UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

Project Title	**Economic Data Analysis**
Tools	Python, ML, SQL, Excel
Domain	Finance Analyst
Project Difficulties level	intermediate

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

The dataset contains information about sales transactions, including details such as the customer's age, gender, location, and the products sold.

The dataset includes data on both the cost of the product and the revenue generated from its sale, allowing for calculations of profit and profit margins.

The quantity column provides information on the volume of products sold, which could be used to analyze sales trends over time.

The dataset includes information on customer age and gender, which could be used to analyze purchasing behavior across different demographic groups.

The dataset likely includes both numeric and categorical data, which would require different types of analysis and visualization techniques.
Overall, the dataset appears to provide a comprehensive view of sales transactions, with the potential for analysis at multiple levels, including by product, customer, and location.

Column Descriptors

1. Year: This column represents the year in which the transaction occurred. It could be used to track trends over time or to filter the data based on a specific year or range of years.
2. Month: This column represents the month in which the transaction occurred. It could be used to track trends over time or to filter the data based on a specific month or range of months.
3. Customer Age: This column represents the age of the customer. It could be used to segment customers based on age ranges or to analyze the purchasing behavior of different age groups.
4. Customer Gender: This column represents the gender of the customer. It could be used to segment customers based on gender or to analyze the purchasing behavior of different genders.
5. Country: This column represents the country where the transaction occurred. It could be used to analyze sales by country or to filter the data based on a specific country or range of countries.
6. State: This column represents the state where the transaction occurred. It could be used to analyze sales by the state or to filter the data based on a specific state or range of states.

7. Product Category: This column represents the broad category of the product sold. It could be used to analyze sales by product category or to filter the data based on a specific product category.
8. Sub Category: This column represents the specific subcategory of the product sold. It could be used to analyze sales by subcategory or to filter the data based on a specific subcategory.
9. Quantity: This column represents the quantity of the product sold. It could be used to analyze sales volume or to calculate the total revenue generated from a particular product or product category.
10. Unit Cost: This column represents the cost of producing or acquiring one unit of the product. It could be used to calculate profit margins or to compare the costs of different products or product categories.
11. Unit Price: This column represents the price at which one unit of the product was sold. It could be used to analyze pricing strategies or to compare the prices of different products or product categories.
12. Cost: This column represents the total cost of the products sold, which is calculated as the product of the quantity and the unit cost. It could be used to analyze the cost structure of the business or to calculate the profit margin of each sale.
13. Revenue: This column represents the total revenue generated by the sales, which is calculated as the product of the quantity and the unit price. It could be used to analyze the overall sales performance of the business or to calculate the profit generated by each sale.

Economic Data Analysis Project

Project Overview

Objective: To analyze macroeconomic data to understand economic trends and their impact on markets.

Steps to Follow:

1. **Define the Scope and Objective**:
   - Identify the specific economic indicators to analyze (e.g., GDP, unemployment rate, inflation rate, interest rates).
   - Define the time frame and geographical scope (e.g., US economy over the past 10 years).
2. **Data Collection**:
   - Gather relevant data from reliable sources such as government databases, financial websites, and international organizations.
   - For this project, we'll use data from the World Bank and Federal Reserve Economic Data (FRED).
3. **Data Preparation**:
   - Clean the data to remove any inconsistencies or errors.
   - Combine data from different sources into a single dataset.
   - Use tools like Pandas for data cleaning and preparation.
4. **Exploratory Data Analysis (EDA)**:
   - Perform EDA to understand the data distribution and identify patterns.
   - Use visualization tools like Matplotlib and Seaborn to visualize the data.
5. **Statistical Analysis**:
   - Perform statistical analysis to identify correlations and trends.
   - Use tools like Python (Pandas, Statsmodels) for this purpose.
6. **Predictive Modeling**:
   - Build predictive models to forecast future economic trends.

   - Use machine learning algorithms like Linear Regression, ARIMA, or SARIMA.
7. **Reporting**:
   - Summarize the findings in a comprehensive report.
   - Use visualizations to support the analysis and make the report more engaging.

Example: You can get the basic idea how you can create a project from here

Detailed Python Code Example

Step-by-Step Implementation

1. **Data Collection**:
    - Assume you have downloaded the GDP, unemployment rate, inflation rate, and interest rates data from the World Bank and FRED.

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Load the datasets
gdp = pd.read_csv('gdp.csv')
unemployment = pd.read_csv('unemployment_rate.csv')
inflation = pd.read_csv('inflation_rate.csv')
interest_rate = pd.read_csv('interest_rate.csv')

# Display the first few rows of each dataset
print(gdp.head())
print(unemployment.head())
print(inflation.head())
print(interest_rate.head())
```

2. **Data Preparation**:

```
# Convert date columns to datetime format
gdp['Date'] = pd.to_datetime(gdp['Date'])
unemployment['Date'] = pd.to_datetime(unemployment['Date'])
inflation['Date'] = pd.to_datetime(inflation['Date'])
interest_rate['Date'] = pd.to_datetime(interest_rate['Date'])

# Merge the datasets on the Date column
data = pd.merge(gdp, unemployment, on='Date', how='inner')
data = pd.merge(data, inflation, on='Date', how='inner')
data = pd.merge(data, interest_rate, on='Date', how='inner')

# Rename columns for clarity
data.columns = ['Date', 'GDP', 'Unemployment_Rate', 'Inflation_Rate', 'Interest_Rate']

# Display the first few rows of the merged dataset
print(data.head())
```

3. **Exploratory Data Analysis (EDA)**:

```
# Set the date column as the index
data.set_index('Date', inplace=True)

# Plot the time series data
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.plot(data['GDP'], label='GDP')
plt.title('GDP Over Time')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(data['Unemployment_Rate'], label='Unemployment Rate')
plt.title('Unemployment Rate Over Time')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(data['Inflation_Rate'], label='Inflation Rate')
plt.title('Inflation Rate Over Time')
plt.legend()

plt.subplot(2, 2, 4)
plt.plot(data['Interest_Rate'], label='Interest Rate')
plt.title('Interest Rate Over Time')
plt.legend()

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

4. **Statistical Analysis**:

```
# Compute correlations between the economic indicators
correlation_matrix = data.corr()

# Plot the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

5. **Predictive Modeling**:

```
# Perform seasonal decomposition on GDP
decomposition = seasonal_decompose(data['GDP'], model='multiplicative',
period=12)
decomposition.plot()
plt.show()
```

```
# Fit an ARIMA model to the GDP data
model = ARIMA(data['GDP'], order=(5, 1, 0))
model_fit = model.fit(disp=0)
print(model_fit.summary())

# Make predictions
predictions = model_fit.forecast(steps=12)[0]

# Plot the predictions
plt.figure(figsize=(10, 6))
plt.plot(data['GDP'], label='Actual GDP')
plt.plot(pd.date_range(start=data.index[-1], periods=12, freq='M'), predictions,
label='Predicted GDP', color='red')
plt.title('GDP Forecast')
plt.legend()
plt.show()

# Evaluate the model
mse = mean_squared_error(data['GDP'][-12:], predictions)
print(f'Mean Squared Error: {mse}')
```

6. **Reporting**:

```
# Generate a summary report
```

```
report = f"""
Economic Data Analysis Report
==============================

1. Data Overview
----------------
- Time Frame: {data.index.min()} to {data.index.max()}
- Indicators: GDP, Unemployment Rate, Inflation Rate, Interest Rate

2. Exploratory Data Analysis
----------------------------
- GDP, Unemployment Rate, Inflation Rate, and Interest Rate trends were plotted
over time.
- Correlation analysis revealed the following relationships:
  - GDP and Unemployment Rate: {correlation_matrix.loc['GDP',
'Unemployment_Rate']:.2f}
  - GDP and Inflation Rate: {correlation_matrix.loc['GDP', 'Inflation_Rate']:.2f}
  - GDP and Interest Rate: {correlation_matrix.loc['GDP', 'Interest_Rate']:.2f}

3. Statistical Analysis
-----------------------
- Seasonal decomposition of GDP showed clear seasonal patterns.
- Correlation analysis showed strong relationships between the indicators.

4. Predictive Modeling
----------------------
- An ARIMA model was used to forecast GDP for the next 12 months.
- The model's Mean Squared Error (MSE) was: {mse:.2f}
```

```
5. Conclusions
--------------
- The analysis provided insights into the trends and relationships between key economic indicators.
- The predictive model can be used to forecast future GDP trends, aiding in economic planning and decision-making.

"""

print(report)
```

Conclusion

This project provides a comprehensive analysis of economic data, including data collection, preparation, exploratory analysis, statistical analysis, and predictive modeling. The resulting report summarizes key findings and insights, which can be useful for economic planning and decision-making.

Example: You can get the basic idea how you can create a project from here

Sample code with output

```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing
Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory
(/kaggle/working/) that gets preserved as output when you create
```

```
a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they
won't be saved outside of the current session
```

```
/kaggle/input/sales-data-for-economic-data-analysis/salesforcou
rse-4fe2kehu.csv
/kaggle/input/sales-data-for-economic-data-analysis/salesforcou
rse-4fe2kehu.xlsx
```

Overall, the dataset appears to provide a comprehensive view of sales transactions, with the potential for analysis at multiple levels, including by product, customer, and location.

Year: This column represents the year in which the transaction occurred. It could be used to track trends over time or to filter the data based on a specific year or range of years.

Month: This column represents the month in which the transaction occurred. It could be used to track trends over time or to filter the data based on a specific month or range of months.

Customer Age: This column represents the age of the customer. It could be used to segment customers based on age ranges or to analyze the purchasing behavior of different age groups.

Customer Gender: This column represents the gender of the customer. It could be used to segment customers based on gender or to analyze the purchasing behavior of different genders.

Country: This column represents the country where the transaction occurred. It could be used to analyze sales by country or to filter the data based on a specific country or range of countries.

State: This column represents the state where the transaction occurred. It could be used to analyze sales by the state or to filter the data based on a specific state or range of states.

Product Category: This column represents the broad category of the product sold. It could be used to analyze sales by product category or to filter the data based on a specific product category.

Sub Category: This column represents the specific subcategory of the product sold. It could be used to analyze sales by subcategory or to filter the data based on a specific subcategory.

Quantity: This column represents the quantity of the product sold. It could be used to analyze sales volume or to calculate the total revenue generated from a particular product or product category.

Unit Cost: This column represents the cost of producing or acquiring one unit of the product. It could be used to calculate profit margins or to compare the costs of different products or product categories.

Unit Price: This column represents the price at which one unit of the product was sold. It could be used to analyze pricing strategies or to compare the prices of different products or product categories.

Cost: This column represents the total cost of the products sold, which is calculated as the product of the quantity and the unit cost. It could be used to analyze the cost structure of the business or to calculate the profit margin of each sale.

Revenue: This column represents the total revenue generated by the sales, which is calculated as the product of the quantity and the unit price. It could be used to analyze the overall sales performance of the business or to calculate the profit generated by each sale.

```
In [2]:
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import statsmodels.api as sm
```

```
In [3]:
df =
pd.read_csv("/kaggle/input/sales-data-for-economic-data-analysi
s/salesforcourse-4fe2kehu.csv")
df.head()
```

```
Out[3]:
```

	ind	Date	Ye	Month	Custome	Customer	Count	State	Product Cate	Sub Cat	Quanti	Unit	Unit Pric	Co	Revenu	Column

	ex		ar		r Age	Gender	ry		gory	egory	ty	Cost	e	st	e	1
0	0	2/19/2016	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.000000	80.0	109.0	NaN
1	1	2/20/2016	2016.0	February	29.0	F	United States	Washington	Clothing	Gloves	2.0	24.50	28.500000	49.0	57.0	NaN
2	2	2/27/2016	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	3.0	3.67	5.000000	11.0	15.0	NaN

3	3	3/12/2016	2016.0	March	29.0	F	United States	Washington	Accessories	Tires and Tubes	2.0	87.50	116.500000	175.0	233.0	NaN
4	4	3/12/2016	2016.0	March	29.0	F	United States	Washington	Accessories	Tires and Tubes	3.0	35.00	41.666667	105.0	125.0	NaN

In [4]:

```
##Identifies Non-Null and data types in datasets
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34867 entries, 0 to 34866
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   index             34867 non-null  int64
 1   Date              34866 non-null  object
 2   Year              34866 non-null  float64
 3   Month             34866 non-null  object
 4   Customer Age      34866 non-null  float64
 5   Customer Gender   34866 non-null  object
 6   Country           34866 non-null  object
 7   State             34866 non-null  object
 8   Product Category  34866 non-null  object
 9   Sub Category      34866 non-null  object
 10  Quantity          34866 non-null  float64
 11  Unit Cost         34866 non-null  float64
 12  Unit Price        34866 non-null  float64
 13  Cost              34866 non-null  float64
 14  Revenue           34867 non-null  float64
 15  Column1           2574 non-null   float64
dtypes: float64(8), int64(1), object(7)
memory usage: 4.3+ MB
```

In [5]:

```
df.tail()
```

Out[5]:

	index	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Column1
34862	34862	2/7/2016	2016.0	February	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	2.0	1160.0	985.500000	2320.0	1971.000000	NaN
34863	34863	3/13/2015	2015.0	March	38.0	M	France	Hauts de	Bikes	Mountain Bikes	1.0	2049.0	1583.000000	2049.0	1583.000000	NaN

								Seine								
34864	34864	4/5/2015	2015.0	April	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	3.0	683.0	560.666667	2049.0	1682.000000	NaN
34865	34865	8/30/2015	2015.0	August	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	1.0	2320.0	1568.000000	2320.0	1568.000000	NaN

34866	34866	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	641.532095	NaN

In [6]:

```
#Removes data in row 34866
df = df.drop(34866, axis=0)
```

In [7]:

```
#Delete data in column "Column1"
df = df.drop('Column1', axis=1)
```

In [8]:

```
# Convert "Customer Age, Year,and Quantity " column to integer
format
df['Customer Age'] = df['Customer Age'].astype(int)
df['Year'] = df['Year'].astype(int)
df['Quantity'] = df['Quantity'].astype(int)
```

In [9]:

```
# Convert "Date" column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
```

In [10]:

```
#Extract the year and month from the "Date" column
df['Year']=df['Date'].dt.year
df['Year_Month'] = df['Date'].dt.strftime('%Y-%m')
```

In [11]:

```
# Calculate profit for each product
df['profit'] = df['Revenue'] - df['Cost']
# Calculate profit margin for each product
df['profit_margin'] = df['profit'] / df['Revenue']
```

In [12]:

```
df.describe()
```

Out[12]:

	index	Date	Year	Customer Age	Quantity	Unit Cost	Unit Price	Cost	Revenue	profit	profit_margin
count	34866.000000	34866	34866.000000	34866.000000	34866.000000	34866.000000	34866.000000	34866.000000	34866.000000	34866.000000	34866.000000
mean	17432.500000	2016-01-19 18:35:05.110996224	2015.569237	36.382895	2.002524	349.880567	389.232485	576.004532	640.870074	64.865542	0.134077
min	0.000000	2015-01-01 00:00:00	2015.000000	17.000000	1.000000	0.670000	0.666667	2.000000	2.000000	-937.000000	-0.686747
25	8716.250	2015-10-26	2015.000	28.00000	1.00	45.00000	53.66666	85.00000	102.0000	5.00	0.06

%	000	00:00:00	000	0	0000	0	7	0	00	0000	1679
50%	17432.500000	2016-01-28 00:00:00	2016.000000	35.000000	2.000000	150.000000	179.000000	261.000000	319.000000	27.000000	0.147963
75%	26148.750000	2016-04-26 00:00:00	2016.000000	44.000000	3.000000	455.000000	521.000000	769.000000	902.000000	96.000000	0.225677
max	34865.000000	2016-07-31 00:00:00	2016.000000	87.000000	3.000000	3240.000000	5082.000000	3600.000000	5082.000000	1842.000000	0.500000
std	10065.091579	NaN	0.495190	11.112902	0.813936	490.015846	525.319091	690.500395	736.650597	152.879908	0.135445

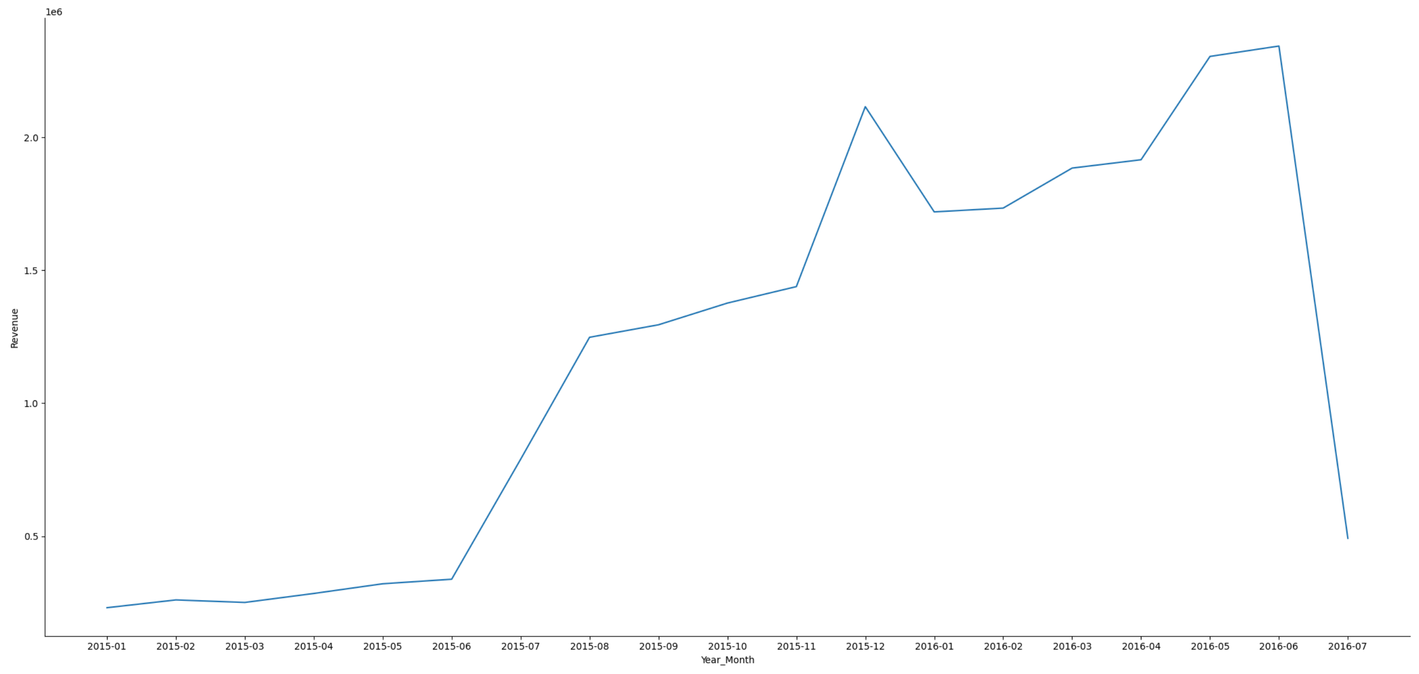
```
In [13]:
```

```
# Group data by Revenue and month
Month_Revenue =
df.groupby(['Year_Month'])['Revenue'].sum().reset_index()
sns.relplot(data=Month_Revenue, x="Year_Month", y="Revenue",
kind="line", height =10, aspect = 2.1)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118
: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
Out[13]:

<seaborn.axisgrid.FacetGrid at 0x7a8224913490>
```



Revenue from sales tended to increase from month to month in 2015 and early 2016, then decreased drastically in July 2016. December 2015 was the month with the highest sales revenue, while July 2016 was the month with the lowest sales revenue

Then let's look at the costs incurred each month

In [14]:

```
monthly_cost =
df.groupby(['Year_Month'])['Cost'].sum().reset_index()
```

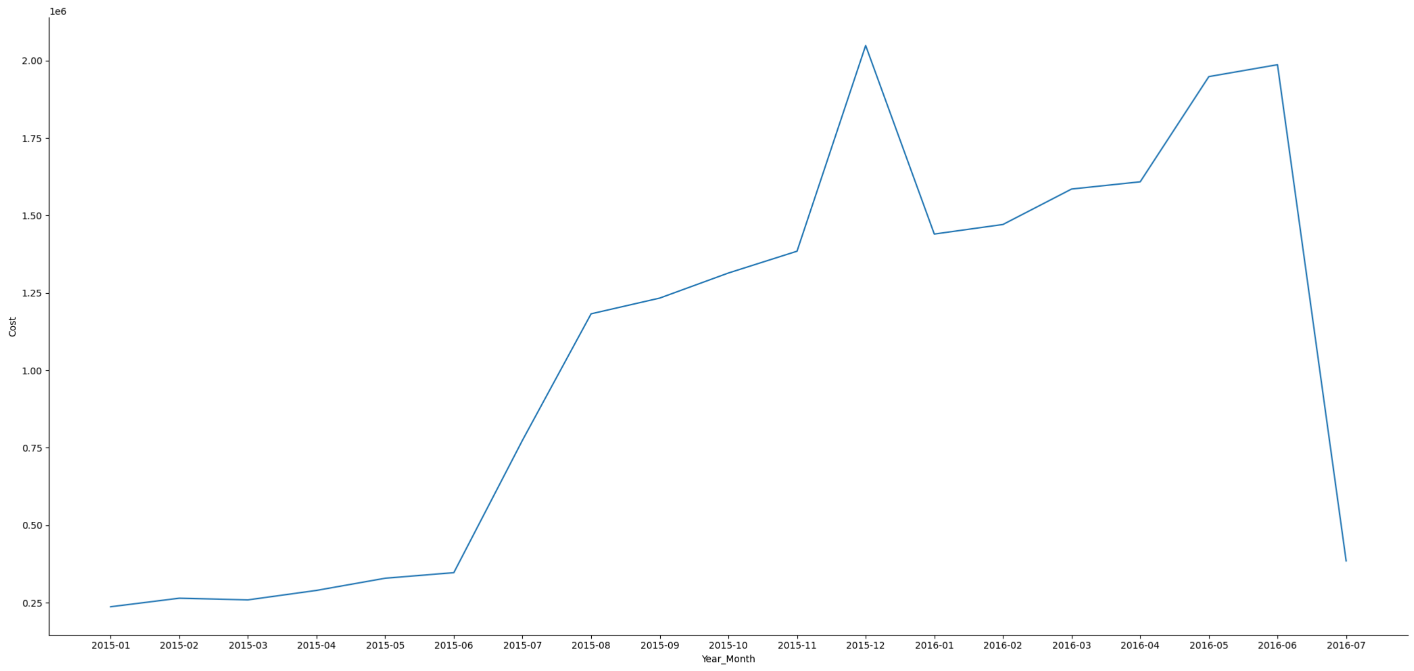
In [15]:

```
sns.relplot(data=monthly_cost, x="Year_Month", y="Cost",
kind="line", height =10, aspect = 2.1)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118
: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x7a8205826a10>
```



From this graph it can be seen that the trend of Cost and Revenue tends to be the same, namely increasing every month, but Cost from January 2015 to June 2015 tends to be higher than Revenue

In [16]:

```
grouped = df.groupby(["Year_Month"])[['Cost', 'Revenue', 'profit']].sum().reset_index()
grouped
```

Out[16]:

	Year_Month	Cost	Revenue	profit

0	2015-01	236328.0	230549.0	-5779.0
1	2015-02	263937.0	259857.0	-4080.0
2	2015-03	258522.0	250358.0	-8164.0
3	2015-04	289089.0	284143.0	-4946.0
4	2015-05	328431.0	320629.0	-7802.0
5	2015-06	346447.0	337756.0	-8691.0
6	2015-07	773950.0	789054.0	15104.0

7	2015-08	1182259.0	1248185.0	65926.0
8	2015-09	1233074.0	1295246.0	62172.0
9	2015-10	1314018.0	1376969.0	62951.0
10	2015-11	1384447.0	1438928.0	54481.0
11	2015-12	2048649.0	2116097.0	67448.0
12	2016-01	1439868.0	1720072.0	280204.0
13	2016-02	1470736.0	1734376.0	263640.0

14	2016-03	1585201.0	1884978.0	299777.0
15	2016-04	1608601.0	1916347.0	307746.0
16	2016-05	1948277.0	2305191.0	356914.0
17	2016-06	1986680.0	2344229.0	357549.0
18	2016-07	384460.0	491612.0	107152.0

In [17]:

```
fig = px.line(grouped, x='Year_Month', y=['Cost', 'Revenue',
'profit'],
              title='Monthly Performance')
fig.update_xaxes(title='Year-Month')
fig.update_yaxes(title='Amount ($)')
```

```
# menampilkan plot
fig.show()
```

Jan 2015Mar 2015May 2015Jul 2015Sep 2015Nov 2015Jan 2016Mar 2016May 2016Jul 201600.5M1M1.5M2M

variableCostRevenueprofitMonthly PerformanceYear-MonthAmount ($)

There is a clear seasonal pattern in this data, where profits tend to increase at the end of the year and decrease at the beginning of the next.

In [18]:

```
# Group by sub category and calculate total quantity sold
category_sales = df.groupby('Sub
Category')['Quantity'].sum().reset_index()
```

In [19]:

```
fig = px.bar(category_sales, y='Sub Category', x='Quantity',
text_auto='.2s',
            title="Product Sales Quantity Based on Sub
Categories")
fig.show()
```

20030011k3.0k1.1k1.5k9108.4k7904.0k5.5k6.1k1.1k75022k2.7k64005k1

0k15k20kBike RacksBike StandsBottles and CagesCapsCleanersFendersGlovesHelmetsHydration PacksJerseysMountain BikesRoad BikesShortsSocksTires and TubesTouring BikesVests

Product Sales Quantity Based on Sub CategoriesQuantitySub Category

In [20]:

```
category_profit= df.groupby('Sub
Category')['profit'].sum().reset_index()
```

In [21]:

```
fig = px.bar(category_profit, y='Sub Category', x='profit',
text_auto='.2s',
            title="Profit by Sub Category")
fig.update_xaxes(title='Profit($)')
fig.show()
```

35k25k130k44k15k71k46k520k72k300k140k98k87k9.5k510k95k58k0100k200k300k400k500kBike RacksBike StandsBottles and CagesCapsCleanersFendersGlovesHelmetsHydration PacksJerseysMountain BikesRoad BikesShortsSocksTires and TubesTouring BikesVests

Profit by Sub CategoryProfit($)Sub Category

In [22]:

```
category_margin = df.groupby('Sub
Category')['profit_margin'].mean().reset_index()
```

```
In [23]:
```

```
fig = px.bar(category_margin, y='Sub Category',
x='profit_margin',
            title="Profit Margin by Sub Category")
fig.show()
```

00.050.10.150.2Bike RacksBike StandsBottles and CagesCapsCleanersFendersGlovesHelmetsHydration PacksJerseysMountain BikesRoad BikesShortsSocksTires and TubesTouring BikesVests

Profit Margin by Sub Categoryprofit_marginSub Category

Bike Racks had the highest profit margin of 22.7%, followed by Fenders with a profit margin of 20.7%. Meanwhile, Road Bikes and Mountain Bikes have very low profit margins, respectively 0.5% and 1.0%.

```
In [24]:
```

```
fig = px.histogram(df, x="Customer Age")
fig.show()
```

20304050607080020040060080010001200

Customer Agecount

What products are purchased the most based on the age of the customer?

In [25]:

```
# Group by Customer Age and product category, sum quantity sold
df_grouped = df.groupby(["Customer Age", "Product
Category"])["Quantity"].sum().reset_index()

# Find top selling product for each Customer Age
top_products = df_grouped.groupby("Customer Age").apply(lambda
x: x.loc[x.Quantity.idxmax()])

# Create bar chart
fig = px.bar(top_products, x="Customer Age", y="Quantity",
color="Product Category", title="Top Selling Products by
Customer Age")
fig.show()
```

2030405060708002004006008001000120014001600

Product CategoryAccessoriesClothingTop Selling Products by Customer AgeCustomer AgeQuantity

In [26]:

```
# Group by Customer Age and product category, sum quantity sold
df_grouped = df.groupby(["Customer Age", "Sub
Category"])["Quantity"].sum().reset_index()

# Find top selling product for each Customer Age
top_products = df_grouped.groupby("Customer Age").apply(lambda
x: x.loc[x.Quantity.idxmax()])

# Create bar chart
fig = px.bar(top_products, x="Customer Age", y="Quantity",
color="Sub Category", title="Top Selling Sub Category Products
by Customer Age")
fig.show()
```

2030405060708001002003004005006007008OO

Sub CategoryTires and TubesHelmetsHydration PacksBottles and CagesShortsTop Selling Sub Category Products by Customer AgeCustomer AgeQuantity

Which country has the highest profit?

```
In [27]:
country_sales =
df.groupby('Country')['profit'].sum().reset_index()
```

```
In [28]:
```

```
fig = px.pie(df, values='profit', names='Country',
color_discrete_sequence=px.colors.sequential.RdBu)
fig.show()
```

42.4%31%14.5%12.1%

GermanyUnited StatesUnited KingdomFrance

**What products are purchased the most in each country **

```
In [29]:
# Group by country and sub category, sum quantity sold
df_grouped = df.groupby(["Country", "Sub
Category"])["Quantity"].sum().reset_index()

# Find top selling product for each country
top_products = df_grouped.groupby("Country").apply(lambda x:
x.loc[x.Quantity.idxmax()])

# Create bar chart
fig = px.bar(top_products, x="Country", y="Quantity",
color="Sub Category", title="Top Selling Sub Category Products
by Country")
fig.show()
```

FranceGermanyUnited KingdomUnited States02k4k6k8k10k12k

Sub CategoryTires and TubesTop Selling Sub Category Products by CountryCountryQuantity

"Tires and Tubes" are the most frequently purchased products in all countries in the dataset. However, what are the products with the highest profit margins in each country?

```
In [30]:
```

Reference link