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

In [6]:
```
df = pd.read_csv('sales_data.csv')
# Display first 5 rows
df.head()
```

```
C:\Users\Sujith\AppData\Local\Temp\ipykernel_16296\500141145.py:1: DtypeWarning: Columns (0,2,5,6,7,8,9,10,11) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('sales_data.csv')
```

Out[6]:

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	O
0	26-11-2013	26.0	November	2013.0	19.0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	
1	26-11-2015	26.0	November	2015.0	19.0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	
2	23-03-2014	23.0	March	2014.0	49.0	Adults (35-64)	M	Australia	New South Wales	Accessories	Bike Racks	Hitch Rack - 4-Bike	
3	23-03-2016	23.0	March	2016.0	49.0	Adults (35-64)	M	Australia	New South Wales	Accessories	Bike Racks	Hitch Rack - 4-Bike	
4	15-05-2014	15.0	May	2014.0	47.0	Adults (35-64)	F	Australia	New South Wales	Accessories	Bike Racks	Hitch Rack - 4-Bike	

In [7]:
```
# Check for missing values
print("Missing Values:\n", df.isnull().sum())
 # Fill missing values with mean/median (for numerical) or mode (for categor
df.fillna(df.mean(), inplace=True)
df.fillna(df.mode().iloc[0], inplace=True)
```

```
Missing Values:
 Date                83036
Day                 83036
Month               83036
Year                83036
Customer_Age        83036
Age_Group           83036
Customer_Gender     83036
Country             83036
State               83036
Product_Category    83036
Sub_Category        83036
Product             83036
Order_Quantity      83036
Unit_Cost           83036
Unit_Price          83036
Profit              83036
Cost                83036
Revenue             83036
dtype: int64

C:\Users\Sujith\AppData\Local\Temp\ipykernel_16296\1463389262.py:4: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.fillna(df.mean(), inplace=True)
```

In [9]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# --- 1. Load your data ---

# Here, we'll assume you already have your DataFrame 'df'
# Make sure Customer_Age and Order_Quantity are numeric:
df['Customer_Age'] = pd.to_numeric(df['Customer_Age'], errors='coerce')
df['Order_Quantity'] = pd.to_numeric(df['Order_Quantity'], errors='coerce')

# Drop rows with missing values (if any)
df.dropna(subset=['Customer_Age', 'Country', 'Order_Quantity'], inplace=True)


# --- 2. Encode the Country Column ---
# We'll use LabelEncoder to convert country names into numeric codes.
le = LabelEncoder()
df['Country_encoded'] = le.fit_transform(df['Country'])

# Create a mapping for country selection (options 1, 2, 3, …)
unique_countries = sorted(df['Country'].unique())
country_options = {i+1: country for i, country in enumerate(unique_countries)}

print("Select a country from the following options:")
for option, country in country_options.items():
    print(f"{option}. {country}")

# --- 3. Prepare the Features and Target ---
# Our features will be Customer_Age and the encoded Country.
X = df[['Customer_Age', 'Country_encoded']]
y = df['Order_Quantity']

# Split the data into training and testing sets (optional, but recommended)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# --- 4. Train the Model ---
model = LinearRegression()
model.fit(X_train, y_train)

# --- 5. Get User Input and Make a Prediction ---
# Ask the user to select a country by number.
selected_option = int(input("Enter the option number for the country: "))
if selected_option not in country_options:
    print("Invalid option selected. Exiting.")
    exit()

selected_country = country_options[selected_option]
# Encode the selected country using the same LabelEncoder
encoded_country = le.transform([selected_country])[0]

# Ask for the customer's age
age_input = float(input("Enter the customer's age: "))

# Create the feature array for prediction
input_features = [[age_input, encoded_country]]
predicted_quantity = model.predict(input_features)

print(f"\nFor a customer aged {age_input} from {selected_country}, the predicted order quantity is approximately: {pre
```

```
Select a country from the following options:
1. Australia
2. Canada
3. France
4. Germany
5. United Kingdom
6. United States
Enter the option number for the country: 1
Enter the customer's age: 36

For a customer aged 36.0 from Australia, the predicted order quantity is approximately: 14.98

C:\Users\Sujith\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]: