In [1]:
```
from google.cloud import bigquery
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.cloud import bigquery
```

```
/home/kryala/myenv/lib/python3.11/site-packages/matplotlib/projections/__in
it__.py:63: UserWarning: Unable to import Axes3D. This may be due to multip
le versions of Matplotlib being installed (e.g. as a system package and as
a pip package). As a result, the 3D projection is not available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versi
ons of "
```

In [3]:
```
import os
from google.cloud import bigquery

# Set the environment variable for the Google Cloud service account key
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/home/kryala/mgmtfinal-42061

# Create a BigQuery client
client = bigquery.Client()

# Now you can use the client to interact with BigQuery
```

In [4]:
```
# List datasets in the project
print(list(client.list_datasets()))
```

```
[<google.cloud.bigquery.dataset.DatasetListItem object at 0x7f3f51d7bc90>]
```

In [5]:
```
# List datasets in the project and print their IDs
datasets = list(client.list_datasets())
if datasets:
    print("Datasets in project {}:".format(client.project))
    for dataset in datasets:
        print("\t{}".format(dataset.dataset_id))
else:
    print("{} project does not contain any datasets.".format(client.project)
```

```
Datasets in project mgmtfinal-420614:
        cdc_analysis
```

In [6]:
```
# Replace 'your_dataset' with the actual dataset ID
dataset_id = 'cdc_analysis'
tables = client.list_tables(dataset_id)

print("Tables contained in '{}':".format(dataset_id))
for table in tables:
    print("\t{}".format(table.table_id))
```

```
Tables contained in 'cdc_analysis':
        cdc_table
```

In [8]:
```
!pip install db-dtypes
```

```
Requirement already satisfied: db-dtypes in /home/kryala/myenv/lib/python3.
11/site-packages (1.2.0)
Requirement already satisfied: packaging>=17.0 in /home/kryala/myenv/lib/py
thon3.11/site-packages (from db-dtypes) (24.0)
Requirement already satisfied: pandas>=0.24.2 in /home/kryala/myenv/lib/pyt
hon3.11/site-packages (from db-dtypes) (2.2.2)
Requirement already satisfied: pyarrow>=3.0.0 in /home/kryala/myenv/lib/pyt
hon3.11/site-packages (from db-dtypes) (15.0.2)
Requirement already satisfied: numpy>=1.16.6 in /home/kryala/myenv/lib/pyth
on3.11/site-packages (from db-dtypes) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/kryala/myen
v/lib/python3.11/site-packages (from pandas>=0.24.2->db-dtypes) (2.9.0.post
0)
Requirement already satisfied: pytz>=2020.1 in /home/kryala/myenv/lib/pytho
n3.11/site-packages (from pandas>=0.24.2->db-dtypes) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/kryala/myenv/lib/pyt
hon3.11/site-packages (from pandas>=0.24.2->db-dtypes) (2024.1)
Requirement already satisfied: six>=1.5 in /home/kryala/myenv/lib/python3.1
1/site-packages (from python-dateutil>=2.8.2->pandas>=0.24.2->db-dtypes)
(1.16.0)
```

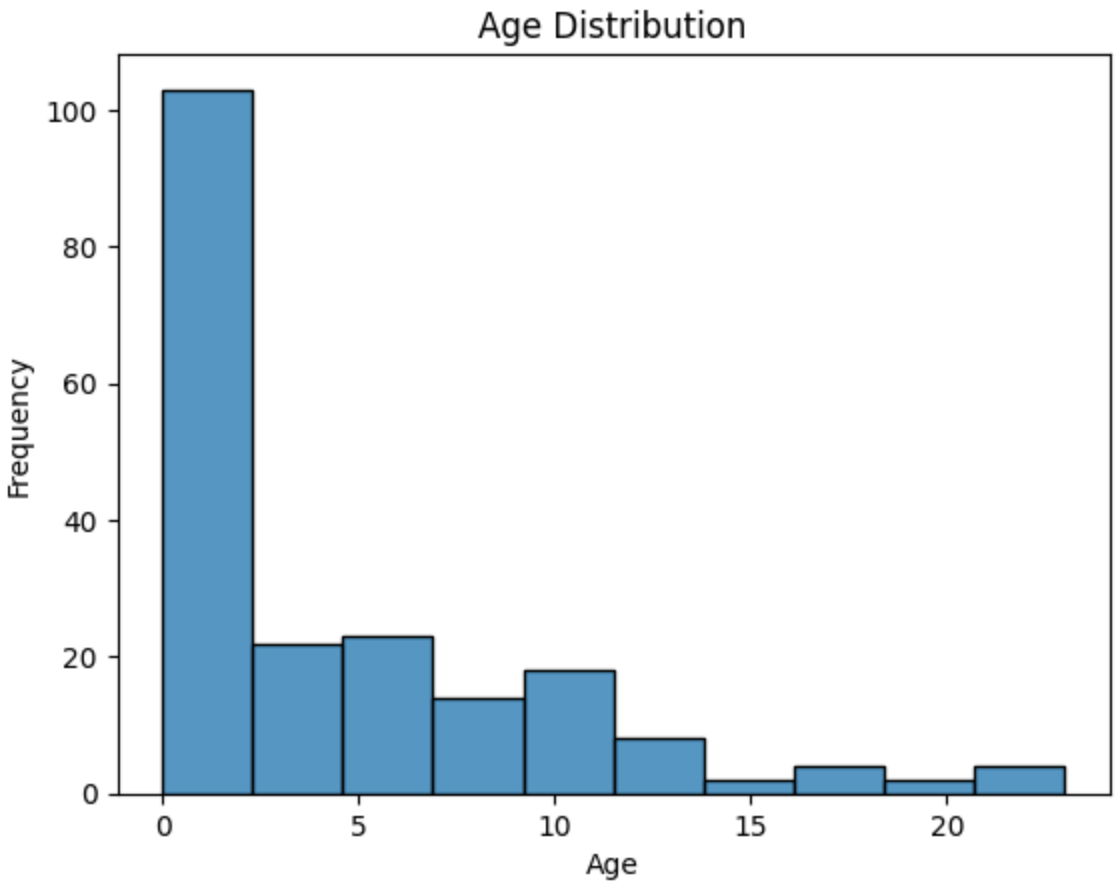
In [47]:

```
from google.cloud import bigquery

# Initialize a BigQuery client
client = bigquery.Client()

# Replace 'your_dataset' and 'your_table' with the actual dataset and table
query = """
SELECT *
FROM `mgmtfinal-420614.cdc_analysis.cdc_table`
LIMIT 200
"""
query_job = client.query(query)  # Start the query job
df = query_job.to_dataframe()  # Convert the results to a pandas DataFrame

# Now you can display the first few rows of the DataFrame
df.head()
```

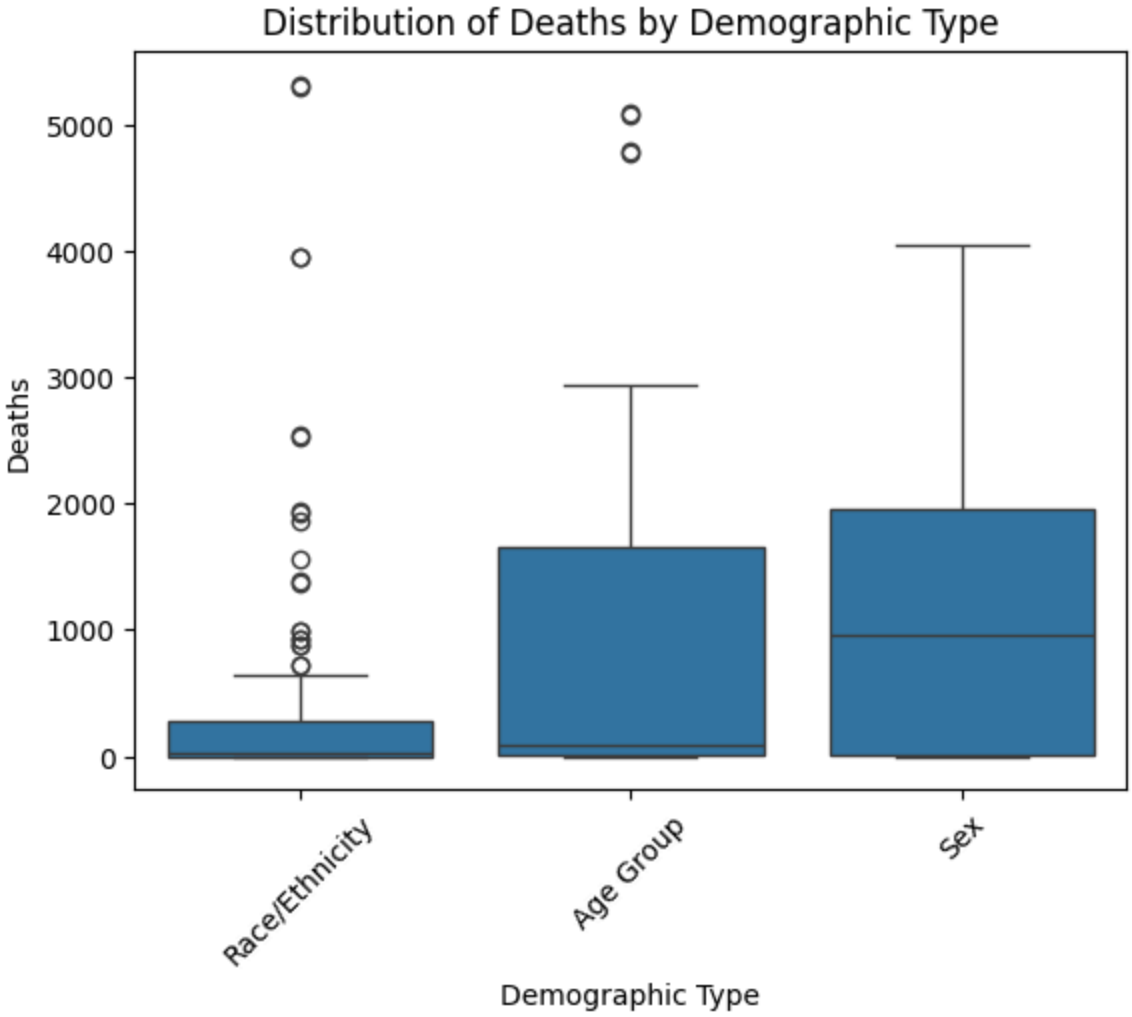
Out[47]:

	data_as_of	start_date	end_date	group	year	month	mmwr_week	weekending_date	sta
0	2024-04-12	2021-11-07	2021-11-13	By Week	2021	11	45	2021-11-13	Unite State
1	2024-04-12	2021-11-07	2021-11-13	By Week	2021	11	45	2021-11-13	Unite State
2	2024-04-12	2021-11-07	2021-11-13	By Week	2021	11	45	2021-11-13	Unite State
3	2024-04-12	2021-11-07	2021-11-13	By Week	2021	11	45	2021-11-13	Unite State
4	2024-04-12	2021-11-07	2021-11-13	By Week	2021	11	45	2021-11-13	Unite State

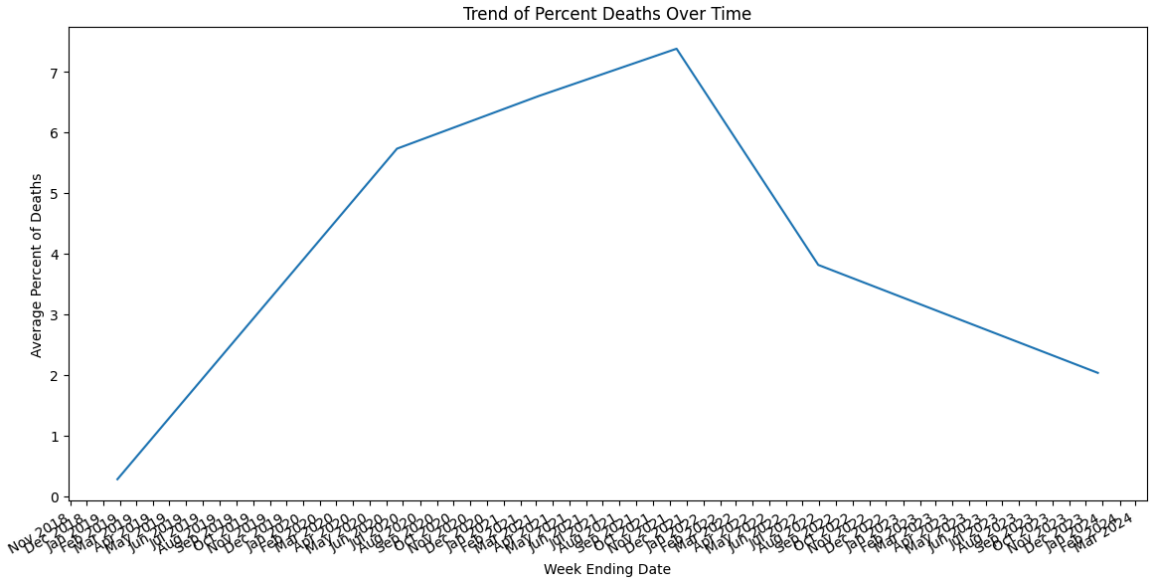
In [48]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'age' is a numeric column in your DataFrame
sns.histplot(df['percent_deaths'])
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



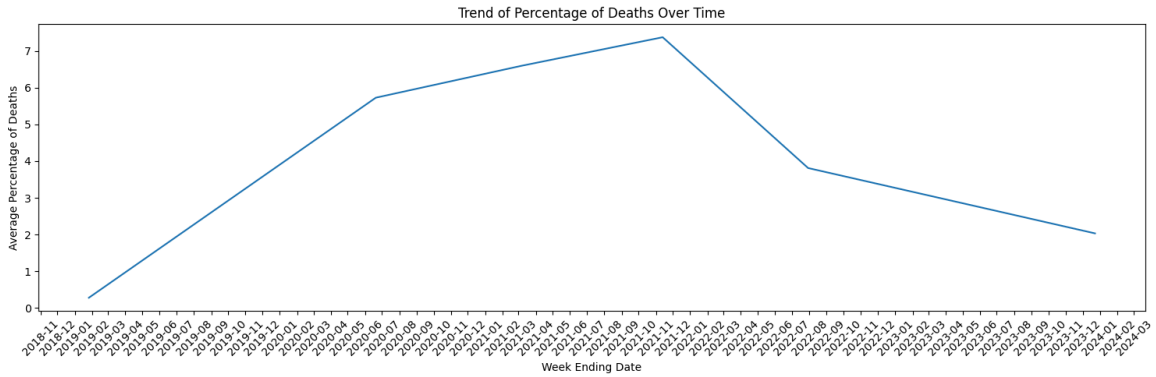
In [49]:
```
sns.boxplot(x='demographic_type', y='deaths', data=df)
plt.title('Distribution of Deaths by Demographic Type')
plt.xticks(rotation=45)
plt.xlabel('Demographic Type')
plt.ylabel('Deaths')
plt.show()
```



In [50]:
```
# Let's assume the data has been loaded into the dataframe 'df'
# We can start by visualizing trends over time. We'll assume 'start_date' an
# It's common to visualize trends using line plots, where the x-axis represe
# For simplicity, let's assume we are interested in the trend of 'percent_de
# You might need to parse 'weekending_date' to datetime if it's not already

import matplotlib.dates as mdates

# Convert 'weekending_date' to datetime if it's not already.
df['weekending_date'] = pd.to_datetime(df['weekending_date'])

# Group the data by 'weekending_date' and calculate the mean of 'percent_dea
weekly_trend = df.groupby('weekending_date')['percent_deaths'].mean().reset_

# Now let's plot the trend of 'percent_deaths' over time
plt.figure(figsize=(14,7))
sns.lineplot(data=weekly_trend, x='weekending_date', y='percent_deaths')
plt.title('Trend of Percent Deaths Over Time')
plt.xlabel('Week Ending Date')
plt.ylabel('Average Percent of Deaths')

# Formatting the date to make it more readable
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gcf().autofmt_xdate()  # Rotation

plt.show()
```



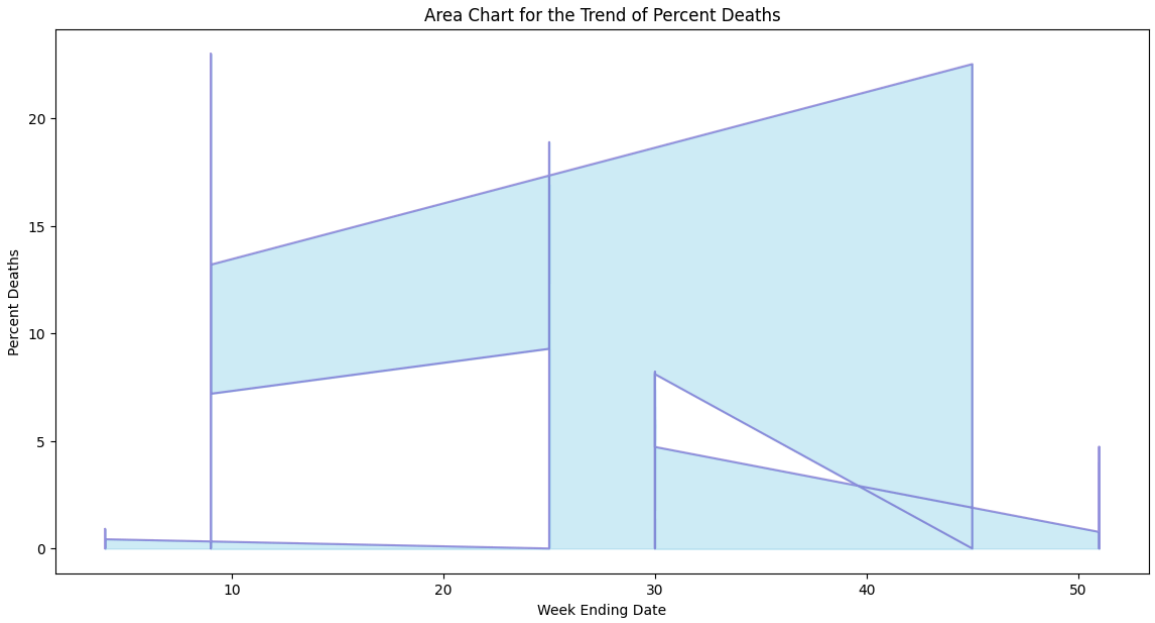
In [51]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Assuming 'df' is your dataframe and 'weekending_date' and 'percent_deaths'
# First, make sure 'weekending_date' is a datetime type
df['weekending_date'] = pd.to_datetime(df['weekending_date'])

# Group the data by week and calculate the mean percentage of deaths
weekly_trend = df.groupby(df['weekending_date']).agg({'percent_deaths':'mean

# Plotting the trend of percentage of deaths over time
plt.figure(figsize=(15, 5))
sns.lineplot(data=weekly_trend, x='weekending_date', y='percent_deaths')
plt.title('Trend of Percentage of Deaths Over Time')
plt.xlabel('Week Ending Date')
plt.ylabel('Average Percentage of Deaths')

# Improve the formatting of dates on the x-axis
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.xticks(rotation=45)

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



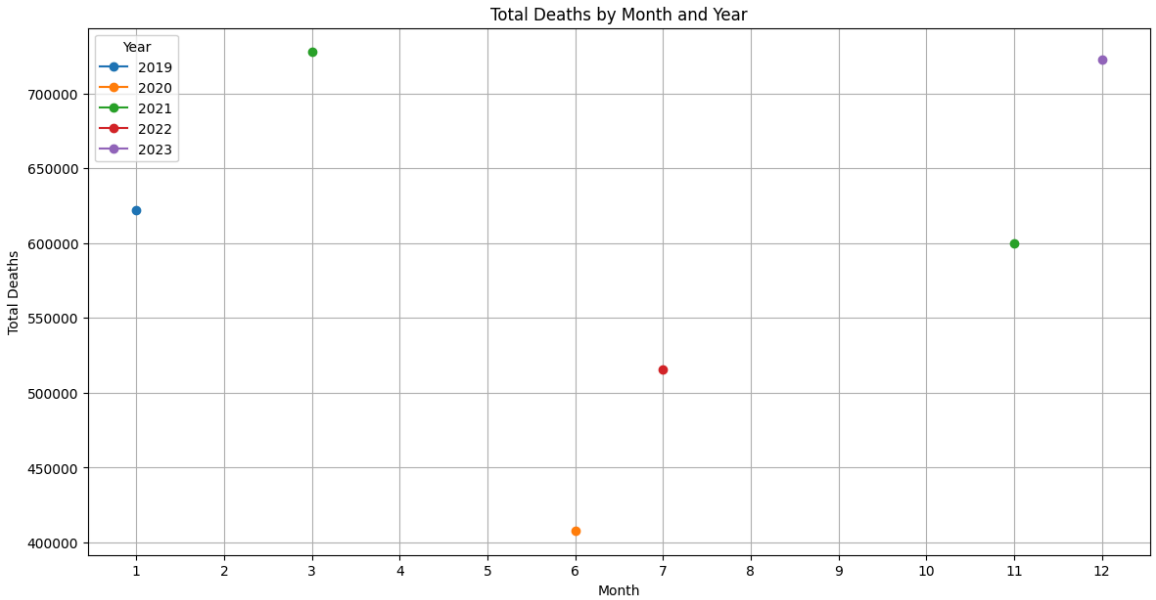
In [20]:
```
print(df.columns)
```

```
Index(['data_as_of', 'start_date', 'end_date', 'group', 'year', 'month',
       'mmwr_week', 'weekending_date', 'state', 'demographic_type',
       'demographic_values', 'pathogen', 'deaths', 'total_deaths',
       'percent_deaths', 'provisional', 'suppressed'],
      dtype='object')
```

In [52]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Ensure your dates are sorted
df.sort_values('weekending_date', inplace=True)

# Area Chart for the trend of percent of deaths
plt.figure(figsize=(14,7))
plt.fill_between(df['mmwr_week'], df['percent_deaths'], color="skyblue", alp
plt.plot(df['mmwr_week'], df['percent_deaths'], color="Slateblue", alpha=0.6
plt.title('Area Chart for the Trend of Percent Deaths')
plt.xlabel('Week Ending Date')
plt.ylabel('Percent Deaths')
plt.show()
```

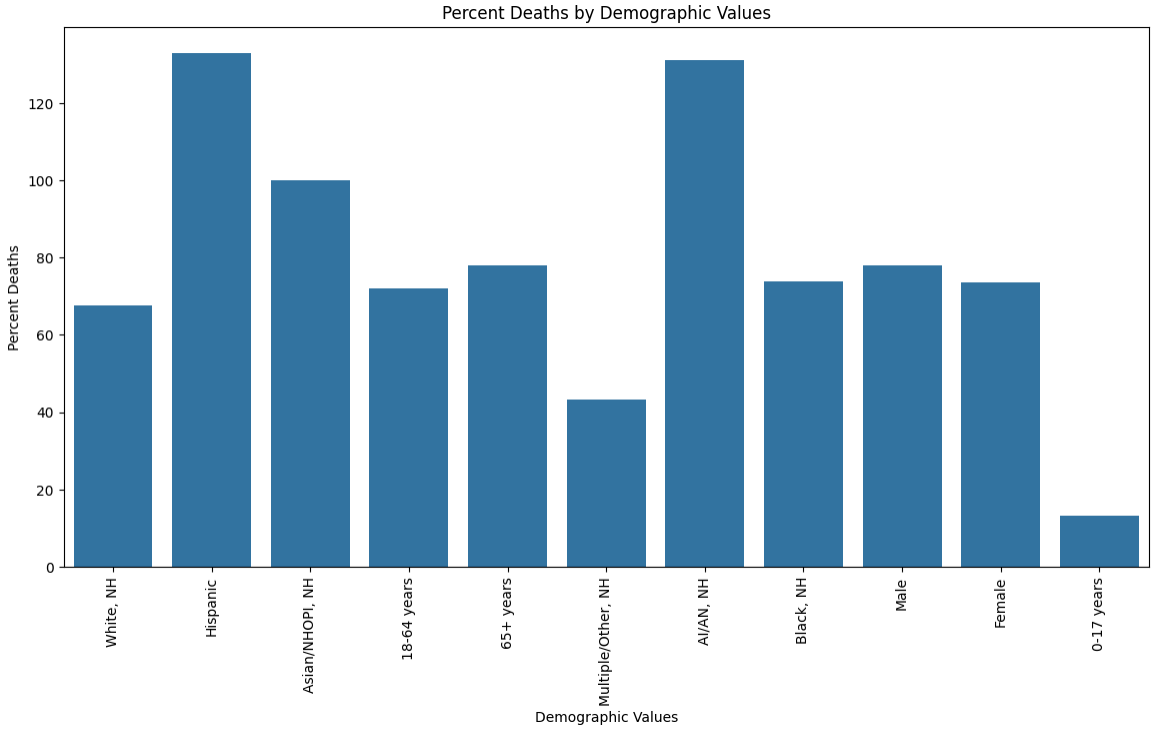


In [53]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Pivot table to summarize deaths by month and year
pivot = df.pivot_table(index='month', columns='year', values='total_deaths',

# Line plot
pivot.plot(figsize=(14, 7), marker='o')
plt.title('Total Deaths by Month and Year')
plt.xlabel('Month')
plt.ylabel('Total Deaths')
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.grid(True)
plt.show()
```

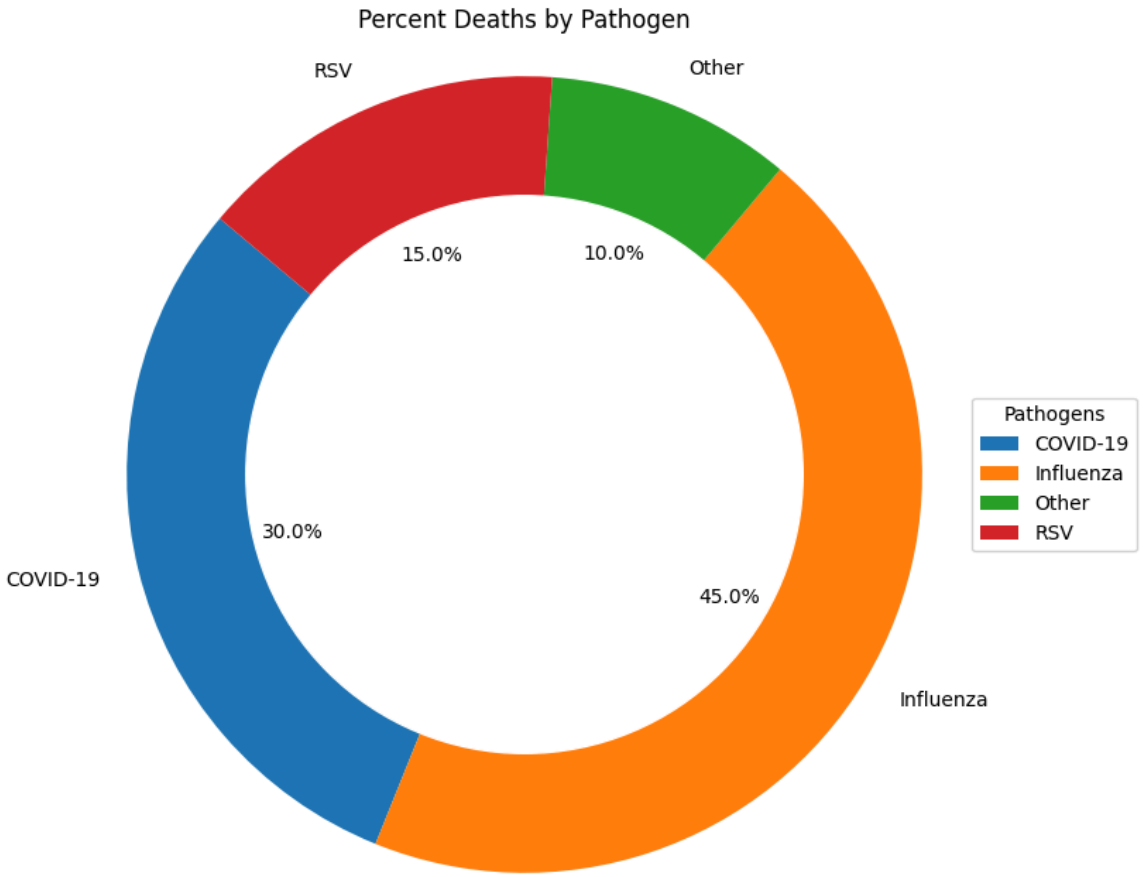


In [54]:
```
plt.figure(figsize=(14, 7))
sns.barplot(x='demographic_values', y='percent_deaths', data=df, estimator=s
plt.title('Percent Deaths by Demographic Values')
plt.xticks(rotation=90)
plt.xlabel('Demographic Values')
plt.ylabel('Percent Deaths')
plt.show()
```

```
/tmp/ipykernel_7196/3956338554.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='demographic_values', y='percent_deaths', data=df, estimato
r=sum, ci=None)
```



In [55]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Sample DataFrame, replace it with your actual data
df = pd.DataFrame({
    'pathogen': ['COVID-19', 'Influenza', 'RSV', 'Other'],
    'percent_deaths': [30, 45, 15, 10]
})

# Group by 'pathogen' and sum 'percent_deaths'
pathogen_deaths = df.groupby('pathogen')['percent_deaths'].sum().reset_index

# Create a pie chart
fig, ax = plt.subplots(figsize=(8, 8))
wedges, texts, autotexts = ax.pie(pathogen_deaths['percent_deaths'], labels=

# Draw a circle at the center of pie to make it look like a donut
centre_circle = plt.Circle((0, 0), 0.70, color='white')
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that pie is drawn as a circle.
ax.axis('equal')
ax.set_title('Percent Deaths by Pathogen')

# Add legend
ax.legend(wedges, pathogen_deaths['pathogen'], title="Pathogens", loc="cente

plt.show()
```


Percent Deaths by Pathogen
RSV
Other
15.0%
10.0%
30.0%
45.0%
COVID-19
Influenza
Pathogens
COVID-19
Influenza
Other
RSV

In [ ]: