Section 1: MySQL Tasks

1. Customer Insights

Query to get the total number of customers who signed up in 2023:

```sql
SELECT COUNT(*) as total_customers
FROM customers
WHERE YEAR(signup_date) = 2023;
```

2. Sales Analysis

Query to calculate total revenue for each month in 2023, grouped by month:

```sql
SELECT MONTH(order_date) as month,
       SUM(total_amount) as total_revenue
FROM orders
WHERE YEAR(order_date) = 2023
GROUP BY MONTH(order_date)
ORDER BY month;
```

3. Top Selling Products

Query to find the top 3 most sold products based on order quantities:

```sql
SELECT p.product_name,
       SUM(oi.quantity) as total_quantity
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_quantity DESC
LIMIT 3;
```

4. Customer Order Value

Query to retrieve customers with total spend more than $500:

```sql
SELECT c.name,
       SUM(o.total_amount) as total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
HAVING total_spent > 500;
```

5. Stock Alert

Query to list products with stock quantity below 50:

```sql
SELECT product_name,
       ck_quan as stock_quantity
FROM products
WHERE ck_quan < 50;
```

6. Refund Handling

Query to identify completed orders with no matching payment records:

```sql
SELECT o.order_id,
       o.order_date,
       o.total_amount
FROM orders o
LEFT JOIN payments p ON o.order_id = p.order_id
WHERE o.status = 'Completed'
AND p.payment_id IS NULL;
```

Section 2: Python Data Analysis Challenges

2. Revenue Calculation:

Create a new column total_price (quantity × price).Find the total revenue generated per product category

```python
from matplotlib.ticker import ScalarFormatter
revenue_per_category = df.groupby('category')['total_price'].sum().reset_index()
revenue_per_category.sort_values(by = 'total_price' ,  ascending = False )
print(revenue_per_category)
```

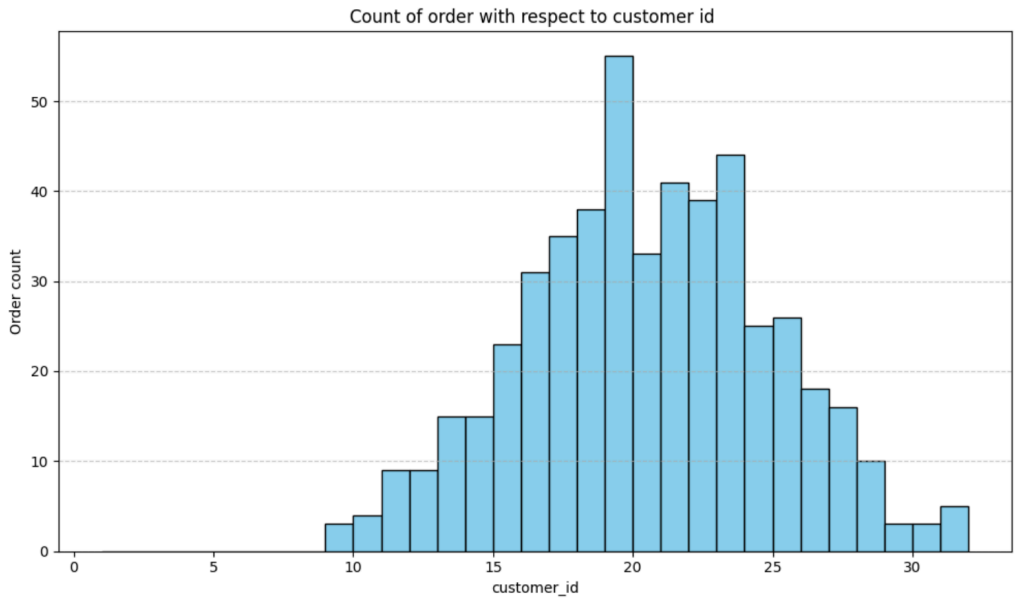
3. Customer Retention:

Identify repeat customers who placed more than 2 orders.

[25]:
```
repeat_customers = df.groupby('customer_id')['order_id'].count().reset_index()
print(repeat_customers)
```

```
     customer_id  order_id
0              1        17
1              2        17
2              3        16
3              4        15
4              5        15
..           ...       ...
495          496        13
496          497        17
497          498        20
498          499        23
499          500        25

[500 rows x 2 columns]
```



4. Sales Trend:

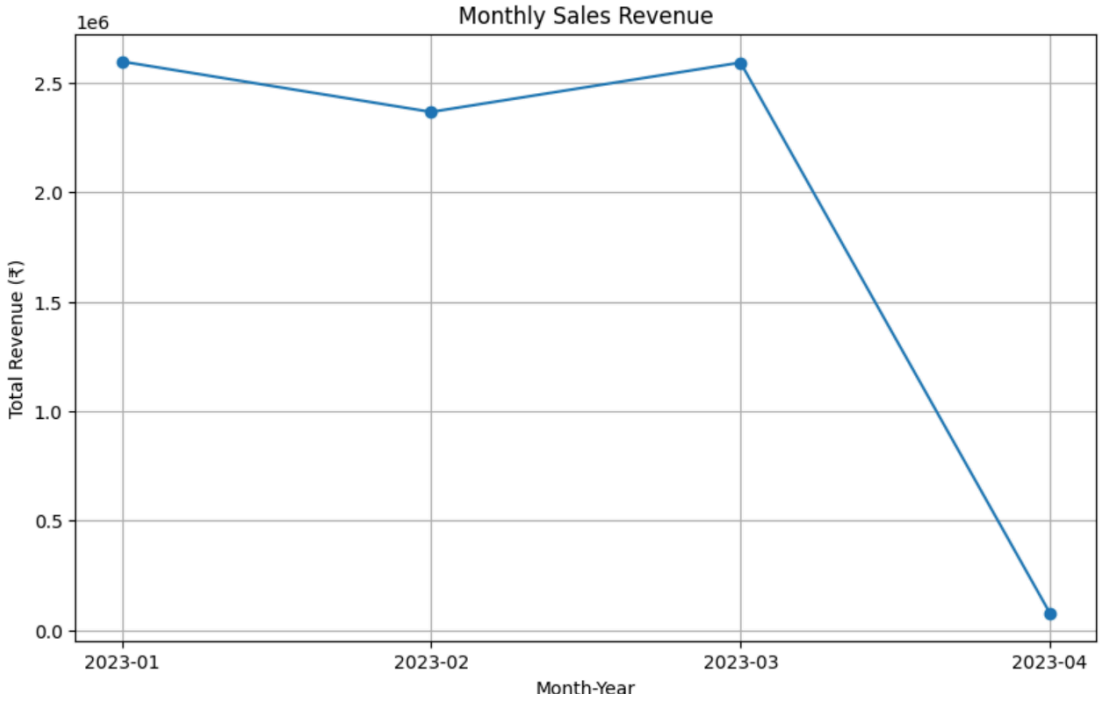
Create a time series plot of monthly sales revenue using Matplotlib/Seaborn.

```
df['month_year'] = df['order_date'].dt.to_period('M')

# Group by month_year and calculate total revenue per month
monthly_sales = df.groupby('month_year')['total_price'].sum().reset_index()

# Plot the time series of monthly sales revenue
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(monthly_sales['month_year'].astype(str), monthly_sales['total_price'], marker='o'
plt.title('Monthly Sales Revenue')
plt.xlabel('Month-Year')
plt.ylabel('Total Revenue (₹)')
plt.xticks(rotation=0)
plt.grid(True)
plt.show()
```



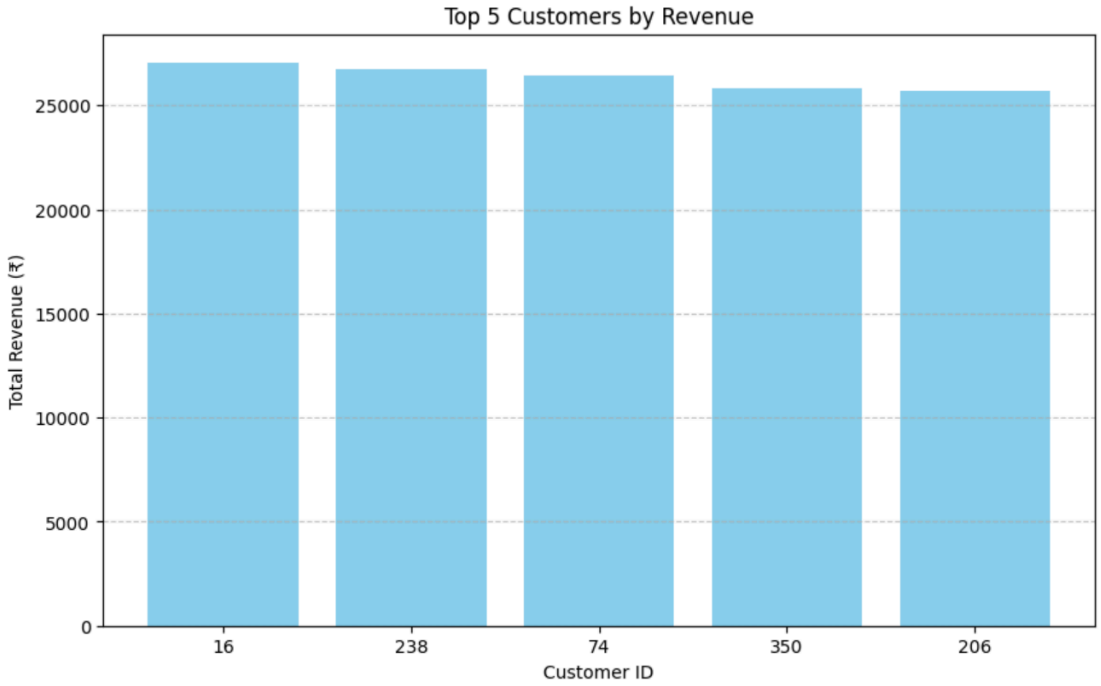
5 .Top 5 Customers by Revenue:

```python
# Group by 'customer_id' and calculate the total revenue for each customer
top_customers = df.groupby('customer_id')['total_price'].sum().reset_index()

# Sort the customers by total revenue in descending order and select the top 5
top_5_customers = top_customers.sort_values(by='total_price', ascending=False).head(5)

# Plot the top 5 customers by revenue
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(top_5_customers['customer_id'].astype(str), top_5_customers['total_price'], color='skyblue')
plt.title('Top 5 Customers by Revenue')
plt.xlabel('Customer ID')
plt.ylabel('Total Revenue (₹)')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



6. Category Performance:

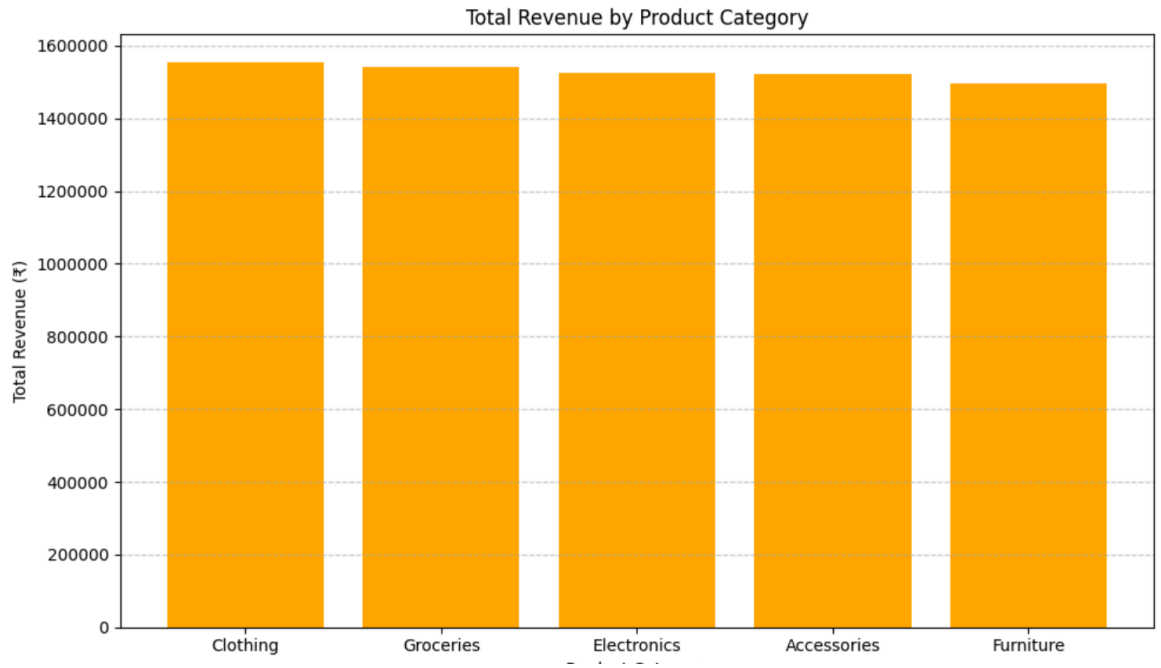
Create a bar chart comparing total revenue across different product categories

```python
# Group by 'category' and calculate total revenue
category_revenue = df.groupby('category')['total_price'].sum().reset_index()

# Sort categories by revenue (optional)
category_revenue = category_revenue.sort_values(by='total_price', ascending=False)

# Plot the bar chart
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(category_revenue['category'], category_revenue['total_price'], color='orange')
plt.title('Total Revenue by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Revenue (₹)')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.gca().yaxis.set_major_formatter(ScalarFormatter(useMathText=False))
plt.ticklabel_format(style='plain', axis='y')  # Ensure y-axis uses plain formatting
plt.tight_layout()
plt.show()
```



Section 3: Advanced Task (Bonus)

🚀 Predictive Analysis using Machine Learning (Optional but Recommended!)

- Use Python's sklearn to predict future sales using a simple Linear Regression model.
- The input features should be order date and product category, and the target variable should be total revenue.

```python
# Step 1: Prepare the data
df['order_date'] = pd.to_datetime(df['order_date'])
df['order_month'] = df['order_date'].dt.month
df['order_year'] = df['order_date'].dt.year

# Step 2: Group data to get total revenue
df['total_price'] = df['quantity'] * df['price']
revenue_data = df.groupby(['order_year', 'order_month', 'category'])['total_price'].sum().reset_index()

# Step 3: Encode categorical variable (category)
encoder = OneHotEncoder()
category_encoded = encoder.fit_transform(revenue_data[['category']]).toarray()
category_df = pd.DataFrame(category_encoded, columns=encoder.get_feature_names_out(['category']))

# Step 4: Combine features
X = pd.concat([revenue_data[['order_year', 'order_month']], category_df], axis=1)
y = revenue_data['total_price']

# Step 5: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 7: Predictions and Evaluation
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("R² Score:", round(r2*100,4))
print("RMSE:", round(rmse, 2))

# Step 8: Visualization
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7, color='green')
plt.xlabel("Actual Revenue")
```