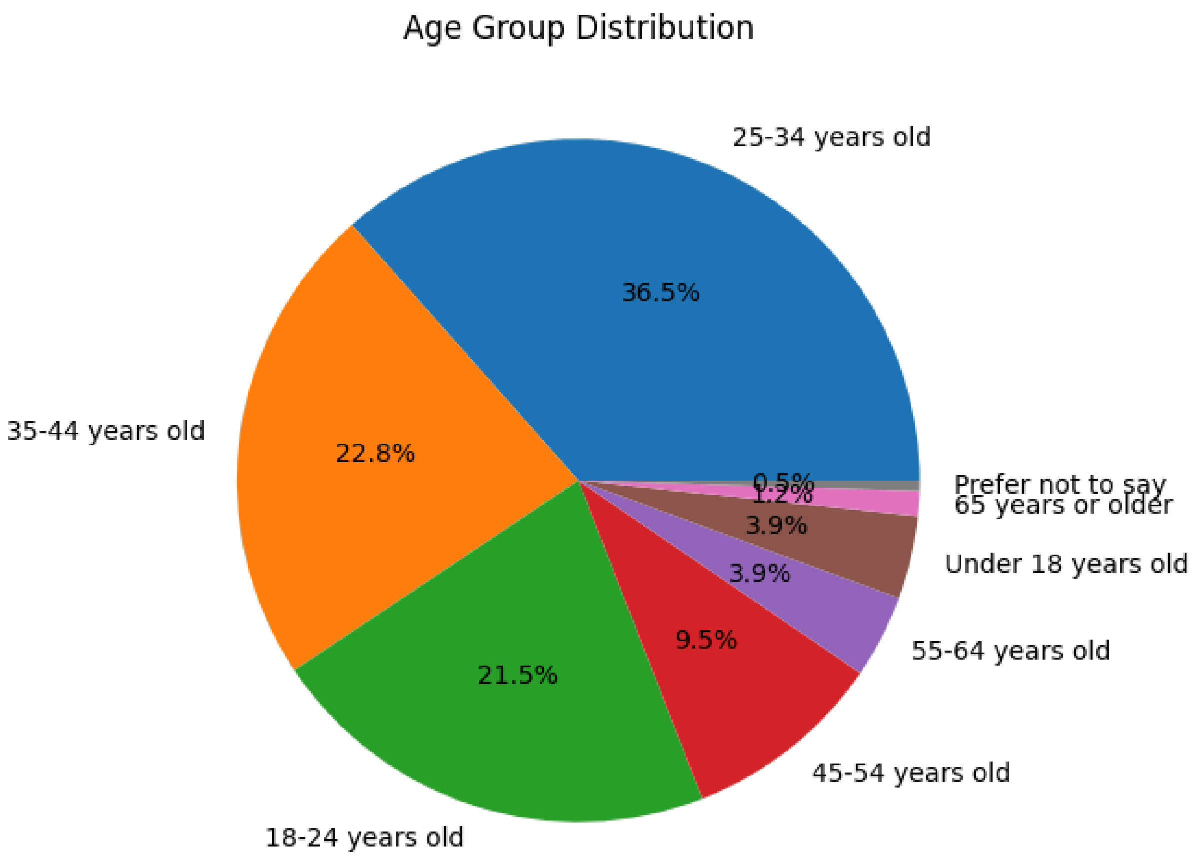
In [75]:
```
print("Respondents by Age Group; ")
df['Age'].value_counts()
```

```
Respondents by Age Group;
```

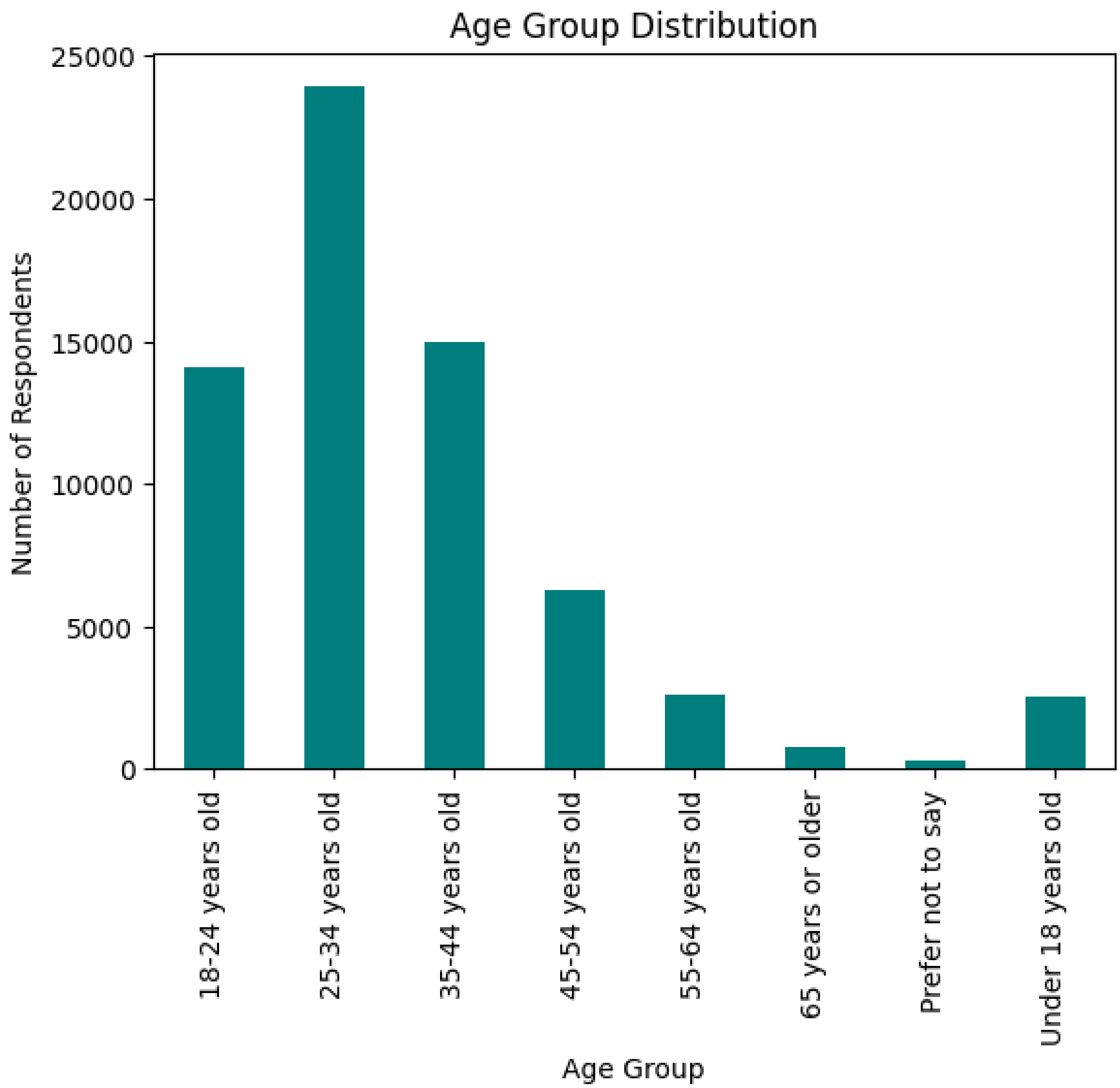
Out[75]:
```
Age
25-34 years old       23911
35-44 years old       14942
18-24 years old       14098
45-54 years old        6249
55-64 years old        2575
Under 18 years old     2568
65 years or older       772
Prefer not to say       322
Name: count, dtype: int64
```

In [76]:
```
# Pie Chart

df['Age'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(10, 6), title='Age Gr
```

```
plt.ylabel('')
plt.show()
```

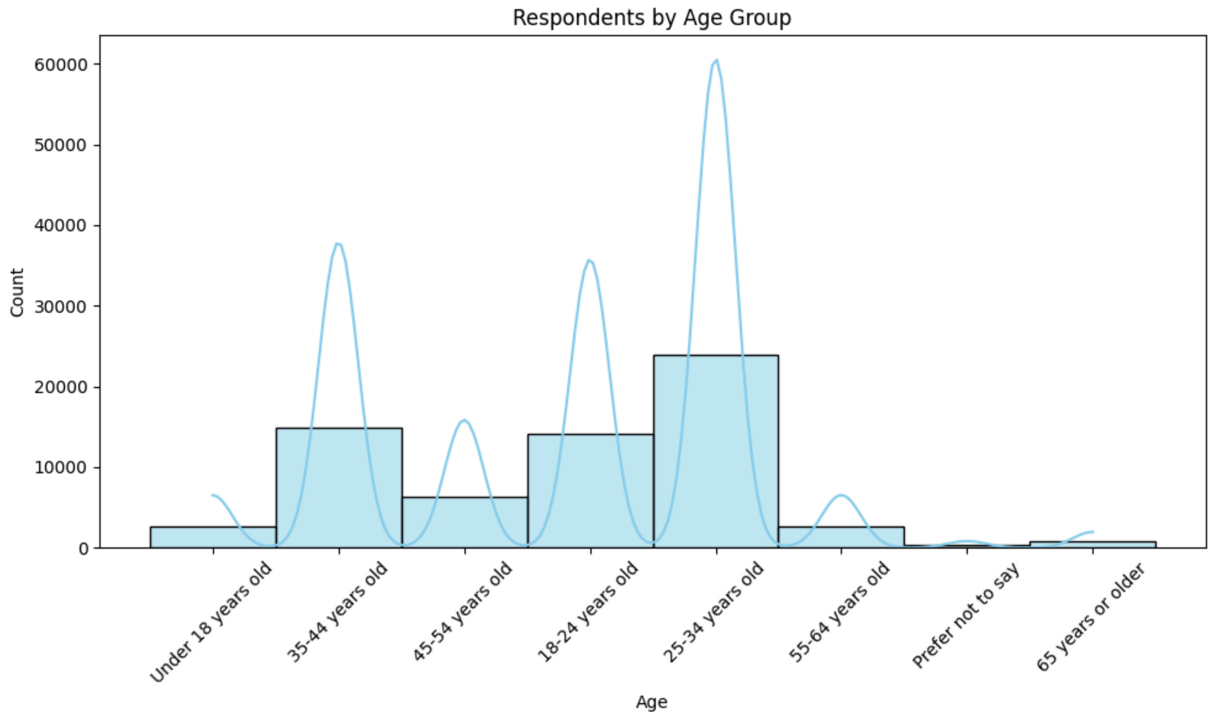


In [77]:
```
# Bar Chart

df['Age'].value_counts().sort_index().plot(kind='bar', color='teal')
plt.title('Age Group Distribution')
plt.xlabel('Age Group')
plt.ylabel('Number of Respondents')
plt.show()
```



In [78]:

```
# Plot Histogram

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', bins=20, kde=True, color='skyblue')
plt.title('Respondents by Age Group')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Obs:

Majority are `25–34` (23.9K), `35-44` (14.9k) follwed closely by `18–24` (14K) years old.

- The survey reflects a younger, early-career audience as well as experienced professionals.
- Tech industry is heavily youth-driven with fast-learning cohorts.

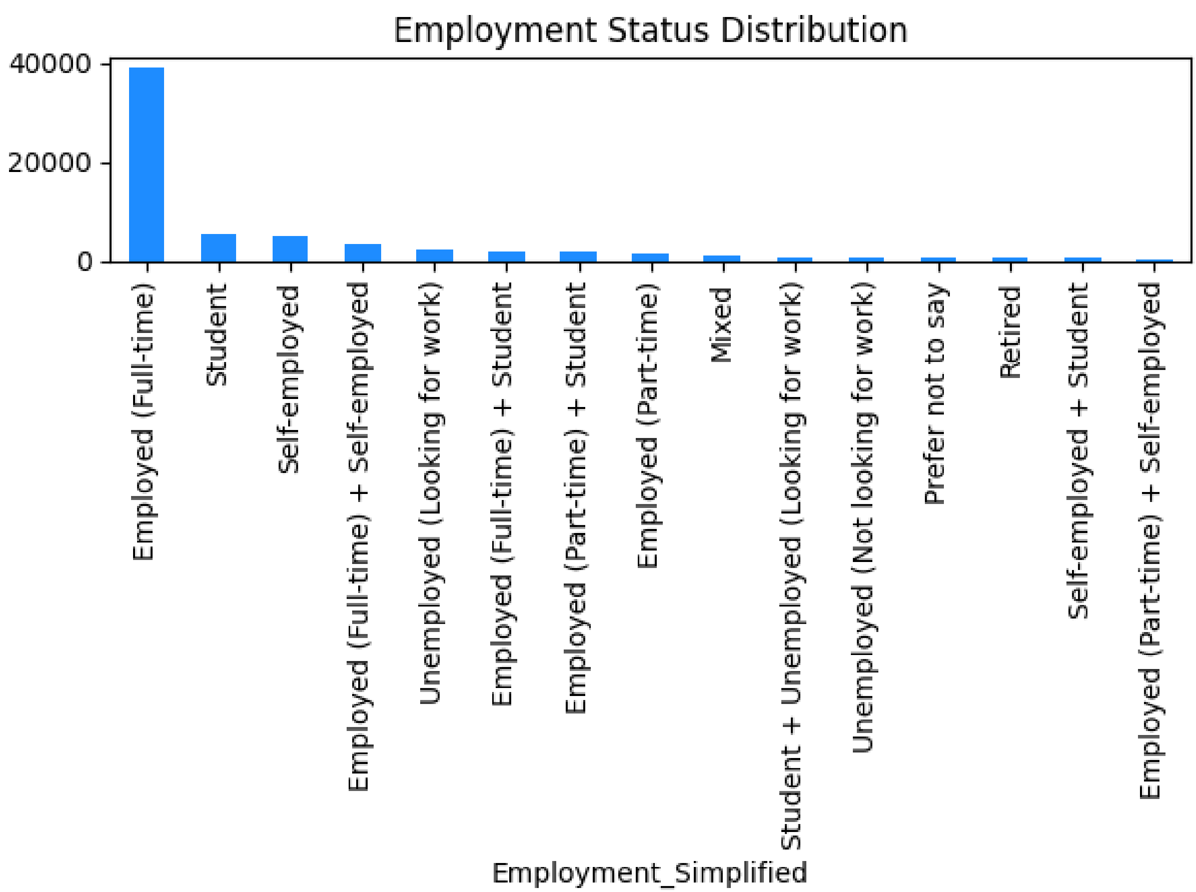
`Employment` **Status Distribution.**

In [79]:
```
print('Top Employment Status Distribution: ')
df['Employment_Simplified'].value_counts().head(15)
```

```
Top Employment Status Distribution:
```

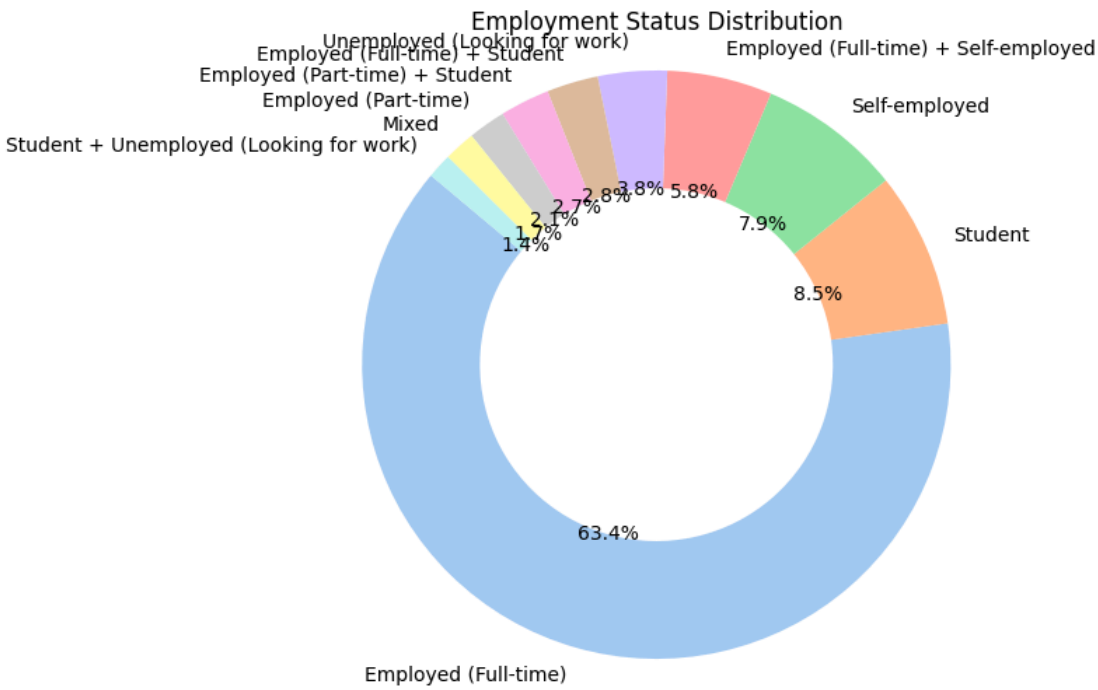
Out[79]:
```
Employment_Simplified
Employed (Full-time)                            39041
Student                                          5254
Self-employed                                    4846
Employed (Full-time) + Self-employed             3557
Unemployed (Looking for work)                    2341
Employed (Full-time) + Student                   1738
Employed (Part-time) + Student                   1680
Employed (Part-time)                             1266
Mixed                                            1059
Student + Unemployed (Looking for work)           839
Unemployed (Not looking for work)                 633
Prefer not to say                                 546
Retired                                           525
Self-employed + Student                           517
Employed (Part-time) + Self-employed              401
Name: count, dtype: int64
```

In [80]:
```python
# Plot Bar Chart fir top Emoployment types

df['Employment_Simplified'].value_counts().head(15).plot(kind='bar', color='dodgerb
plt.title('Employment Status Distribution')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



In [81]:
```python
emp_counts = df['Employment_Simplified'].value_counts().head(10)
```

```python
plt.figure(figsize=(10, 6))
plt.pie(emp_counts, labels=emp_counts.index, autopct='%1.1f%%', startangle=140, col
plt.title('Employment Status Distribution')
plt.axis('equal')
plt.show()
```



Obs:

Most are `Employed Full-time` (39K), followed by `Students` and `Self-employed`.

- Majority of respondents are actively employed professionals.
- Employers can target this group for upskilling or product adoption.

Show the number of respondents from various `Countries`.

In [82]:

```python
df['Country'].value_counts()
```

Out[82]:

```
Country
USA                  11095
Not Specified         6550
Germany               4947
India                 4231
United Kingdom        3224
                     ...
Niger                    1
Guinea                   1
Dominica                 1
Papua New Guinea         1
Solomon Islands          1
Name: count, Length: 183, dtype: int64
```

In [83]:
```
print('Top Countries with Max Respondents: ')
df['Country'].value_counts().head(40)
```

```
Top Countries with Max Respondents:
```

Out[83]:
```
Country
USA               11095
Not Specified      6550
Germany            4947
India              4231
United Kingdom     3224
Ukraine            2672
France             2110
Canada             2104
Poland             1534
Netherlands        1449
Brazil             1375
Italy              1341
Australia          1260
Spain              1123
Sweden             1016
Russia              925
Switzerland         876
Austria             791
Czech Republic      714
Israel              604
Turkey              546
Belgium             526
Denmark             504
Portugal            470
Norway              450
Romania             439
Pakistan            415
Iran                411
China               406
Mexico              402
New Zealand         396
Hungary             396
Greece              389
Finland             386
South Africa        358
Indonesia           354
Argentina           345
Bangladesh          327
Bulgaria            319
Nigeria             305
Name: count, dtype: int64
```

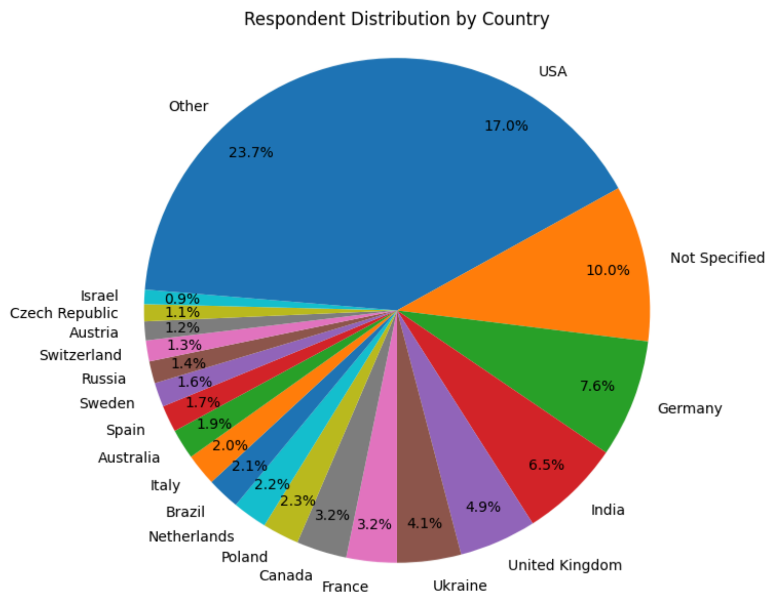
In [84]:
```
n = 20    # Number of top countries to display

# Get top 25 countries
top_countries = df['Country'].value_counts().head(n)

# Calculate sum of remaining countries
others = df['Country'].value_counts().iloc[n:].sum()
```

```python
# Conditionally concatenate 'Other' if there are more countries than N
if others > 0:     # Add 'Other' since there are actually countries to group
    country_data = pd.concat([top_countries, pd.Series({'Other': others})])
else:
    country_data = top_countries     # If all countries are within top n, 'Others'

# Ensure data is ready for plotting
labels = country_data.index
sizes = country_data.values

print("Data for Pie Chart:\n", country_data)
```

```
Data for Pie Chart:
 USA                11095
Not Specified      6550
Germany            4947
India              4231
United Kingdom     3224
Ukraine            2672
France             2110
Canada             2104
Poland             1534
Netherlands        1449
Brazil             1375
Italy              1341
Australia          1260
Spain              1123
Sweden             1016
Russia              925
Switzerland         876
Austria             791
Czech Republic      714
Israel              604
Other             15496
dtype: int64
```

In [85]:

```python
# Plot the pie chart

plt.figure(figsize=(12, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, pctdistance=0.85, c
plt.title('Respondent Distribution by Country')
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Obs:

`USA`, `Germany`, `India`, and `UK` have the highest respondent counts.

- Tech talent is globally distributed with high representation from both western and emerging markets.
- International collaboration and talent acquisition is more viable than ever.

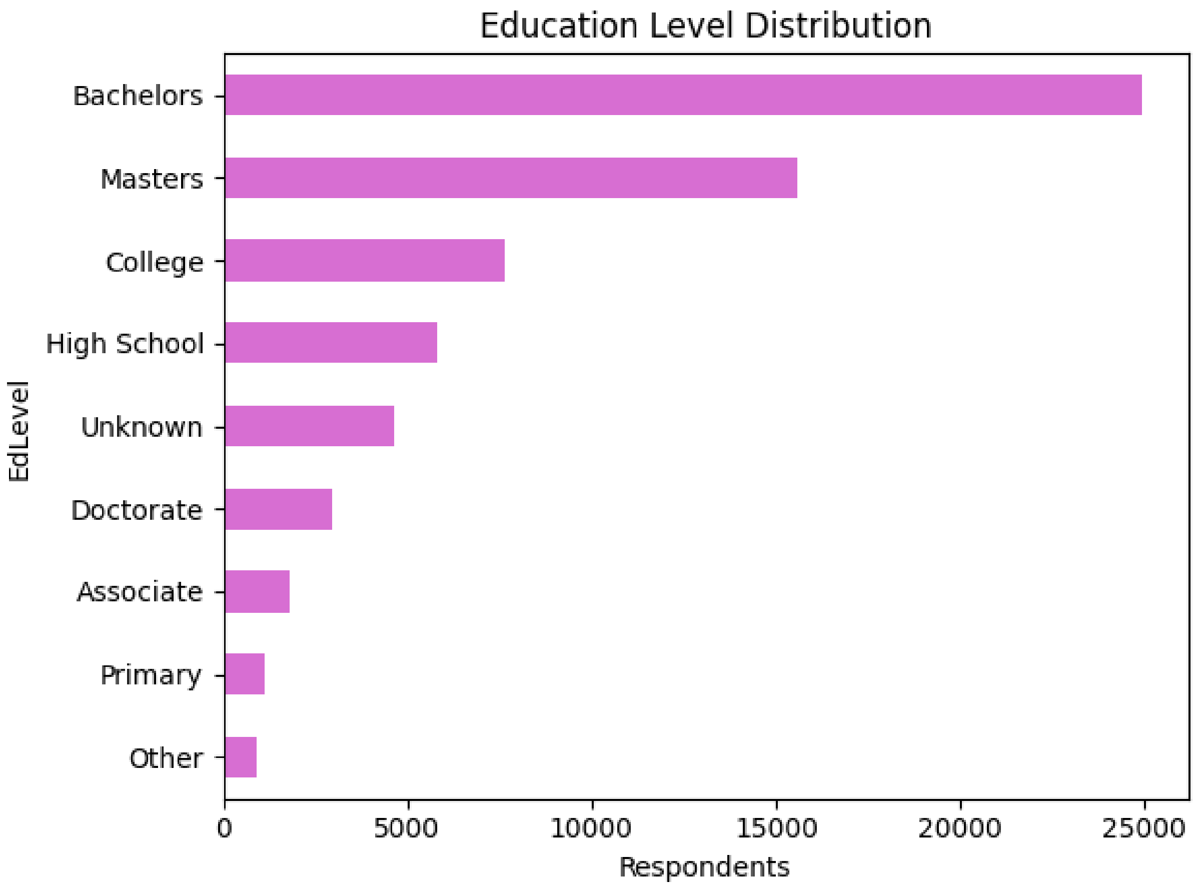
Visualize respondents classified by their highest `Education Level`.

In [86]:
```
print('Education Level Distribution of Respondents: ')
df['EdLevel'].value_counts()
```

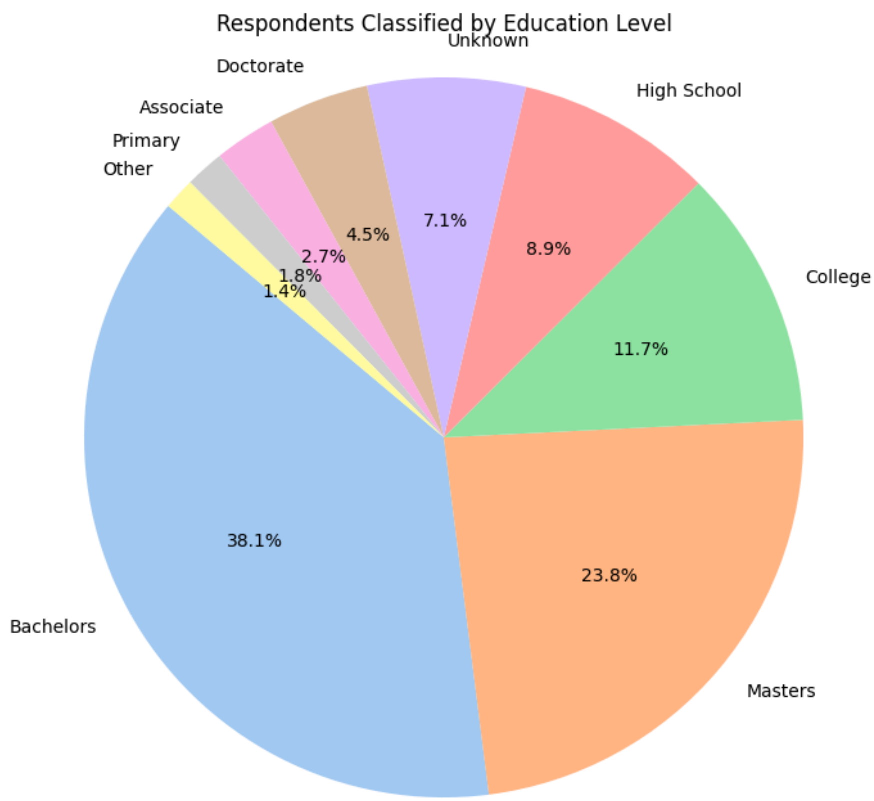
```
Education Level Distribution of Respondents:
```

Out[86]:
```
EdLevel
Bachelors      24942
Masters        15557
College         7651
High School     5793
Unknown         4653
Doctorate       2970
Associate       1793
Primary         1146
Other            932
Name: count, dtype: int64
```

In [87]:
```
# Plot Bar Chart

df['EdLevel'].value_counts().sort_values().plot(kind='barh', color='orchid')
plt.title('Education Level Distribution')
plt.xlabel('Respondents')
plt.tight_layout()
plt.show()
```



In [88]:
```
# Plot Pie Chart

edu_counts = df['EdLevel'].value_counts()

plt.figure(figsize=(12, 8))
plt.pie(edu_counts, labels=edu_counts.index, autopct='%1.1f%%', startangle=140, col
plt.title('Respondents Classified by Education Level')
plt.axis('equal')
plt.show()
```



Obs:

`Bachelor’s` (24.9K) and `Master’s` (15.5K) are most common.

- Majority hold advanced academic qualifications.
- Strong foundational knowledge base; skill programs can assume technical depth.

Further Analysis

Educational Background and Employment Type

Explore how educational background (EdLevel) relates to employment type (Employment). Use cross-tabulation and visualizations to understand if higher education correlates with specific employment types.

In [89]:
```
# Cross-tabulation
edu_emp_ct = pd.crosstab(df['EdLevel'], df['Employment_Simplified'], normalize='ind
edu_emp_ct = edu_emp_ct.round(1)
print(edu_emp_ct)
```

```
Employment_Simplified  Employed (Full-time)  Employed (Full-time) + Retired  \
EdLevel
Associate                              59.1                             0.0
Bachelors                              67.4                             0.0
College                                46.8                             0.0
Doctorate                              69.8                             0.1
High School                            25.2                             0.0
Masters                                70.8                             0.0
Other                                  40.5                             0.0
Primary                                14.0                             0.0
Unknown                                54.1                             0.0

Employment_Simplified  Employed (Full-time) + Self-employed  \
EdLevel
Associate                                               5.8
Bachelors                                               5.5
College                                                 6.4
Doctorate                                               5.4
High School                                             3.1
Masters                                                 6.2
Other                                                   4.4
Primary                                                 2.2
Unknown                                                 4.5

Employment_Simplified  Employed (Full-time) + Student  \
EdLevel
Associate                                         3.7
Bachelors                                         2.5
College                                           4.1
Doctorate                                         1.8
High School                                       2.8
Masters                                           2.1
Other                                             1.8
Primary                                           0.5
Unknown                                           3.5

Employment_Simplified  Employed (Full-time) + Unemployed (Looking for work)  \
EdLevel
Associate                                                            0.0
Bachelors                                                            0.1
College                                                              0.0
Doctorate                                                            0.0
High School                                                          0.1
Masters                                                              0.1
Other                                                                0.1
Primary                                                              0.0
Unknown                                                              0.1

Employment_Simplified  Employed (Full-time) + Unemployed (Not looking for work)  \
EdLevel
Associate                                                            0.0
Bachelors                                                            0.0
College                                                              0.0
Doctorate                                                            0.0
High School                                                          0.0
Masters                                                              0.0
```

```
Other                                                                   0.0
Primary                                                                 0.0
Unknown                                                                 0.0

Employment_Simplified  Employed (Part-time)  Employed (Part-time) + Retired  \
EdLevel
Associate                               2.2                             0.0
Bachelors                               1.6                             0.0
College                                 1.8                             0.0
Doctorate                               2.8                             0.1
High School                             1.9                             0.0
Masters                                 2.1                             0.0
Other                                   2.7                             0.0
Primary                                 2.3                             0.0
Unknown                                 2.6                             0.0

Employment_Simplified  Employed (Part-time) + Self-employed  \
EdLevel
Associate                                               0.6
Bachelors                                               0.5
College                                                 0.8
Doctorate                                               1.0
High School                                             0.5
Masters                                                 0.7
Other                                                   0.9
Primary                                                 0.6
Unknown                                                 0.4

Employment_Simplified  Employed (Part-time) + Student  ...  Student + Retired  \
EdLevel                                                ...
Associate                                         3.0  ...                0.0
Bachelors                                         2.1  ...                0.0
College                                           4.6  ...                0.0
Doctorate                                         0.4  ...                0.0
High School                                       6.9  ...                0.0
Masters                                           1.0  ...                0.0
Other                                             2.6  ...                0.1
Primary                                           2.7  ...                0.1
Unknown                                           2.5  ...                0.0

Employment_Simplified  Student + Unemployed (Looking for work)  \
EdLevel
Associate                                                  1.6
Bachelors                                                  1.0
College                                                    2.2
Doctorate                                                  0.1
High School                                                4.0
Masters                                                    0.3
Other                                                      1.9
Primary                                                    3.5
Unknown                                                    1.4

Employment_Simplified  Student + Unemployed (Looking for work) + Self-employed  \
EdLevel
Associate                                                             0.1
Bachelors                                                             0.2
```

```
College                                                                    0.4
Doctorate                                                                  0.1
High School                                                                0.6
Masters                                                                    0.0
Other                                                                      0.2
Primary                                                                    0.5
Unknown                                                                    0.3

Employment_Simplified  Student + Unemployed (Not looking for work)  \
EdLevel
Associate                                                       0.2
Bachelors                                                       0.1
College                                                         0.7
Doctorate                                                       0.1
High School                                                     2.8
Masters                                                         0.1
Other                                                           1.4
Primary                                                         5.3
Unknown                                                         0.8

Employment_Simplified  Student + Unemployed (Not looking for work) + Self-employed
\
EdLevel
Associate                                                                  0.0
Bachelors                                                                  0.0
College                                                                    0.0
Doctorate                                                                  0.0
High School                                                                0.2
Masters                                                                    0.0
Other                                                                      0.0
Primary                                                                    0.1
Unknown                                                                    0.0

Employment_Simplified  Unemployed (Looking for work)  \
EdLevel
Associate                                        4.1
Bachelors                                        3.9
College                                          4.1
Doctorate                                        2.2
High School                                      3.5
Masters                                          2.5
Other                                            5.5
Primary                                          4.5
Unknown                                          4.6

Employment_Simplified  Unemployed (Looking for work) + Retired  \
EdLevel
Associate                                                  0.0
Bachelors                                                  0.0
College                                                    0.0
Doctorate                                                  0.0
High School                                                0.0
Masters                                                    0.0
Other                                                      0.0
Primary                                                    0.0
Unknown                                                    0.0
```

```
Employment_Simplified  Unemployed (Looking for work) + Unemployed (Not looking for w
ork)  \
EdLevel
Associate                                                                     0.1
Bachelors                                                                     0.1
College                                                                       0.1
Doctorate                                                                     0.0
High School                                                                   0.1
Masters                                                                       0.1
Other                                                                         0.0
Primary                                                                       0.1
Unknown                                                                       0.0

Employment_Simplified  Unemployed (Not looking for work)  \
EdLevel
Associate                                            0.5
Bachelors                                            0.6
College                                              1.1
Doctorate                                            0.5
High School                                          2.5
Masters                                              0.4
Other                                                3.1
Primary                                              7.1
Unknown                                              1.2

Employment_Simplified  Unemployed (Not looking for work) + Retired
EdLevel
Associate                                                      0.0
Bachelors                                                      0.0
College                                                        0.0
Doctorate                                                      0.0
High School                                                    0.0
Masters                                                        0.0
Other                                                          0.1
Primary                                                        0.0
Unknown                                                        0.0

[9 rows x 31 columns]
```

In [90]:

```python
# Visualization (Stacked Bar Plot)

edu_emp_ct.plot(kind='bar', stacked=True, figsize=(10,6), colormap='tab20')
plt.title('Employment Type Distribution by Education Level')
plt.ylabel('Percentage')
plt.xlabel('Education Level')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Employment Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Distribution of `Industry`

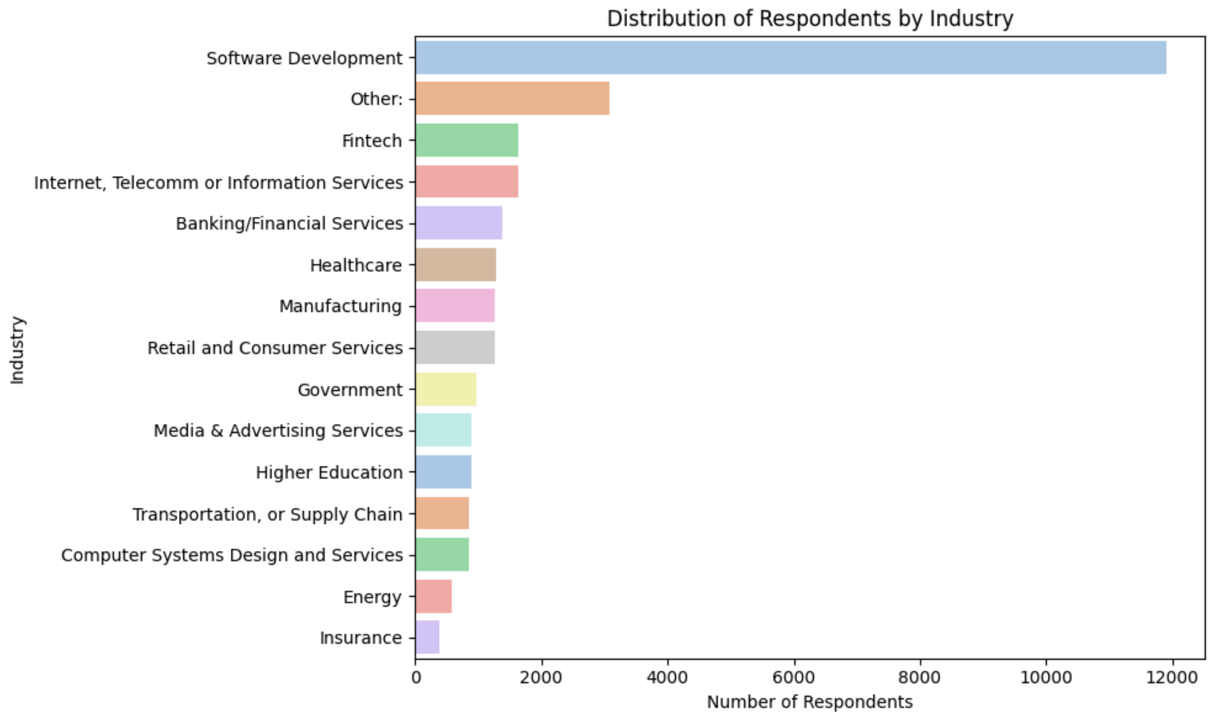
Explore how respondents are distributed across different industries.

In [91]:
```
print('Distribution of Respondents by Industry', df['Industry'].value_counts())
```

```
Distribution of Respondents by Industry Industry
Software Development                         11918
Other:                                        3077
Fintech                                       1641
Internet, Telecomm or Information Services    1629
Banking/Financial Services                    1371
Healthcare                                    1277
Manufacturing                                 1265
Retail and Consumer Services                  1264
Government                                     962
Media & Advertising Services                   894
Higher Education                               890
Transportation, or Supply Chain                859
Computer Systems Design and Services           844
Energy                                         578
Insurance                                      389
Name: count, dtype: int64
```

In [92]:
```
# Barplot

plt.figure(figsize=(10, 6))
sns.barplot(y=df['Industry'].value_counts().index, x=df['Industry'].value_counts().
```

```
plt.title('Distribution of Respondents by Industry')
plt.xlabel('Number of Respondents')
plt.ylabel('Industry')
plt.tight_layout()
plt.show()
```



Obs:

Dominated by `Software Development` (11.9K), followed by `Fintech`, `Internet/IT`, and `Healthcare`.

- Primary audience comes from core tech and related industries.
- Skill demand aligns with trends in digital transformation and tech innovation.

Save the Data

In [93]:
```
df.to_csv(r"./clean_so_survey_data.csv", index=False)
print("File Saved Successfully !")
```

```
File Saved Successfully !
```

Observation

- Open-source ecosystems (PostgreSQL, Python, Linux, React) dominate current and future preferences.
- There's a visible shift towards cloud-native and full-stack JS technologies.

- Youth-driven participation suggests rapid technology adoption and a learning-centric mindset.
- Cloud computing, DevOps tools, and flexible work environments are now fundamental expectations in the tech world.

Overall Findings & Implications

- PostgreSQL, React, Python, and AWS are clear leaders in both usage and future preference.
- Professionals are eager to explore scalable, flexible, and cloud-compatible technologies.
- There's growing interest in Rust, Go, and NoSQL solutions like MongoDB and Redis.
- Companies need to realign L&D programs toward these new areas of interest and demand.
- Most professionals are young, formally educated, and actively employed—creating a highly trainable and adaptable workforce.

Usefullness of these findings

How these Insights are Helpful ?

What are they about?

1. Skill Demand Awareness

- Reveals which programming languages, databases, frameworks, and platforms are in actual industry use and which are gaining popularity.
- Helps forecast future trends in technology adoption.

2. Workforce Trends

- Provides insight into work models (remote, hybrid, in-person), employment types, education levels, and demographics.

3. Technology Alignment

- Indicates the most relevant tools and stacks for building products or services aligned with developer interest and productivity.

4. Talent and Hiring Strategy

- Informs companies about where to focus hiring, upskilling, and recruitment outreach.

Real-World Implications

For Businesses

- **Product Strategy**: SaaS or developer-focused tools should prioritize support for popular languages (e.g., Python, JavaScript) and databases (PostgreSQL, MySQL).
- **Cloud Adoption**: Heavy usage and interest in AWS and Azure show that cloud-native design is no longer optional.
- **Remote Work Policy**: Demand for hybrid/remote models implies businesses need flexible work structures to retain top talent.
- **Hiring Insight**: With most professionals aged 25–34, hiring strategies should cater to millennial and Gen Z expectations.

For Students & Career Starters

- **Learning Priorities**: Focus on in-demand technologies like Python, React, PostgreSQL, and cloud platforms (AWS).
- **Competitive Edge**: Learning Rust, Go, or tools like MongoDB or Redis can set them apart in niche roles.
- **Career Path Clarity**: Insights into industry preferences help in choosing a specialization (e.g., backend, full-stack, DevOps).

For Job Seekers

- **Upskilling Focus**: Enables targeted learning to meet market demands and improve employability.
- **Platform Familiarity**: Proficiency in cloud and web frameworks aligns with job descriptions in modern tech roles.
- **Better Career Decisions**: Helps assess which tools and domains are future-proof and industry-relevant.

For Employers & Recruiters

- **Training Roadmaps**: Align internal L&D programs to popular languages, tools, and platforms.
- **Tech Stack Choices**: Adopt tech stacks that are widely known — making hiring and onboarding easier.
- **Global Hiring Strategies**: Understand which countries have the most tech talent and adjust sourcing plans accordingly.

Other Strategic Implications

Curriculum Design

- **For educators and bootcamps**: Aligning course offerings with in-demand languages (Python, SQL) and platforms (AWS, React) increases student placement success.

Market Forecasting

- **For investors and strategists**: Helps identify emerging tech (Rust, Next.js, DynamoDB) that may drive new markets or ventures.

Global Talent Insights

- High representation from countries like USA, India, Germany shows where remote hiring or tech expansion might be most fruitful.

Tool/Framework Development

- Frameworks like React and Node.js being popular suggest tool builders should focus on integrating or enhancing compatibility with these.

Summary of Value

Stakeholder	Value Derived from Insights
Businesses	Better tech and hiring decisions, cloud alignment
Students	Clear skill priorities, better job readiness
Job Seekers	Targeted upskilling and tool mastery
Employers	Smarter recruitment, tech stack planning
Educators	Curriculum aligned with job market
Tool Makers	Product strategy based on developer needs