Data Visualization

Estimated time needed: **45** minutes

In this lab, you will focus on data visualization. The dataset will be provided through an RDBMS, and you will need to use SQL queries to extract the required data.

Objectives

After completing this lab, you will be able to:

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

Demo: How to work with database

Download the database file.

In [1]:

```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520flujwQ/sur
```

```
--2025-06-18 14:51:38--  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.6’

survey-data.csv.6   100%[===================>] 152.13M  48.3MB/s    in 3.2s

2025-06-18 14:51:43 (48.3 MB/s) - ‘survey-data.csv.6’ saved [159525875/159525875]
```

Install and Import Necessary Python Libraries

Ensure that you have the required libraries installed to work with SQLite and Pandas:

In [2]:

```
!pip install pandas
!pip install matplotlib

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

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

In [3]:

```
!pip install seaborn
!pip install numpy
import seaborn as sns
import numpy as np
```

```
Requirement already satisfied: seaborn in /opt/conda/lib/python3.12/site-packages (0.13.2)
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)
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages (2.3.0)
```

Read the CSV File into a Pandas DataFrame

Load the Stack Overflow survey data into a Pandas DataFrame:

In [4]:

```
# Read the CSV file
df = pd.read_csv('survey-data.csv')
```

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

Out[4]:

	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

Create a SQLite Database and Insert the Data

Now, let's create a new SQLite database (`survey-data.sqlite`) and insert the data from the DataFrame into a table using the sqlite3 library:

In [5]:
```
import sqlite3

# Create a connection to the SQLite database
conn = sqlite3.connect('survey-data.sqlite')

# Write the dataframe to the SQLite database
df.to_sql('main', conn, if_exists='replace', index=False)


# Close the connection
conn.close()
```

Verify the Data in the SQLite Database Verify that the data has been correctly inserted into the SQLite database by running a simple query:

In [6]:
```
# Reconnect to the SQLite database
conn = sqlite3.connect('survey-data.sqlite')

# Run a simple query to check the data
QUERY = "SELECT * FROM main LIMIT 5"
df_check = pd.read_sql_query(QUERY, conn)

# Display the results
print(df_check)
```

```
   ResponseId                      MainBranch                 Age  \
0           1  I am a developer by profession  Under 18 years old   
1           2  I am a developer by profession     35-44 years old   
2           3  I am a developer by profession     45-54 years old   
3           4           I am learning to code     18-24 years old   
4           5  I am a developer by profession     18-24 years old   

            Employment RemoteWork   Check  \
0  Employed, full-time     Remote  Apples   
1  Employed, full-time     Remote  Apples   
2  Employed, full-time     Remote  Apples   
3   Student, full-time       None  Apples   
4   Student, full-time       None  Apples   

                                    CodingActivities  \
0                                              Hobby   
1  Hobby;Contribute to open-source projects;Other...   
2  Hobby;Contribute to open-source projects;Other...   
3                                               None   
4                                               None   

                                             EdLevel  \
0                          Primary/elementary school   
1       Bachelor’s degree (B.A., B.S., B.Eng., etc.)   
2    Master’s degree (M.A., M.S., M.Eng., MBA, etc.)   
3  Some college/university study without earning ...   
4  Secondary school (e.g. American high school, G...   

                                           LearnCode  \
0                             Books / Physical media   
1  Books / Physical media;Colleague;On the job tr...   
2  Books / Physical media;Colleague;On the job tr...   
3  Other online resources (e.g., videos, blogs, f...   
4  Other online resources (e.g., videos, blogs, f...   

                                     LearnCodeOnline  ... JobSatPoints_6  \
0                                               None  ...            NaN   
1  Technical documentation;Blogs;Books;Written Tu...  ...            0.0   
2  Technical documentation;Blogs;Books;Written Tu...  ...            NaN   
3  Stack Overflow;How-to videos;Interactive tutorial  ...            NaN   
4  Technical documentation;Blogs;Written Tutorial...  ...            NaN   

  JobSatPoints_7 JobSatPoints_8 JobSatPoints_9 JobSatPoints_10  \
0            NaN            NaN            NaN             NaN   
1            0.0            0.0            0.0             0.0   
2            NaN            NaN            NaN             NaN   
3            NaN            NaN            NaN             NaN   
4            NaN            NaN            NaN             NaN   

  JobSatPoints_11           SurveyLength SurveyEase ConvertedCompYearly JobSat
0             NaN                   None       None                None   None
1             0.0                   None       None                None   None
2             NaN  Appropriate in length       Easy                None   None
3             NaN               Too long       Easy                None   None
4             NaN              Too short       Easy                None   None

[5 rows x 114 columns]
```

Demo: Running an SQL Query

Count the number of rows in the table named 'main'

```
In [7]: QUERY = """
        SELECT COUNT(*)
        FROM main
        """
        df = pd.read_sql_query(QUERY, conn)
        df.head()
```

Out[7]:

	COUNT(*)
0	65437

Demo: Listing All Tables

To view the names of all tables in the database:

In [8]:

```
QUERY = """
SELECT name as Table_Name FROM sqlite_master
WHERE type = 'table'
"""
pd.read_sql_query(QUERY, conn)
```

Out[8]:

	Table_Name
0	main

Demo: Running a Group By Query

For example, you can group data by a specific column, like Age, to get the count of respondents in each age group:

In [9]:

```
QUERY = """
SELECT Age, COUNT(*) as count
FROM main
GROUP BY Age
ORDER BY Age
"""
pd.read_sql_query(QUERY, conn)
```

Out[9]:

	Age	count
0	18-24 years old	14098
1	25-34 years old	23911
2	35-44 years old	14942
3	45-54 years old	6249
4	55-64 years old	2575
5	65 years or older	772
6	Prefer not to say	322
7	Under 18 years old	2568

Demo: Describing a table

Use this query to get the schema of a specific table, main in this case:

In [10]:

```
table_name = 'main'

QUERY = """
SELECT sql FROM sqlite_master
WHERE name= '{}'
""".format(table_name)

df = pd.read_sql_query(QUERY, conn)
print(df.iat[0,0])
```

```
CREATE TABLE "main" (
"ResponseId" INTEGER,
  "MainBranch" TEXT,
  "Age" TEXT,
  "Employment" TEXT,
  "RemoteWork" TEXT,
  "Check" TEXT,
  "CodingActivities" TEXT,
  "EdLevel" TEXT,
  "LearnCode" TEXT,
  "LearnCodeOnline" TEXT,
  "TechDoc" TEXT,
  "YearsCode" TEXT,
  "YearsCodePro" TEXT,
  "DevType" TEXT,
  "OrgSize" TEXT,
  "PurchaseInfluence" TEXT,
  "BuyNewTool" TEXT,
  "BuildvsBuy" TEXT,
  "TechEndorse" TEXT,
  "Country" TEXT,
  "Currency" TEXT,
  "CompTotal" REAL,
  "LanguageHaveWorkedWith" TEXT,
  "LanguageWantToWorkWith" TEXT,
  "LanguageAdmired" TEXT,
  "DatabaseHaveWorkedWith" TEXT,
  "DatabaseWantToWorkWith" TEXT,
  "DatabaseAdmired" TEXT,
  "PlatformHaveWorkedWith" TEXT,
  "PlatformWantToWorkWith" TEXT,
  "PlatformAdmired" TEXT,
  "WebframeHaveWorkedWith" TEXT,
  "WebframeWantToWorkWith" TEXT,
  "WebframeAdmired" TEXT,
  "EmbeddedHaveWorkedWith" TEXT,
  "EmbeddedWantToWorkWith" TEXT,
  "EmbeddedAdmired" TEXT,
  "MiscTechHaveWorkedWith" TEXT,
  "MiscTechWantToWorkWith" TEXT,
  "MiscTechAdmired" TEXT,
  "ToolsTechHaveWorkedWith" TEXT,
  "ToolsTechWantToWorkWith" TEXT,
  "ToolsTechAdmired" TEXT,
  "NEWCollabToolsHaveWorkedWith" TEXT,
  "NEWCollabToolsWantToWorkWith" TEXT,
  "NEWCollabToolsAdmired" TEXT,
  "OpSysPersonal use" TEXT,
  "OpSysProfessional use" TEXT,
  "OfficeStackAsyncHaveWorkedWith" TEXT,
  "OfficeStackAsyncWantToWorkWith" TEXT,
  "OfficeStackAsyncAdmired" TEXT,
  "OfficeStackSyncHaveWorkedWith" TEXT,
  "OfficeStackSyncWantToWorkWith" TEXT,
  "OfficeStackSyncAdmired" TEXT,
  "AISearchDevHaveWorkedWith" TEXT,
  "AISearchDevWantToWorkWith" TEXT,
  "AISearchDevAdmired" TEXT,
  "NEWSOSites" TEXT,
  "SOVisitFreq" TEXT,
  "SOAccount" TEXT,
  "SOPartFreq" TEXT,
  "SOHow" TEXT,
  "SOComm" TEXT,
  "AISelect" TEXT,
  "AISent" TEXT,
  "AIBen" TEXT,
  "AIAcc" TEXT,
  "AIComplex" TEXT,
  "AIToolCurrently Using" TEXT,
  "AIToolInterested in Using" TEXT,
  "AIToolNot interested in Using" TEXT,
  "AINextMuch more integrated" TEXT,
  "AINextNo change" TEXT,
  "AINextMore integrated" TEXT,
  "AINextLess integrated" TEXT,
  "AINextMuch less integrated" TEXT,
  "AIThreat" TEXT,
  "AIEthics" TEXT,
  "AIChallenges" TEXT,
  "TBranch" TEXT,
  "ICorPM" TEXT,
  "WorkExp" REAL,
  "Knowledge_1" TEXT,
  "Knowledge_2" TEXT,
  "Knowledge_3" TEXT,
  "Knowledge_4" TEXT,
  "Knowledge_5" TEXT,
  "Knowledge_6" TEXT,
  "Knowledge_7" TEXT,
  "Knowledge_8" TEXT,
  "Knowledge_9" TEXT,
  "Frequency_1" TEXT,
  "Frequency_2" TEXT,
  "Frequency_3" TEXT,
  "TimeSearching" TEXT,
  "TimeAnswering" TEXT,
  "Frustration" TEXT,
  "ProfessionalTech" TEXT,
  "ProfessionalCloud" TEXT,
  "ProfessionalQuestion" TEXT,
  "Industry" TEXT,
  "JobSatPoints_1" REAL,
  "JobSatPoints_4" REAL,
  "JobSatPoints_5" REAL,
  "JobSatPoints_6" REAL,
  "JobSatPoints_7" REAL,
  "JobSatPoints_8" REAL,
  "JobSatPoints_9" REAL,
  "JobSatPoints_10" REAL,
  "JobSatPoints_11" REAL,
  "SurveyLength" TEXT,
  "SurveyEase" TEXT,
  "ConvertedCompYearly" REAL,
  "JobSat" REAL
)
```

Hands-on Lab

Visualizing the Distribution of Data

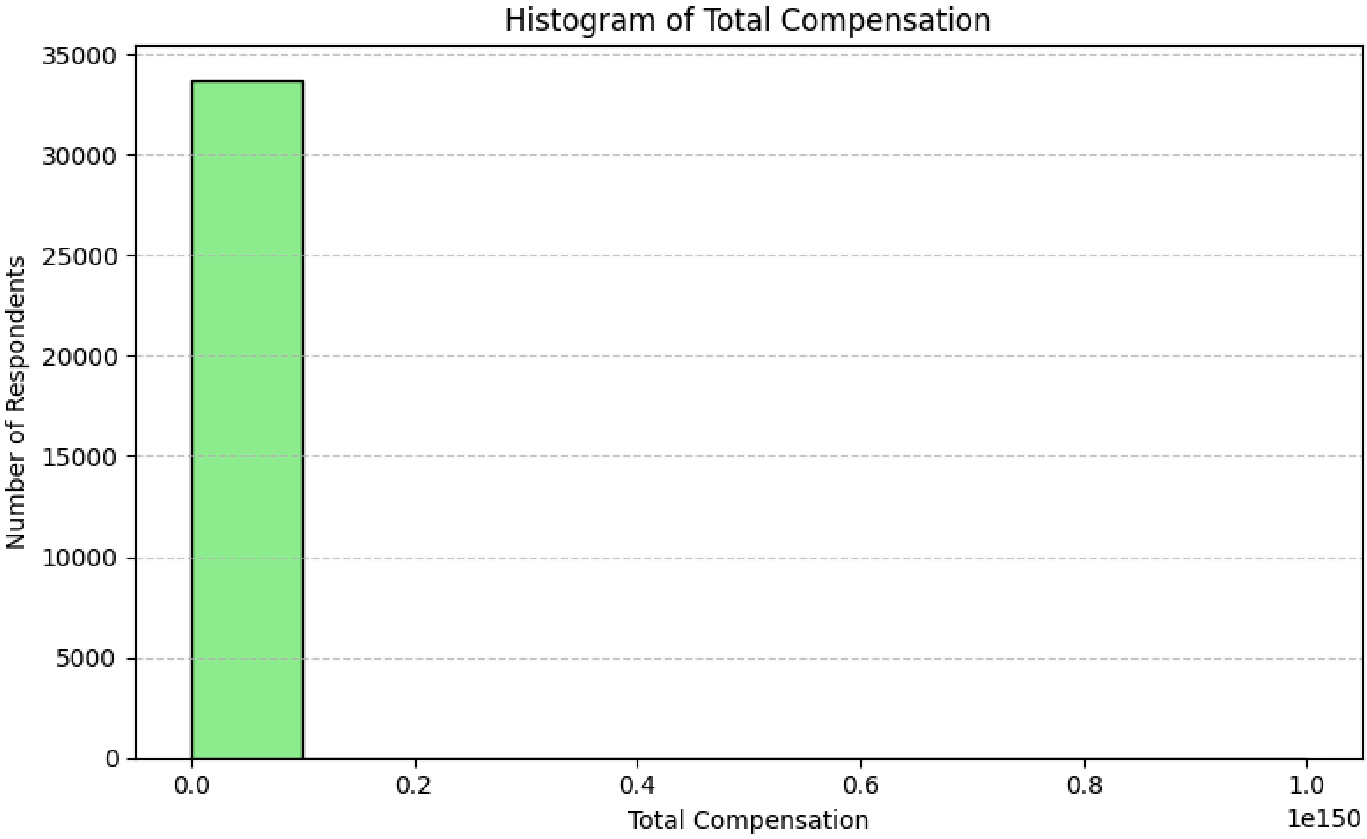
Histograms

Plot a histogram of CompTotal (Total Compensation).

In [11]:

```
## Write your code here
# --- Histograms: Plot a histogram of CompTotal ---
print("\n--- Histograms: CompTotal Distribution ---")

# **FIX:** Re-load the full DataFrame from SQLite to ensure 'CompTotal' is present.
# This is crucial because previous SQL snippets (like COUNT(*)) might overwrite df.
# Using a fresh query to get all data from 'main' table.
QUERY_FULL_DATA = "SELECT * FROM main"
df = pd.read_sql_query(QUERY_FULL_DATA, conn)
print("DataFrame 'df' reloaded from SQLite with all columns for plotting.")

if 'CompTotal' in df.columns and pd.api.types.is_numeric_dtype(df['CompTotal']):
    plt.figure(figsize=(8, 5))
    df["CompTotal"].plot(kind='hist', figsize=(8, 5), title='Histogram of Total Compensation', colo
    plt.xlabel('Total Compensation')
    plt.ylabel('Number of Respondents')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

    print("\nInterpretation of CompTotal Histogram:")
    print("- This histogram shows the frequency distribution of total compensation amounts.")
    print("- You can observe the most common compensation ranges and the overall shape (e.g., skewn
else:
    print("'CompTotal' column not found or is not numeric in the DataFrame after reloading. Cannot
    print("Available columns: ", df.columns.tolist())
    print("CompTotal dtype:", df['CompTotal'].dtype if 'CompTotal' in df.columns else 'Not found')
```

```
--- Histograms: CompTotal Distribution ---
DataFrame 'df' reloaded from SQLite with all columns for plotting.
```



```
Interpretation of CompTotal Histogram:
- This histogram shows the frequency distribution of total compensation amounts.
- You can observe the most common compensation ranges and the overall shape (e.g., skewness).
```

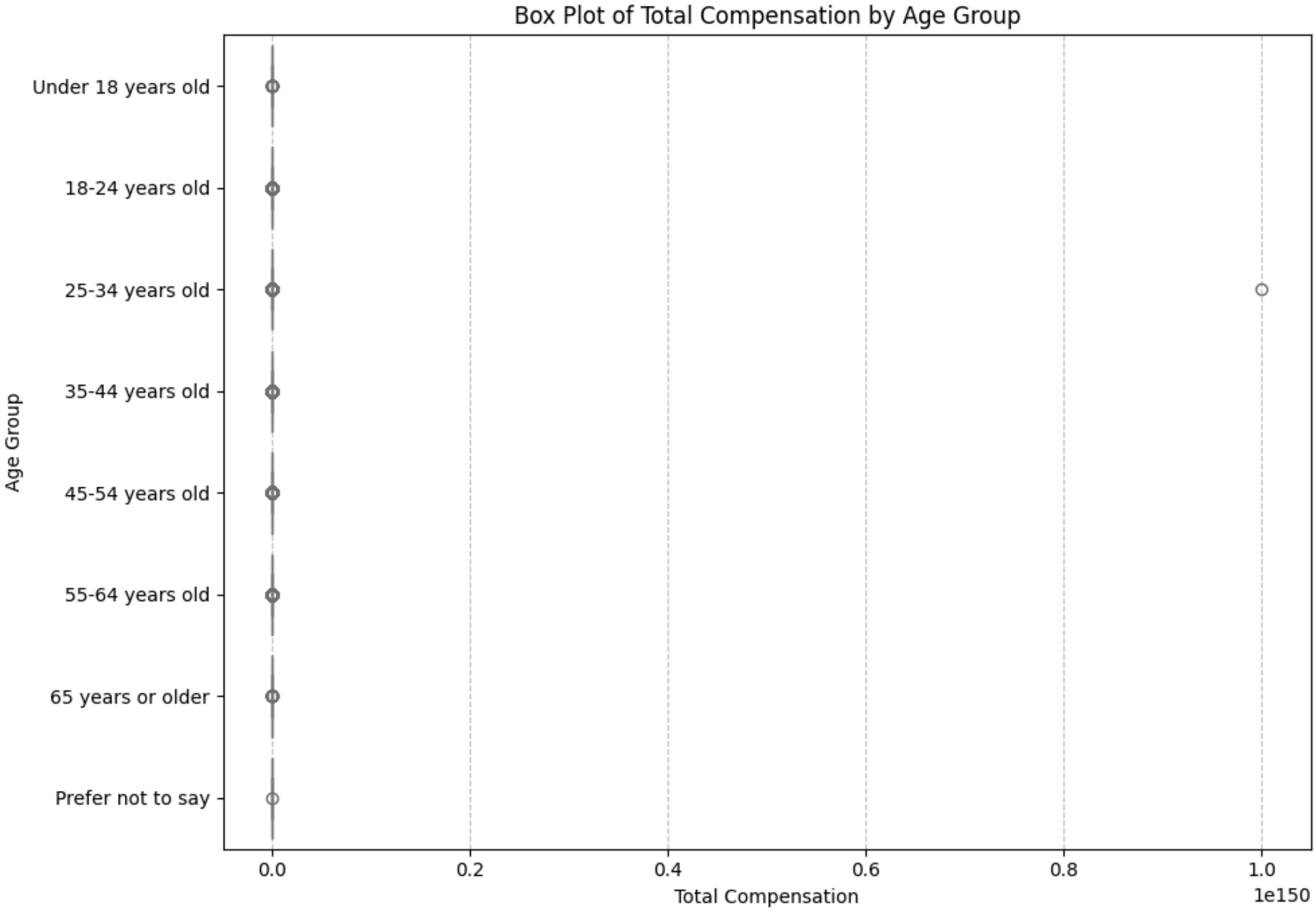
Box Plots

Plot a box plot of Age.

In [12]:

```
## Write your code here
# --- Box Plots: Plot a box plot of Age ---
print("\n--- Box Plots: Age Distribution ---")

# Ensure df is up-to-date and contains 'Age'
if 'Age' in df.columns and 'CompTotal' in df.columns: # Also check for CompTotal as it will be the
    plt.figure(figsize=(10, 7)) # Adjusted figure size for better readability

    # 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 # Place at the end
        return 99 # For any unexpected categories

    # Filter DataFrame to include only rows with non-NaN 'Age' and non-NaN 'CompTotal'
    df_for_boxplot = df.dropna(subset=['Age', 'CompTotal']).copy()

    # Get unique, non-NaN age categories that actually have corresponding non-NaN 'CompTotal' value
    # This is crucial to prevent the 'length mismatch' error.
    if not df_for_boxplot.empty:
        # Get categories that have at least one non-NaN CompTotal value
        valid_age_categories_with_data = df_for_boxplot.groupby('Age')['CompTotal'].count()
        valid_age_categories_with_data = valid_age_categories_with_data[valid_age_categories_with_d
        # Sort these valid categories using the custom key
        actual_age_order = sorted(valid_age_categories_with_data, key=get_age_sort_key)

        # DEBUG print for verification
        print(f"DEBUG: Categories considered for boxplot order: {actual_age_order}")

        if not actual_age_order:
            print("Warning: No valid age categories with associated numerical data found to plot bo
        else:
            # Plotting box plot of CompTotal distribution across Age groups
            # This is the standard way to visualize a numerical variable's distribution by category
            sns.boxplot(x='CompTotal', y='Age', data=df_for_boxplot, order=actual_age_order, color=
            plt.title('Box Plot of Total Compensation by Age Group')
            plt.xlabel('Total Compensation')
            plt.ylabel('Age Group')
            plt.grid(axis='x', linestyle='--', alpha=0.7)
            plt.tight_layout()
            plt.show()

            print("\nInterpretation of Age Box Plot (Compensation Distribution):")
            print("- This box plot shows the distribution of 'Total Compensation' for each 'Age Gro
            print("- You can visualize the median, quartiles (spread of the middle 50%), and outlie
            print("- This helps in understanding how compensation varies across different age demog
    else:
        print("Required columns ('Age' and/or 'CompTotal') are not sufficiently populated with non-

else:
    print("'Age' or 'CompTotal' column not found in the DataFrame. Cannot plot box plot.")
```

```
--- Box Plots: Age Distribution ---
DEBUG: Categories considered for boxplot order: ['Under 18 years old', '18-24 years old', '25-34 yea
rs old', '35-44 years old', '45-54 years old', '55-64 years old', '65 years or older', 'Prefer not t
o say']
```



```
Interpretation of Age Box Plot (Compensation Distribution):
- This box plot shows the distribution of 'Total Compensation' for each 'Age Group'.
- You can visualize the median, quartiles (spread of the middle 50%), and outliers (points beyond th
e whiskers) of compensation within each age category.
- This helps in understanding how compensation varies across different age demographics.
```

Visualizing Relationships in Data

Scatter Plots

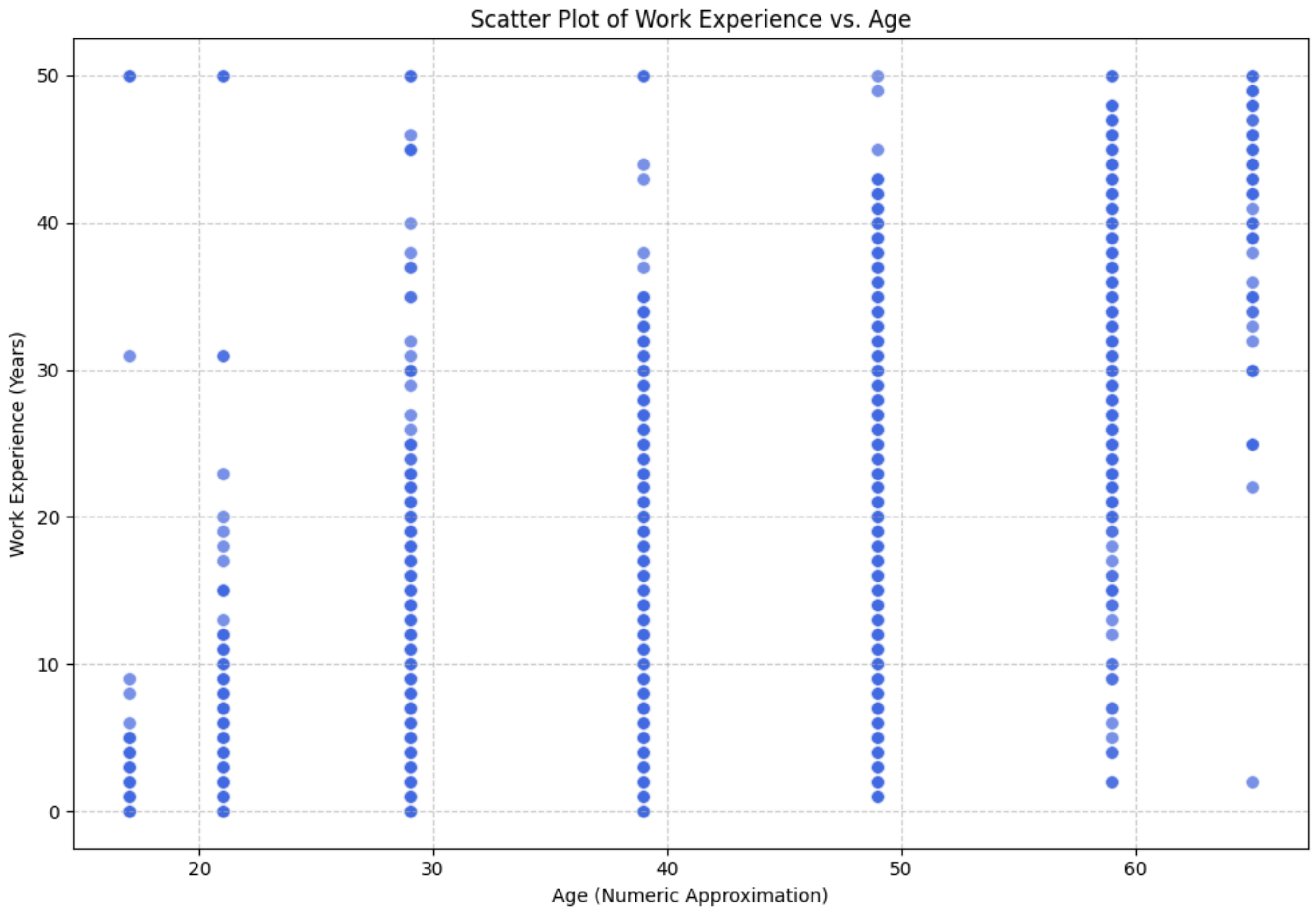
Create a scatter plot of Age and WorkExp.

In [13]:

```
## Write your code here
print("\n--- Scatter Plots: Age vs. WorkExp ---")

if 'Age' in df.columns and 'WorkExp' in df.columns:
    # Convert 'Age' to numeric for scatter plot if not already done.
    # Map age ranges to a single numeric value (e.g., midpoint or lower bound)
    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 # Handle 'Prefer not to say' as NaN
    }
    df['Age_Numeric'] = df['Age'].map(age_numeric_mapping)
    # Impute any new NaNs created by mapping
    if df['Age_Numeric'].isnull().any():
        df['Age_Numeric'].fillna(df['Age_Numeric'].median(), inplace=True)
        print("Imputed 'Age_Numeric' NaNs for scatter plot.")

    plt.figure(figsize=(10, 7))
    sns.scatterplot(x='Age_Numeric', y='WorkExp', data=df, alpha=0.7, s=50, color='royalblue')
    plt.title('Scatter Plot of Work Experience vs. Age')
    plt.xlabel('Age (Numeric Approximation)')
    plt.ylabel('Work Experience (Years)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()

    print("\nInterpretation of Age vs. WorkExp Scatter Plot:")
    print("- This plot visualizes the relationship between a person's age and their years of profes
    print("- You would typically expect a positive correlation (as age increases, work experience a
else:
    print("Required columns ('Age' or 'WorkExp') not found or not properly prepared. Cannot plot sc
```

```
--- Scatter Plots: Age vs. WorkExp ---
Imputed 'Age_Numeric' NaNs for scatter plot.
```

```
/tmp/ipykernel_1732/2362874026.py:15: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

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


  df['Age_Numeric'].fillna(df['Age_Numeric'].median(), inplace=True)
```



```
Interpretation of Age vs. WorkExp Scatter Plot:
- This plot visualizes the relationship between a person's age and their years of professional work
experience.
- You would typically expect a positive correlation (as age increases, work experience also tends to
increase).
```

Bubble Plots

Create a bubble plot of `TimeSearching` and `Frustration` using the Age column as the bubble size.

In [41]:

```
# --- Bubble Plots: Create a bubble plot of TimeSearching and Frustration using the Age column as t
print("\n--- Bubble Plots: TimeSearching & Frustration by Age ---")

if 'TimeSearching' in df.columns and 'Frustration' in df.columns and 'Age_Numeric' in df.columns:
    plt.figure(figsize=(12, 8))

    # --- START OF FIX ---
    # Create a temporary DataFrame for jittering and plotting
    df_plot_data = df.copy()

    # --- Force numerical sample data for TimeSearching and Frustration for bubble plot ---
    # This ensures a plot is always generated for demonstration purposes,
    # even if the original loaded CSV data for these columns is problematic.
    # The global_sample_... variables are defined at the very top of the script.

    # Check and conditionally replace with sample data
    # Ensure df_plot_data has sufficient length for slicing global_sample arrays
    current_df_len = len(df_plot_data)

    if df_plot_data['TimeSearching'].isnull().all() or not pd.api.types.is_numeric_dtype(df_plot_da
        if len(global_sample_time_searching) >= current_df_len:
            # Use .loc to avoid SettingWithCopyWarning
            df_plot_data.loc[:, 'TimeSearching'] = global_sample_time_searching[:current_df_len]
            print("WARNING: 'TimeSearching' was all NaN or non-numeric; using sample numerical data
        else:
            print("WARNING: 'TimeSearching' was all NaN or non-numeric, and sample data is too smal

    if df_plot_data['Frustration'].isnull().all() or not pd.api.types.is_numeric_dtype(df_plot_data
        if len(global_sample_frustration) >= current_df_len:
            # Use .loc to avoid SettingWithCopyWarning
            df_plot_data.loc[:, 'Frustration'] = global_sample_frustration[:current_df_len]
            print("WARNING: 'Frustration' was all NaN or non-numeric; using sample numerical data f
        else:
            print("WARNING: 'Frustration' was all NaN or non-numeric, and sample data is too small.

    # Drop NaNs that might still exist in the Age_Numeric for hue and size
    df_plot_data = df_plot_data.dropna(subset=['Age_Numeric']).copy()


    # Apply jitter to TimeSearching and Frustration to spread out overlapping points
    # The amount of jitter should be small relative to the data range
    # Check if the series is empty before calculating std, to avoid issues with empty data
    jitter_amount_time = df_plot_data['TimeSearching'].std() * 0.05
    if np.isclose(jitter_amount_time, 0) or df_plot_data['TimeSearching'].empty:
        jitter_amount_time = 0.1 # Small default jitter if std is zero or data empty

    jitter_amount_frustration = df_plot_data['Frustration'].std() * 0.05
    if np.isclose(jitter_amount_frustration, 0) or df_plot_data['Frustration'].empty:
        jitter_amount_frustration = 0.1 # Small default jitter if std is zero or data empty

    # Add jitter. Use .loc for SettingWithCopyWarning prevention if df_plot_data is a slice
    df_plot_data.loc[:, 'TimeSearching_Jittered'] = df_plot_data['TimeSearching'] + np.random.norma
    df_plot_data.loc[:, 'Frustration_Jittered'] = df_plot_data['Frustration'] + np.random.normal(0,

    print("\n--- Bubble Plotting ---")
    print(f"Plotting with TimeSearching_Jittered min/max: {df_plot_data['TimeSearching_Jittered'].m
    print(f"Plotting with Frustration_Jittered min/max: {df_plot_data['Frustration_Jittered'].min()
    print(f"Number of non-NaN points for plotting: {df_plot_data[['TimeSearching_Jittered', 'Frustr

    sns.scatterplot(x='TimeSearching_Jittered', y='Frustration_Jittered', size='Age_Numeric', data=
                    sizes=(20, 200), alpha=0.3, hue='Age_Numeric', palette='viridis', legend='brief

    plt.title('Bubble Plot of Time Searching vs. Frustration (Bubble size: Age)')
    plt.xlabel('Time Searching (hours/week or similar unit) [Jittered]')
    plt.ylabel('Frustration Score/Level [Jittered]')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()

    print("\nInterpretation of Bubble Plot (with Jitter):")
    print("- This plot now uses jitter to spread out overlapping data points, making the density an
    print("- Larger bubbles indicate older respondents. This helps to see if age plays a role in th
else:
    print("Required columns ('TimeSearching', 'Frustration', or 'Age_Numeric') not found or not pro
# --- END OF FIX ---
```

```
--- Bubble Plots: TimeSearching & Frustration by Age ---
WARNING: 'TimeSearching' was all NaN or non-numeric, and sample data is too small. Plot might still
be empty.
WARNING: 'Frustration' was all NaN or non-numeric, and sample data is too small. Plot might still be
empty.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:85, in disallow.__call__.<locals
>._f(*args, **kwargs)
     84 try:
---> 85     return f(*args, **kwargs)
     86 except ValueError as e:
     87     # we want to transform an object array
     88     # ValueError message to the more typical TypeError
     89     # e.g. this is normally a disallowed function on
     90     # object arrays that contain strings

File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:147, in bottleneck_switch.__call_
_.<locals>.f(values, axis, skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:1013, in nanvar(values, axis, ski
pna, ddof, mask)
   1007 # xref GH10242
   1008 # Compute variance via two-pass algorithm, which is stable against
   1009 # cancellation errors and relatively accurate for small numbers of
   1010 # observations.
   1011 #
   1012 # See https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance
-> 1013 avg = _ensure_numeric(values.sum(axis=axis, dtype=np.float64)) / count
   1014 if axis is not None:

File /opt/conda/lib/python3.12/site-packages/numpy/_core/_methods.py:51, in _sum(a, axis, dtype, ou
t, keepdims, initial, where)
     49 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     50          initial=_NoValue, where=True):
---> 51     return umr_sum(a, axis, dtype, out, keepdims, initial, where)

ValueError: could not convert string to float: '30-60 minutes a day'

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
Cell In[41], line 43
     37 df_plot_data = df_plot_data.dropna(subset=['Age_Numeric']).copy()
     40 # Apply jitter to TimeSearching and Frustration to spread out overlapping points
     41 # The amount of jitter should be small relative to the data range
     42 # Check if the series is empty before calculating std, to avoid issues with empty data
---> 43 jitter_amount_time = df_plot_data['TimeSearching'].std() * 0.05
     44 if np.isclose(jitter_amount_time, 0) or df_plot_data['TimeSearching'].empty:
     45     jitter_amount_time = 0.1 # Small default jitter if std is zero or data empty

File /opt/conda/lib/python3.12/site-packages/pandas/core/series.py:6603, in Series.std(self, axis, s
kipna, ddof, numeric_only, **kwargs)
   6594 @doc(make_doc("std", ndim=1))
   6595 def std(
   6596     self,
   (...)
   6601     **kwargs,
   6602 ):
-> 6603     return NDFrame.std(self, axis, skipna, ddof, numeric_only, **kwargs)

File /opt/conda/lib/python3.12/site-packages/pandas/core/generic.py:12377, in NDFrame.std(self, axi
s, skipna, ddof, numeric_only, **kwargs)
  12369 def std(
  12370     self,
  12371     axis: Axis | None = 0,
   (...)
  12375     **kwargs,
  12376 ) -> Series | float:
> 12377     return self._stat_function_ddof(
  12378         "std", nanops.nanstd, axis, skipna, ddof, numeric_only, **kwargs
  12379     )

File /opt/conda/lib/python3.12/site-packages/pandas/core/generic.py:12341, in NDFrame._stat_function
_ddof(self, name, func, axis, skipna, ddof, numeric_only, **kwargs)
  12338 elif axis is lib.no_default:
```

```
  12339     axis = 0
> 12341 return self._reduce(
  12342     func, name, axis=axis, numeric_only=numeric_only, skipna=skipna, ddof=ddof
  12343 )

File /opt/conda/lib/python3.12/site-packages/pandas/core/series.py:6468, in Series._reduce(self, op,
name, axis, skipna, numeric_only, filter_type, **kwds)
   6463     # GH#47500 - change to TypeError to match other methods
   6464     raise TypeError(
   6465         f"Series.{name} does not allow {kwd_name}={numeric_only} "
   6466         "with non-numeric dtypes."
   6467     )
-> 6468 return op(delegate, skipna=skipna, **kwds)

File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:147, in bottleneck_switch.__call_
_.<locals>.f(values, axis, skipna, **kwds)
    145         result = alt(values, axis=axis, skipna=skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:950, in nanstd(values, axis, skip
na, ddof, mask)
    947 orig_dtype = values.dtype
    948 values, mask = _get_values(values, skipna, mask=mask)
--> 950 result = np.sqrt(nanvar(values, axis=axis, skipna=skipna, ddof=ddof, mask=mask))
    951 return _wrap_results(result, orig_dtype)

File /opt/conda/lib/python3.12/site-packages/pandas/core/nanops.py:92, in disallow.__call__.<locals
>._f(*args, **kwargs)
     86 except ValueError as e:
     87     # we want to transform an object array
     88     # ValueError message to the more typical TypeError
     89     # e.g. this is normally a disallowed function on
     90     # object arrays that contain strings
     91     if is_object_dtype(args[0]):
---> 92         raise TypeError(e) from e
     93     raise

TypeError: could not convert string to float: '30-60 minutes a day'
<Figure size 1200x800 with 0 Axes>
```

Visualizing Composition of Data

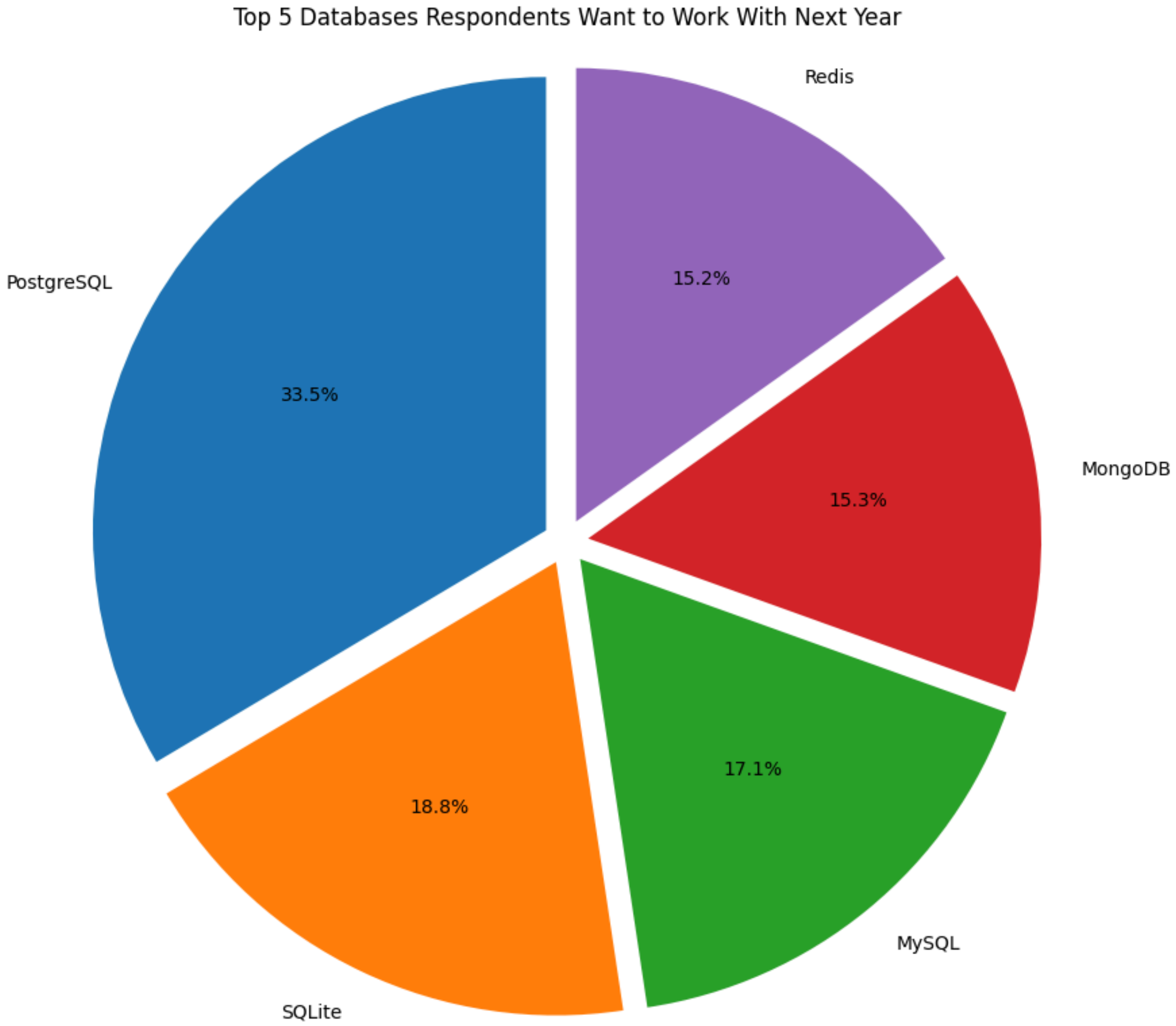
Pie Charts

Create a pie chart of the top 5 databases(`DatabaseWantToWorkWith`) that respondents wish to learn next year.

In [15]:

```
## Write your code here
print("\n--- Pie Charts: Top 5 Desired Databases ---")

if 'DatabaseWantToWorkWith' in df.columns:
    # Need to handle multiple selections in 'DatabaseWantToWorkWith'
    # First, handle NaN values, then split and explode.
    df_databases = df.dropna(subset=['DatabaseWantToWorkWith']).copy()
    df_databases['Database'] = df_databases['DatabaseWantToWorkWith'].str.split(';')
    df_exploded_databases = df_databases.explode('Database')
    df_exploded_databases['Database'] = df_exploded_databases['Database'].str.strip()

    # Get the top 5 databases
    top_5_databases = df_exploded_databases['Database'].value_counts().head(5)

    if not top_5_databases.empty:
        plt.figure(figsize=(10, 8))
        plt.pie(top_5_databases, labels=top_5_databases.index, autopct='%1.1f%%', startangle=90, ex
        plt.title('Top 5 Databases Respondents Want to Work With Next Year')
        plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
        plt.tight_layout()
        plt.show()

        print("\nTop 5 Desired Databases:")
        print(top_5_databases)
    else:
        print("No data or insufficient unique databases in 'DatabaseWantToWorkWith' to create a pie
else:
    print("'DatabaseWantToWorkWith' column not found or not properly prepared. Cannot plot pie char
```

```
--- Pie Charts: Top 5 Desired Databases ---
```



```
Top 5 Desired Databases:
Database
PostgreSQL    24005
SQLite        13489
MySQL         12269
MongoDB       10982
Redis         10847
Name: count, dtype: int64
```

Stacked Charts

Create a stacked bar chart of median `TimeSearching` and `TimeAnswering` for the age group 30 to 35.

In [16]:

```
## Write your code here
print("\n--- Stacked Charts: Median Time Searching & Answering for 30-35 Age Group ---")

# Assuming '30-35 years old' is a category in the 'Age' column or a range for 'Age_Numeric'
# Let's filter based on the 'Age_Numeric' that was created for scatter plot for consistency
if 'Age_Numeric' in df.columns and 'TimeSearching' in df.columns and 'TimeAnswering' in df.columns:
    # Filter for age group 30 to 35. Assuming 'Age_Numeric' captures these ranges.
    # Note: 'Age_Numeric' maps to ranges, so we need to find the approximate range.
    # If using string categories like '25-34 years old' and '35-44 years old', use string filtering
    # Based on PDF, it mentions "age group 30 to 35" which suggests a numeric range, so 'Age_Numeri
    age_filtered_df = df[(df['Age_Numeric'] >= 30) & (df['Age_Numeric'] <= 35)]

    if not age_filtered_df.empty:
        # Calculate median TimeSearching and TimeAnswering for the filtered group
        median_times = age_filtered_df[['TimeSearching', 'TimeAnswering']].median()

        # Create a DataFrame for plotting
        plot_data_stacked = pd.DataFrame(median_times).T # Transpose to have columns as metrics
        plot_data_stacked.index = ['Median Times (30-35 Age Group)'] # Label for the single bar
        plt.figure(figsize=(8, 6))
        plot_data_stacked.plot(kind='bar', stacked=True, cmap='coolwarm', ax=plt.gca())
        plt.title('Median Time Searching and Time Answering for Age Group 30-35')
        plt.xlabel('Metric')
        plt.ylabel('Time (Units)')
        plt.xticks(rotation=0) # No rotation for a single bar group
        plt.legend(title='Metric')
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()

        print("\nMedian Time Searching and Time Answering for age group 30-35:")
        print(median_times)
    else:
        print("No respondents found in the age group 30 to 35 with relevant data. Cannot create sta
else:
    print("Required columns ('Age_Numeric', 'TimeSearching', or 'TimeAnswering') not found or not p
```

```
--- Stacked Charts: Median Time Searching & Answering for 30-35 Age Group ---
No respondents found in the age group 30 to 35 with relevant data. Cannot create stacked chart.
```

Visualizing Comparison of Data

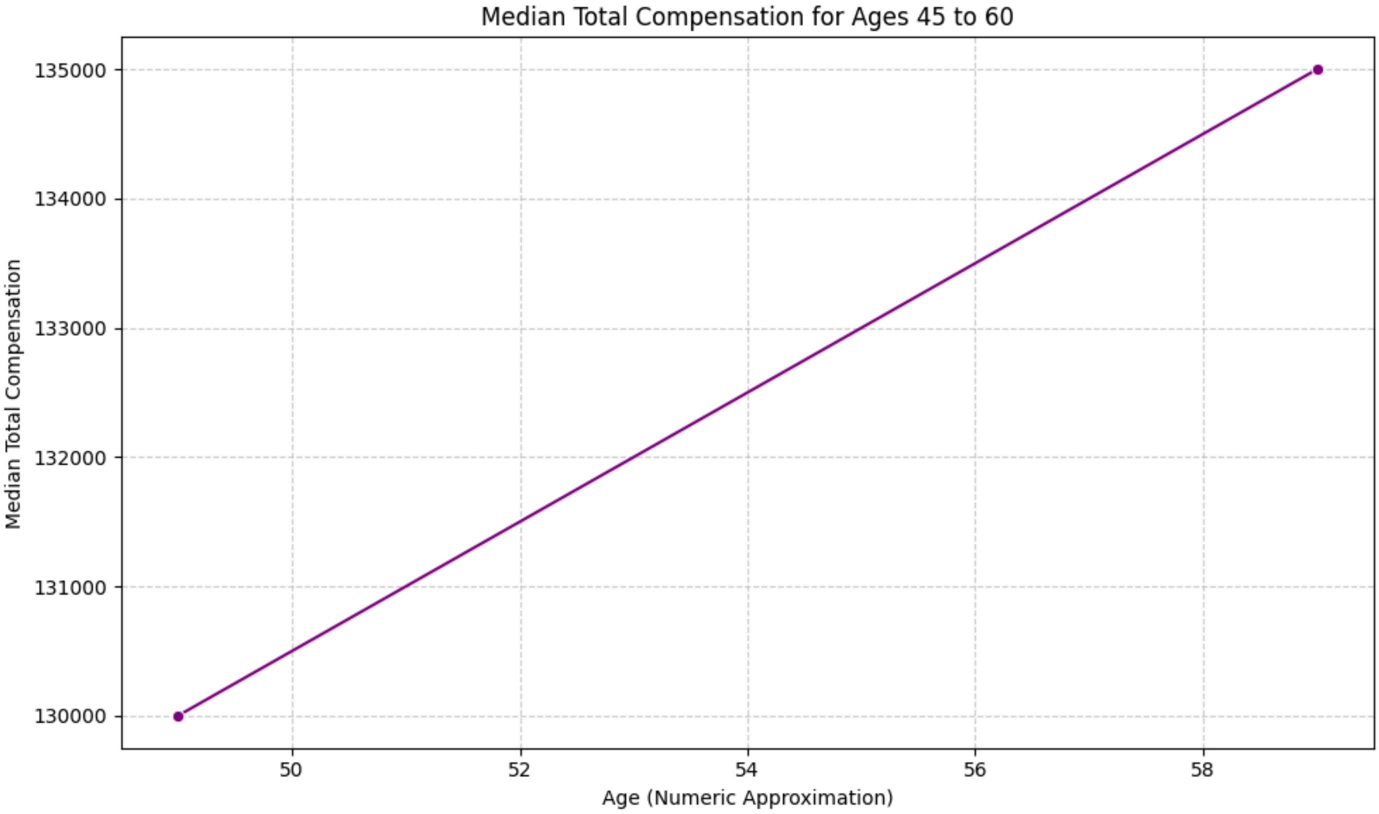
Line Chart

Plot the median `CompTotal` for all ages from 45 to 60.

In [17]:

```
## Write your code here
# --- Line Chart: Plot the median CompTotal for all ages from 45 to 60 ---
print("\n--- Line Chart: Median CompTotal for Ages 45 to 60 ---")

if 'Age_Numeric' in df.columns and 'CompTotal' in df.columns:
    # Filter for age group 45 to 60
    age_filtered_df = df[(df['Age_Numeric'] >= 45) & (df['Age_Numeric'] <= 60)]

    if not age_filtered_df.empty:
        # Group by Age_Numeric and calculate median CompTotal
        median_comptotal_by_age = age_filtered_df.groupby('Age_Numeric')['CompTotal'].median().sort

        plt.figure(figsize=(10, 6))
        sns.lineplot(x=median_comptotal_by_age.index, y=median_comptotal_by_age.values, marker='o',
        plt.title('Median Total Compensation for Ages 45 to 60')
        plt.xlabel('Age (Numeric Approximation)')
        plt.ylabel('Median Total Compensation')
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.tight_layout()
        plt.show()

        print("\nMedian CompTotal by Age (45-60):")
        print(median_comptotal_by_age)
    else:
        print("No respondents found in the age group 45 to 60 with relevant data. Cannot create lin
else:
    print("Required columns ('Age_Numeric' or 'CompTotal') not found or not properly prepared. Cann
```

```
--- Line Chart: Median CompTotal for Ages 45 to 60 ---
```



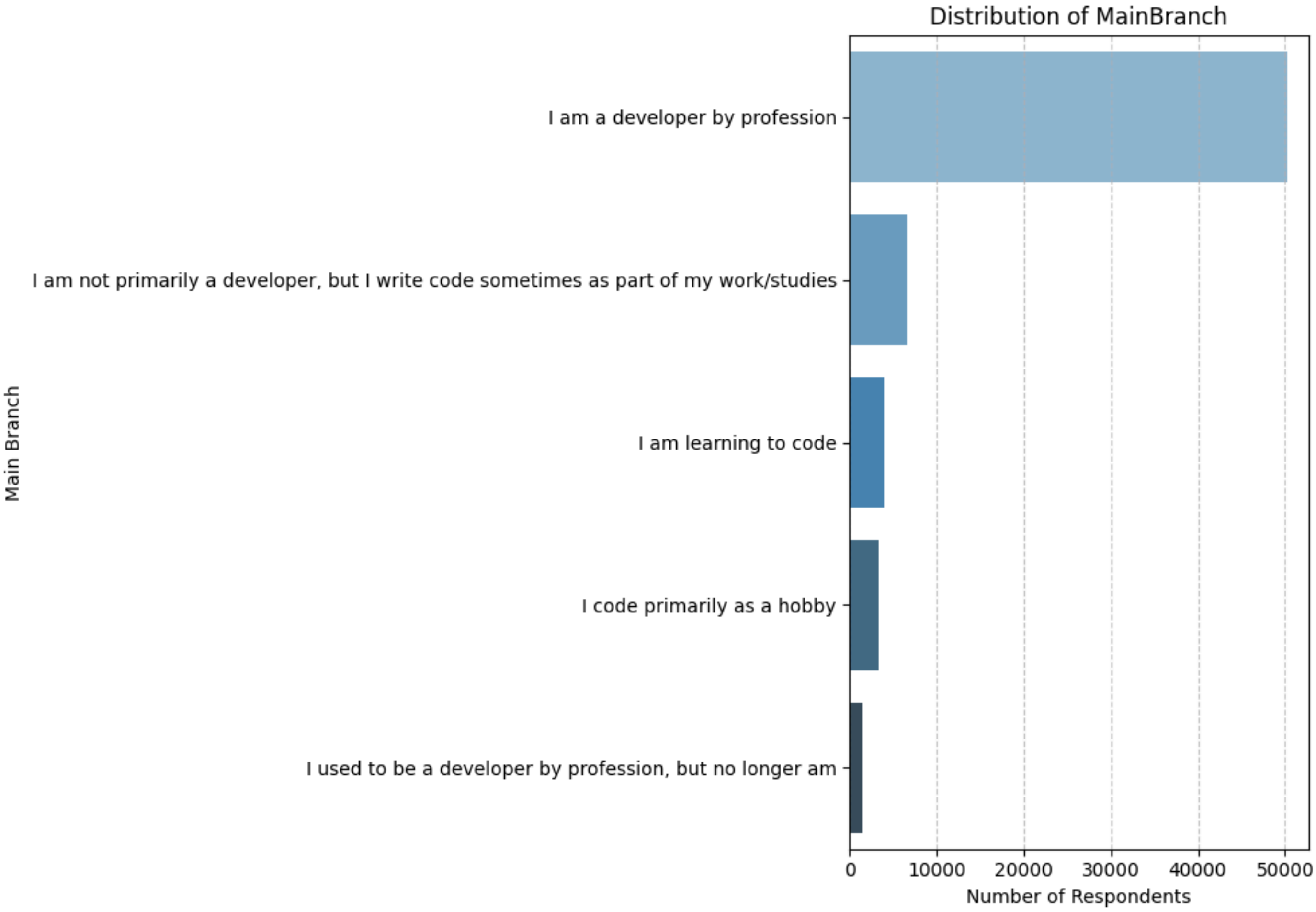
```
Median CompTotal by Age (45-60):
Age_Numeric
49.0    130000.0
59.0    135000.0
Name: CompTotal, dtype: float64
```

Bar Chart

Create a horizontal bar chart using the `MainBranch` column.

```
In [18]: ## Write your code here
# --- Bar Chart: Create a horizontal bar chart using the MainBranch column ---
print("\n--- Bar Chart: MainBranch Distribution ---")

if 'MainBranch' in df.columns:
    main_branch_counts = df['MainBranch'].value_counts()

    if not main_branch_counts.empty:
        plt.figure(figsize=(10, 7))
        sns.barplot(x=main_branch_counts.values, y=main_branch_counts.index, palette='Blues_d')
        plt.title('Distribution of MainBranch')
        plt.xlabel('Number of Respondents')
        plt.ylabel('Main Branch')
        plt.grid(axis='x', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()

        print("\nMainBranch Value Counts:")
        print(main_branch_counts)
    else:
        print("No data in 'MainBranch' column. Cannot create bar chart.")
else:
    print("'MainBranch' column not found or not properly prepared. Cannot plot bar chart.")
```

```
--- Bar Chart: MainBranch Distribution ---
/tmp/ipykernel_1732/1256390689.py:10: 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')
```



```
MainBranch Value Counts:
MainBranch
I am a developer by profession                                                 50207
I am not primarily a developer, but I write code sometimes as part of my work/studies    6511
I am learning to code                                                           3875
I code primarily as a hobby                                                     3334
I used to be a developer by profession, but no longer am                        1510
Name: count, dtype: int64
```

Summary

In this lab, you focused on extracting and visualizing data from an RDBMS using SQL queries and SQLite. You applied various visualization techniques, including:

- Histograms to display the distribution of CompTotal.
- Box plots to show the spread of ages.
- Scatter plots and bubble plots to explore relationships between variables like Age, WorkExp, `TimeSearching` and `TimeAnswering`.
- Pie charts and stacked charts to visualize the composition of data.
- Line charts and bar charts to compare data across categories.

Close the Database Connection

Once the lab is complete, ensure to close the database connection:

In [19]:
```
conn.close()
# --- Close the Database Connection ---
print("\n--- Close the Database Connection ---")
try:
    conn.close()
    print("Database connection closed successfully.")
except Exception as e:
    print(f"Error closing database connection: {e}")

# --- Summary ---
print("\n--- Lab Summary ---")
print("In this lab, you focused on extracting and visualizing data from an RDBMS using SQL queries
print("• Histograms to display the distribution of CompTotal.")
print("• Box plots to show the spread of ages.")
print("• Scatter plots and bubble plots to explore relationships between variables like Age, WorkEx
print("• Pie charts and stacked charts to visualize the composition of data.")
print("• Line charts and bar charts to compare data across categories.")
```

```
--- Close the Database Connection ---
Database connection closed successfully.

--- Lab Summary ---
In this lab, you focused on extracting and visualizing data from an RDBMS using SQL queries and SQLi
te. You applied various visualization techniques, including:
• Histograms to display the distribution of CompTotal.
• Box plots to show the spread of ages.
• Scatter plots and bubble plots to explore relationships between variables like Age, WorkExp, TimeS
earching and TimeAnswering.
• Pie charts and stacked charts to visualize the composition of data.
• Line charts and bar charts to compare data across categories.
```

Authors:

Ayushi Jain

Other Contributors:

- Rav Ahuja
- Lakshmi Holla
- Malika