Case 1) Identify the months with the highest and lowest customer acquisition count. What strategies could be implemented to address the fluctuations and ensure consistent growth throughout the year?

```
import pandas as pd

# Load sales data
sales = pd.read_csv('/content/sample_data/Online_Sales.csv')
customer = pd.read_excel('/content/sample_data/CustomersData.xlsx')
sales['Transaction_Date'] = pd.to_datetime(sales['Transaction_Date'])
sales['sale_month'] = sales['Transaction_Date'].dt.month

# Find acquisition date per customer
minData = sales.groupby('CustomerID').agg({'Transaction_Date':'min'}).reset_index()
minData = minData.rename(columns={'CustomerID' :'Customer_Count'})
minData['sale_month'] = minData['Transaction_Date'].dt.month

# Count customers acquired per month
monthCount = minData.groupby('sale_month').agg({'Customer_Count':'count'}).reset_index()

print(monthCount.sort_values('sale_month').reset_index(drop=True))

print("------------------Min-------------")
print(monthCount.loc[[monthCount['Customer_Count'].idxmin()]].reset_index(drop=True))

print("------------------Max-------------")
print(monthCount.loc[[monthCount['Customer_Count'].idxmax()]].reset_index(drop=True))
```

```
    sale_month  Customer_Count
0            1             215
1            2              96
2            3             177
3            4             163
4            5             112
5            6             137
6            7              94
7            8             135
8            9              78
9           10              87
10          11              68
11          12             106
------------------Min-------------
   sale_month  Customer_Count
0          11              68
------------------Max-------------
   sale_month  Customer_Count
0           1             215
```

Case 2) Analyze the data to determine if certain months consistently show higher or lower acquisition. How can the company capitalize on high-performing months and improve performance during slower periods?

```
coupon = pd.read_csv('/content/sample_data/Discount_Coupon.csv')
marketing = pd.read_csv('/content/sample_data/Marketing_Spend.csv')

#Marketing data analysys start

maxmonth = monthCount.loc[monthCount['Customer_Count'].idxmax() , 'sale_month']
minMonth = monthCount.loc[monthCount['Customer_Count'].idxmin() , 'sale_month']
marketing['Date'] = pd.to_datetime(marketing['Date'])
marketing['month'] = marketing['Date'].dt.month

maxMonthData = marketing[marketing['month'] == maxmonth]
minMonthData = marketing[marketing['month'] == minMonth]
```

```
print("=== Marketing Spend Comparison ===")
print("Offline Spend - Max Month:", maxMonthData['Offline_Spend'].sum(),
      "| Min Month:", minMonthData['Offline_Spend'].sum())
print("Online Spend  - Max Month:", maxMonthData['Online_Spend'].sum(),
      "| Min Month:", minMonthData['Online_Spend'].sum())

print('-' * 100)

# --- Step 3: Coupon data analysis ---
coupon['month_num'] = pd.to_datetime(coupon['Month'], format='%b').dt.month

maxMonthCoupon = coupon[coupon['month_num'] == maxmonth]
minMonthCoupon = coupon[coupon['month_num'] == minMonth]

print("=== Coupon Discount Comparison ===")
print("Total Discount % - Max Month:", maxMonthCoupon['Discount_pct'].mean(),
      "| Min Month:", minMonthCoupon['Discount_pct'].mean())


#higher acquisition month we spend more amount we spend on Offline_Spend we increase the sales
maxMonthData['Offline_Spend'].sum()
```

```
=== Marketing Spend Comparison ===
Offline Spend - Max Month: 96600 | Min Month: 93000
Online Spend  - Max Month: 58328.950000000004 | Min Month: 68144.96
----------------------------------------------------------------------------------------------------
=== Coupon Discount Comparison ===
Total Discount % - Max Month: 10.0 | Min Month: 20.0
np.int64(96600)
```

```
Start coding or generate with AI.
```

3. Identify periods with the strongest and weakest retention rates. What strategies could be implemented to improve retention during weaker months?

```
import numpy as np
import pandas as pd

online_Sale = sales.copy()
online_Sale['sale_month'] = online_Sale['Transaction_Date'].dt.month

# Grouping customers by month
month_customers = online_Sale.groupby('sale_month')['CustomerID'].unique().reset_index()

def calculateRetention(customer):
    # create empty column for retention rate
    customer['retention_rate'] = np.nan

    for i in range(len(customer) - 1):
        similerCustomer = np.intersect1d(customer.loc[i, 'CustomerID'],
                                         customer.loc[i + 1, 'CustomerID'])
        customer.loc[i, 'retention_rate'] = round(
            (len(similerCustomer) / len(customer.loc[i, 'CustomerID'])) * 100, 2
        )

    return customer

# Calculate retention
month_customers = calculateRetention(month_customers)

# Print retention table
print("Retention Table:\n", month_customers)

# Strongest retention month
print("\nStrongest Retention Month:")
print(month_customers.loc[month_customers['retention_rate'].idxmax()])

# Weakest retention month
print("\nWeakest Retention Month:")
print(month_customers.loc[month_customers['retention_rate'].idxmin()])
```

```
# Get month numbers
maxmonth1 = month_customers.loc[month_customers['retention_rate'].idxmax(), 'sale_month']
minMonth1 = month_customers.loc[month_customers['retention_rate'].idxmin(), 'sale_month']

# --- Step 2: Marketing spend comparison ---
marketing['Date'] = pd.to_datetime(marketing['Date'])
marketing['month'] = marketing['Date'].dt.month

maxMonthData1 = marketing[marketing['month'] == maxmonth1]
minMonthData1 = marketing[marketing['month'] == minMonth1]

print("\n=== Marketing Spend Comparison ===")
print("Offline Spend - Max Month:", maxMonthData1['Offline_Spend'].sum(),
      "| Min Month:", minMonthData1['Offline_Spend'].sum())
print("Online Spend  - Max Month:", maxMonthData1['Online_Spend'].sum(),
      "| Min Month:", minMonthData1['Online_Spend'].sum())

print('-' * 100)

# --- Step 3: Coupon data analysis ---
coupon['month_num'] = pd.to_datetime(coupon['Month'], format='%b').dt.month

maxMonthCoupon1 = coupon[coupon['month_num'] == maxmonth1]
minMonthCoupon1 = coupon[coupon['month_num'] == minMonth1]

print("=== Coupon Discount Comparison ===")
print("Total Discount % - Max Month:", maxMonthCoupon1['Discount_pct'].sum(),
      "| Min Month:", minMonthCoupon1['Discount_pct'].sum())

# Strongest retention month: July (27.54%)
# Weakest retention month: January (6.05%)

# In January, retention was poor despite higher marketing spend, while July achieved better retention with lower spend. Coupons we
```

```
Retention Table:
     sale_month                                         CustomerID  \
0            1  [17850, 13047, 12583, 13748, 15100, 15291, 146...
1            2  [13370, 16883, 13520, 12841, 16905, 12427, 179...
2            3  [14606, 15953, 16143, 12720, 12748, 16422, 127...
3            4  [16016, 14060, 17722, 17324, 17341, 14051, 159...
4            5  [15514, 13448, 13787, 17865, 17220, 13798, 165...
5            6  [17841, 17017, 15107, 12829, 14051, 13097, 128...
6            7  [17841, 14472, 17358, 15581, 14911, 16686, 182...
7            8  [13081, 13280, 15845, 13488, 16656, 13715, 151...
8            9  [14646, 17685, 15159, 15033, 15570, 14329, 153...
9           10  [13854, 18061, 18223, 17504, 14320, 14646, 178...
10          11  [12474, 17389, 16714, 13098, 15661, 17027, 123...
11          12  [15602, 17619, 12778, 14502, 13089, 17256, 167...

    retention_rate
0             6.05
1            10.09
2            11.54
3            11.16
4            18.50
5            22.39
6            27.54
7            14.67
8            15.03
9            14.76
10           14.89
11             NaN

Strongest Retention Month:
sale_month                                                      7
CustomerID       [17841, 14472, 17358, 15581, 14911, 16686, 182...
retention_rate                                               27.54
Name: 6, dtype: object

Weakest Retention Month:
sale_month                                                      1
CustomerID       [17850, 13047, 12583, 13748, 15100, 15291, 146...
retention_rate                                                6.05
Name: 0, dtype: object

=== Marketing Spend Comparison ===
Offline Spend - Max Month: 67500 | Min Month: 96600
Online Spend  - Max Month: 52717.85 | Min Month: 58328.950000000004
----------------------------------------------------------------------------------------------------
```

```
=== Coupon Discount Comparison ===
Total Discount % - Max Month: 170 | Min Month: 170
```

4. Analyze customer behavior during high-retention months and suggest ways to replicate this success throughout the year.# New section

```
# 1. Find high and low retention months
highRetentionMonth = month_customers.loc[month_customers['retention_rate'].idxmax(), 'sale_month']
lowRetentionMonth  = month_customers.loc[month_customers['retention_rate'].idxmin(), 'sale_month']

# 2. Filter sales for those months
maxMonthSale = sales[sales['sale_month'] == highRetentionMonth]
minMonthSale = sales[sales['sale_month'] == lowRetentionMonth]

# 3. Group by Product & Coupon status
max_group = maxMonthSale.groupby(['Product_Category', 'Coupon_Status']).size().reset_index(name='MaxMonth_Count')
min_group = minMonthSale.groupby(['Product_Category', 'Coupon_Status']).size().reset_index(name='MinMonth_Count')

# 4. Merge for comparison
comparison = pd.merge(max_group, min_group, on=['Product_Category', 'Coupon_Status'], how='outer').fillna(0)

# 5. Get the top category where coupon usage is highest
usedCoupon = comparison[comparison['Coupon_Status'] == 'Used']
top_used   = usedCoupon.loc[usedCoupon['MaxMonth_Count'].idxmax()]

print(top_used)

# 📊 Analysis (auto-insertable if you want later)
print(f"""
During the high-retention month, {int(top_used['MaxMonth_Count'])} transactions were made with coupons.
In the low-retention month, even though overall sales were weaker, {int(top_used['MinMonth_Count'])} transactions still used coupc
This indicates coupon usage plays a significant role in driving sales for {top_used['Product_Category']}.
""")
```

```
Product_Category    Apparel
Coupon_Status          Used
MaxMonth_Count        720.0
MinMonth_Count        338.0
Name: 8, dtype: object

During the high-retention month, 720 transactions were made with coupons.
In the low-retention month, even though overall sales were weaker, 338 transactions still used coupons.
This indicates coupon usage plays a significant role in driving sales for Apparel.
```

5. Compare the revenue generated by new and existing customers month-over-month. What does this trend suggest about the balance between acquisition and retention efforts?

```
# Calculate total revenue for new and existing customers each month

# Create a dataframe with the first transaction date for each customer
first_transaction_date = sales.groupby('CustomerID')['Transaction_Date'].min().reset_index()
first_transaction_date.columns = ['CustomerID', 'First_Transaction_Date']

# Merge the first transaction date with the sales data
sales_with_first_transaction = pd.merge(sales, first_transaction_date, on='CustomerID')

# Identify new and existing customers for each transaction
sales_with_first_transaction['Customer_Type'] = sales_with_first_transaction.apply(
    lambda row: 'New' if row['Transaction_Date'].month == row['First_Transaction_Date'].month else 'Existing', axis=1
)

# Calculate revenue for each transaction (Quantity * Avg_Price)
sales_with_first_transaction['Revenue'] = sales_with_first_transaction['Quantity'] * sales_with_first_transaction['Avg_Price']

# Group by month and customer type to calculate total revenue
monthly_revenue = sales_with_first_transaction.groupby(['sale_month', 'Customer_Type'])['Revenue'].sum().reset_index()
```

```
# Pivot the table for easier comparison
monthly_revenue_pivot = monthly_revenue.pivot(index='sale_month', columns='Customer_Type', values='Revenue').fillna(0)

print("Monthly Revenue Comparison (New vs. Existing Customers):\n")
display(monthly_revenue_pivot)
```

```
Monthly Revenue Comparison (New vs. Existing Customers):
```

Customer_Type	Existing	New
sale_month		
1	0.00	403624.58
2	39230.64	271589.16
3	54741.54	294866.55
4	168105.33	233513.09
5	110574.61	197188.81
6	128924.97	192156.41
7	222701.53	149936.54
8	209665.62	191544.75
9	213977.90	146570.50
10	192164.80	217516.48
11	291746.31	217196.31
12	276448.22	246809.97

Next steps: Generate code with `monthly_revenue_pivot` New interactive sheet

6. Analyze the relationship between coupon usage and revenue generation. How can discount strategies be optimized to maximize revenue while maintaining profitability?

```
# Copy sales data
coupon_revenue = sales.copy()

# Handle missing values
coupon_revenue['Avg_Price'] = coupon_revenue['Avg_Price'].fillna(0)
coupon_revenue['Quantity']  = coupon_revenue['Quantity'].fillna(0)


# Calculate revenue
coupon_revenue['Revenue'] = (
    coupon_revenue['Quantity'] * coupon_revenue['Avg_Price']
)

# Simplify coupon usage status (binary split: Used / Not Used)
coupon_revenue['Coupon_Usage'] = coupon_revenue['Coupon_Status'].apply(
    lambda x: "Used" if x == "Used" else "Not Used"
)

# Aggregate revenue metrics by coupon usage
revenue_table = (
    coupon_revenue
    .groupby('Coupon_Usage', as_index=False)['Revenue']
    .agg(
        Avg_Revenue='mean',
        Total_Revenue='sum',
        Transactions='count'
    )
)

# Calculate contribution percentage
revenue_table['Revenue_Share(%)'] = (
    revenue_table['Total_Revenue'] /
    revenue_table['Total_Revenue'].sum() * 100
```

```
)

# Format results for readability
revenue_table['Avg_Revenue'] = revenue_table['Avg_Revenue'].round(2)
revenue_table['Total_Revenue'] = revenue_table['Total_Revenue'].round(2)
revenue_table['Revenue_Share(%)'] = revenue_table['Revenue_Share(%)'].round(2)

print(revenue_table)
```

```
  Coupon_Usage  Avg_Revenue  Total_Revenue  Transactions  Revenue_Share(%)
0     Not Used        88.81     3109976.52         35020             66.58
1         Used        87.18     1560818.10         17904             33.42
```

Coupon transactions account for one-third of total revenue, with average spend per transaction nearly equal to non-coupon users. This suggests that coupons effectively drive sales volume without significantly lowering basket value. To optimize, the business should implement targeted discounts (e.g., for new or dormant customers) and introduce minimum spend thresholds to lift order sizes. Additionally, shifting some offers toward non-monetary incentives such as free shipping or loyalty points can help sustain revenue growth while maintaining profitability.

7. Identify the top-performing products and analyze the factors driving their success. How can this insight inform inventory management and promotional

```
# Aggregate product performance metrics by category
product_performance = (
    coupon_revenue
    .groupby('Product_Category')
    .agg(
        Total_Revenue=('Revenue', 'sum'),
        Transaction_Count=('Revenue', 'count'),
        Total_Units_Sold=('Quantity', 'sum')
    )
    .sort_values('Total_Revenue', ascending=False)
)

# Round total revenue for clean presentation
product_performance['Total_Revenue'] = product_performance['Total_Revenue'].round(2)

# Calculate average revenue earned per unit sold (premium vs budget indicator)
product_performance['Revenue_Per_Unit'] = (
    product_performance['Total_Revenue'] / product_performance['Total_Units_Sold']
).round(2)

# Calculate percentage contribution of each category to overall revenue
product_performance['Revenue_Share_%'] = (
    (product_performance['Total_Revenue'] / product_performance['Total_Revenue'].sum()) * 100
).round(2)

# Display top 5 product categories by revenue
product_performance.head(5)
```

Product_Category	Total_Revenue	Transaction_Count	Total_Units_Sold	Revenue_Per_Unit	Revenue_Share_%
Nest-USA	2554202.39	14013	21430	119.19	54.68
Apparel	591145.80	18126	32438	18.22	12.66
Nest	518193.50	2198	2837	182.66	11.09
Office	276794.40	6513	88383	3.13	5.93
Drinkware	200707.83	3483	30501	6.58	4.30

Next steps: Generate code with `product_performance` New interactive sheet

8 Analyze the relationship between monthly marketing spend and revenue. Are there any months where marketing efforts yielded disproportionately high or low returns? How can marketing strategies be adjusted to improve ROI?

```
sales_group = coupon_revenue.groupby('sale_month')['Revenue'].sum()
marketing_group = marketing.groupby('month').agg(
    Total_Offline_Spend=('Offline_Spend', 'sum'),
    Total_Online_Spend=('Online_Spend', 'sum')
)
marketing_group['total_marketing_spend'] = marketing_group['Total_Offline_Spend'] + marketing_group['Total_Online_Spend']

ROI_table = pd.merge(sales_group, marketing_group, left_index=True, right_index=True)
ROI_table['ROI'] = ((ROI_table['Revenue'] - ROI_table['total_marketing_spend'])*100 / ROI_table['total_marketing_spend']).round(2)

low = ROI_table['ROI'].quantile(0.25)
high = ROI_table['ROI'].quantile(0.75)

def classify(x):
    if x <= low:
        return 'Low'
    elif x <= high:
        return 'Medium'
    else:
        return 'High'

ROI_table['ROI_Status'] = ROI_table['ROI'].apply(classify)


print(ROI_table)
```

```
              Revenue  Total_Offline_Spend  Total_Online_Spend  \
sale_month
1           403624.58                96600            58328.95
2           310819.80                81300            55807.92
3           349608.09                73500            48750.09
4           401618.42                96000            61026.83
5           307763.42                65500            52759.64
6           321081.38                80500            53818.14
7           372638.07                67500            52717.85
8           401210.37                85500            57404.15
9           360548.40                83000            52514.54
10          409681.28                93500            57724.65
11          508942.62                93000            68144.96
12          523258.19               122000            76648.75

            total_marketing_spend     ROI ROI_Status
sale_month
1                      154928.95  160.52     Medium
2                      137107.92  126.70        Low
3                      122250.09  185.98       High
4                      157026.83  155.76        Low
5                      118259.64  160.24     Medium
6                      134318.14  139.05        Low
7                      120217.85  209.97       High
8                      142904.15  180.75     Medium
9                      135514.54  166.06     Medium
10                     151224.65  170.91     Medium
11                     161144.96  215.83       High
12                     198648.75  163.41     Medium
```

9. Evaluate the effectiveness of marketing campaigns by comparing marketing spend to revenue generated. Are there opportunities to reallocate resources for better results?

```
# 1. Keep a copy to avoid overwriting ROI_table
marketingPercentageTable = ROI_table.copy()

# 2. Monthly spend % calculation
marketingPercentageTable['Market_spend_%'] = (
    (marketingPercentageTable['total_marketing_spend'] * 100) / marketingPercentageTable['Revenue']
```

```
).round(2)

# 3. Reset index to work with sale_month
marketingPercentageTable = marketingPercentageTable.reset_index()

# 4. Convert sale_month → proper datetime (attach a dummy year for quarter extraction)
marketingPercentageTable['Date'] = pd.to_datetime(
    marketingPercentageTable['sale_month'].astype(str) + "-2025",
    format="%m-%Y",
    errors="coerce"
)

# 5. Extract quarter
marketingPercentageTable['Quarter'] = marketingPercentageTable['Date'].dt.to_period('Q')

# 6. Quarterly aggregation
Quarter_analysis = marketingPercentageTable.groupby('Quarter').agg(
    Quarter_revenue=('Revenue', 'sum'),
    Quarter_spend=('total_marketing_spend', 'sum')
).reset_index()

Quarter_analysis['Quarter_Market_spend_%'] = (
    (Quarter_analysis['Quarter_spend'] * 100) / Quarter_analysis['Quarter_revenue']
).round(2)

# ------------------------------
# 7. Final outputs
# ------------------------------
print("=== Monthly Analysis ===")
print(marketingPercentageTable[['sale_month', 'Revenue', 'total_marketing_spend', 'Market_spend_%']])

print("\n=== Quarterly Analysis ===")
print(Quarter_analysis)
```

```
=== Monthly Analysis ===
    sale_month    Revenue  total_marketing_spend  Market_spend_%
0            1  403624.58              154928.95           38.38
1            2  310819.80              137107.92           44.11
2            3  349608.09              122250.09           34.97
3            4  401618.42              157026.83           39.10
4            5  307763.42              118259.64           38.43
5            6  321081.38              134318.14           41.83
6            7  372638.07              120217.85           32.26
7            8  401210.37              142904.15           35.62
8            9  360548.40              135514.54           37.59
9           10  409681.28              151224.65           36.91
10          11  508942.62              161144.96           31.66
11          12  523258.19              198648.75           37.96

=== Quarterly Analysis ===
  Quarter  Quarter_revenue  Quarter_spend  Quarter_Market_spend_%
0  2025Q1       1064052.47      414286.96                   38.93
1  2025Q2       1030463.22      409604.61                   39.75
2  2025Q3       1134396.84      398636.54                   35.14
3  2025Q4       1441882.09      511018.36                   35.44
```

10. Segment customers into groups such as Premium, Gold, Silver, and Standard. What targeted strategies can be developed for each segment to improve retention and revenue? (Use RFM segmentation techniques)

```
segment_table  = sales
LastTransactionDate = sales['Transaction_Date'].max();
segment_table['Revenue'] = segment_table['Quantity'] * segment_table['Avg_Price']
RFM_Segmentation = segment_table.groupby('CustomerID').agg(
    Recent_purcheded = ('Transaction_Date' , 'max'),
    Frequency_count = ('Transaction_ID' , 'count'),
    Total_purches_amount = ('Revenue' , 'sum')

    ).reset_index()
RFM_Segmentation['Recent_purcheded'] = pd.to_datetime(RFM_Segmentation['Recent_purcheded'])
RFM_Segmentation['Days_ago'] =  (pd.to_datetime(LastTransactionDate) - RFM_Segmentation['Recent_purcheded']).dt.days
RFM_Segmentation['R_Score'] = pd.qcut(RFM_Segmentation['Days_ago'],4, labels=[ 4, 3, 2, 1]).astype(int)
RFM_Segmentation['F_Score'] = pd.qcut(RFM_Segmentation['Frequency_count'],4, labels=[ 1, 2, 3, 4]).astype(int)
```

```
RFM_Segmentation['M_Score'] = pd.qcut(RFM_Segmentation['Total_purches_amount'] ,4, labels=[ 1, 2, 3, 4]).astype(int)
RFM_Segmentation['RFM_Score'] = RFM_Segmentation['R_Score'] + RFM_Segmentation['F_Score']+ RFM_Segmentation['M_Score']

def classify_customer(score):
    if score <= 6:
        return 'Standard'
    elif score <= 9:
        return 'Silver'
    elif score <= 11:
        return 'Gold'
    else:
        return 'Premium'

RFM_Segmentation['Customer_Segment'] = RFM_Segmentation['RFM_Score'].apply(classify_customer)

print('Customer Segment Result')
print(RFM_Segmentation)
```

```
Customer Segment Result
      CustomerID Recent_purcheded  Frequency_count  Total_purches_amount  \
0          12346       2019-09-15                2                 30.99
1          12347       2019-11-02               60              13834.90
2          12348       2019-10-19               23               1442.12
3          12350       2019-12-14               17               1360.07
4          12356       2019-09-15               36               1442.47
...          ...              ...              ...                   ...
1463       18259       2019-04-05                7                544.34
1464       18260       2019-10-05               40               2363.05
1465       18269       2019-06-20                8                101.56
1466       18277       2019-10-23                1                298.00
1467       18283       2019-10-10              102               6362.77

      Days_ago  R_Score  F_Score  M_Score  RFM_Score Customer_Segment
0          107        3        1        1          5         Standard
1           59        3        4        4         11             Gold
2           73        3        3        2          8           Silver
3           17        4        2        2          8           Silver
4          107        3        3        2          8           Silver
...        ...      ...      ...      ...        ...              ...
1463       270        1        1        1          3         Standard
1464        87        3        3        3          9           Silver
1465       194        2        1        1          4         Standard
1466        69        3        1        1          5         Standard
1467        82        3        4        4         11             Gold

[1468 rows x 10 columns]
```

11.Analyze the revenue contribution of each customer segment. How can the company focus its efforts on high-value segments while nurturing lower-value

```
# Aggregate revenue and customer count by segment
revenue_segments = (
    RFM_Segmentation
    .groupby('Customer_Segment')['Total_purches_amount']
    .agg(['sum', 'count'])
    .rename(columns={'sum': 'Revenue_Contribution', 'count': 'Customer_Count'})
    .sort_values(by='Revenue_Contribution', ascending=False)
)

# Calculate revenue contribution percentage
revenue_segments['Contribution_Percentage'] = (
    (revenue_segments['Revenue_Contribution'] / revenue_segments['Revenue_Contribution'].sum()) * 100
).round(2)

# Display results
print(revenue_segments)

# Key business insight
print(    "Premium and Gold customers (~50% of the base) drive ~86% of total revenue. ")
```

```
                 Revenue_Contribution  Customer_Count  \
Customer_Segment
Gold                      1596428.02             270
```

```
Premium                    1376595.36            120
Silver                     1331599.65            498
Standard                    366171.59            580

                  Contribution_Percentage
Customer_Segment
Gold                                34.18
Premium                             29.47
Silver                              28.51
Standard                             7.84
Premium and Gold customers (~50% of the base) drive ~86% of total revenue.
```

12. Group customers by their month of first purchase and analyze retention rates over time. Which cohorts exhibit the highest and lowest retention rates? What strategies can be implemented to improve retention for weaker cohorts?

```
# ===== Cohort Retention Analysis =====

# Step 1: Prepare data
cohort_data = sales.copy()
cohort_data['purchase_month'] = cohort_data['Transaction_Date'].dt.month

# Step 2: Assign each customer to their cohort (first purchase month)
first_purchase = cohort_data.groupby('CustomerID')['purchase_month'].min()
cohort_data['cohort_month'] = cohort_data['CustomerID'].map(first_purchase)

# Step 3: Count unique customers by cohort and purchase month
cohort_counts = (
    cohort_data
    .groupby(['cohort_month', 'purchase_month'])['CustomerID']
    .nunique()
    .reset_index()
)

# Step 4: Pivot to create cohort matrix
cohort_matrix = (
    cohort_counts
    .pivot(index='purchase_month', columns='cohort_month', values='CustomerID')
    .fillna(0)
)

# Step 5: Get initial cohort sizes (baseline for retention)
cohort_sizes = cohort_matrix[cohort_matrix != 0].max()

# Step 6: Calculate retention rates
retention_matrix = cohort_matrix.divide(cohort_sizes, axis=1)

# Step 7: Convert to percentage format
retention_matrix = (retention_matrix * 100).round(2).astype(str) + '%'

# Final Result
print("\n=== Cohort Retention Table (in %) ===\n")
retention_matrix
print(retention_matrix.to_string())
```

```
=== Cohort Retention Table (in %) ===

cohort_month        1       2       3       4       5       6       7       8       9       10      11      12
purchase_month
1               100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
2                6.05%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
3               11.16%   7.29%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
4               15.81%   9.38%  10.17%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
5                10.7%  16.67%  19.77%   8.59%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
6               20.47%  17.71%  14.12%  14.72%  10.71%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
7               16.28%  22.92%  18.08%  14.72%   8.04%   14.6%  100.0%    0.0%    0.0%    0.0%    0.0%    0.0%
8               21.86%  19.79%  18.64%  11.04%  11.61%  16.06%  13.83%  100.0%    0.0%    0.0%    0.0%    0.0%
9                10.7%  15.62%  12.43%    9.2%   8.93%   8.76%   4.26%  10.37%  100.0%    0.0%    0.0%    0.0%
10              13.02%   12.5%  12.43%   6.13%  11.61%   8.03%   6.38%  11.11%   7.69%  100.0%    0.0%    0.0%
11                9.3%  11.46%   8.47%   9.82%   12.5%  10.22%   11.7%   7.41%   3.85%    6.9%  100.0%    0.0%
12              15.81%  16.67%  10.73%   7.36%   7.14%   8.03%   9.57%   5.93%   2.56%    4.6%  10.29%  100.0%
```

13. Analyze the lifetime value of customers acquired in different months. How can this insight inform acquisition and retention strategies?

```
# ===== Customer Lifetime Value (LTV) Analysis =====

# Step 1: Prepare data
ltv_data = sales.copy()
ltv_data['purchase_month'] = ltv_data['Transaction_Date'].dt.to_period('M')

# Step 2: Assign each customer to their acquisition cohort (first purchase month)
first_purchase_month = ltv_data.groupby('CustomerID')['purchase_month'].min()
ltv_data['acquisition_cohort'] = ltv_data['CustomerID'].map(first_purchase_month)

# Step 3: Calculate revenue per transaction
ltv_data['Revenue'] = ltv_data['Quantity'] * ltv_data['Avg_Price']

# Step 4: Group by acquisition cohort and purchase month to get monthly revenue per cohort
monthly_cohort_revenue = ltv_data.groupby(['acquisition_cohort', 'purchase_month'])['Revenue'].sum().reset_index()

# Step 5: Calculate cumulative revenue for each cohort over time
def cumulative_revenue_by_cohort(df, cohort_col, period_col, revenue_col):
    # Create a period number for each month relative to the acquisition month
    df['period_number'] = (df[period_col] - df[cohort_col]).apply(lambda x: x.n + 1)

    # Group by cohort and period number and sum revenue
    cohort_revenue = df.groupby([cohort_col, 'period_number'])[revenue_col].sum().reset_index()

    # Pivot to get cumulative revenue in wide format
    cumulative_revenue = cohort_revenue.pivot_table(index=cohort_col, columns='period_number', values=revenue_col).fillna(0)

    # Calculate cumulative sum across periods
    cumulative_revenue = cumulative_revenue.cumsum(axis=1)
    return cumulative_revenue

cohort_ltv = cumulative_revenue_by_cohort(ltv_data, 'acquisition_cohort', 'purchase_month', 'Revenue')

# Step 6: Calculate Average LTV for each cohort
# Get the size of each cohort
cohort_sizes = ltv_data.groupby('acquisition_cohort')['CustomerID'].nunique()

# Divide cumulative revenue by cohort size to get average LTV
average_cohort_ltv = cohort_ltv.divide(cohort_sizes, axis=0)

print("\n=== Average Customer Lifetime Value (LTV) by Acquisition Cohort ===\n")
display(average_cohort_ltv)
```

```
=== Average Customer Lifetime Value (LTV) by Acquisition Cohort ===
```

period_number	1	2	3	4	5	6	7	8	9
acquisition_cohort									
2019-01	1877.323628	2059.791721	2276.749581	2808.574744	2943.837070	3182.871628	3569.685070	3800.791256	3988.29414C
2019-02	2829.053750	2913.382917	3045.777604	3274.898333	3519.261458	3948.308958	4180.742812	4648.890000	4827.494896
2019-03	1665.912712	1897.850734	2103.304124	2276.207684	2537.720282	2937.629379	3208.363842	3432.959774	3572.845028
2019-04	1432.595644	1574.512025	1676.334233	1858.651902	2013.559264	2198.521840	2294.831350	2645.068098	2756.633497
2019-05	1760.614375	1821.978036	1928.119375	2054.107054	2208.261429	2530.009554	2868.239821	2970.523214	2970.523214
2019-06	1402.601533	1478.913504	1571.670000	1683.650511	1756.316642	2024.275401	2137.232190	2137.232190	2137.232190
2019-07	1595.069574	1752.644787	1830.643723	1988.048511	2247.219787	2555.910000	2555.910000	2555.910000	2555.910000
2019-08	1418.850000	1498.222074	1580.295037	1794.784519	1918.606444	1918.606444	1918.606444	1918.606444	1918.606444
2019-09	1879.108974	1903.148205	1935.922051	1944.413333	1944.413333	1944.413333	1944.413333	1944.413333	1944.413333
2019-10	2500.189425	2611.512184	2643.410690	2643.410690	2643.410690	2643.410690	2643.410690	2643.410690	2643.410690
2019-11	3194.063382	3260.171029	3260.171029	3260.171029	3260.171029	3260.171029	3260.171029	3260.171029	3260.171029
2019-12	2328.395943	2328.395943	2328.395943	2328.395943	2328.395943	2328.395943	2328.395943	2328.395943	2328.395943

Next steps: Generate code with `average_cohort_ltv` New interactive sheet

14. Identify seasonal trends in sales by category and location. How can the company prepare for peak and off-peak seasons to maximize revenue?

```
# ----------------------------
# Step 1: Merge Customer & Sales Data
# ----------------------------
customer_sale = pd.merge(sales, customer, on='CustomerID', how='outer')

# ----------------------------
# Step 2: Feature Engineering
# ----------------------------
# Calculate Revenue per transaction
customer_sale['Revenue'] = customer_sale['Quantity'] * customer_sale['Avg_Price']

# Assign Calendar Quarters from Sale Month
customer_sale['Period'] = (customer_sale['sale_month'] - 1) // 3 + 1
customer_sale['Period'] = customer_sale['Period'].map({1: 'Q1', 2: 'Q2', 3: 'Q3', 4: 'Q4'})

# ----------------------------
# Step 3: Seasonal Trends (Category × Location × Quarter)
# ----------------------------
customer_group = (
    customer_sale.groupby(['Product_Category', 'Location', 'Period'])
    .agg(
        Location_Revenue=('Revenue', 'sum'),
        Location_Quantity=('Quantity', 'count')
    )
    .sort_values(by='Location_Revenue', ascending=False)
)

print("\n--- Seasonal Trends (Category × Location × Quarter) ---")
print("Max Sales Combination:\n", customer_group.loc[customer_group['Location_Revenue'].idxmax()])
print("Min Sales Combination:\n", customer_group.loc[customer_group['Location_Revenue'].idxmin()])

# ----------------------------
# Step 4: Overall by Category × Location
# ----------------------------
customer_revenue = (
    customer_sale.groupby(['Product_Category', 'Location'])
    .agg(
        Location_Revenue=('Revenue', 'sum'),
        Location_Quantity=('Quantity', 'count')
```

```
    )
    .sort_values(by='Location_Revenue', ascending=False)
)

print("\n--- Overall by Category × Location ---")
print("Max Combination:\n", customer_revenue.loc[customer_revenue['Location_Revenue'].idxmax()])
print("Min Combination:\n", customer_revenue.loc[customer_revenue['Location_Revenue'].idxmin()])

# ----------------------------
# Step 5: Seasonal by Quarter × Location
# ----------------------------
Period_revenue = (
    customer_sale.groupby(['Period', 'Location'])
    .agg(
        Location_Revenue=('Revenue', 'sum'),
        Location_Quantity=('Quantity', 'count')
    )
    .sort_values(by=['Period', 'Location'], ascending=True)
)

print("\n--- Quarterly Revenue by Location ---")
print("Max Sales:\n", Period_revenue.loc[Period_revenue['Location_Revenue'].idxmax()])
print("Min Sales:\n", Period_revenue.loc[Period_revenue['Location_Revenue'].idxmin()])

# ----------------------------
# Step 6: Yearly by Location
# ----------------------------
Year_revenue = (
    customer_sale.groupby(['Location'])
    .agg(
        Location_Revenue=('Revenue', 'sum'),
        Location_Quantity=('Quantity', 'count')
    )
    .sort_values(by='Location_Revenue', ascending=False)
)

print("\n--- Yearly Revenue by Location ---")
print("Max Location:\n", Year_revenue.loc[Year_revenue['Location_Revenue'].idxmax()])
print("Min Location:\n", Year_revenue.loc[Year_revenue['Location_Revenue'].idxmin()])

# ----------------------------
# Step 7: Yearly by Category
# ----------------------------
Category_revenue = (
    customer_sale.groupby(['Product_Category'])
    .agg(
        Location_Revenue=('Revenue', 'sum'),
        Location_Quantity=('Quantity', 'count')
    )
    .sort_values(by='Location_Revenue', ascending=False)
)

print("\n--- Yearly Revenue by Category ---")
print("Max Category:\n", Category_revenue.loc[Category_revenue['Location_Revenue'].idxmax()])
print("Min Category:\n", Category_revenue.loc[Category_revenue['Location_Revenue'].idxmin()])

# ----------------------------
# Step 8: Final Output
# ----------------------------
print("\n--- Full Seasonal Grouped Data ---")
print(customer_group)
```

```
Name: (Housewares, Washington DC, Q2), dtype: float64

--- Overall by Category × Location ---
Max Combination:
 Location_Revenue     882041.55
Location_Quantity      4855.00
```

```
Location_Quantity      4387.00
Name: (Q4, Chicago), dtype: float64
Min Sales:
 Location_Revenue     37030.38
Location_Quantity      465.00
Name: (Q2, Washington DC), dtype: float64

--- Yearly Revenue by Location ---
Max Location:
 Location_Revenue     1625885.58
Location_Quantity      18380.00
Name: Chicago, dtype: float64
Min Location:
 Location_Revenue     255772.24
Location_Quantity      2732.00
Name: Washington DC, dtype: float64

--- Yearly Revenue by Category ---
Max Category:
 Location_Revenue     2554202.39
Location_Quantity      14013.00
Name: Nest-USA, dtype: float64
Min Category:
 Location_Revenue     711.03
Location_Quantity     43.00
Name: Android, dtype: float64

--- Full Seasonal Grouped Data ---
                                      Location_Revenue  Location_Quantity
Product_Category Location      Period
Nest-USA         Chicago       Q4             269260.24               1450
                 California    Q1             224091.02               1223
                 Chicago       Q1             216516.31               1198
                               Q3             215037.00               1210
                 California    Q4             209542.82               1154
...                                                ...                ...
Gift Cards       New Jersey    Q3                 10.00                  1
Housewares       Washington DC Q3                  9.80                  2
Bottles          New Jersey    Q4                  8.50                  3
Fun              New York      Q4                  7.20                  1
Housewares       Washington DC Q2                  2.00                  1

[341 rows x 2 columns]
```

15. Analyze daily sales trends to identify high-performing and low-performing days. What strategies can be implemented to boost sales on slower days?

```
days_revenue = sales.copy()
days_revenue['sales_day'] = days_revenue['Transaction_Date'].dt.dayofweek  # 0=Monday, 6=Sunday
days_revenue['Revenue'] = days_revenue['Quantity'] * days_revenue['Avg_Price']

# Aggregate daily sales performance
grouped_days = (
    days_revenue.groupby('sales_day')
    .agg(
        Total_Revenue=('Revenue', 'sum'),
        Transaction_Count=('Transaction_Date', 'count'),
        Total_Units_Sold=('Quantity', 'sum')
    )
    .reset_index()
)

# Identify high-performing and low-performing sales days
high_day = grouped_days.loc[grouped_days['Total_Units_Sold'].idxmax()]
low_day = grouped_days.loc[grouped_days['Total_Units_Sold'].idxmin()]

# Filter transactions for high and low performing days
low_day_filter = (
    days_revenue[days_revenue['sales_day'] == low_day['sales_day']]
    .groupby('Coupon_Status')['Coupon_Status']
    .size()
    .reset_index(name='low_day_count')
)

high_day_filter = (
    days_revenue[days_revenue['sales_day'] == high_day['sales_day']]
    .groupby('Coupon_Status')['Coupon_Status']
    .size()
```

```
    .reset_index(name='high_day_count')
)

# Compare coupon usage on high vs low performing days
coupon_comparison = pd.merge(
    high_day_filter,
    low_day_filter,
    on='Coupon_Status',
    how='inner'
)

print("Daily Sales Performance:")
print(grouped_days)
print("\nCoupon Usage Comparison (High vs Low Performing Days):")
print(coupon_comparison)

# --- Business Insight ---
# Observation:
#   Coupon usage is higher on high-performing days.
# Recommendation:
#   Increase coupon distribution and promotions on low-performing days
#   to boost sales and balance demand across the week.
```

```
Daily Sales Performance:
   sales_day  Total_Revenue  Transaction_Count  Total_Units_Sold
0          0      365626.90               4464             11983
1          1      396819.65               4611             11317
```