In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
order_details = pd.read_csv("order_details_zim.csv")
orders = pd.read_csv('orders_zim.csv', parse_dates=['orderDate', 'requiredDate', 'shippedDate'], dayfirst=True)
product_details = pd.read_csv("product_details_zim.csv")
productline = pd.read_csv("ProductLine_zim.csv")
customer = pd.read_csv("Zim_customer.csv")
employee = pd.read_csv("Zimson_employee.csv")
offices = pd.read_csv("Zimson_offices.csv")
transaction = pd.read_csv("zimson_transaction_dataset.csv")
```

In [9]:
```
#merging orders and order_details
merged_orders = pd.merge(orders, order_details, on='orderNumber', how='inner')
```

In [10]:
```
merged_orders.head()
```

Out[10]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	orderLineNumber
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	1
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	1
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	1
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	1
4	20030	2025-05-07	2025-05-12	2025-05-14	Shipped	Shipped order	1002	WAT0030	4	499	1

In [12]:
```
# Merge with customers
merged_orders_customers = pd.merge(merged_orders, customer, on='customerNumber', how='inner')
```

In [13]:
```
merged_orders_customers.head()
```

Out[13]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	contactFirstName	phone	addressLine1	addressLine2	cit
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	Ankit	9876543210	12 MG Road	Near Park	Mumb
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	Ankit	9876543210	12 MG Road	Near Park	Mumb
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	Priya	9123456789	34 Nehru Place	Opp. Metro	Kolkat
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	Priya	9123456789	34 Nehru Place	Opp. Metro	Kolkat
4	20030	2025-05-07	2025-05-12	2025-05-14	Shipped	Shipped order	1002	WAT0030	4	499	...	Priya	9123456789	34 Nehru Place	Opp. Metro	Kolkat

5 rows × 23 columns

In [15]:
```
#Merge with products
merged_orders_products = pd.merge(merged_orders_customers, product_details, on='productCode', how='inner')
```

In [16]:
```
merged_orders_products.head()
```

Out[16]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	country	salesRepEmployeeNumbers	creditLimit	productName	pr
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	India	201	500000.0	Tag Heuer Promaster	
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	India	201	500000.0	Tag Heuer Promaster	
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	India	202	450000.0	Seiko Le Locle	
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	India	202	450000.0	Seiko Le Locle	
4	20030	2025-05-07	2025-05-12	2025-05-14	Shipped	Shipped order	1002	WAT0030	4	499	...	India	202	450000.0	Citizen G-Shock	

5 rows × 30 columns

In [20]:
```
#Merge with product_lines
merged_orders_productline = pd.merge(merged_orders_products, productline, on='productLine', how='inner')
```

In [21]:
```
merged_orders_productline.head()
```

Out[21]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	salesRepEmployeeNumbers	creditLimit	productName	productLine
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	201	500000.0	Tag Heuer Promaster	Limitec Editior
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	201	500000.0	Tag Heuer Promaster	Limitec Editior
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	202	450000.0	Seiko Le Locle	Limitec Editior
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	202	450000.0	Seiko Le Locle	Limitec Editior
4	20104	2024-09-17	2024-09-24	2024-09-25	Shipped	Shipped order	1007	WAT0104	1	10995	...	207	650000.0	Patek Philippe Royal Oak	Limitec Editior

5 rows × 31 columns

In [22]:
```
# Merge with employees
merged_orders_employee = pd.merge(merged_orders_productline, employee, left_on='salesRepEmployeeNumbers', right_on='employeeNumber', how='left')
```

In [23]:
```
merged_orders_employee.head()
```

Out[23]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	lastName	firstName	extension	email	offic
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	Ahamed	Mir Wasim	x9828	mirwasimahamed@gmail.com	
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	Ahamed	Mir Wasim	x9828	mirwasimahamed@gmail.com	
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	Mansoor	Mansoor	x8842	mansoor@gmail.com	
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	Mansoor	Mansoor	x8842	mansoor@gmail.com	
4	20104	2024-09-17	2024-09-24	2024-09-25	Shipped	Shipped order	1007	WAT0104	1	10995	...	Elamaran	Elamaran	x1438	elamaran@gmail.com	

5 rows × 42 columns

In [28]:
```
#Merge with offices
merged_orders_offices_mergedfully = pd.merge(merged_orders_employee, offices, on='officeCode', how='left')
```

In [29]:
```
merged_orders_offices_mergedfully.head()
```

Out[29]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	City	Country	city_y	phone_y	addressLine1_y	addressLine2_y
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
4	20104	2024-09-17	2024-09-24	2024-09-25	Shipped	Shipped order	1007	WAT0104	1	10995	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar

5 rows × 50 columns

In [30]:
```
# Basic structure and types
merged_orders_offices_mergedfully.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 50 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   orderNumber              300 non-null    int64
 1   orderDate                300 non-null    datetime64[ns]
 2   requiredDate             300 non-null    datetime64[ns]
 3   shippedDate              300 non-null    datetime64[ns]
 4   status                   300 non-null    object
 5   comments                 300 non-null    object
 6   customerNumber           300 non-null    int64
 7   productCode              300 non-null    object
 8   quantityOrdered          300 non-null    int64
 9   priceEach                300 non-null    int64
 10  orderLineNumber          300 non-null    int64
 11  customerName             300 non-null    object
 12  contactLastName          300 non-null    object
 13  contactFirstName         300 non-null    object
 14  phone_x                  300 non-null    int64
 15  addressLine1_x           300 non-null    object
 16  addressLine2_x           294 non-null    object
 17  city_x                   300 non-null    object
 18  state_x                  300 non-null    object
 19  postalCode_x             300 non-null    int64
 20  country_x                300 non-null    object
 21  salesRepEmployeeNumbers  300 non-null    int64
 22  creditLimit              300 non-null    float64
 23  productName              300 non-null    object
 24  productLine              300 non-null    object
 25  productVendor            300 non-null    object
 26  productDescription       300 non-null    object
 27  quantityInStock          300 non-null    int64
 28  buyPrice                 300 non-null    int64
 29  MSRP                     300 non-null    int64
 30  textDescription          300 non-null    object
 31  employeeNumber           63 non-null     float64
 32  lastName                 63 non-null     object
 33  firstName                63 non-null     object
 34  extension                63 non-null     object
 35  email                    63 non-null     object
 36  officeCode               63 non-null     float64
 37  reportsTo                61 non-null     float64
 38  jobTitle                 63 non-null     object
 39  User Name                63 non-null     object
 40  City                     63 non-null     object
 41  Country                  63 non-null     object
 42  city_y                   63 non-null     object
 43  phone_y                  63 non-null     float64
 44  addressLine1_y           63 non-null     object
 45  addressLine2_y           63 non-null     object
 46  state_y                  63 non-null     object
 47  country_y                63 non-null     object
 48  postalCode_y             63 non-null     float64
 49  territory                63 non-null     object
dtypes: datetime64[ns](3), float64(6), int64(11), object(30)
memory usage: 117.3+ KB
```

In [32]:
```
# Descriptive statistics
merged_orders_offices_mergedfully.describe()
```

Out[32]:

	orderNumber	orderDate	requiredDate	shippedDate	customerNumber	quantityOrdered	priceEach	orderLineNumber	phone_x	postalCode_x	salesRepEmployeeNumbers	creditLimit	qua
count	300.000000	300	300	300	300.00000	300.000000	300.000000	300.0	3.000000e+02	300.000000	300.000000	300.000000	
mean	20091.423333	2024-07-09 13:02:24	2024-07-15 10:48:00	2024-07-17 17:40:48	1169.57000	5.376667	17605.146667	1.0	5.441129e+09	268068.753333	807.243333	286222.581667	
min	20001.000000	2023-01-01 00:00:00	2023-01-06 00:00:00	2023-01-10 00:00:00	1001.00000	1.000000	499.000000	1.0	-1.064600e+04	10974.000000	201.000000	25165.960000	
25%	20046.000000	2023-09-26 06:00:00	2023-10-02 06:00:00	2023-10-03 12:00:00	1028.00000	3.000000	2495.000000	1.0	-6.114000e+03	66724.000000	228.000000	62484.460000	
50%	20092.000000	2024-07-08 12:00:00	2024-07-14 00:00:00	2024-07-21 00:00:00	1050.00000	5.000000	6295.000000	1.0	9.812346e+09	110024.000000	250.000000	340000.000000	
75%	20136.250000	2025-04-24 06:00:00	2025-05-01 06:00:00	2025-05-03 06:00:00	1323.00000	8.000000	20495.000000	1.0	9.845105e+09	500033.000000	1504.000000	470000.000000	
max	20180.000000	2025-12-28 00:00:00	2026-01-02 00:00:00	2026-01-04 00:00:00	1348.00000	10.000000	404400.000000	1.0	9.988771e+09	700091.000000	1507.000000	650000.000000	
std	52.477778	NaN	NaN	NaN	148.71217	2.913718	37501.019483	0.0	4.897764e+09	239323.441382	629.317598	213455.019434	

In [34]:
```
# Check missing values
merged_orders_offices_mergedfully.isnull().sum()
```

Out[34]:
```
orderNumber                0
orderDate                  0
requiredDate               0
shippedDate                0
status                     0
comments                   0
customerNumber             0
productCode                0
quantityOrdered            0
priceEach                  0
orderLineNumber            0
customerName               0
contactLastName            0
contactFirstName           0
phone_x                    0
addressLine1_x             0
addressLine2_x             6
city_x                     0
state_x                    0
postalCode_x               0
country_x                  0
salesRepEmployeeNumbers    0
creditLimit                0
productName                0
productLine                0
productVendor              0
productDescription         0
quantityInStock            0
buyPrice                   0
MSRP                       0
textDescription            0
employeeNumber           237
lastName                 237
firstName                237
extension                237
email                    237
officeCode               237
reportsTo                239
jobTitle                 237
User Name                237
City                     237
Country                  237
city_y                   237
phone_y                  237
addressLine1_y           237
addressLine2_y           237
state_y                  237
country_y                237
postalCode_y             237
territory                237
dtype: int64
```

In [36]:
```
# Fill missing values for text columns
for col in merged_orders_offices_mergedfully.select_dtypes(include='object').columns:
    merged_orders_offices_mergedfully[col].fillna("Not Available", inplace=True)
```

In [37]:
```
# Check missing values
merged_orders_offices_mergedfully.isnull().sum()
```

Out[37]:
```
orderNumber               0
orderDate                 0
requiredDate              0
shippedDate               0
status                    0
comments                  0
customerNumber            0
productCode               0
quantityOrdered           0
priceEach                 0
orderLineNumber           0
customerName              0
contactLastName           0
contactFirstName          0
phone_x                   0
addressLine1_x            0
addressLine2_x            0
city_x                    0
state_x                   0
postalCode_x              0
country_x                 0
salesRepEmployeeNumbers   0
creditLimit               0
productName               0
productLine               0
productVendor             0
productDescription        0
quantityInStock           0
buyPrice                  0
MSRP                      0
textDescription           0
employeeNumber            0
lastName                  0
firstName                 0
extension                 0
email                     0
officeCode                0
reportsTo                 0
jobTitle                  0
User Name                 0
City                      0
Country                   0
city_y                    0
phone_y                   0
addressLine1_y            0
addressLine2_y            0
state_y                   0
country_y                 0
postalCode_y              0
territory                 0
dtype: int64
```

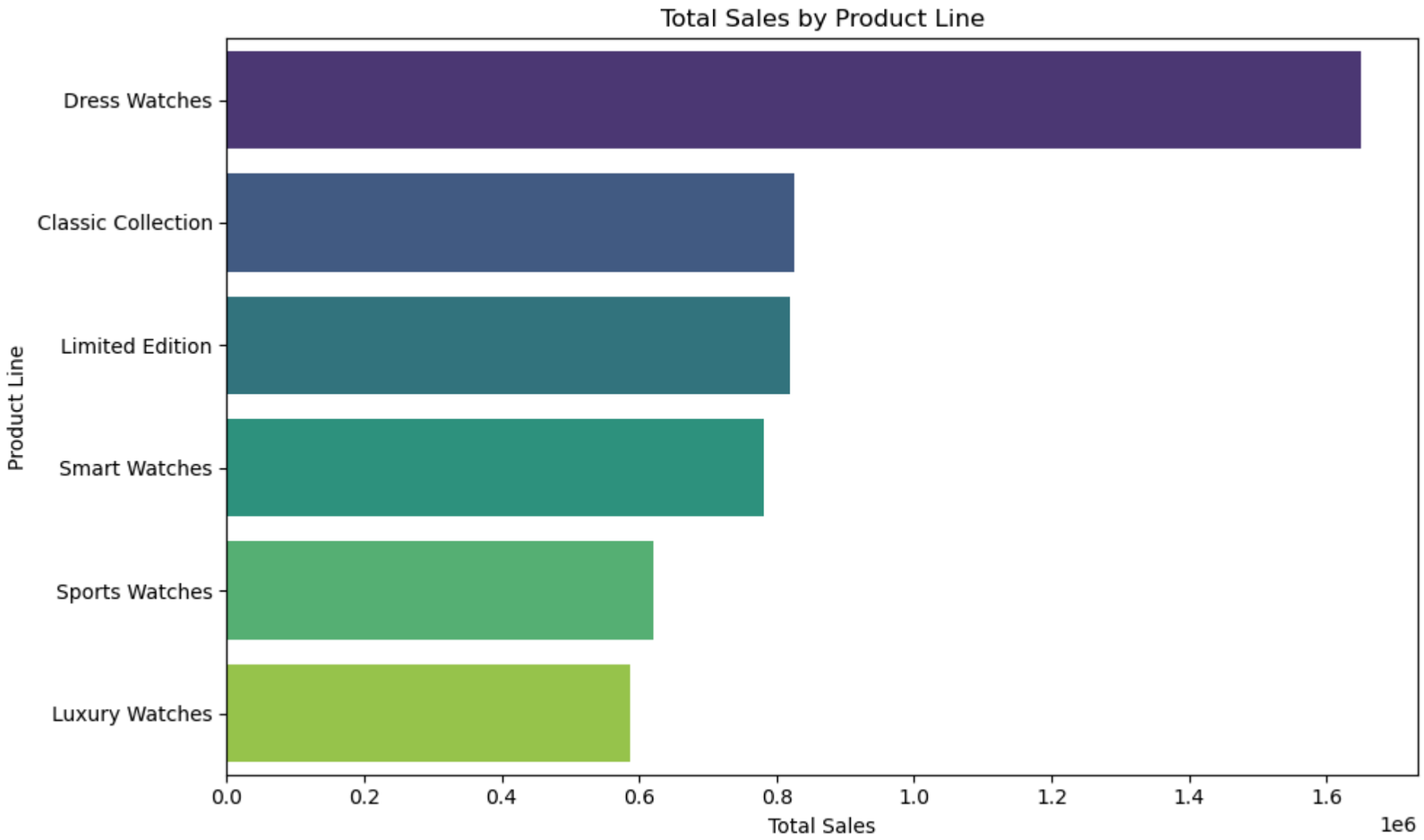
In [39]:
```
# Peek at data
merged_orders_full.head()
```

Out[39]:

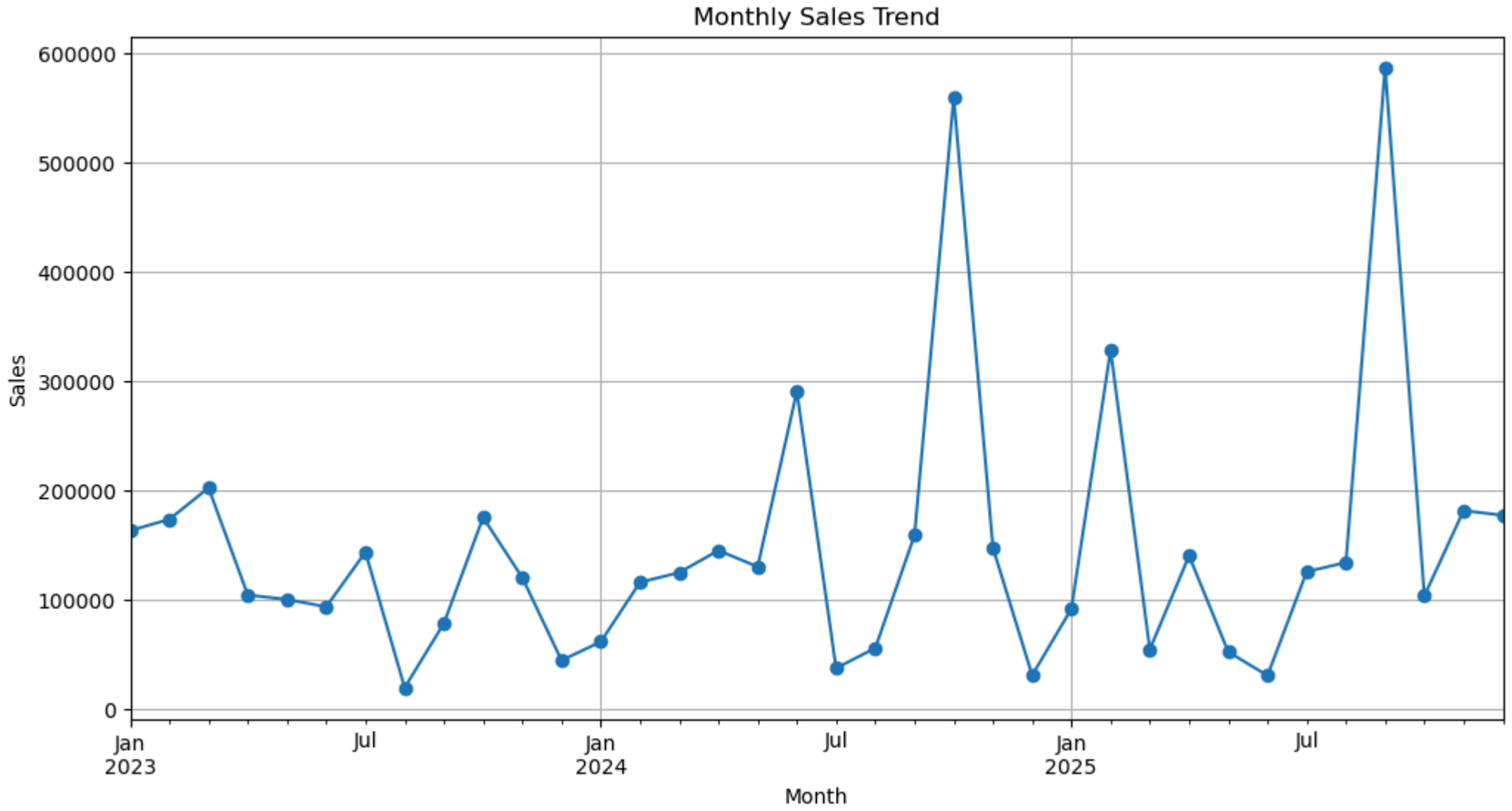
	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode	quantityOrdered	priceEach	...	City	Country	city_y	phone_y	addressLine1_y	addressLine2_y
0	20078	2023-10-17	2023-10-23	2023-10-25	Shipped	Shipped order	1001	WAT0078	9	2495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
1	20078	2025-08-25	2025-08-31	2025-09-03	Shipped	Shipped order	1001	WAT0078	9	2495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
2	20161	2025-12-07	2025-12-14	2025-12-16	Shipped	Shipped order	1002	WAT0161	2	5495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
3	20161	2023-08-28	2023-09-03	2023-09-06	Shipped	Shipped order	1002	WAT0161	2	5495	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar
4	20104	2024-09-17	2024-09-24	2024-09-25	Shipped	Shipped order	1007	WAT0104	1	10995	...	Trichy	India	Trichy	-2779239.0	C102-A, Shastri Road	Thillai Nagar

5 rows × 50 columns

In [43]:
```
#visualizing total Total Sales by Product Line:
import matplotlib.pyplot as plt
import seaborn as sns

product_sales = merged_orders_offices_mergedfully.groupby('productLine')['priceEach'].sum().sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=product_sales.values, y=product_sales.index, palette='viridis')
plt.title('Total Sales by Product Line')
plt.xlabel('Total Sales')
plt.ylabel('Product Line')
plt.tight_layout()
plt.show()
```



In [44]:
```
#visualizing Monthly Sales Trend

merged_orders_offices_mergedfully['orderDate'] = pd.to_datetime(merged_orders_offices_mergedfully['orderDate'])
merged_orders_offices_mergedfully['Month'] = merged_orders_offices_mergedfully['orderDate'].dt.to_period('M')
monthly_sales = merged_orders_offices_mergedfully.groupby('Month')['priceEach'].sum()

plt.figure(figsize=(12,6))
monthly_sales.plot(kind='line', marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```



In [45]:
```
# Add revenue column
merged_orders_offices_mergedfully['Revenue'] = merged_orders_offices_mergedfully['quantityOrdered'] * merged_orders_offices_mergedfully['priceEach']
```

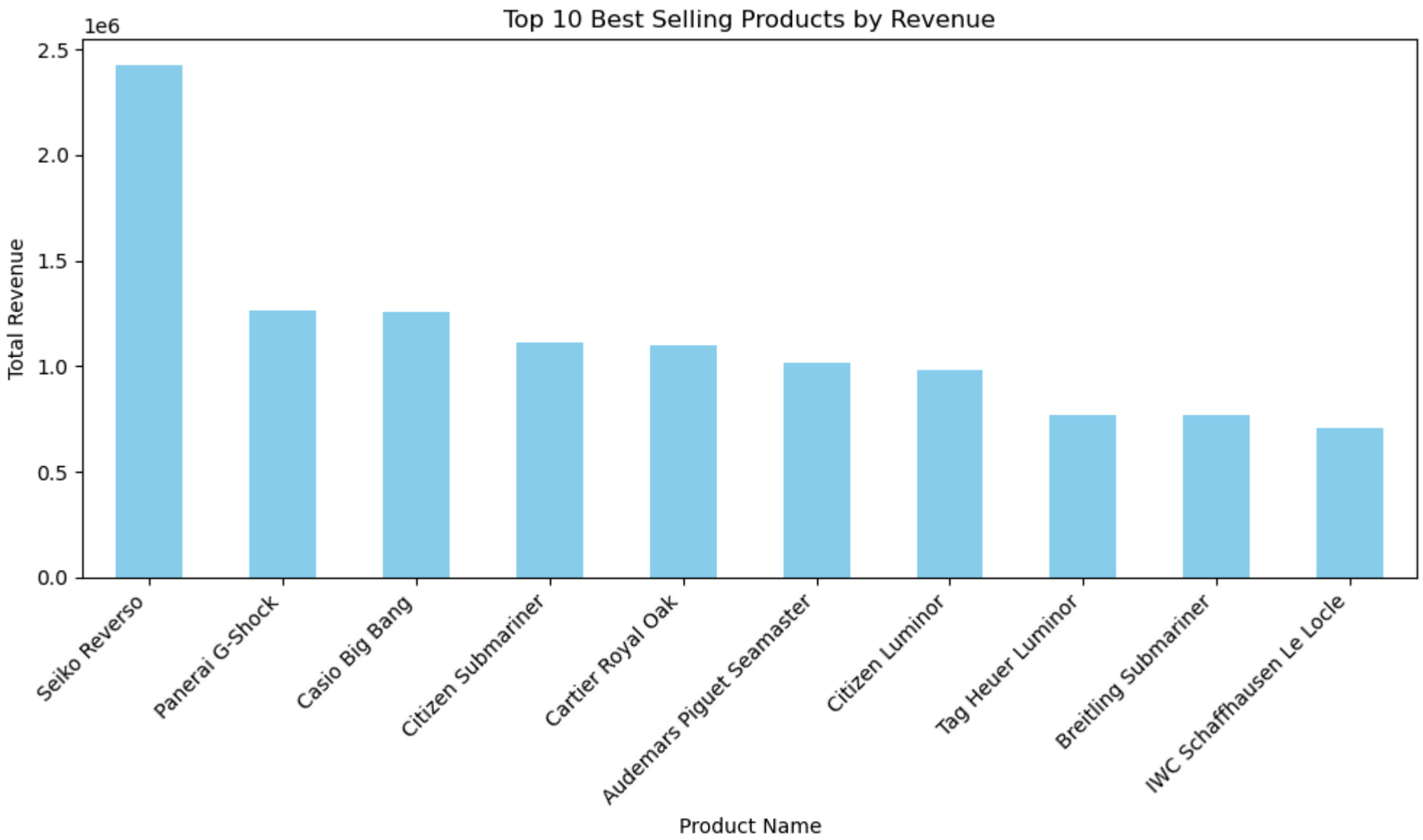
In [46]:
```
# now Finding top 10 products by total revenue
top_products = merged_orders_offices_mergedfully.groupby('productName')['Revenue'].sum().sort_values(ascending=False).head(10)
```

In [48]:
```
print(top_products)
```

```
productName
Seiko Reverso               2426400
Panerai G-Shock             1264800
Casio Big Bang              1253990
Citizen Submariner          1113950
Cartier Royal Oak           1098900
Audemars Piguet Seamaster   1013950
Citizen Luminor              979200
Tag Heuer Luminor            768000
Breitling Submariner         767000
IWC Schaffhausen Le Locle    709380
Name: Revenue, dtype: int64
```

In [50]:
```
#visualizing the Top 10 Best Selling Products by revenue:

import matplotlib.pyplot as plt

# Plot top 10 selling products
```

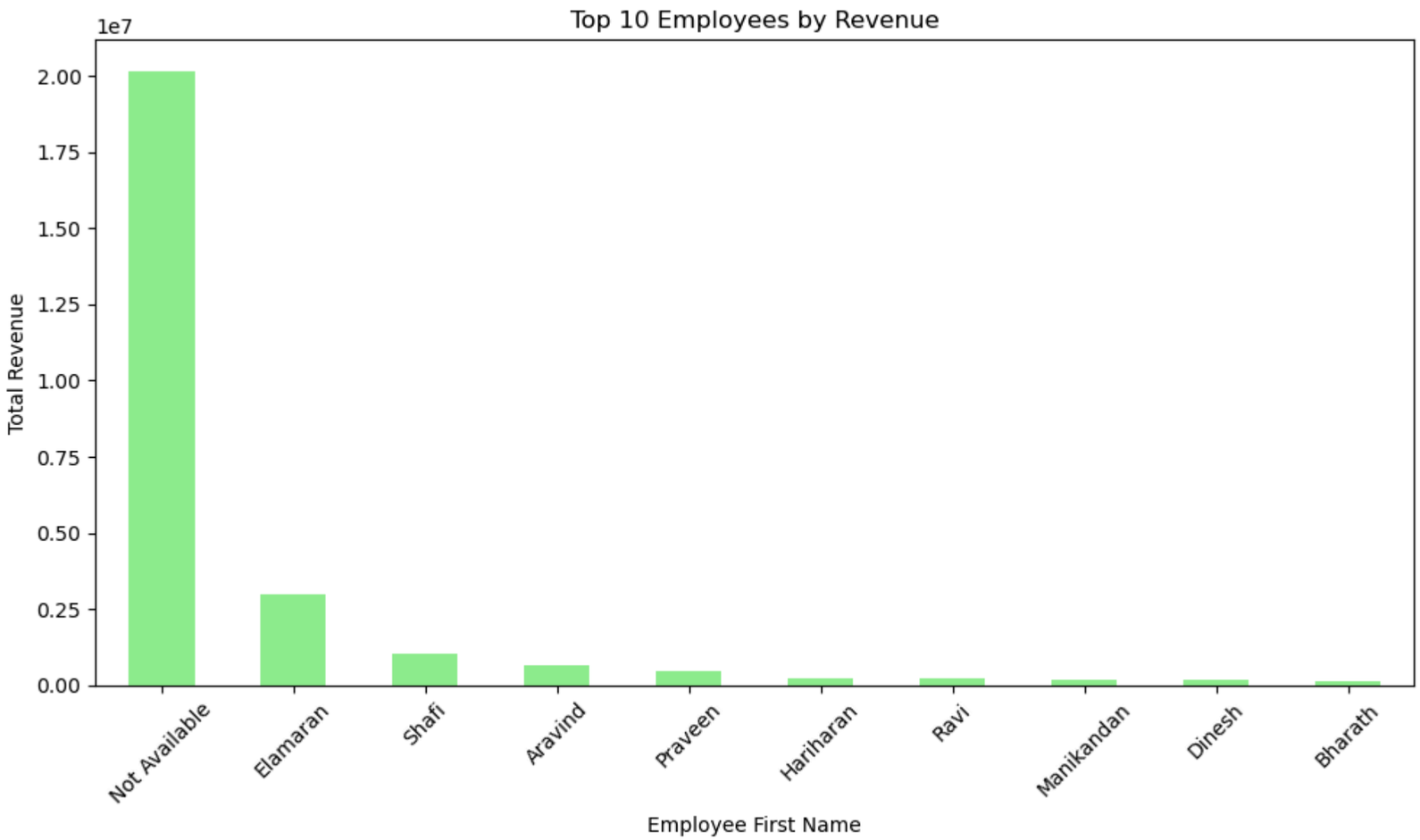
```
top_products.plot(kind='bar', color='skyblue', figsize=(10,6))

# Chart details
plt.title('Top 10 Best Selling Products by Revenue')
plt.xlabel('Product Name')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Show chart
plt.show()
```



In [52]:
```
# Create a new column for total revenue
merged_orders_offices_mergedfully['total_revenue'] = merged_orders_offices_mergedfully['quantityOrdered'] * merged_orders_offices_mergedfully['priceEach']
```

In [53]:
```
#now visualizing the Best Performing Salesperson (Employee)
# Group by employee name and calculate revenue
employee_sales = merged_orders_offices_mergedfully.groupby('firstName')['total_revenue'].sum().sort_values(ascending=False).head(10)

# Plot it
employee_sales.plot(kind='bar', color='lightgreen', figsize=(10,6))

# Chart labels
plt.title('Top 10 Employees by Revenue')
```

```
plt.xlabel('Employee First Name')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [54]:
```
merged_orders_offices_mergedfully['firstName'].unique()
```

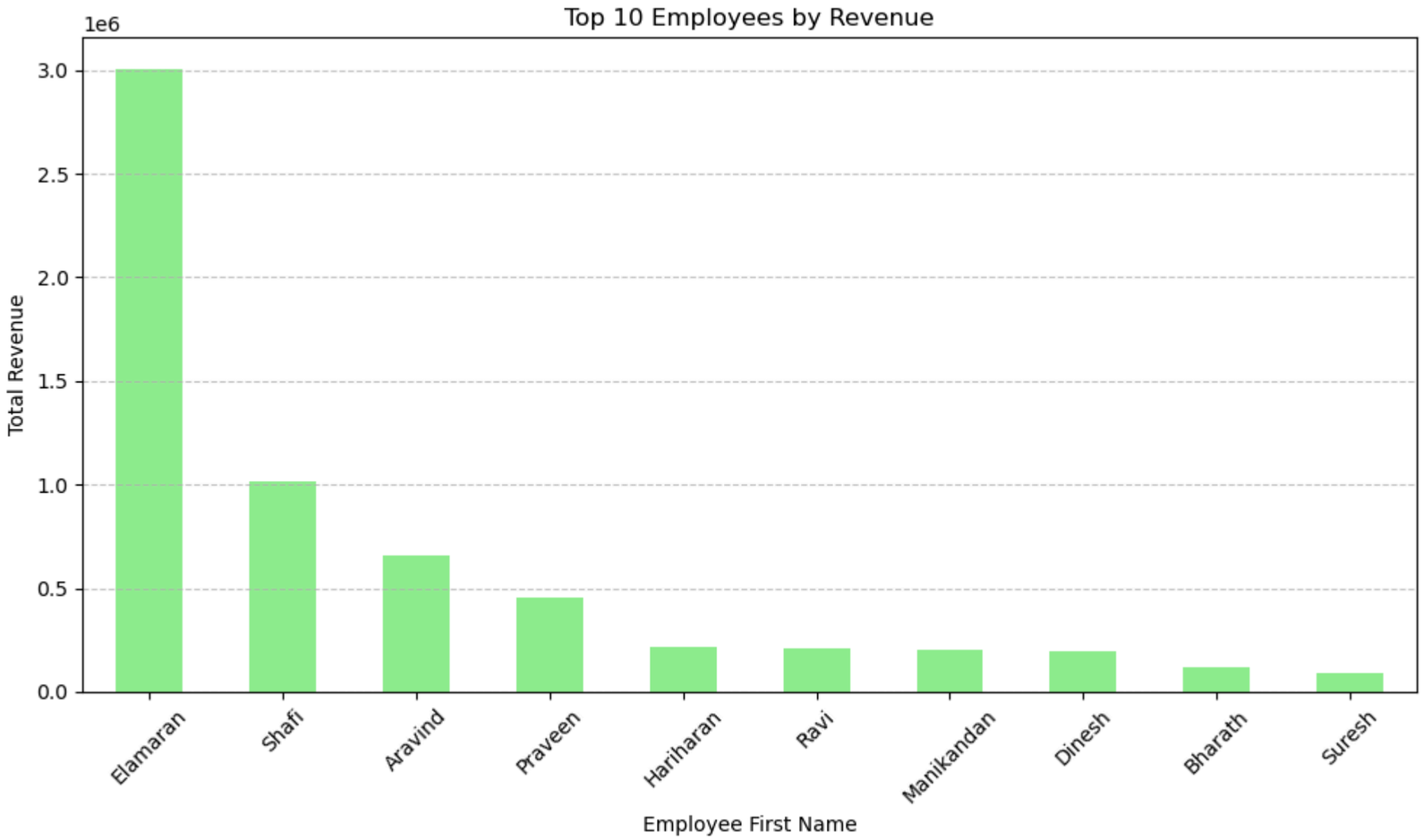
Out[54]:
```
array(['Mir Wasim', 'Mansoor', 'Elamaran', 'Manikandan', 'Suresh',
       'Not Available', 'Shafi', 'Bharath', 'Balavishnu', 'Praveen',
       'Dinesh', 'Hariharan', 'Aravind', 'Ravi', 'Karthik', 'Vignesh',
       'Kumar'], dtype=object)
```

In [55]:
```
merged_orders_offices_mergedfully['firstName'].value_counts()
```

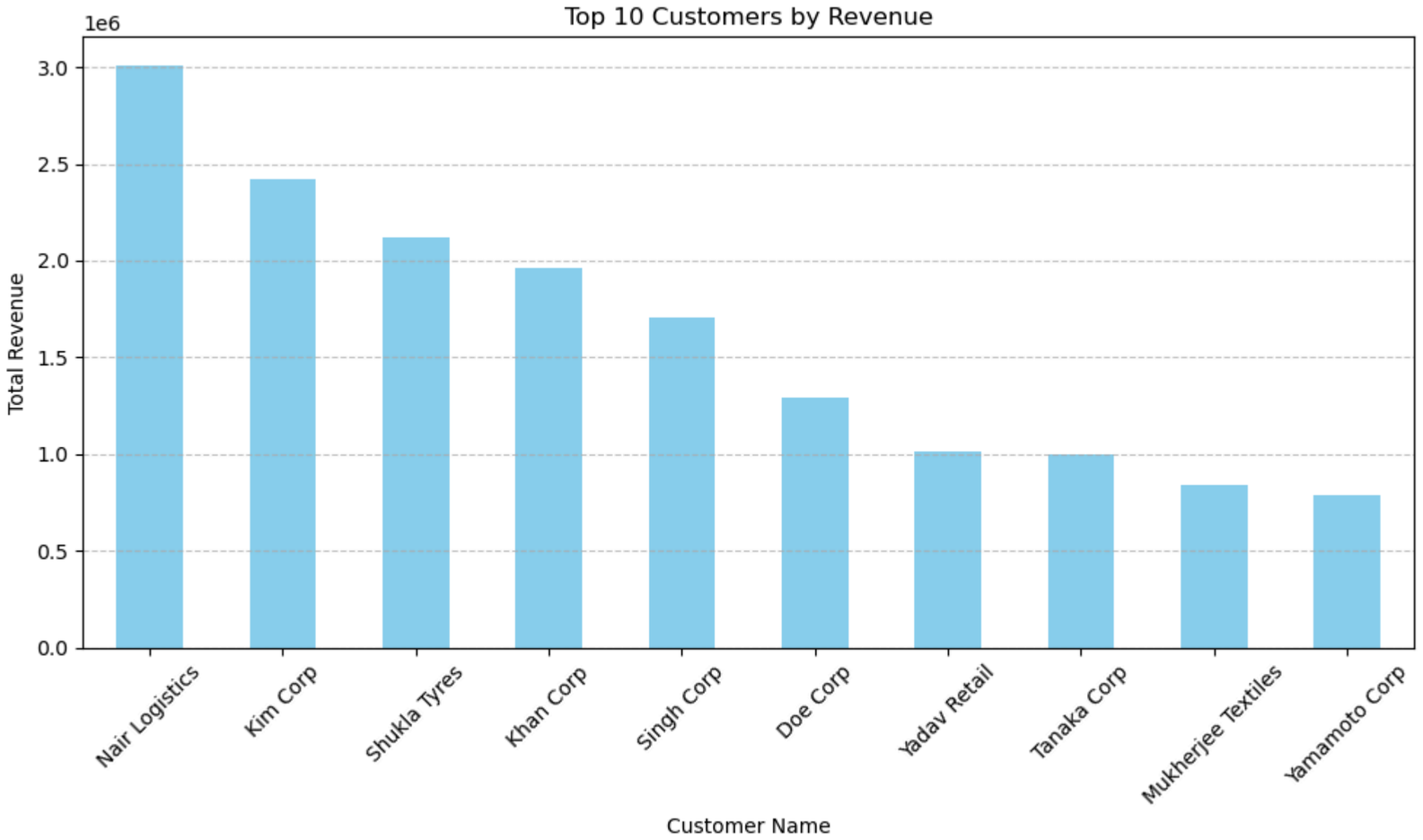
Out[55]:
```
firstName
Not Available    237
Elamaran           9
Balavishnu         6
Manikandan         6
Shafi              6
Dinesh             6
Bharath            4
Praveen            4
Aravind            4
Suresh             3
Mansoor            3
Hariharan          3
Karthik            2
Vignesh            2
Mir Wasim          2
Ravi               2
Kumar              1
Name: count, dtype: int64
```

In [56]:
```
# Removing rows with 'Not Available' as employee first name
valid_employee_data = merged_orders_offices_mergedfully[merged_orders_offices_mergedfully['firstName'] != 'Not Available']
```

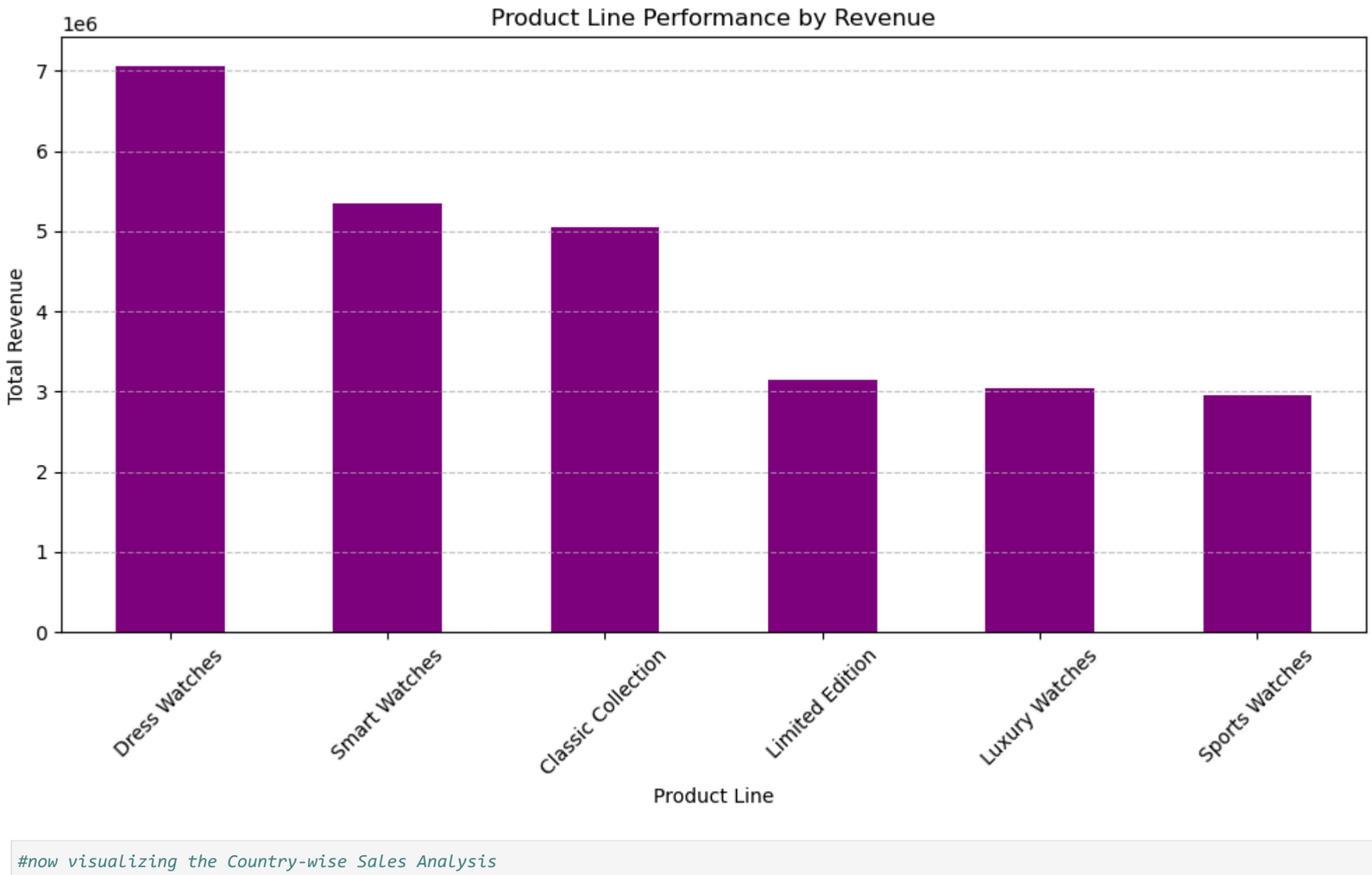
In [60]:
```
# Group by employee name and calculate revenue again
employee_sales = valid_employee_data.groupby('firstName')['total_revenue'].sum().sort_values(ascending=False).head(10)

# Plot
employee_sales.plot(kind='bar', color='lightgreen', figsize=(10,6))
plt.title('Top 10 Employees by Revenue')
plt.xlabel('Employee First Name')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



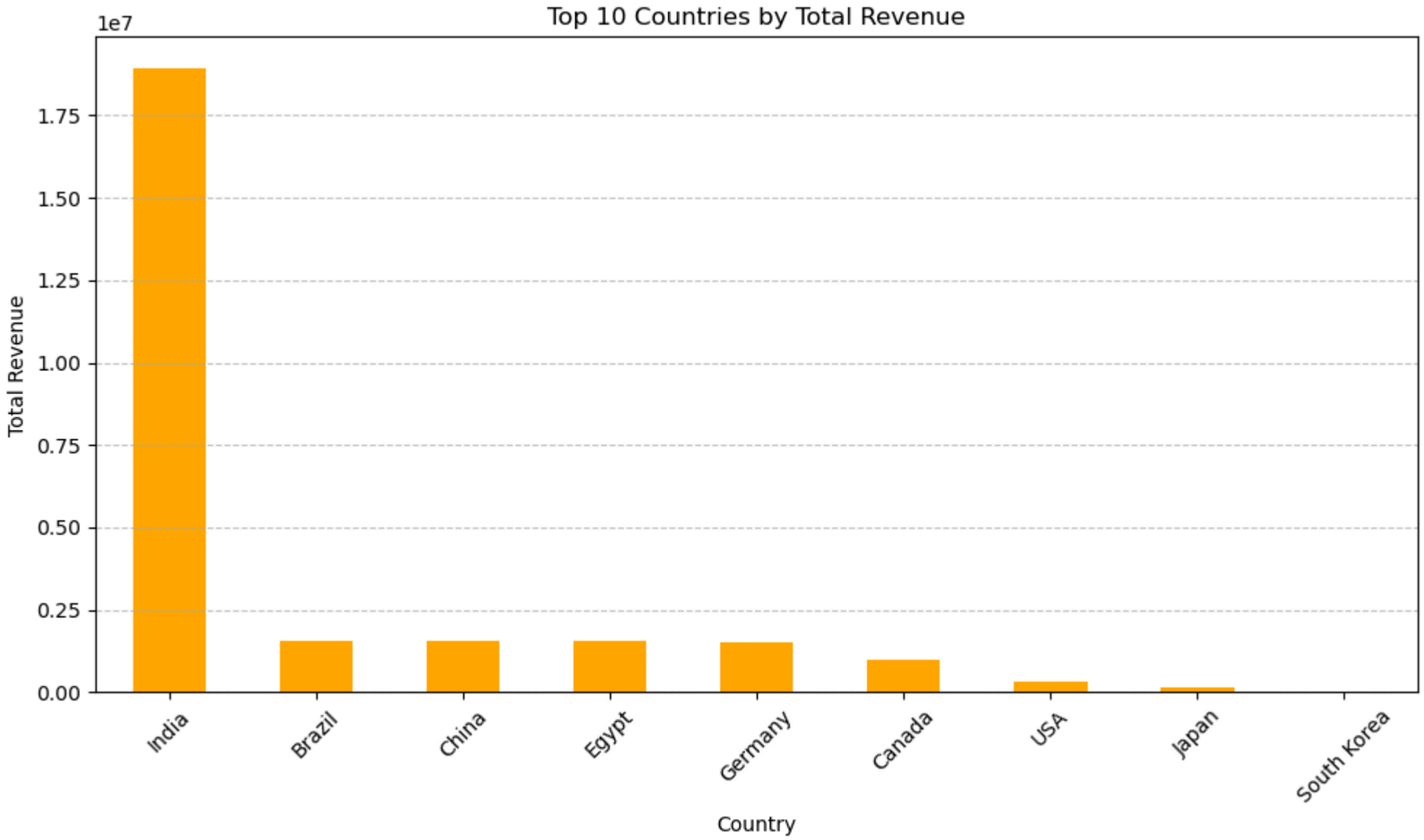
In [61]:
```
#Top 10 Customers by Revenue
top_customers = merged_orders_offices_mergedfully.groupby('customerName')['total_revenue'].sum().sort_values(ascending=False).head(10)

# Plot it
top_customers.plot(kind='bar', color='skyblue', figsize=(10,6))
plt.title('Top 10 Customers by Revenue')
plt.xlabel('Customer Name')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [62]:
```
#now visualizing the Product Line Performance (to see which product category sells the best)
# Group by product line and calculate total revenue
product_line_perf = merged_orders_offices_mergedfully.groupby('productLine')['total_revenue'].sum().sort_values(ascending=False)

# Plot it
product_line_perf.plot(kind='bar', color='purple', figsize=(10,6))
plt.title('Product Line Performance by Revenue')
plt.xlabel('Product Line')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [63]:
```
#now visualizing the Country-wise Sales Analysis
# Group by country and sum the revenue
country_sales = merged_orders_offices_mergedfully.groupby('country_x')['total_revenue'].sum().sort_values(ascending=False).head(10)

# Plot
country_sales.plot(kind='bar', color='orange', figsize=(10,6))
plt.title('Top 10 Countries by Total Revenue')
plt.xlabel('Country')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [64]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
import pandas as pd

 #Selecting relevant features:
features = merged_orders_offices_mergedfully[['quantityOrdered', 'priceEach', 'productLine', 'country_x']].dropna()
```

In [65]:
```
# Creatimg a new column for total revenue
features['total_revenue'] = features['quantityOrdered'] * features['priceEach']
```

In [66]:
```
# Input features (X) and Target variable (y)
X = features[['total_revenue', 'productLine', 'country_x']]
y = merged_orders_offices_mergedfully.loc[features.index, 'status']
```

In [67]:
```
#Converting categorical features to numbers
X = pd.get_dummies(X)
```

In [68]:
```
#spliting the traingset and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [69]:
```
#Train Decision Tree model
model = DecisionTreeClassifier(random_state=42)
```

```
model.fit(X_train, y_train)
```

Out[69]:

DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)

In [70]:
```
#redicting and checking accuracy:
y_pred = model.predict(X_test)
```

In [71]:
```
y_pred
```

Out[71]:
```
array(['Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'On Hold', 'On Hold', 'On Hold',
       'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'In Process', 'Shipped',
       'Shipped', 'Shipped', 'On Hold', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped', 'Shipped',
       'Shipped', 'Shipped', 'On Hold', 'Shipped', 'Shipped', 'Shipped',
       'Shipped'], dtype=object)
```

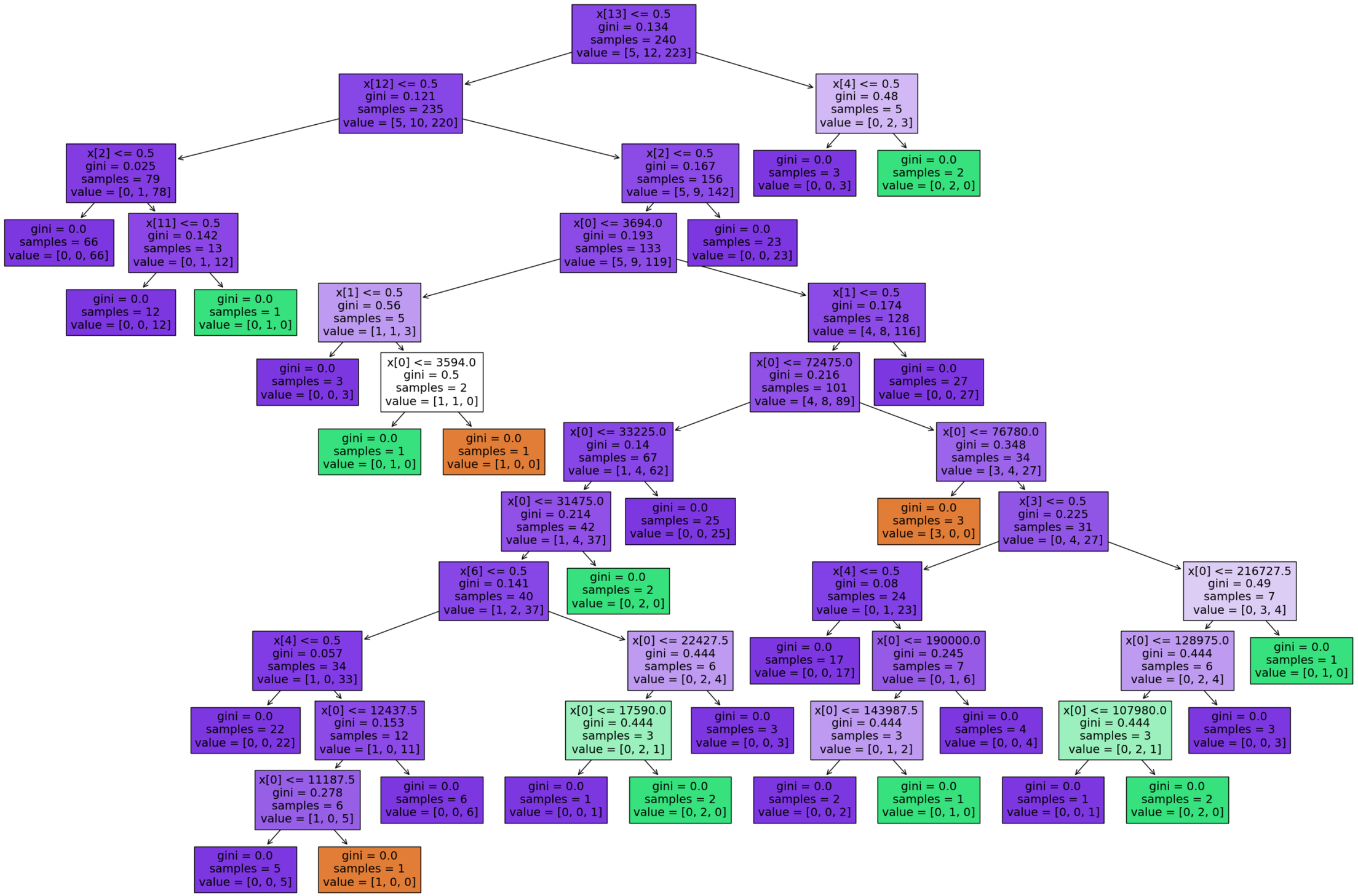
In [72]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.95
```

In [75]:
```
print(" Classification Report:\n", classification_report(y_test, y_pred))
```

```
 Classification Report:
                precision    recall  f1-score   support

    In Process       1.00      1.00      1.00         1
       On Hold       0.60      0.75      0.67         4
       Shipped       0.98      0.96      0.97        55

      accuracy                           0.95        60
     macro avg       0.86      0.90      0.88        60
  weighted avg       0.96      0.95      0.95        60
```

In [81]:
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(30, 20))
plot_tree(model, filled=True)
plt.show()
```



In [82]:
```
merged_orders_offices_mergedfully.to_excel("Zimson_Final_Data.xlsx", index=False)
```

In [83]:
```
import joblib
joblib.dump(model, "zimson_decision_tree_model.pkl")
```

Out[83]: ['zimson_decision_tree_model.pkl']

In [84]:
```
merged_orders_offices_mergedfully.to_csv("zimson_final_dataset.csv", index=False)
```

In [ ]: