ASSESMENT OF MARGINAL WORKERS IN TN

APPLIED DATASCIENCE (PHASE-4)

To perform the demographic analysis and calculate the distribution of marginal workers based on age, industrial category, and sex, you'll need a dataset that contains the necessary information. Once you have the dataset, you can use various data manipulation and aggregation techniques, such as the pandas library in Python, to perform the analysis.

Step 1: Import necessary libraries

```
import pandas as pd
```

Step 2: Load the dataset into a pandas DataFrame

```
# Assuming you have a dataset named 'data.csv'

data = pd.read_csv('data.csv')
```

Step 3: Data manipulation and aggregation

```
# Calculate the distribution of marginal workers based on age, industrial category, and sex

distribution = data.groupby(['Age', 'Industrial_Category',
'Sex']).size().reset_index(name='Count')


# Print the distribution

print(distribution)
```

This will provide you with a DataFrame that shows the distribution of marginal workers based on age, industrial category, and sex.

1. Data Collection: Obtain the relevant dataset that includes information on workers, their age, industrial category, and sex, and their employment status (marginal worker or not).

2. Data Preprocessing: Ensure the data is clean and structured. Remove any missing or irrelevant data and format it properly for analysis.

3. Data Aggregation: Group the data based on the criteria of age, industrial category, and sex. Calculate the counts of marginal workers within each group.

4. Data Manipulation and Analysis: Use the grouped data to calculate the distribution of marginal workers. You can use tools like Microsoft Excel, Google Sheets, or a programming language like Python (using libraries like Pandas) for this purpose. I'll provide examples using Python and Pandas.

Assuming you have a dataset in a CSV file named "workers.csv," here's how you can perform the analysis using Python.

```
# Load the dataset
data = pd.read_csv("workers.csv")

# Group the data by age, industrial category, and sex, and count marginal workers in each
group

grouped_data = data.groupby(['Age', 'Industrial_Category',
'Sex'])['Marginal_Worker'].sum().reset_index()

# Calculate the total number of workers in each group

grouped_data['Total_Workers'] = data.groupby(['Age', 'Industrial_Category',
'Sex'])['Marginal_Worker'].count().reset_index()['Marginal_Worker']

# Calculate the distribution of marginal workers as a percentage
grouped_data['Distribution'] = (grouped_data['Marginal_Worker'] /
grouped_data['Total_Workers']) * 100

# Print or export the results
print(grouped_data)
```

To calculate the distribution of marginal workers based on age, industrial category, and sex using data aggregation and manipulation

```
# Sample data (replace this with your dataset)
data = {
    'Age': [25, 30, 25, 35, 30, 35, 25, 30, 35],
    'Industrial_Category': ['Manufacturing', 'Services', 'Manufacturing', 'Services',
'Manufacturing', 'Services', 'Manufacturing', 'Services', 'Manufacturing'],
    'Sex': ['Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male'],
    'Marginal_Worker': [1, 0, 1, 1, 0, 1, 0, 1, 1]
# Assuming 1 represents marginal worker and 0 represents non-marginal worker
}

# Create a DataFrame
df = pd.DataFrame(data)

# Group the data by 'Age', 'Industrial_Category', and 'Sex'
grouped_data = df.groupby(['Age', 'Industrial_Category', 'Sex'])

# Calculate the distribution of marginal workers
distribution = grouped_data['Marginal_Worker'].mean() * 100

# Display the results
print(distribution)
```

In this example, we created a sample dataset with columns 'Age', 'Industrial_Category', 'Sex', and 'Marginal_Worker' to demonstrate the process. You should replace this data with your actual dataset.

The script groups the data by 'Age', 'Industrial_Category', and 'Sex', then calculates the mean of the 'Marginal_Worker' column within each group, effectively giving the percentage of marginal workers. Finally, it displays the distribution. Adjust the script as needed based on the structure and specifics of your dataset.

Create Visualization using data visualization Library:

To create visualizations using data visualization libraries such as Matplotlib and Seaborn in Python, we can continue building on the previous example and generate some basic plots. Here's an example of how you can visualize the data using these libraries:

```
import matplotlib.pyplot as plt

import seaborn as sns


# Resetting the index for easy plotting

distribution = distribution.reset_index()


# Create a bar plot

plt.figure(figsize=(12, 6))

sns.barplot(x='Age', y='Marginal_Worker', hue='Industrial_Category',
data=distribution)

plt.title('Distribution of Marginal Workers Based on Age and Industrial Category')

plt.xlabel('Age')

plt.ylabel('Percentage of Marginal Workers')

plt.show()
```

In this code snippet, we've added the necessary imports for Matplotlib and Seaborn. The script creates a bar plot that displays the distribution of marginal workers based on age and industrial category.