# Consumer Goods Ad_Hoc Insights

# SQL Requests

## Question

1. Provide the list of markets in which customer "Atliq Exclusive" operates its business in the APAC region

## SQL Code

```
SELECT DISTINCT market
FROM gdb023.dim_customer
WHERE region = "APAC" AND customer = "Atliq Exclusive"
ORDER BY market ASC;
```

## Output

| market |
| --- |
| Australia |
| Bangladesh |
| India |
| Indonesia |
| Japan |
| Newzealand |
| Philiphines |
| South Korea |

# SQL Requests

## Question

2. What is the percentage of unique product increase in 2021 vs. 2020?-- The final output contains these fields: unique_products_2020, unique_products_2021, percentage_chg

## SQL Code

```sql
WITH details AS (
    SELECT
        fiscal_year,
        COUNT(DISTINCT product_code) AS unique_products_count
    FROM
        gdb023.fact_sales_monthly
    GROUP BY
        fiscal_year
)

, details_with_lag AS (
    SELECT
        fiscal_year,
        unique_products_count,
        LAG(unique_products_count) OVER (ORDER BY fiscal_year) AS unique_products_count_LY
    FROM
        details
)
```

```sql
SELECT
    unique_products_count AS unique_products_2021,
    unique_products_count_LY AS unique_products_2020,
    CASE
        WHEN unique_products_count_LY IS NOT NULL AND unique_products_count_LY != 0 THEN
            (unique_products_count - unique_products_count_LY) * 100.0
            / CAST(unique_products_count_LY AS DECIMAL)
        ELSE
            NULL
    END AS percentage_chg
FROM
    details_with_lag
WHERE
    fiscal_year = 2021;
```

# SQL Requests

## Output

| | unique_products_2021 | unique_products_2020 | percentage_chg |
|---|---|---|---|
| ▶ | 334 | 245 | 36.32653 |

## Question

3. Provide a report with all the unique product counts for each segment and sort them in descending order of product counts.

The final output contains these fields: segment, product_count

# SQL Requests

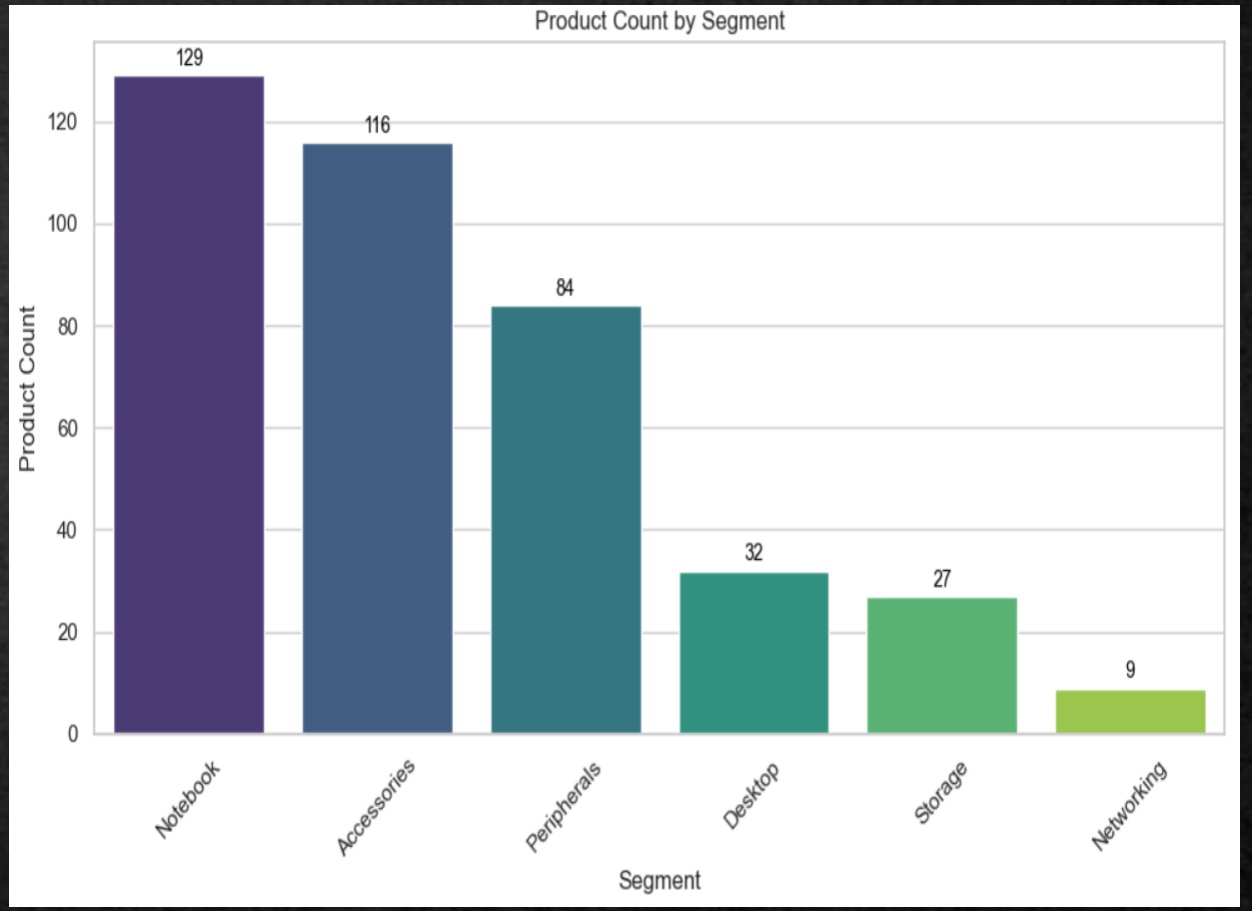
## SQL Code

```sql
SELECT
    segment,
    COUNT(DISTINCT product_code) AS product_count
FROM
    gdb023.dim_product
GROUP BY
    segment
ORDER BY
    product_count DESC;
```

## Output

| segment | product_count |
|---|---|
| Notebook | 129 |
| Accessories | 116 |
| Peripherals | 84 |
| Desktop | 32 |
| Storage | 27 |
| Networking | 9 |

## Conversion of Output to Visual



# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns

# Connecting to database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)

query = """
SELECT
    segment,
    COUNT(DISTINCT product_code) AS product_count
FROM
    gdb023.dim_product
GROUP BY
    segment
ORDER BY
    product_count DESC;
"""

df = pd.read_sql(query, conn)
conn.close()
```

```python
# Plotting
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='segment', y='product_count', data=df, palette='viridis')

# Adding value labels on top of the bars
for index, row in df.iterrows():
    ax.text(
        index,                          # x position
        row['product_count'] + 1,       # y position slightly above the bar
        row['product_count'],           # text to display
        color='black',                  # text color
        ha='center',                    # horizontal alignment
        va='bottom',                    # vertical alignment
        fontsize=10                     # font size
    )

plt.xlabel('Segment')
plt.ylabel('Product Count')
plt.title('Product Count by Segment')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

# SQL Requests

## Question

4. Which segment had the most increase in unique products in 2021 vs 2020?
The final output contains these fields: segment, product_count_2020, product_count_2021, difference

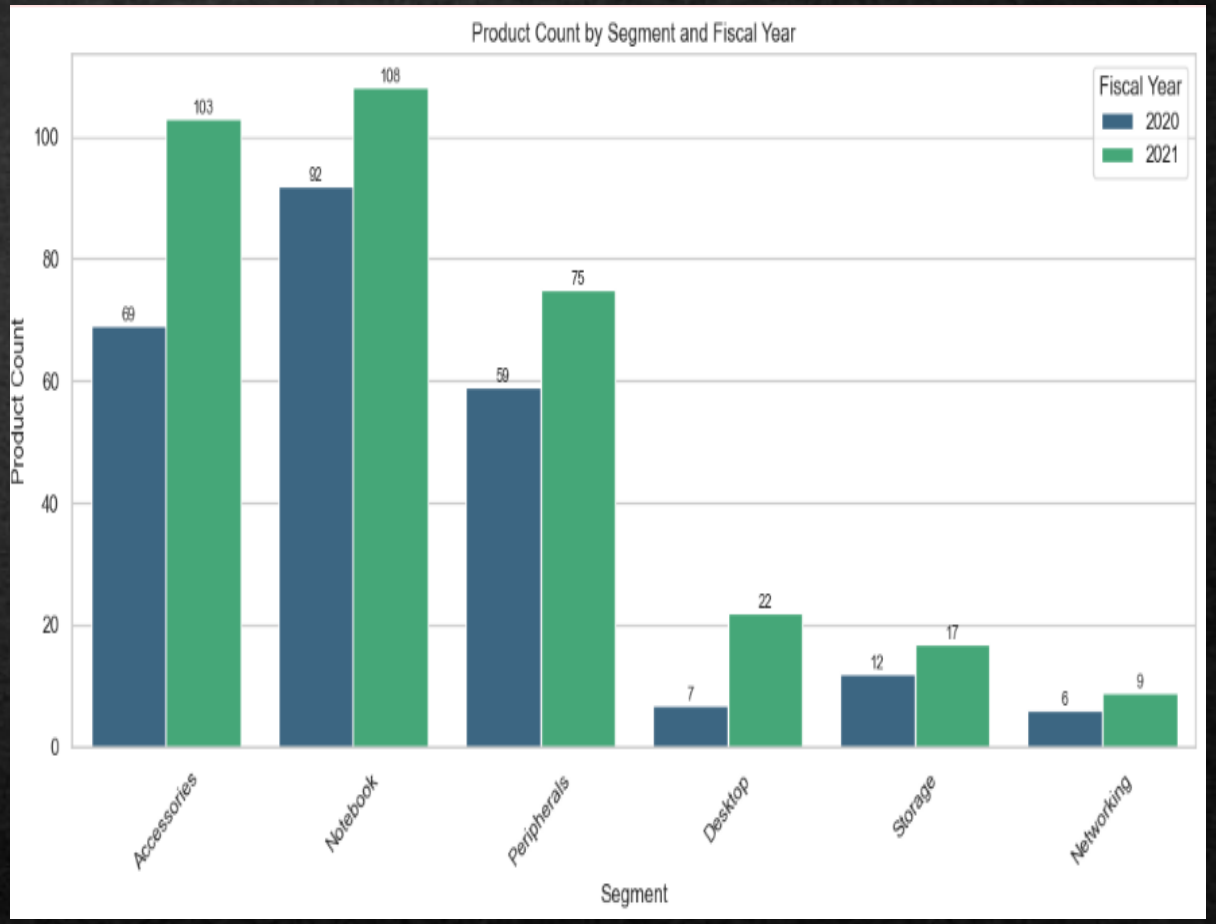
## SQL Code

```sql
SELECT
    dp.segment AS segment,
    COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2020 THEN dp.product_code END) AS product_count_2020,
    COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2021 THEN dp.product_code END) AS product_count_2021,
    (COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2021 THEN dp.product_code END) -
     COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2020 THEN dp.product_code END)) AS difference
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.dim_product dp ON fsm.product_code = dp.product_code
GROUP BY
    dp.segment
ORDER BY
    difference DESC;
```

## Output

| segment | product_count_2020 | product_count_2021 | difference |
|---|---|---|---|
| Accessories | 69 | 103 | 34 |
| Notebook | 92 | 108 | 16 |
| Peripherals | 59 | 75 | 16 |
| Desktop | 7 | 22 | 15 |
| Storage | 12 | 17 | 5 |
| Networking | 6 | 9 | 3 |

## Conversion of Output to Visual



# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns

# Connecting to the MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)

# SQL query
query = """
SELECT
    dp.segment AS segment,
    COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2020 THEN dp.product_code END) AS product_count_2020,
    COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2021 THEN dp.product_code END) AS product_count_2021,
    (COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2021 THEN dp.product_code END) -
     COUNT(DISTINCT CASE WHEN fsm.fiscal_year = 2020 THEN dp.product_code END)) AS difference
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.dim_product dp ON fsm.product_code = dp.product_code
GROUP BY
    dp.segment
ORDER BY
    difference DESC;
"""
```

```python
# Fetching data into a pandas DataFrame
df = pd.read_sql(query, conn)

# Closing the database connection
conn.close()

# Reshapeing the DataFrame from wide to long format for seaborn
df_melted = pd.melt(
    df,
    id_vars=['segment'],
    value_vars=['product_count_2020', 'product_count_2021'],
    var_name='Fiscal Year',
    value_name='Product Count'
)
# Renaming fiscal years for clearer labels
df_melted['Fiscal Year'] = df_melted['Fiscal Year'].replace({
    'product_count_2020': '2020',
    'product_count_2021': '2021'
})

# Plotting with seaborn
plt.figure(figsize=(12, 6))
ax = sns.barplot(
    x='segment',
    y='Product Count',
    hue='Fiscal Year',
    data=df_melted,
    palette='viridis'
)
# Adding value labels on each bar
for container in ax.containers:
    ax.bar_label(container, fontsize=9)

# Customize the plot
plt.xlabel('Segment')
plt.ylabel('Product Count')
plt.title('Product Count by Segment and Fiscal Year')
plt.xticks(rotation=45)
plt.legend(title='Fiscal Year')
plt.tight_layout()

# Show the plot
plt.show()
```

# SQL Requests

## Question

5. Get the products that have the highest and lowest manufacturing costs.
 The final output should contain these fields: product_code, product, manufacturing_costWhich segment had the most increase in unique products in 2021 vs 2020?

## SQL Code

```sql
WITH result AS (
    SELECT
        fmc.product_code,
        dp.product AS product_name,
        fmc.manufacturing_cost,
        ROW_NUMBER() OVER (ORDER BY fmc.manufacturing_cost DESC) AS ranked_desc,
        ROW_NUMBER() OVER (ORDER BY fmc.manufacturing_cost ASC) AS ranked_asc
    FROM
        gdb023.fact_manufacturing_cost fmc
    JOIN
        gdb023.dim_product dp ON dp.product_code = fmc.product_code)
SELECT
    product_code,
    product_name,
    manufacturing_cost
FROM
    result
WHERE
    ranked_desc = 1 OR ranked_asc = 1
ORDER BY
    ranked desc ASC;
```

## Output

| | product_code | product_name | manufacturing_cost |
|---|---|---|---|
| ▶ | A6120110206 | AQ HOME Allin1 Gen 2 | 240.5364 |
| | A2118150101 | AQ Master wired x1 Ms | 0.8920 |

# SQL Requests

## Question

6. Generate a report which contains the top 5 customers who received an average high pre_invoice_discount_pct for the fiscal year 2021 and in the Indian market.
The final output contains these fields: customer_code, customer, average_discount_percentage

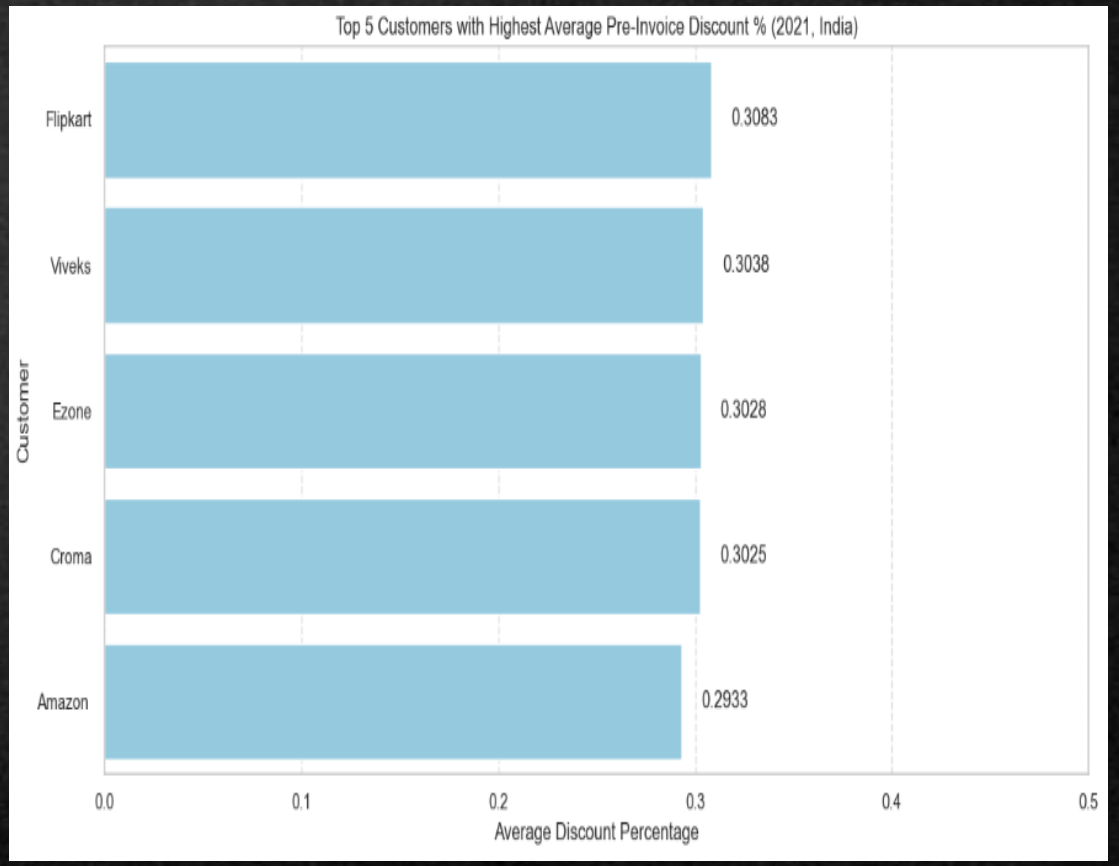
## SQL Code

```sql
SELECT
    fpid.customer_code,
    dc.customer,
    AVG(fpid.pre_invoice_discount_pct) AS average_discount_percentage
FROM
    gdb023.fact_pre_invoice_deductions fpid
JOIN
    gdb023.dim_customer dc ON dc.customer_code = fpid.customer_code
WHERE
    fpid.fiscal_year = 2021 AND dc.market = "India"
GROUP BY
    fpid.customer_code, dc.customer
ORDER BY
    average_discount_percentage DESC
LIMIT 5;
```

## Output

| customer_code | customer | average_discount_percentage |
|---|---|---|
| 90002009 | Flipkart | 0.30830000 |
| 90002006 | Viveks | 0.30380000 |
| 90002003 | Ezone | 0.30280000 |
| 90002002 | Croma | 0.30250000 |
| 90002016 | Amazon | 0.29330000 |

## Conversion of Output to Visual



# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns

# Connecting to the database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)

# Defining the SQL query
query = """
SELECT
    fpid.customer_code,
    dc.customer,
    AVG(fpid.pre_invoice_discount_pct) AS average_discount_percentage
FROM
    gdb023.fact_pre_invoice_deductions fpid
JOIN
    gdb023.dim_customer dc
    ON fpid.customer_code = dc.customer_code
WHERE
    fpid.fiscal_year = 2021
    AND dc.market = 'India'
GROUP BY
    fpid.customer_code, dc.customer
ORDER BY
    average_discount_percentage DESC
LIMIT 5;
"""
```

```python
# Executing the query and load the results into a DataFrame
df = pd.read_sql(query, conn)
# Closing the database connection
conn.close()
# Plotting
plt.figure(figsize=(12, 6))
# Horizontal bar plot of average discount percentage by customer
sns.barplot(
    x='average_discount_percentage',
    y='customer',
    data=df,
    color='skyblue'
)
# Annotating each bar with its value (to 4 decimal places)
for index, row in df.iterrows():
    plt.text(
        row['average_discount_percentage'] + 0.01,  # offset for clarity
        index,
        f"{row['average_discount_percentage']:.4f}",
        va='center'
    )
# Customizing plot labels and title
plt.xlabel('Average Discount Percentage')
plt.ylabel('Customer')
plt.title('Top 5 Customers with Highest Average Pre-Invoice Discount % (2021, India)')
# Adding gridlines for improved readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
# Setting x-axis limit to improve visual appearance
plt.xlim(0, 0.5)
# Ensuring layout fits well
plt.tight_layout()
# Displaying the plot
plt.show()
```

# SQL Requests

## Question

7. Get the complete report of the Gross sales amount for the customer “Atliq Exclusive” for each month.The final report contains these columns: Month, Year, Gross sales Amount

## SQL Code

```sql
SELECT
    fsm.fiscal_year AS Fiscal_Year,
    MONTHNAME(fsm.date) AS Month,
    ROUND(SUM(fgp.gross_price * fsm.sold_quantity), 0)
    AS Gross_sales_Amount
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.fact_gross_price fgp ON fsm.product_code = fgp.product_code
    AND fsm.fiscal_year = fgp.fiscal_year
JOIN
    gdb023.dim_customer dc ON dc.customer_code = fsm.customer_code
WHERE
    dc.customer = "Atliq Exclusive"
GROUP BY
    fsm.fiscal_year, MONTH(fsm.date), MONTHNAME(fsm.date)
ORDER BY
    YEAR(fsm.date), MONTH(fsm.date);
```

## Output

| Fiscal_Year | Month | Gross_sales_Amount |
|---|---|---|
| 2020 | September | 4496260 |
| 2020 | October | 5135902 |
| 2020 | November | 7522893 |
| 2020 | December | 4830405 |
| 2020 | January | 4740600 |
| 2020 | February | 3996228 |
| 2020 | March | 378771 |
| 2020 | April | 395035 |
| 2020 | May | 783813 |
| 2020 | June | 1695217 |
| 2020 | July | 2551159 |
| 2020 | August | 2786648 |
| 2021 | September | 12353510 |
| 2021 | October | 13218636 |
| 2021 | November | 20464999 |
| 2021 | December | 12944660 |
| 2021 | January | 12399393 |
| 2021 | February | 10129736 |
| 2021 | March | 12144061 |
| 2021 | April | 7312000 |
| 2021 | May | 12150225 |
| 2021 | June | 9824521 |
| 2021 | July | 12092346 |
| 2021 | August | 7178708 |

# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
# Step 1: Connecting to the Database and Retrieve the Data
# Establishing database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)
# Defining the SQL query
query = """ SELECT
    fsm.fiscal_year AS Fiscal_Year,
    MONTHNAME(fsm.date) AS Month,
    ROUND(SUM(fgp.gross_price * fsm.sold_quantity), 0) AS Gross_sales_Amount
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.fact_gross_price fgp
    ON fsm.product_code = fgp.product_code
    AND fsm.fiscal_year = fgp.fiscal_year
JOIN
    gdb023.dim_customer dc
    ON fsm.customer_code = dc.customer_code
WHERE
    dc.customer = 'Atliq Exclusive'
GROUP BY
    fsm.fiscal_year, MONTH(fsm.date), MONTHNAME(fsm.date)
ORDER BY
    YEAR(fsm.date), MONTH(fsm.date);
"""
```

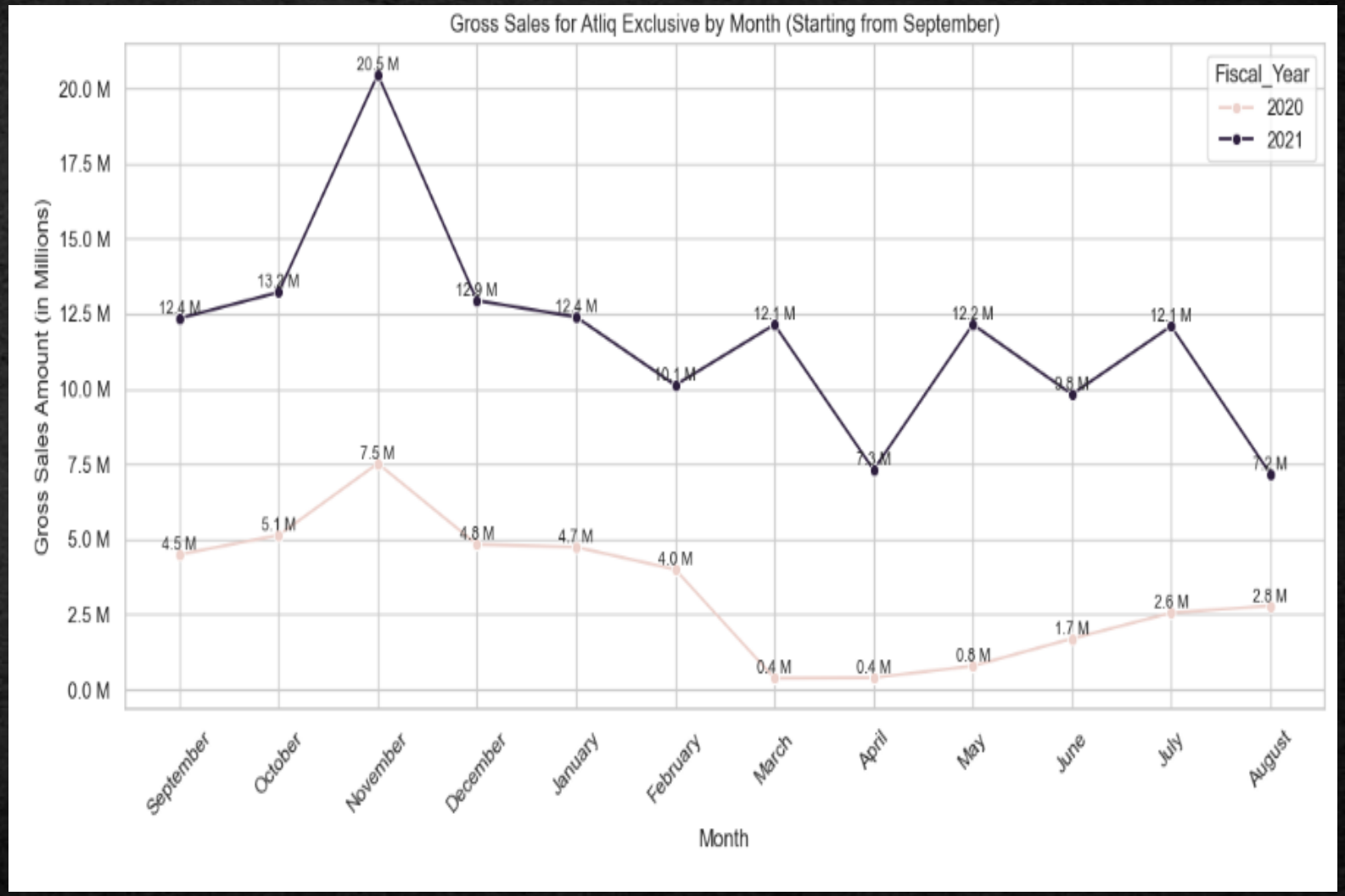
```python
# Executing the query and loading the result into a DataFrame
df = pd.read_sql(query, conn)
# Closing the database connection
conn.close()
# Step 2: Preparing and Visualizing the Data
# Defining the month order starting from September
month_order = [
    'September', 'October', 'November', 'December',
    'January', 'February', 'March', 'April',
    'May', 'June', 'July', 'August']
# Converting the Month column to a categorical type with the specified order
df['Month'] = pd.Categorical(df['Month'], categories=month_order, ordered=True)
# Sorting the DataFrame by Fiscal Year and Month
df = df.sort_values(['Fiscal_Year', 'Month'])
# Creating the line plot
plt.figure(figsize=(12, 6))
sns.lineplot(
    data=df,
    x='Month',
    y='Gross_sales_Amount',
    hue='Fiscal_Year',
    marker='o')
# Annotating each point with its value (in millions)
for i in range(df.shape[0]):
    value_million = df['Gross_sales_Amount'].iloc[i] / 1_000_000
    plt.text(
        x=df['Month'].iloc[i],
        y=df['Gross_sales_Amount'].iloc[i],
        s=f"{value_million:.1f} M",
        ha='center',
        va='bottom',
        fontsize=9
    )
```

# SQL Requests

## Python code for visual creation

```
# Formating Y-axis labels as millions
plt.gca().yaxis.set_major_formatter
(mtick.FuncFormatter(lambda x, _: f'{x/1_000_000:.1f} M'))

# Seting labels and title
plt.xlabel('Month')
plt.ylabel('Gross Sales Amount (in Millions)')
plt.title('Gross Sales for Atliq Exclusive by Month
(Starting from September)')

# Rotating x-axis labels for readability
plt.xticks(rotation=45)

# Ensuring layout is tight
plt.tight_layout()

# Showing the plot
plt.show()
```

## Conversion of Output to Visual



# SQL Requests

## Question

8. In which quarter of 2020 did we get the maximum total_sold_quantity?
The final output contains these fields: Quarter, total_sold_quantity

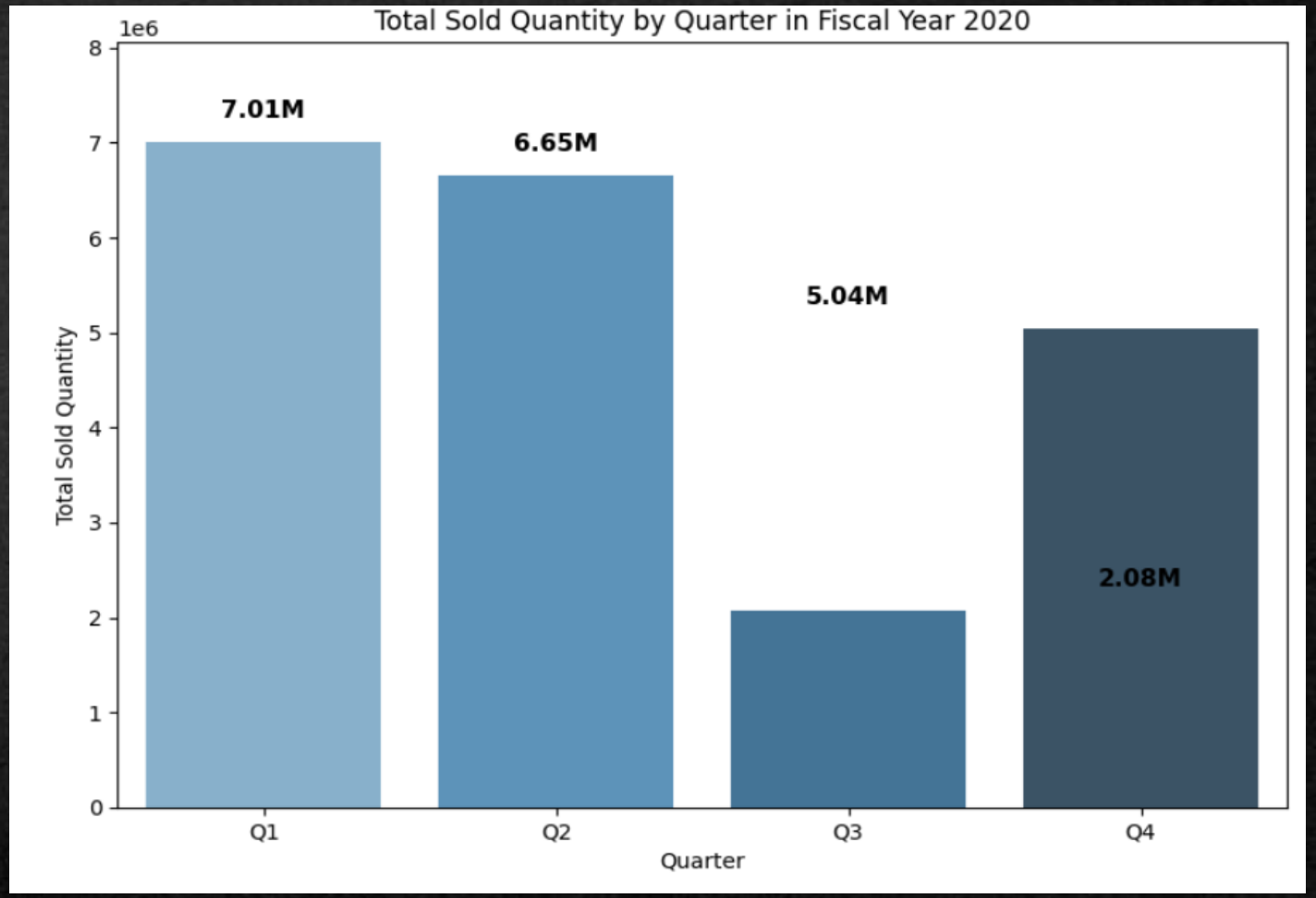
## SQL Code

```sql
SELECT
    CASE
        WHEN MONTH(date) IN (9, 10, 11) THEN 'Q1'
        WHEN MONTH(date) IN (12, 1, 2) THEN 'Q2'
        WHEN MONTH(date) IN (3, 4, 5) THEN 'Q3'
        WHEN MONTH(date) IN (6, 7, 8) THEN 'Q4'
    END AS Quarter,
    SUM(sold_quantity) AS total_sold_quantity
FROM
    gdb023.fact_sales_monthly
WHERE
    fiscal_year = 2020
GROUP BY
    Quarter
ORDER BY
    total_sold_quantity DESC;
```

## Output

| | Quarter | total_sold_quantity |
|---|---|---|
| ▶ | Q1 | 7005619 |
| | Q2 | 6649642 |
| | Q4 | 5042541 |
| | Q3 | 2075087 |

## Conversion of Output to Visual



# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns
# Step 1: Connecting to the Database and Retrieving the Data
# Establishing database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)
# Defining the SQL query
query = """
SELECT
    CASE
        WHEN MONTH(date) IN (9, 10, 11) THEN 'Q1'
        WHEN MONTH(date) IN (12, 1, 2) THEN 'Q2'
        WHEN MONTH(date) IN (3, 4, 5) THEN 'Q3'
        WHEN MONTH(date) IN (6, 7, 8) THEN 'Q4'
    END AS Quarter,
    SUM(sold_quantity) AS total_sold_quantity
FROM
    gdb023.fact_sales_monthly
WHERE
    fiscal_year = 2020
GROUP BY
    Quarter
ORDER BY
    total_sold_quantity DESC;
"""
# Executing the query and loading the result
#into a DataFrame
df = pd.read_sql(query, conn)
# Closing the database connection
conn.close()
# Step 2: Preparint and Visualizing the Data
# Defining the correct quarter order
quarter_order = ['Q1', 'Q2', 'Q3', 'Q4']
# Converting the Quarter column to a categorical
#type with the specified order
df['Quarter'] = pd.Categorical(df['Quarter'],
categories=quarter_order, ordered=True)
# Sorting the DataFrame by Quarter order
df = df.sort_values('Quarter')
# Creating the bar plot
plt.figure(figsize=(8, 6))
sns.barplot(
    data=df,
    x='Quarter',
    y='total_sold_quantity',
    palette='Blues_d'
)
# Seting labels and title
plt.xlabel('Quarter')
plt.ylabel('Total Sold Quantity')
plt.title('Total Sold Quantity by Quarter in Fiscal Year 2020')
# Seting y-axis limiting slightly above the maximum bar height
max_height = df['total_sold_quantity'].max()
plt.ylim(0, max_height * 1.15)
# Annotating each bar with its total sold quantity in millions
for index, row in df.iterrows():
    height = row['total_sold_quantity']
    plt.text(
        x=index,
        y=height + max_height * 0.03,
        # position the label above the bar
        s=f"{height / 1_000_000:.2f}M",
        ha='center',
        va='bottom',
        color='black',
        fontsize=11,
        fontweight='bold'
    )

# Improving layout
plt.tight_layout()

# Displaying the plot
plt.show()
```

# SQL Requests

## Question

9. Which channel helped bring more gross sales in the fiscal year 2021 and the percentage of contribution?-- The final output contains these fields: channel, gross_sales_mln, percentage

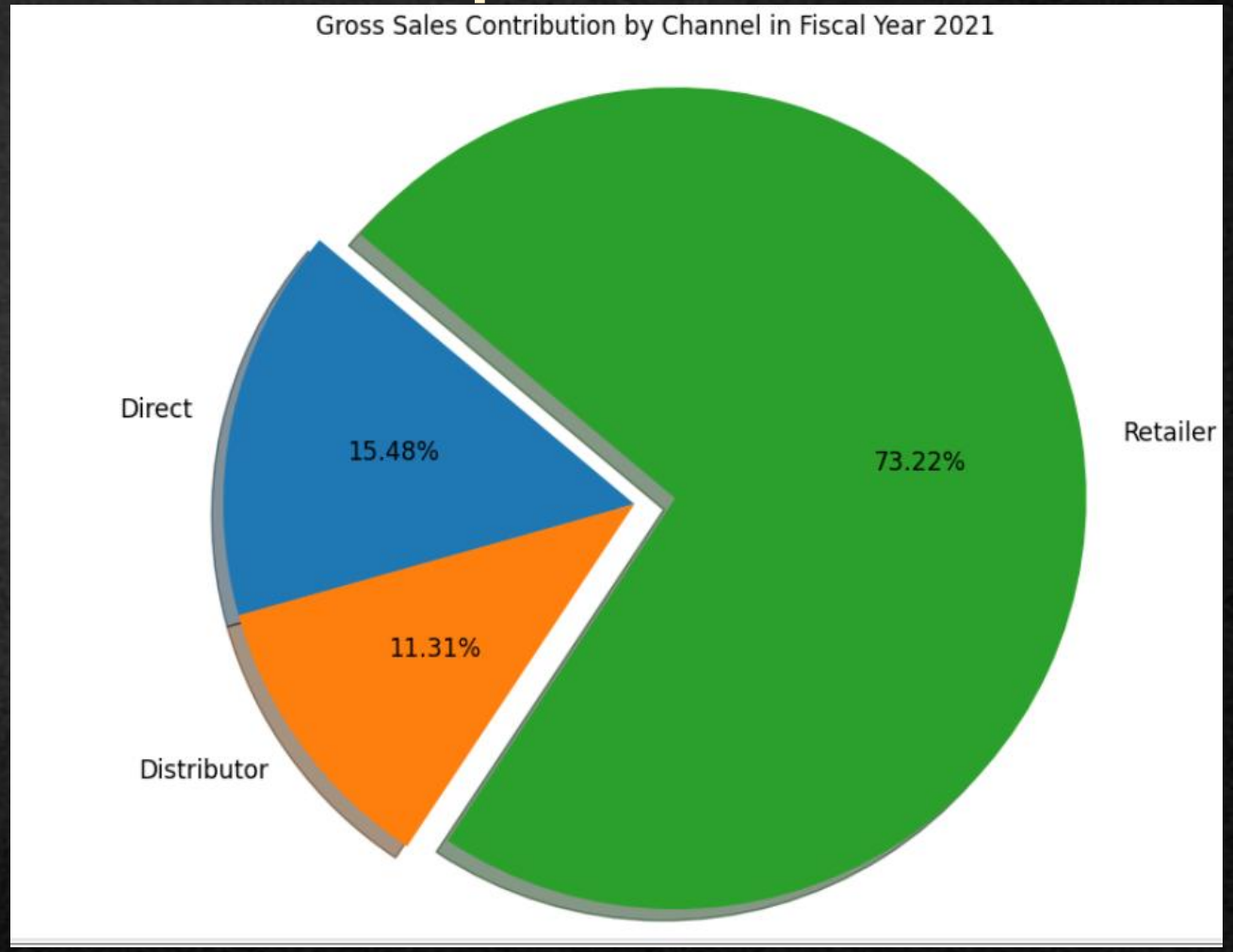
## SQL Code

```sql
SELECT
    dc.channel,
    ROUND(SUM(fgp.gross_price * fsm.sold_quantity) / 1000000, 2) AS gross_sales_mln,
    ROUND(SUM(fgp.gross_price * fsm.sold_quantity) * 100 /
    SUM(SUM(fgp.gross_price * fsm.sold_quantity)) OVER (), 2) AS percentage
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.dim_customer dc ON fsm.customer_code = dc.customer_code
JOIN
    gdb023.fact_gross_price fgp ON fsm.product_code = fgp.product_code
WHERE
    fsm.fiscal_year = 2021
GROUP BY
    dc.channel;
```

## Output

| | channel | gross_sales_mln | percentage |
|---|---|---|---|
| ▶ | Direct | 406.69 | 15.47 |
| | Distributor | 297.18 | 11.31 |
| | Retailer | 1924.17 | 73.22 |

## Conversion of Output to Visual



# SQL Requests

## Python code for visual creation

```python
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
# Step 1: Connecting to the Database and Retrieving the Data
# Establising database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="gdb023"
)
# Defining the SQL query
query = """
SELECT
    dc.channel,
    ROUND(SUM(fgp.gross_price * fsm.sold_quantity) / 1000000, 2) AS gross_sales_mln,
    ROUND(
        SUM(fgp.gross_price * fsm.sold_quantity) * 100 /
        SUM(SUM(fgp.gross_price * fsm.sold_quantity)) OVER (), 2
    ) AS percentage
FROM
    gdb023.fact_sales_monthly fsm
JOIN
    gdb023.dim_customer dc ON fsm.customer_code = dc.customer_code
JOIN
    gdb023.fact_gross_price fgp
    ON fsm.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE
    fsm.fiscal_year = 2021
GROUP BY
    dc.channel;
"""
# Executing the query and load the result into a DataFrame
df = pd.read_sql(query, conn)
# Closing the database connection
conn.close()
```

```python
# Step 2: Preparing and Visualizing the Data
# Defining labels and sizes for the pie chart
labels = df['channel']
sizes = df['gross_sales_mln']
# Highlighting the largest slice (channel with highest gross sales)
explode = [
    0.1 if i == sizes.idxmax() else 0
    for i in range(len(sizes))
]
# Creating the pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    sizes,
    labels=labels,
    autopct='%1.2f%%',
    startangle=140,
    shadow=True,
    explode=explode,
    textprops={'fontsize': 12}
)
# Seting title and make sure pie is circular
plt.title('Gross Sales Contribution by Channel in Fiscal Year 2021')
plt.axis('equal')  # Equal aspect ratio ensures a circular pie chart
# Displaying the plot
plt.show()
```

# SQL Requests

## Question

10. Get the Top 3 products in each division that have a high total_sold_quantity in the fiscal year 2021. The final output contains these fields: division, product_code, product, total_sold_quantity, rank_order

## SQL Code

```sql
WITH results AS (
    SELECT
        dp.division,
        fsm.product_code,
        dp.product,
        SUM(fsm.sold_quantity) AS total_sold_quantity,
        RANK() OVER (PARTITION BY dp.division ORDER BY
        SUM(fsm.sold_quantity) DESC) AS rank_order
    FROM gdb023.fact_sales_monthly fsm
    JOIN gdb023.dim_product dp ON fsm.product_code = dp.product_code
    WHERE fsm.fiscal_year = 2021
    GROUP BY dp.division, fsm.product_code, dp.product
)
SELECT *
FROM
    results
WHERE
    rank_order <= 3;
```

## Output

| division | product_code | product | total_sold_quantity | rank_order |
|---|---|---|---|---|
| N & S | A6720160103 | AQ Pen Drive 2 IN 1 | 701373 | 1 |
| N & S | A6818160202 | AQ Pen Drive DRC | 688003 | 2 |
| N & S | A6819160203 | AQ Pen Drive DRC | 676245 | 3 |
| P & A | A2319150302 | AQ Gamers Ms | 428498 | 1 |
| P & A | A2520150501 | AQ Maxima Ms | 419865 | 2 |
| P & A | A2520150504 | AQ Maxima Ms | 419471 | 3 |
| PC | A4218110202 | AQ Digit | 17434 | 1 |
| PC | A4319110306 | AQ Velocity | 17280 | 2 |
| PC | A4218110208 | AQ Digit | 17275 | 3 |