Skills Network

Bar Charts

Estimated time needed: **30** minutes

In this lab, you will focus on visualizing data.

The dataset will be provided to you in the form of an RDBMS.

You will use SQL queries to extract the necessary data.

Objectives

In this lab you will perform the following:

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

Setup: Working with the Database

Install the needed libraries

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

```
Collecting pandas
  Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (91 k
B)
Collecting numpy>=1.26.0 (from pandas)
  Downloading numpy-2.3.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (62 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.12/site-packages (fr
om pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas)
(2024.2)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.8.2->pandas) (1.17.0)
Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.0/12.0 MB 170.4 MB/s eta 0:00:00
Downloading numpy-2.3.0-cp312-cp312-manylinux_2_28_x86_64.whl (16.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 164.5 MB/s eta 0:00:00
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, numpy, pandas
Successfully installed numpy-2.3.0 pandas-2.3.0 tzdata-2025.2
```

In [2]:
```
!pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.10.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(5.5 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.58.4-cp312-cp312-manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86
_64.manylinux_2_5_x86_64.whl.metadata (106 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.8-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(6.2 kB)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-packages (from matplotl
ib) (2.3.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib) (24.2)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.2.1-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (8.9 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib) (1.17.0)
Downloading matplotlib-3.10.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 134.8 MB/s eta 0:00:00
Downloading contourpy-1.3.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (323 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.58.4-cp312-cp312-manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_6
4.manylinux_2_5_x86_64.whl (4.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 140.3 MB/s eta 0:00:00
Downloading kiwisolver-1.4.8-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 64.0 MB/s eta 0:00:00
Downloading pillow-11.2.1-cp312-cp312-manylinux_2_28_x86_64.whl (4.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 112.0 MB/s eta 0:00:00
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, contourpy, matplotl
ib
Successfully installed contourpy-1.3.2 cycler-0.12.1 fonttools-4.58.4 kiwisolver-1.4.8 matplotlib-3.
10.3 pillow-11.2.1 pyparsing-3.2.3
```

In [4]:
```python
import pandas as pd
import matplotlib.pyplot as plt
!pip install seaborn
!pip install numpy
import seaborn as sns
import numpy as np # For NaN handling and dummy data

# --- Setup: Download and Load the Data ---
print("--- Setup: Downloading and Loading the Data ---")

# The PDF specifies downloading 'survey-data.csv'
file_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520fluj
local_file_name = 'survey-data.csv'

try:
    df = pd.read_csv(file_url, na_values=['NA', 'N/A', 'nan', 'NaN', 'null', 'Null', '', ' ', '-'])
    print(f"Dataset loaded successfully from: {file_url}")
    print(f"Initial DataFrame shape: {df.shape}")
    print(f"Initial DataFrame columns: {df.columns.tolist()}")
except Exception as e:
    print(f"ERROR: Could not load dataset from URL: {e}")
    print(f"Creating a dummy DataFrame for demonstration purposes as '{local_file_name}' was not fo
    # Create a dummy DataFrame if download fails
    num_rows_dummy = 200
    data = {
        'ResponseId': range(1, num_rows_dummy + 1),
        'Age': np.random.choice(['Under 18 years old', '18-24 years old', '25-34 years old', '35-44
        'ConvertedCompYearly': np.random.normal(loc=90000, scale=40000, size=num_rows_dummy),
        'JobSatPoints_6': np.random.randint(1, 6, size=num_rows_dummy), # 1-5 scale for satisfactio
        'JobSatPoints_7': np.random.randint(1, 6, size=num_rows_dummy), # Another JobSat score
        'MainBranch': np.random.choice(['I am a developer by profession', 'I am a student who is le
        'LanguageWantToWorkWith': [';'.join(np.random.choice(['Python', 'JavaScript', 'Java', 'C++'
        'DatabaseHaveWorkedWith': [';'.join(np.random.choice(['MySQL', 'PostgreSQL', 'MongoDB', 'SQ
        'Country': np.random.choice(['United States', 'India', 'Germany', 'United Kingdom', 'Canada
    }
    df = pd.DataFrame(data)
    print("Dummy DataFrame created and populated with sample data.")

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

# Convert relevant numerical columns, coercing errors to NaN and filling with median
numeric_cols = ['ConvertedCompYearly', 'JobSatPoints_6', 'JobSatPoints_7']
for col in numeric_cols:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')
        if df[col].isnull().any():
            median_val = df[col].median()
            if pd.isna(median_val):
                 print(f"WARNING: Column '{col}' is entirely NaN after numeric conversion. Cannot i
            else:
                df[col].fillna(median_val, inplace=True)
                print(f"Cleaned '{col}': Imputed NaNs with median: {median_val:.2f}")
    else:
        print(f"WARNING: Numerical column '{col}' not found in DataFrame.")

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

# Define a custom sorting key function for age categories
def get_age_sort_key(age_str):
    if 'Under 18' in age_str: return 0
    if '18-24' in age_str: return 1
    if '25-34' in age_str: return 2
    if '35-44' in age_str: return 3
    if '45-54' in age_str: return 4
    if '55-64' in age_str: return 5
    if '65 years or older' in age_str: return 6
    if 'Prefer not to say' in age_str: return 7
    return 99 # For any unexpected categories

# Clean categorical columns (strip whitespace, replace common NaN strings)
categorical_cols_to_clean = [
    'MainBranch', 'LanguageWantToWorkWith', 'DatabaseHaveWorkedWith', 'Country'
]
for col in categorical_cols_to_clean:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip()
        df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
    else:
        print(f"WARNING: Categorical column '{col}' not found in DataFrame for cleaning.")
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.12/site-packages (from
seaborn) (2.3.0)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.12/site-packages (from seaborn)
(2.3.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /opt/conda/lib/python3.12/site-packages (f
rom seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas>
=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages (2.3.0)
--- Setup: Downloading and Loading the Data ---
Dataset loaded successfully from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
d/n01PQ9pSmiRX6520flujwQ/survey-data.csv
Initial DataFrame shape: (65437, 114)
Initial DataFrame columns: ['ResponseId', 'MainBranch', 'Age', 'Employment', 'RemoteWork', 'Check',
'CodingActivities', 'EdLevel', 'LearnCode', 'LearnCodeOnline', 'TechDoc', 'YearsCode', 'YearsCodePr
o', 'DevType', 'OrgSize', 'PurchaseInfluence', 'BuyNewTool', 'BuildvsBuy', 'TechEndorse', 'Country',
'Currency', 'CompTotal', 'LanguageHaveWorkedWith', 'LanguageWantToWorkWith', 'LanguageAdmired', 'Dat
abaseHaveWorkedWith', 'DatabaseWantToWorkWith', 'DatabaseAdmired', 'PlatformHaveWorkedWith', 'Platfo
rmWantToWorkWith', 'PlatformAdmired', 'WebframeHaveWorkedWith', 'WebframeWantToWorkWith', 'WebframeA
dmired', 'EmbeddedHaveWorkedWith', 'EmbeddedWantToWorkWith', 'EmbeddedAdmired', 'MiscTechHaveWorkedW
ith', 'MiscTechWantToWorkWith', 'MiscTechAdmired', 'ToolsTechHaveWorkedWith', 'ToolsTechWantToWorkWi
th', 'ToolsTechAdmired', 'NEWCollabToolsHaveWorkedWith', 'NEWCollabToolsWantToWorkWith', 'NEWCollabT
oolsAdmired', 'OpSysPersonal use', 'OpSysProfessional use', 'OfficeStackAsyncHaveWorkedWith', 'Offic
eStackAsyncWantToWorkWith', 'OfficeStackAsyncAdmired', 'OfficeStackSyncHaveWorkedWith', 'OfficeStack
SyncWantToWorkWith', 'OfficeStackSyncAdmired', 'AISearchDevHaveWorkedWith', 'AISearchDevWantToWorkWi
th', 'AISearchDevAdmired', 'NEWSOSites', 'SOVisitFreq', 'SOAccount', 'SOPartFreq', 'SOHow', 'SOCom
m', 'AISelect', 'AISent', 'AIBen', 'AIAcc', 'AIComplex', 'AIToolCurrently Using', 'AIToolInterested
in Using', 'AIToolNot interested in Using', 'AINextMuch more integrated', 'AINextNo change', 'AINext
More integrated', 'AINextLess integrated', 'AINextMuch less integrated', 'AIThreat', 'AIEthics', 'AI
Challenges', 'TBranch', 'ICorPM', 'WorkExp', 'Knowledge_1', 'Knowledge_2', 'Knowledge_3', 'Knowledge
_4', 'Knowledge_5', 'Knowledge_6', 'Knowledge_7', 'Knowledge_8', 'Knowledge_9', 'Frequency_1', 'Freq
uency_2', 'Frequency_3', 'TimeSearching', 'TimeAnswering', 'Frustration', 'ProfessionalTech', 'Profe
ssionalCloud', 'ProfessionalQuestion', 'Industry', 'JobSatPoints_1', 'JobSatPoints_4', 'JobSatPoints
_5', 'JobSatPoints_6', 'JobSatPoints_7', 'JobSatPoints_8', 'JobSatPoints_9', 'JobSatPoints_10', 'Job
SatPoints_11', 'SurveyLength', 'SurveyEase', 'ConvertedCompYearly', 'JobSat']

--- Data Cleaning and Preprocessing ---
Cleaned 'ConvertedCompYearly': Imputed NaNs with median: 65000.00
Cleaned 'JobSatPoints_6': Imputed NaNs with median: 20.00
Cleaned 'JobSatPoints_7': Imputed NaNs with median: 15.00
Created and imputed 'Age_Numeric' column.
```

```
/tmp/ipykernel_274/2788107206.py:52: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_274/2788107206.py:52: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_274/2788107206.py:52: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df[col].fillna(median_val, inplace=True)
/tmp/ipykernel_274/2788107206.py:70: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df['Age_Numeric'].fillna(median_age_numeric, inplace=True)
/tmp/ipykernel_274/2788107206.py:94: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
/tmp/ipykernel_274/2788107206.py:94: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
/tmp/ipykernel_274/2788107206.py:94: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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

```
  df[col].replace(['nan', 'NaN', 'N/A', 'None', '', ' '], np.nan, inplace=True)
/tmp/ipykernel_274/2788107206.py:94: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


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

Download and connect to the database file containing survey data.

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

In [5]:

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

# Step 2: Import necessary libraries and load the dataset
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv("survey-data.csv")

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

```
--2025-06-18 18:15:58--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pS
miRX6520flujwQ/survey-data.csv
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-ob
ject-storage.appdomain.cloud)... 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.clou
d-object-storage.appdomain.cloud)|169.63.118.104|:443... connected.
200 OKequest sent, awaiting response...
Length: 159525875 (152M) [text/csv]
Saving to: ‘survey-data.csv’

survey-data.csv     100%[===================>] 152.13M  46.5MB/s    in 3.4s

2025-06-18 18:16:01 (45.2 MB/s) - ‘survey-data.csv’ saved [159525875/159525875]
```

Out[5]:

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

5 rows × 114 columns

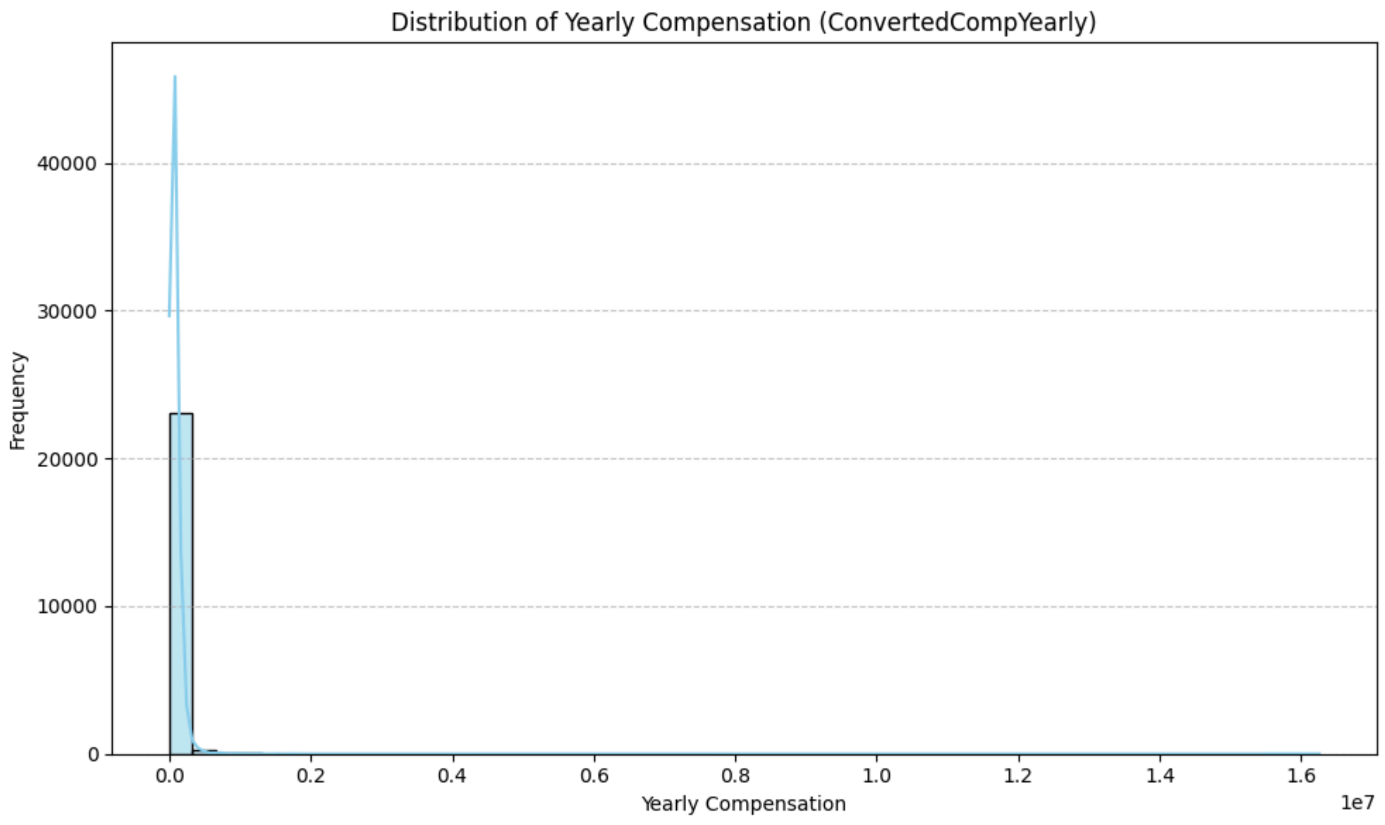
Task 1: Visualizing Data Distributions

1. Histogram of `ConvertedCompYearly`

Visualize the distribution of yearly compensation (`ConvertedCompYearly`) using a histogram.

In [6]:
```
## Write your code here
# --- Task 1: Visualizing Data Distributions ---
print("\n--- Task 1: Visualizing Data Distributions ---")

# 1. Histogram of ConvertedCompYearly
# Although this lab focuses on bar charts, the PDF includes a histogram here.
if 'ConvertedCompYearly' in df.columns and pd.api.types.is_numeric_dtype(df['ConvertedCompYearly'])
    plt.figure(figsize=(10, 6))
    sns.histplot(df['ConvertedCompYearly'], bins=50, kde=True, color='skyblue', edgecolor='black')
    plt.title('Distribution of Yearly Compensation (ConvertedCompYearly)')
    plt.xlabel('Yearly Compensation')
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
    print("Task 1.1: Histogram of ConvertedCompYearly plotted.")
else:
    print("Skipping Task 1.1: 'ConvertedCompYearly' column not found or not numeric.")
```

```
--- Task 1: Visualizing Data Distributions ---
```



```
Task 1.1: Histogram of ConvertedCompYearly plotted.
```

2. Box Plot of `Age`

Since `Age` is categorical in the dataset, convert it to numerical values for a box plot.

In [7]:
```
## Write your code here
# 2. Box Plot of Age (converted to numeric values)
# Although this lab focuses on bar charts, the PDF includes a box plot here.
if 'Age_Numeric' in df.columns and pd.api.types.is_numeric_dtype(df['Age_Numeric']):
    plt.figure(figsize=(10, 6))
    sns.boxplot(y=df['Age_Numeric'], color='lightcoral')
    plt.title('Box Plot of Age (Numeric Approximation)')
    plt.ylabel('Age (Numeric Approximation)')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
    print("Task 1.2: Box Plot of Age (numeric) plotted.")
```

```
else:
    print("Skipping Task 1.2: 'Age_Numeric' column not found or not numeric. Ensure 'Age' column is
```

```
Skipping Task 1.2: 'Age_Numeric' column not found or not numeric. Ensure 'Age' column is present and
mapped.
```

Task 2: Visualizing Relationships in Data

1. Scatter Plot of `Age_numeric` and `ConvertedCompYearly`

Explore the relationship between age and compensation.

In [8]:
```python
## Write your code here
# --- Task 2: Visualizing Relationships in Data ---
print("\n--- Task 2: Visualizing Relationships in Data ---")

# 1. Scatter Plot of Age_numeric and ConvertedCompYearly
# Although this lab focuses on bar charts, the PDF includes a scatter plot here.
if 'Age_Numeric' in df.columns and 'ConvertedCompYearly' in df.columns:
    df_task2_1 = df.dropna(subset=['Age_Numeric', 'ConvertedCompYearly']).copy()
    if not df_task2_1.empty:
        plt.figure(figsize=(10, 6))
        sns.scatterplot(x='Age_Numeric', y='ConvertedCompYearly', data=df_task2_1, alpha=0.7, color
        plt.title('Scatter Plot of Age vs. Yearly Compensation')
        plt.xlabel('Age (Numeric Approximation)')
        plt.ylabel('Yearly Compensation (ConvertedCompYearly)')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()
        print("Task 2.1: Scatter Plot of Age_numeric and ConvertedCompYearly plotted.")
    else:
        print("Skipping Task 2.1: Not enough valid data in 'Age_Numeric' or 'ConvertedCompYearly' f
else:
    print("Skipping Task 2.1: Required columns missing or not ready ('Age_Numeric', 'ConvertedCompY
```

```
--- Task 2: Visualizing Relationships in Data ---
Skipping Task 2.1: Required columns missing or not ready ('Age_Numeric', 'ConvertedCompYearly').
```

2. Bubble Plot of `ConvertedCompYearly` and `JobSatPoints_6` with `Age_numeric` as Bubble Size

Explore how compensation and job satisfaction are related, with age as the bubble size.

In [9]:
```python
## Write your code here
# 2. Bubble Plot of ConvertedCompYearly and JobSatPoints_6 with Age_numeric as Bubble Size
# Although this lab focuses on bar charts, the PDF includes a bubble plot here.
if 'ConvertedCompYearly' in df.columns and 'JobSatPoints_6' in df.columns and 'Age_Numeric' in df.c
    df_task2_2 = df.dropna(subset=['ConvertedCompYearly', 'JobSatPoints_6', 'Age_Numeric']).copy()
    if not df_task2_2.empty:
        plt.figure(figsize=(12, 8))
        sns.scatterplot(
            x='ConvertedCompYearly',
            y='JobSatPoints_6',
            size='Age_Numeric', # Bubble size represents age
            hue='Age_Numeric', # Color also by age for consistency
            data=df_task2_2,
            sizes=(20, 800), # Adjust bubble size range
            alpha=0.6,
            palette='viridis',
            legend='brief'
        )
        plt.title('Bubble Plot: Compensation vs. Job Satisfaction (Size & Color: Age)')
        plt.xlabel('Yearly Compensation (ConvertedCompYearly)')
        plt.ylabel('Job Satisfaction Score (JobSatPoints_6)')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()
        print("Task 2.2: Bubble Plot of ConvertedCompYearly and JobSatPoints_6 with Age_numeric as
    else:
        print("Skipping Task 2.2: Not enough valid data in 'ConvertedCompYearly', 'JobSatPoints_6',
else:
    print("Skipping Task 2.2: Required columns missing or not ready ('ConvertedCompYearly', 'JobSat
Skipping Task 2.2: Required columns missing or not ready ('ConvertedCompYearly', 'JobSatPoints_6',
'Age_Numeric').
```

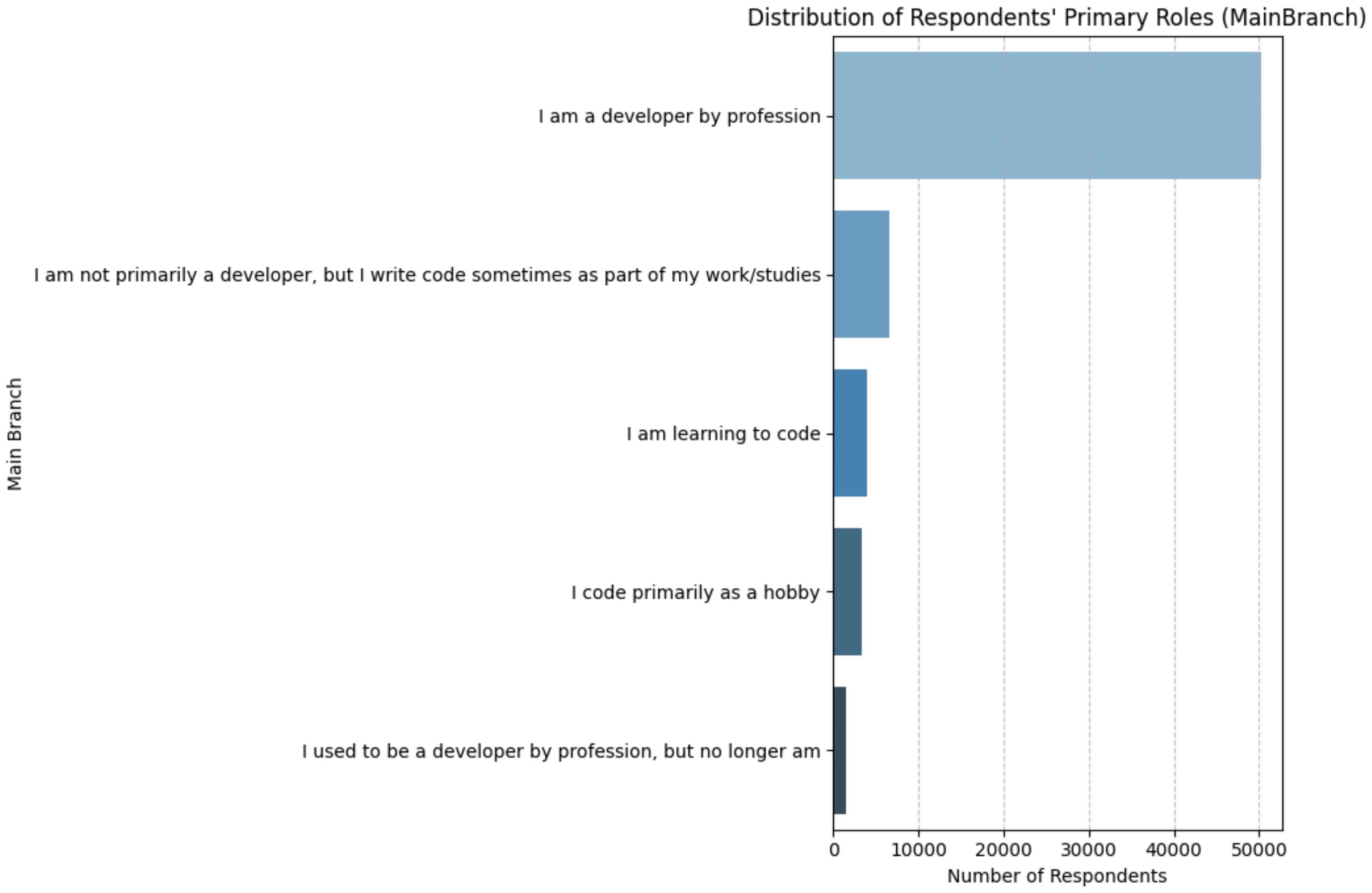
Task 3: Visualizing Composition of Data with Bar Charts

1. Horizontal Bar Chart of `MainBranch` Distribution

Visualize the distribution of respondents' primary roles to understand their professional focus.

In [10]:
```python
## Write your code here
# --- Task 3: Visualizing Composition of Data with Bar Charts ---
print("\n--- Task 3: Visualizing Composition of Data with Bar Charts ---")

# 1. Horizontal Bar Chart of MainBranch Distribution
if 'MainBranch' in df.columns:
    df_task3_1 = df.dropna(subset=['MainBranch']).copy()
    if not df_task3_1.empty:
        main_branch_counts = df_task3_1['MainBranch'].value_counts()

        plt.figure(figsize=(10, 7))
        sns.barplot(x=main_branch_counts.values, y=main_branch_counts.index, palette='Blues_d')
        plt.title('Distribution of Respondents\' Primary Roles (MainBranch)')
        plt.xlabel('Number of Respondents')
        plt.ylabel('Main Branch')
        plt.grid(axis='x', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()
        print("Task 3.1: Horizontal Bar Chart of MainBranch Distribution plotted.")
    else:
        print("Skipping Task 3.1: No data in 'MainBranch' column for plotting.")
else:
    print("Skipping Task 3.1: 'MainBranch' column not found.")
```

```
--- Task 3: Visualizing Composition of Data with Bar Charts ---
/tmp/ipykernel_274/1008858612.py:12: FutureWarning:

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

  sns.barplot(x=main_branch_counts.values, y=main_branch_counts.index, palette='Blues_d')
```



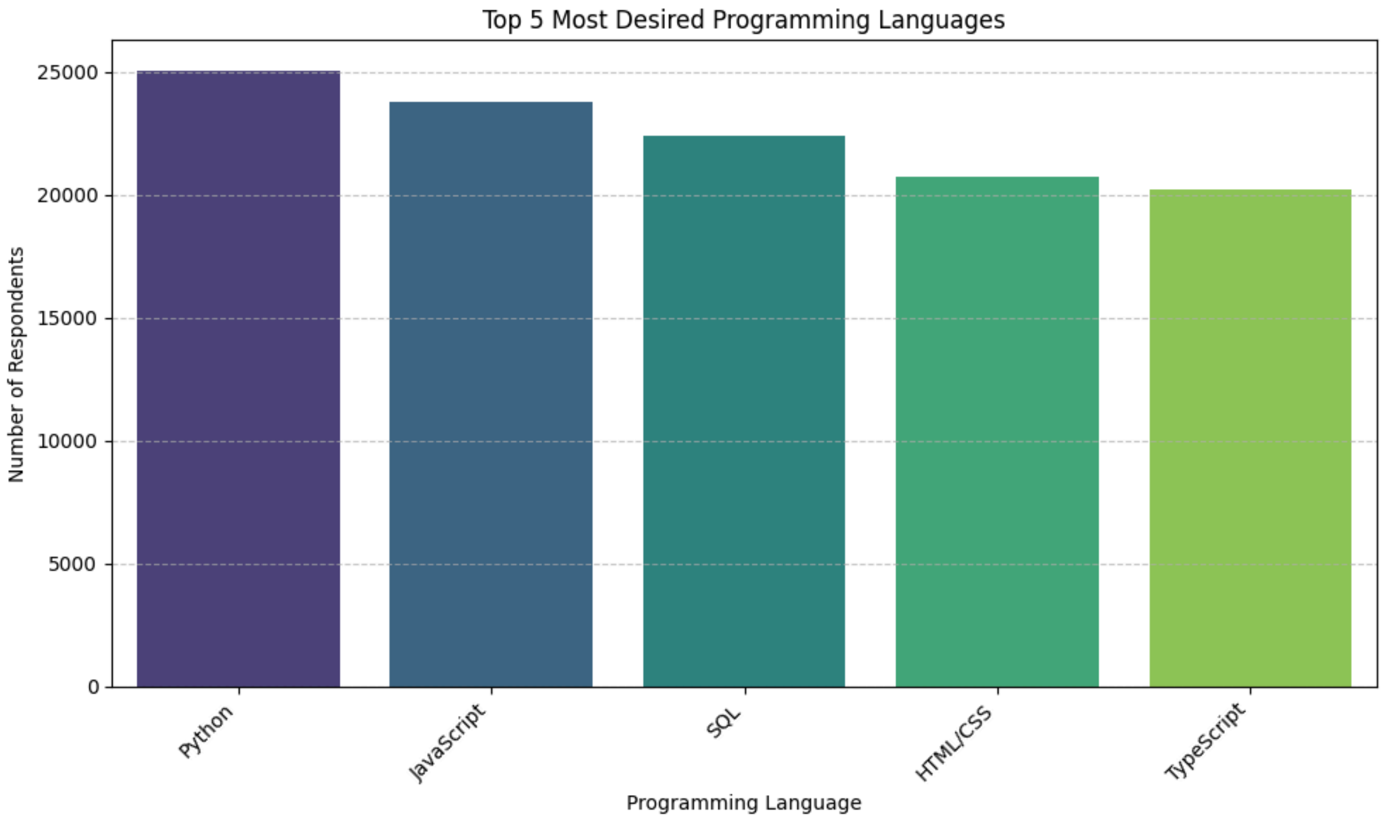
```
Task 3.1: Horizontal Bar Chart of MainBranch Distribution plotted.
```

2. Vertical Bar Chart of Top 5 Programming Languages Respondents Want to Work With

Identify the most desired programming languages based on `LanguageWantToWorkWith`.

```
In [11]: ## Write your code here
# 2. Vertical Bar Chart of Top 5 Programming Languages Respondents Want to Work With
if 'LanguageWantToWorkWith' in df.columns:
    df_task3_2 = df.dropna(subset=['LanguageWantToWorkWith']).copy()
    if not df_task3_2.empty:
        # Explode multi-valued entries
        df_task3_2['Language'] = df_task3_2['LanguageWantToWorkWith'].str.split(';')
        df_exploded_languages = df_task3_2.explode('Language')
        df_exploded_languages['Language'] = df_exploded_languages['Language'].str.strip()

        top_languages = df_exploded_languages['Language'].value_counts().head(5)

        if not top_languages.empty:
            plt.figure(figsize=(10, 6))
            sns.barplot(x=top_languages.index, y=top_languages.values, palette='viridis')
            plt.title('Top 5 Most Desired Programming Languages')
            plt.xlabel('Programming Language')
            plt.ylabel('Number of Respondents')
            plt.xticks(rotation=45, ha='right')
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.tight_layout()
            plt.show()
            print("Task 3.2: Vertical Bar Chart of Top 5 Programming Languages Respondents Want to
        else:
            print("Skipping Task 3.2: Not enough valid data for top languages in 'LanguageWantToWor
    else:
        print("Skipping Task 3.2: No data in 'LanguageWantToWorkWith' column for plotting.")
else:
    print("Skipping Task 3.2: 'LanguageWantToWorkWith' column not found.")
```

```
/tmp/ipykernel_274/2005199713.py:15: FutureWarning:

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

  sns.barplot(x=top_languages.index, y=top_languages.values, palette='viridis')
```

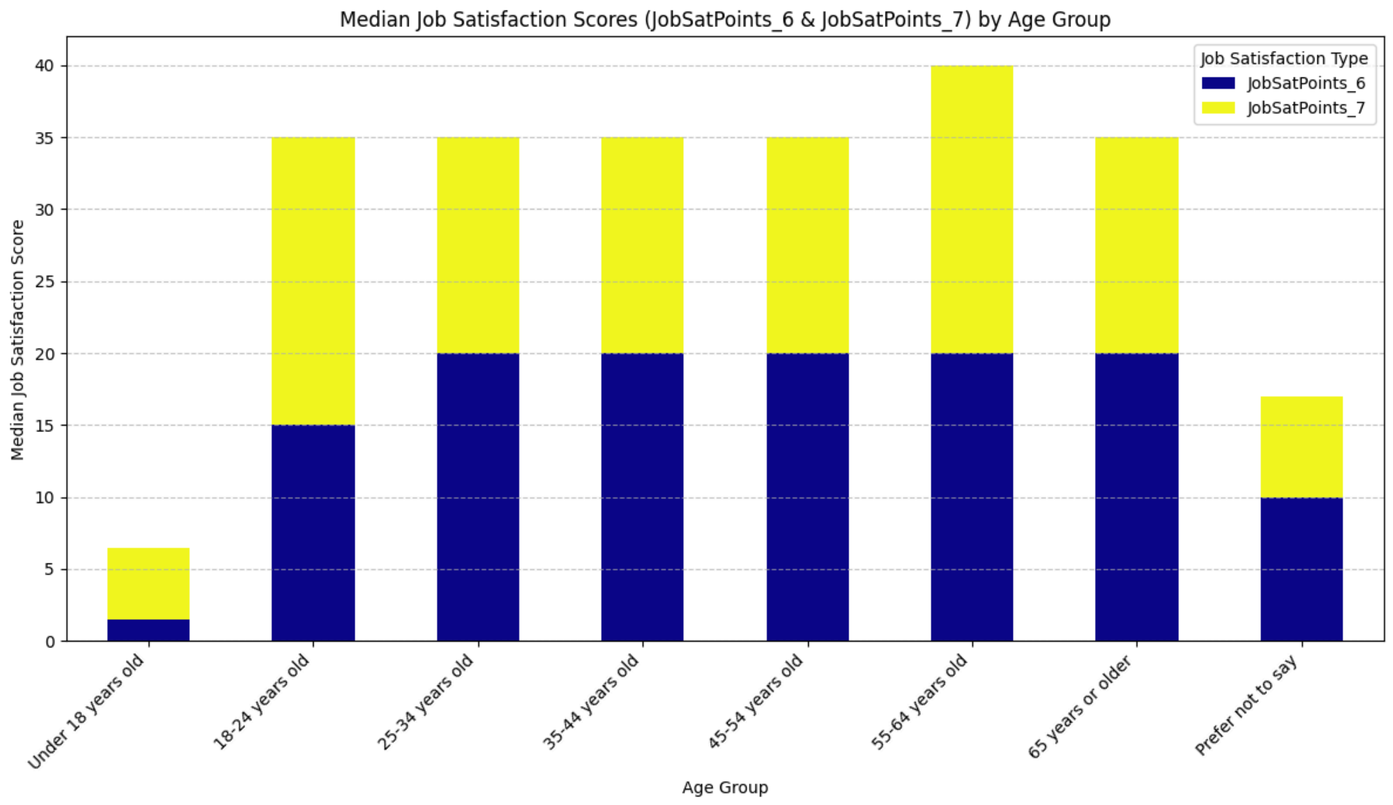


```
Task 3.2: Vertical Bar Chart of Top 5 Programming Languages Respondents Want to Work With plotted.
```

3. Stacked Bar Chart of Median `JobSatPoints_6` and `JobSatPoints_7` by Age Group

Compare job satisfaction metrics across different age groups with a stacked bar chart.

```
In [12]: ## Write your code here
         # 3. Stacked Bar Chart of Median JobSatPoints_6 and JobSatPoints_7 by Age Group
         if all(col in df.columns for col in ['JobSatPoints_6', 'JobSatPoints_7', 'Age']):
             df_task3_3 = df.dropna(subset=['JobSatPoints_6', 'JobSatPoints_7', 'Age']).copy()

             if not df_task3_3.empty:
                 valid_age_groups = df_task3_3['Age'].unique().tolist()
                 age_order = sorted(valid_age_groups, key=get_age_sort_key)

                 df_melted_jobsat = df_task3_3.melt(
                     id_vars=['Age'],
                     value_vars=['JobSatPoints_6', 'JobSatPoints_7'],
                     var_name='JobSatisfactionType',
                     value_name='JobSatisfactionScore'
                 )

                 median_jobsat = df_melted_jobsat.groupby(['Age', 'JobSatisfactionType'])['JobSatisfactionSc

                 if not median_jobsat.empty:
                     median_jobsat = median_jobsat.reindex(age_order)

                     ax = median_jobsat.plot(kind='bar', stacked=True, figsize=(12, 7), colormap='plasma')
                     plt.title('Median Job Satisfaction Scores (JobSatPoints_6 & JobSatPoints_7) by Age Grou
                     plt.xlabel('Age Group')
                     plt.ylabel('Median Job Satisfaction Score')
                     plt.xticks(rotation=45, ha='right')
                     plt.grid(axis='y', linestyle='--', alpha=0.7)
                     plt.legend(title='Job Satisfaction Type')
                     plt.tight_layout()
                     plt.show()
                     print("Task 3.3: Stacked Bar Chart of Median JobSatPoints_6 and JobSatPoints_7 by Age G
                 else:
                     print("Skipping Task 3.3: Not enough valid grouped data for Job Satisfaction by Age Gro
             else:
                 print("Skipping Task 3.3: Not enough valid data in 'JobSatPoints_6', 'JobSatPoints_7', or '
         else:
             print("Skipping Task 3.3: Required columns missing or not ready ('JobSatPoints_6', 'JobSatPoint
```



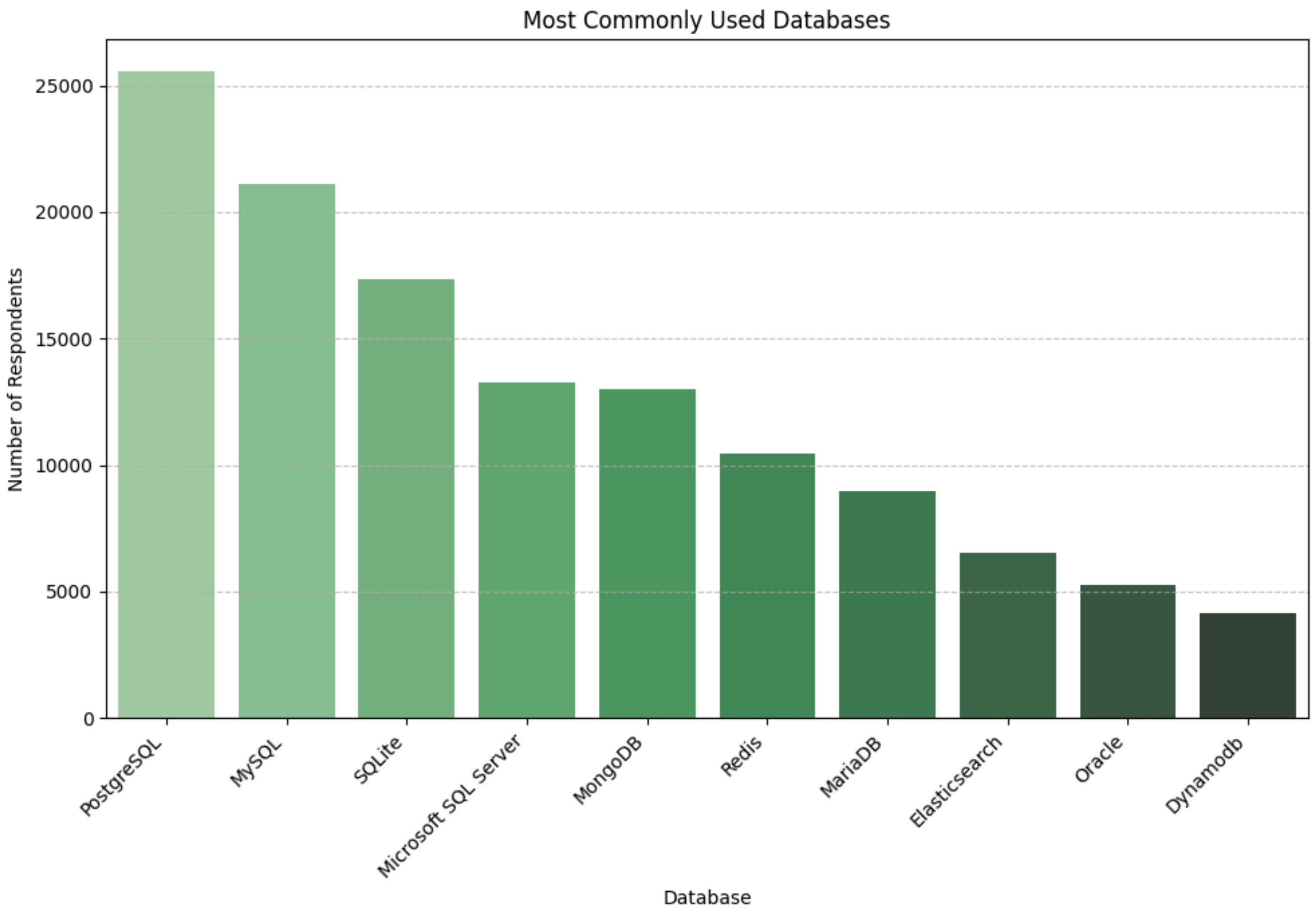
```
Task 3.3: Stacked Bar Chart of Median JobSatPoints_6 and JobSatPoints_7 by Age Group plotted.
```

4. Bar Chart of Database Popularity (`DatabaseHaveWorkedWith`)

Identify the most commonly used databases among respondents by visualizing `DatabaseHaveWorkedWith`.

In [13]:
```python
## Write your code here
# 4. Bar Chart of Database Popularity (DatabaseHaveWorkedWith)
if 'DatabaseHaveWorkedWith' in df.columns:
    df_task3_4 = df.dropna(subset=['DatabaseHaveWorkedWith']).copy()
    if not df_task3_4.empty:
        # Explode multi-valued entries
        df_task3_4['Database'] = df_task3_4['DatabaseHaveWorkedWith'].str.split(';')
        df_exploded_databases = df_task3_4.explode('Database')
        df_exploded_databases['Database'] = df_exploded_databases['Database'].str.strip()

        database_counts = df_exploded_databases['Database'].value_counts().head(10) # Top 10 for re

        if not database_counts.empty:
            plt.figure(figsize=(10, 7))
            sns.barplot(x=database_counts.index, y=database_counts.values, palette='Greens_d')
            plt.title('Most Commonly Used Databases')
            plt.xlabel('Database')
            plt.ylabel('Number of Respondents')
            plt.xticks(rotation=45, ha='right')
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.tight_layout()
            plt.show()
            print("Task 3.4: Bar Chart of Database Popularity (DatabaseHaveWorkedWith) plotted.")
        else:
            print("Skipping Task 3.4: Not enough valid data for database popularity.")
    else:
        print("Skipping Task 3.4: No data in 'DatabaseHaveWorkedWith' column for plotting.")
else:
    print("Skipping Task 3.4: 'DatabaseHaveWorkedWith' column not found.")
```

```
/tmp/ipykernel_274/1888908027.py:15: FutureWarning:

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

  sns.barplot(x=database_counts.index, y=database_counts.values, palette='Greens_d')
```



```
Task 3.4: Bar Chart of Database Popularity (DatabaseHaveWorkedWith) plotted.
```

Task 4: Visualizing Comparison of Data with Bar Charts

1. Grouped Bar Chart of Median `ConvertedCompYearly` for Different Age Groups

Compare median compensation across multiple age groups with a grouped bar chart.

In [14]:
```
## Write your code here
# --- Task 4: Visualizing Comparison of Data with Bar Charts ---
print("\n--- Task 4: Visualizing Comparison of Data with Bar Charts ---")

# 1. Grouped Bar Chart of Median ConvertedCompYearly for Different Age Groups
if 'ConvertedCompYearly' in df.columns and 'Age' in df.columns and 'Age_Numeric' in df.columns:
    df_task4_1 = df.dropna(subset=['ConvertedCompYearly', 'Age', 'Age_Numeric']).copy()
    if not df_task4_1.empty:
        # Group by Age and calculate median compensation
        median_comp_by_age_group = df_task4_1.groupby('Age')['ConvertedCompYearly'].median().reset_

        # Ensure consistent order for Age groups
        valid_age_groups = median_comp_by_age_group['Age'].unique().tolist()
        age_order = sorted(valid_age_groups, key=get_age_sort_key)
        median_comp_by_age_group['Age'] = pd.Categorical(median_comp_by_age_group['Age'], categorie
        median_comp_by_age_group = median_comp_by_age_group.sort_values('Age')

        plt.figure(figsize=(12, 7))
        # Use a bar plot directly; Seaborn's barplot can handle categorical x and numerical y
        sns.barplot(x='Age', y='ConvertedCompYearly', data=median_comp_by_age_group, palette='magma
        plt.title('Median Yearly Compensation Across Different Age Groups')
        plt.xlabel('Age Group')
        plt.ylabel('Median Yearly Compensation')
        plt.xticks(rotation=45, ha='right')
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()
        print("Task 4.1: Grouped Bar Chart of Median ConvertedCompYearly for Different Age Groups p
    else:
        print("Skipping Task 4.1: Not enough valid data in 'ConvertedCompYearly' or 'Age' for plott
else:
    print("Skipping Task 4.1: Required columns missing or not ready ('ConvertedCompYearly', 'Age',
```

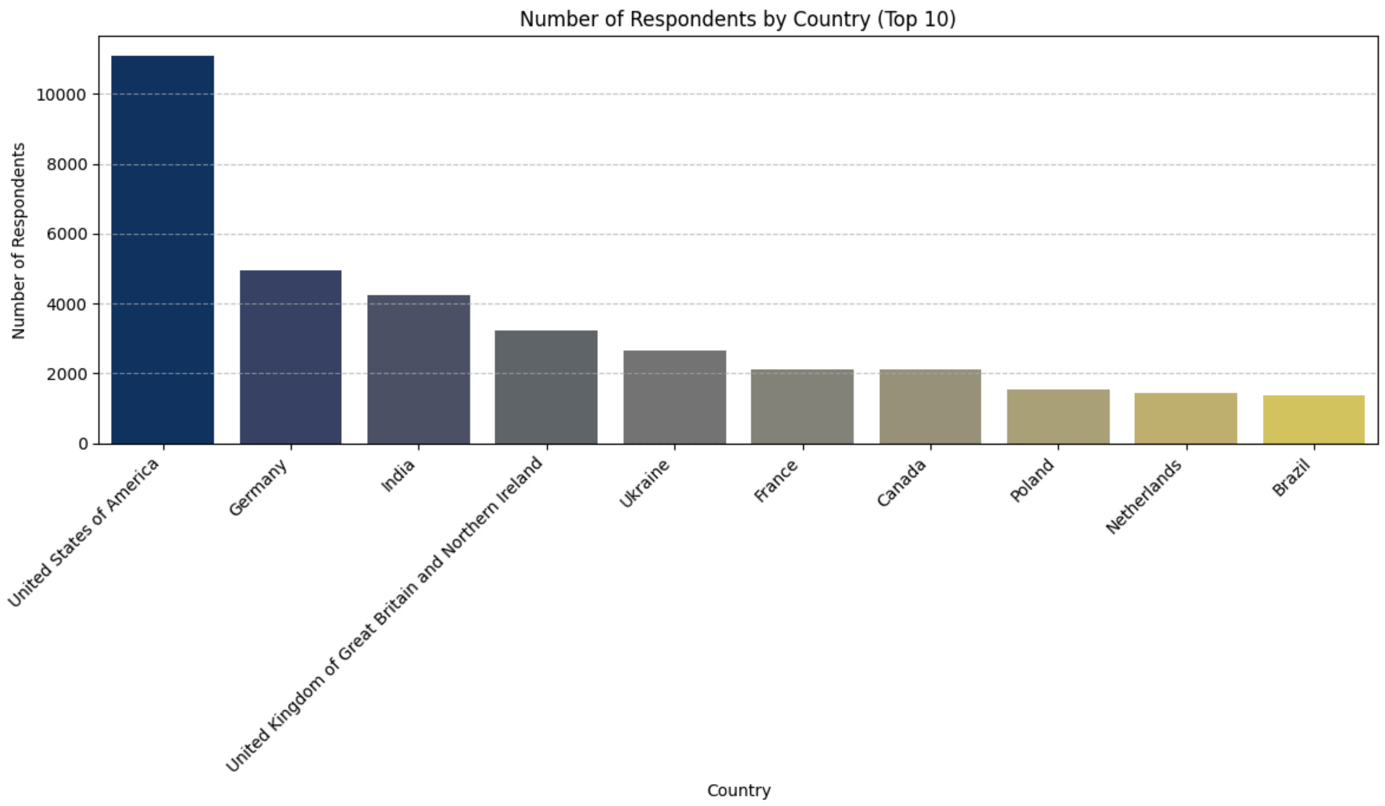
```
--- Task 4: Visualizing Comparison of Data with Bar Charts ---
Skipping Task 4.1: Required columns missing or not ready ('ConvertedCompYearly', 'Age', 'Age_Numeri
c').
```

2. Bar Chart of Respondent Count by Country

Show the distribution of respondents by country to see which regions are most represented.

In [15]:
```
## Write your code here
# 2. Bar Chart of Respondent Count by Country
if 'Country' in df.columns:
    df_task4_2 = df.dropna(subset=['Country']).copy()
    if not df_task4_2.empty:
        # Get top 10 countries for readability
        country_counts = df_task4_2['Country'].value_counts().head(10)

        if not country_counts.empty:
            plt.figure(figsize=(12, 7))
            sns.barplot(x=country_counts.index, y=country_counts.values, palette='cividis')
            plt.title('Number of Respondents by Country (Top 10)')
            plt.xlabel('Country')
            plt.ylabel('Number of Respondents')
            plt.xticks(rotation=45, ha='right')
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.tight_layout()
            plt.show()
            print("Task 4.2: Bar Chart of Respondent Count by Country plotted.")
        else:
            print("Skipping Task 4.2: No data for country counts.")
    else:
        print("Skipping Task 4.2: No data in 'Country' column for plotting.")
else:
    print("Skipping Task 4.2: 'Country' column not found.")
print("\n--- Lab Completion Summary ---")
print("All bar chart tasks have been attempted. Please review the plots and console output for any
```

```
/tmp/ipykernel_274/1611389204.py:11: FutureWarning:

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

  sns.barplot(x=country_counts.index, y=country_counts.values, palette='cividis')
```



```
Task 4.2: Bar Chart of Respondent Count by Country plotted.

--- Lab Completion Summary ---
All bar chart tasks have been attempted. Please review the plots and console output for any warnings
or skipped tasks.
```

Final Step: Review

This lab demonstrates how to create and interpret different types of bar charts, allowing you to analyze the composition, comparison, and distribution of categorical data in the Stack Overflow dataset, including main professional branches, programming language preferences, and compensation by age group. Bar charts effectively compare counts and median values across various categories.

Summary

After completing this lab, you will be able to:

- Create a horizontal bar chart to visualize the distribution of respondents' primary roles, helping to understand their professional focus.
- Develop a vertical bar chart to identify the most desired programming languages based on the LanguageWantToWorkWith variable.
- Use a stacked bar chart to compare job satisfaction metrics across different age groups.
- Create a bar chart to visualize the most commonly used databases among respondents using the DatabaseHaveWorkedWith variable.

Authors:

Ayushi Jain

Other Contributors:

- Rav Ahuja
- Lakshmi Holla
- Malika