San Martin's Store Sales Analysis

Numbers
At A Glance

Total Customers
6954

Order Count
14059

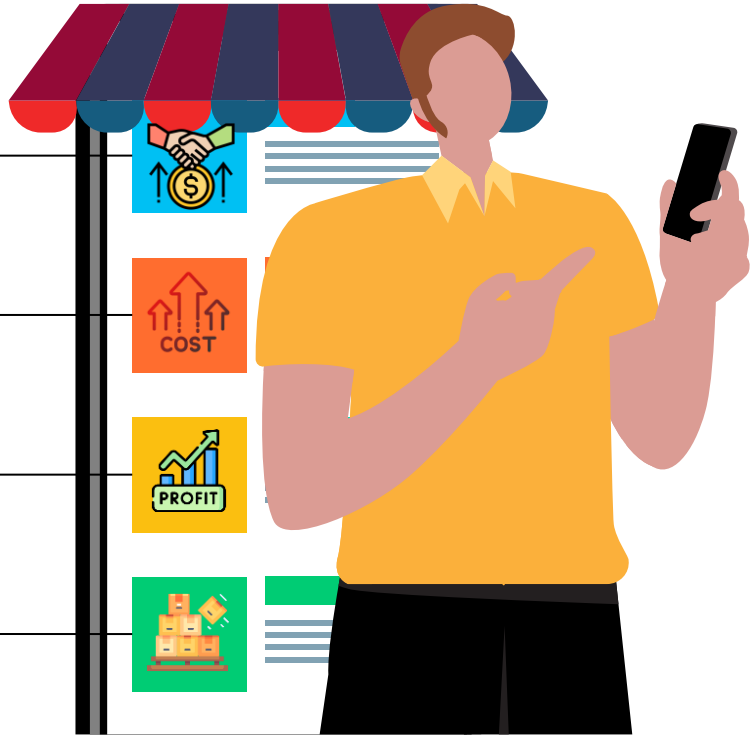
Total Products
246

Total Stores
150

Total Sales Agents
9

METRICS

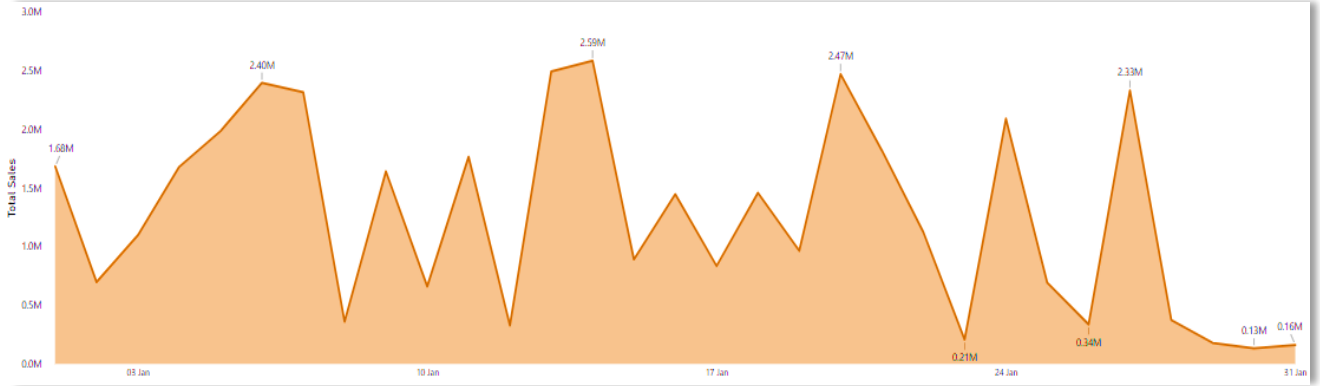
Total Sales	39.2M
Total Cost	28.1M
Total Profit	11.1M
Total Quantity Sold	18.3K



TIME-BASED ANALYSIS

TOTAL SALES OVER TIME

```sql
---1. Total Sales over Time
SELECT [Order Date], sum(Sales.sales) as total_sales
FROM Sales
GROUP BY [Order Date]
ORDER BY [Order Date];
```



TIME-BASED ANALYSIS

SALES GROWTH RATE

```sql
---2. Sales Growth Rate
SELECT [Order Date], sum(Sales.sales) as total_sales,
        round(((sum(sales) - lag(sum(sales), 1) OVER (ORDER BY [Order Date]))/  lag(sum(sales), 1) OVER (ORDER BY [Order Date]))*100,0)
        as growth_rate
FROM Sales
GROUP BY [Order Date]
ORDER BY [Order Date];
```

CUSTOMER ANALYSIS

Customer Segmentation based on **Purchase Frequency**

```sql
--1(a.)Customer Segmentation based on Purchase Frequency -

SELECT c.Customers, count(*) as purchase_freq,
        CASE
            WHEN count(*) > 5 THEN 'High-Frequency Customers'
            WHEN count(*) BETWEEN 2 AND 5 THEN 'Medium-Frequency Customers'
            ELSE 'Low-Frequency Customers'
        END AS segment
FROM Sales s JOIN Customers c on c.[Customer Key] = s.[Customer Key]
group by c.Customers
```

	Customers	purchase_freq	segment
1	Aarón Blanco Rosselló	3	Medium-Frequency Customers
2	Aarón Caballero-Marquez	2	Medium-Frequency Customers
3	Aarón Carrera Iglesias	2	Medium-Frequency Customers
4	Aarón Catalán Martínez	2	Medium-Frequency Customers
5	Aarón de Sanabria	1	Low-Frequency Customers
6	Aarón Ignacio Pi Salom	2	Medium-Frequency Customers
7	Aarón Manuel Galiano	1	Low-Frequency Customers
8	Aarón Marin Castañeda	1	Low-Frequency Customers
9	Aarón Salvador Cases Castañeda	2	Medium-Frequency Customers
10	Aarón Zamorano Requena	1	Low-Frequency Customers
11	Abel Alcázar Botella	2	Medium-Frequency Customers
12	Abel Boix	1	Low-Frequency Customers
13	Abel Carrillo-Trujillo	1	Low-Frequency Customers

Customer Segmentation based on Total Spend

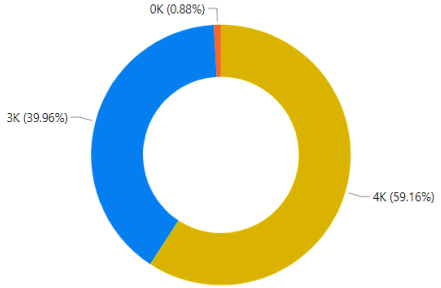
```sql
--2(a.)Customer Segmentation based on Total Spend -
SELECT c.Customers, sum(s.Sales) as total_sales,
        CASE
            WHEN sum(s.Sales) > 20000 THEN 'High-Spend Customers'
            WHEN sum(s.Sales) BETWEEN 10000 AND 20000 THEN 'Medium-Spend Customers'
            ELSE 'Low-Spend Customers'
        END AS segment
FROM Sales s JOIN Customers c on c.[Customer Key] = s.[Customer Key]
GROUP BY c.Customers
ORDER BY total_sales desc;
```

	Customers	total_sales	segment
1	Aarón Blanco Rosselló	5184.76	Low-Spend Customers
2	Aarón Caballero-Marquez	2614.19	Low-Spend Customers
3	Aarón Carrera Iglesias	5917.05	Low-Spend Customers
4	Aarón Catalán Martínez	5583.14	Low-Spend Customers
5	Aarón de Sanabria	2628.28	Low-Spend Customers
6	Aarón Ignacio Pi Salom	8701.75	Low-Spend Customers
7	Aarón Manuel Galiano	3951.24	Low-Spend Customers
8	Aarón Marin Castañeda	1130.79	Low-Spend Customers
9	Aarón Salvador Cases Castañeda	5147.43	Low-Spend Customers
10	Aarón Zamorano Requena	1730.11	Low-Spend Customers
11	Abel Alcázar Botella	5733.93	Low-Spend Customers
12	Abel Boix	1360.6	Low-Spend Customers
13	Abel Carrillo-Trujillo	1247.93	Low-Spend Customers

CUSTOMER ANALYSIS

Distribution of Purchase Frequency Segment

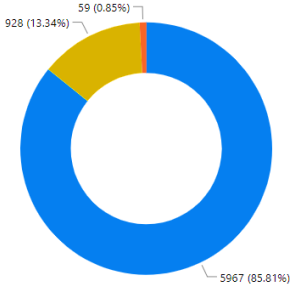
```sql
--1(b.)  Distribution of Purchase Frequency Segment

SELECT  (CASE
            WHEN num_orders > 5 THEN 'High-Frequency Customers'
            WHEN num_orders BETWEEN 2 AND 5 THEN 'Medium-Frequency Customers'
            ELSE 'Low-Frequency Customers' END) AS Frequency, COUNT(*) as num_customers
FROM    (SELECT Sales.[Customer Key], COUNT(*) as num_orders
         FROM Sales JOIN Customers on sales.[Customer Key] = Customers.[Customer Key]
         group by sales.[Customer Key]) o
GROUP BY  (CASE WHEN num_orders > 5 THEN 'High-Frequency Customers'
        WHEN num_orders BETWEEN 2 AND 5 THEN 'Medium-Frequency Customers'
        ELSE 'Low-Frequency Customers' END)
ORDER BY num_customers desc;
```



Frequency ● Medium-Frequency Customers ● Low-Frequency Customers ● High-Frequency Customers

Distribution Of Total Spend Segment

```sql
-- 2(b.) Distribution Of Total Spend Segment
SELECT segment, count(segment) as num_customers
FROM (select c.Customers, sum(s.Sales) as total_sales,
        CASE
            WHEN sum(s.Sales) > 20000 THEN 'High-Spend Customers'
            WHEN sum(s.Sales) BETWEEN 10000 AND 20000 THEN 'Medium-Spend Customers'
            ELSE 'Low-Spend Customers'
        END AS segment
      FROM Sales s JOIN Customers c on c.[Customer Key] = s.[Customer Key]
      group by c.Customers) g
GROUP BY segment;
```

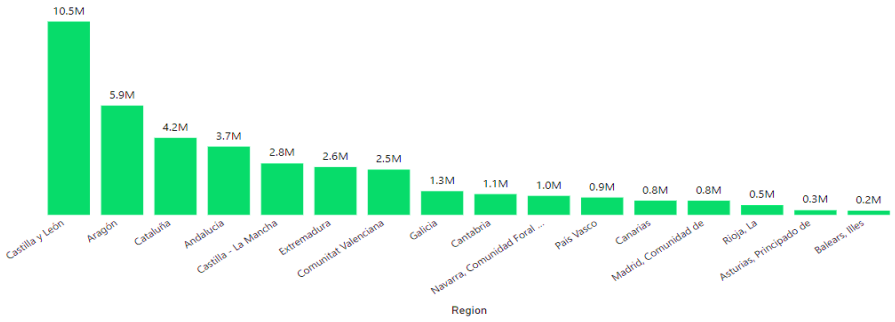


segment ● Low-Spend Customers ● Medium-Spend Customers ● High-Spend Customers

GEOGRAPHIC ANALYSIS

Regional Sales Performance

```sql
-- 1. Regional Sales Performance- Sales by Region

SELECT l.Region, sum(s.sales) as total_sales
FROM Sales s JOIN Locations l on s.[Region Key] = l.[Region Key]
GROUP BY l.Region
ORDER BY total_sales desc;
```



Market Penetration - Percentage of Total Sales by Region

```sql
-- 2. Market Penetration - Percentage of Total Sales by Region

SELECT l.Region, sum(s.sales) as total_sales, round((100 * sum(s.sales)) / sum(sum(s.sales)) over (), 2) as sales_contribution
FROM Sales s JOIN Locations l on s.[Region Key] = l.[Region Key]
GROUP BY l.Region;
```

SALES CHANNEL ANALYSIS

Total/Average Sales for each Store

```sql
---- 1. Total/Average Sales FOR STORE

SELECT sales.[Store Key], Stores.stores , sum(Sales.sales) as total_sales, avg(Sales.sales) as avg_sales
FROM Sales JOIN Stores on sales.[Store Key] = Stores.[Store Key]
GROUP BY sales.[Store Key], Stores.stores
ORDER BY total_sales desc;
```

Stores	Total Sales	Average Sales
Tienda Abiego	2,30,745.54	2,508.10
Tienda Agurain/Salvatierra	2,55,760.53	2,841.78
Tienda Alameda del Valle	2,39,716.90	2,634.25
Tienda Alcubillas	2,22,272.52	2,778.41
Tienda Alfoz de Lloredo	2,25,864.61	2,788.45
Tienda Almoster	2,58,432.70	2,749.28
Tienda Alquézar	2,72,218.46	2,895.94
Tienda Andoain	2,21,727.21	2,671.41
Tienda Anguita	2,82,204.42	2,613.00
Tienda Anievas	3,41,191.81	3,249.45
Tienda Antzuola	2,20,026.64	2,588.55
Tienda Arandilla	2,77,916.89	2,956.56

Total/Average Sales for each Location

```sql
--- 2. Total/Average Sales FOR LOCATION

SELECT Sales.[Region Key], Locations.Region, sum(Sales.sales) as total_sales
FROM Sales JOIN Locations on Sales.[Region Key] = Locations.[Region Key]
GROUP BY Sales.[Region Key], Locations.Region
ORDER BY total_sales desc;
```

Region	Total Sales	Average Sales
Andalucía	37,15,194.49	2,840.36
Aragón	59,46,003.02	2,880.82
Asturias, Principado de	2,53,382.20	2,912.44
Balears, Illes	2,29,947.60	2,673.81
Canarias	7,75,109.49	2,768.25
Cantabria	11,29,520.77	2,956.86
Castilla - La Mancha	28,00,397.94	2,710.94
Castilla y León	1,05,19,745.07	2,748.10
Cataluña	41,91,302.45	2,773.86
Comunitat Valenciana	24,64,335.87	2,816.38
Extremadura	26,04,481.65	2,756.07
Galicia	12,89,277.55	2,766.69
Madrid, Comunidad de	7,72,487.04	2,758.88

PRODUCT PERFORMANCE ANALYSIS

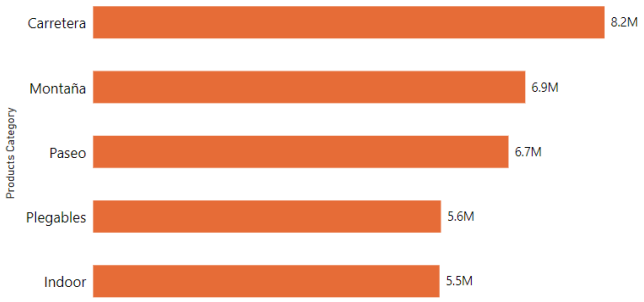
Best-selling Products- Top 10 Products by Sales

```sql
SELECT TOP 10 P.Products, SUM(S.SALES) AS total_sales
FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY P.Products
ORDER BY total_sales desc;
```

Products	Total Sales
BICICLETA CICLISMO INDOOR FFITTECH FUN	5,96,044.36
BICICLETA DAHON MARINER D8	4,36,171.96
BICICLETA DAHON MARINER D8 PLATA	4,41,555.71
BICICLETA INDOOR INXIDE XS08	4,74,128.12
BICICLETA MERIDA CROSSWAY 10 2021	5,78,998.07
BICICLETA MERIDA CROSSWAY 20 2021	4,24,465.42
BICICLETA MONTY 301 2020	5,16,370.97
BICICLETA MONTY SWING 2021	4,80,284.79
BICICLETA ORBEA CARPE 15 2021	4,71,438.00
BICICLETA PLEGABLE MONTY SOURCE 2020	5,10,575.88

Best-selling Product category - Top 5 Product category by Sales

```sql
SELECT P.[Products Category], SUM(S.SALES) AS total_sales
FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY P.[Products Category]
ORDER BY total_sales desc;
```



PRODUCT PERFORMANCE ANALYSIS

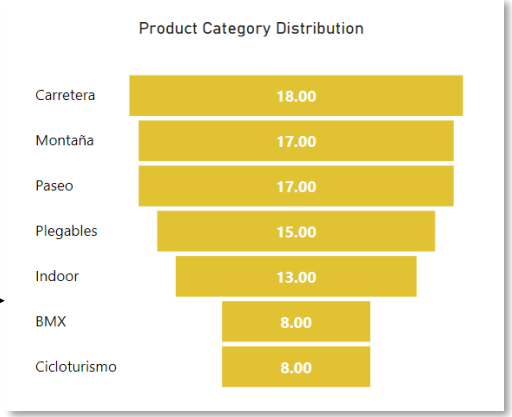
Profitability of Each Product - Product Contribution Margin

```sql
SELECT P.Products, sum(S.Sales) as Total_sales, SUM(S.[Unit Cost] * S.Quantity) AS Total_Var_Cost,
       ((SUM(S.Sales) - SUM(S.[Unit Cost] * S.Quantity)) / SUM(S.Sales)) * 100 as Profit_contribution
FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY P.Products
ORDER BY Profit_contribution desc;

SELECT P.Products, Sum(S.Profit) / sum(S.Sales) * 100 as profit_cont
FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY P.Products
ORDER BY profit_cont desc;
```

	Products	profit_count
1	BICICLETA BERGAMONT GRANDURANCE 4 2021	13
2	BICICLETA BERGAMONT GRANDURANCE 6 2021	13
3	BICICLETA BERGAMONT GRANDURANCE 8 2021	13
4	BICICLETA BERGAMONT REVOX 4 29 2021	29
5	BICICLETA BH EXPERT JUNIOR 26 2021	29
6	BICICLETA BH EXPERT JUNIOR 26 DISC 2021	29
7	BICICLETA BH GLAVELX ALU 1.0 2021	13
8	BICICLETA BH GLAVELX ALU 2.0 2021	13
9	BICICLETA BH IBIZA PRO 2020	40
10	BICICLETA BH OXFORD 2021	31
11	BICICLETA BH OXFORD JET 2021	31
12	BICICLETA BH OXFORD JET LITE 2021	31
13	BICICLETA BH OXFORD LITE 2021	31
14	BICICLETA BH SILVERTIP 2021	31
15	BICICLETA BH SILVERTIP JET 2021	31

Product Category Distribution

```sql
SELECT
    P.[Products Category] AS ProductCategory,
    COUNT(*) AS ProductCount,
    FLOOR((COUNT(*) * 100.0 / SUM(COUNT(*)) OVER ())) AS CategoryDistributionPercentage
FROM
    sales S JOIN Products P on S.[Product Key] = p.[Product Key]
GROUP BY
    P.[Products Category];
```



TIME-TO-FULFILLMENT ANALYSIS

Order Processing Time - Average Time to Fulfill an Order

```sql
--1. Order Processing Time - Average Time to Fulfill an Order
SELECT Avg(DATEDIFF(Day, [Order Date], [Shipping date]))
FROM Sales;
```

1.49

Average Order_Processing_Time (In Days)

Order Processing Time based on each store

```sql
--2. Order Processing Time - Average Time to Fulfill an Order based on each Store
SELECT St.stores, Avg(DATEDIFF(Day, [Order Date], [Shipping date]))
FROM Sales S JOIN Stores St on S.[Store Key] = St.[Store Key]
Group by st.Stores;
```

	stores	order_processing_time
1	Tienda Abiego	1
2	Tienda Agurain/Salvatierra	1
3	Tienda Alameda del Valle	1
4	Tienda Alcubillas	1
5	Tienda Alfoz de Lloredo	1
6	Tienda Almoster	1
7	Tienda Alquézar	1
8	Tienda Andoain	1
9	Tienda Anguita	1
10	Tienda Anievas	1

COST ANALYSIS

Cost of Goods Sold (COGS)

```sql
--1. Cost of Goods Sold (COGS)
SELECT sum(cost) as total_cost, avg(cost) as avg_cost
FROM Sales;
```

Gross profit Margin

```sql
---2. Gross profit Margin
SELECT (sum(Sales) - sum(cost)) * 100 / sum(sales) as goss_profit_margin
FROM Sales;
```

Total Cost

28.07M

Average Cost

2.00K

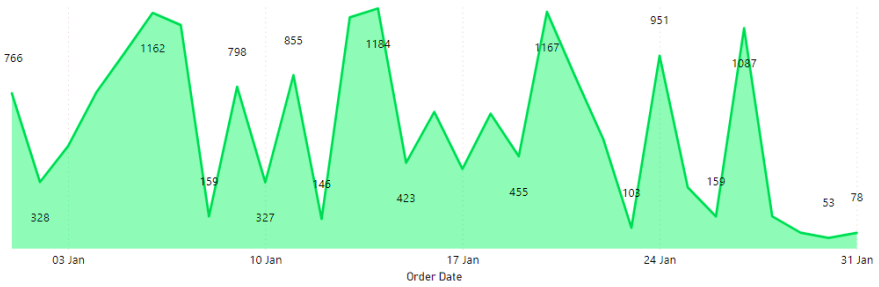
Gross Profit Margin(%)

28.39

QUANTITY ANALYSIS

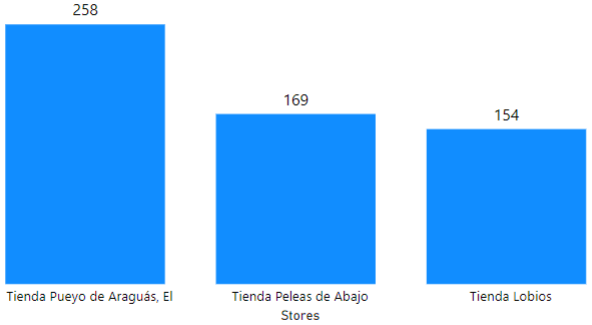
Quantity Trends

```
--1. Quantity Trends Over Time
SELECT [Order Date], sum(Quantity) as total_quantity
FROM Sales
GROUP BY [Order Date]
Order by [Order Date];
```



Highest Quantity bought From which store

```
---4. Highest Quantity bought FROM which store

SELECT st.Stores, sum(s.Quantity) as total_qty
FROM Sales s JOIN Stores st on s.[Store Key] = st.[Store Key]
GROUP BY st.Stores
ORDER BY total_qty desc;
```



SALES AGENTS ANALYSIS

BEST PERFROMING SALES AGENTS

```sql
---1. BEST PERFROMING SALES AGENTS
SELECT SA.[Sales Agent Name], SUM(S.Sales) as total_sales
FROM Sales S JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
GROUP BY SA.[Sales Agent Name]
ORDER BY total_sales DESC;
```

Sales Agent Photo	Sales Agent Name	Total Sales
	Juanito Pacheco Quintero	77,95,632.23
	Ricardo Amat Casals	58,90,232.38
	Evelia Cazorla Girona	47,62,350.45

SALES AGENT WITH HIGHEST NUMBER OF UNIQUE PRODUCTS SOLD

```sql
---3. SALES AGENT WITH HIGHEST NUMBER OF UNIQUE PRODUCTS SOLD
SELECT SA.[Sales Agent Name], COUNT(DISTINCT P.Products) AS NUM_PRODUCTS
FROM Sales S
        JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
        JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY SA.[Sales Agent Name]
ORDER BY NUM_PRODUCTS DESC;
```

	Sales Agent Name	NUM_PRODUCTS
1	Natalia Arellano Gil	186
2	Ricardo Amat Casals	186
3	Juanito Pacheco Quintero	185
4	Josefa Estevez Abella	185
5	Evelia Cazorla Girona	185
6	Toño Prado-Arco	184
7	Teobaldo Peña Tejero	184
8	Aureliano Cabezas Sola	181
9	Eduardo del Azcona	168

SALES AGENTS ANALYSIS

SALES AGENT ASSOCIATED WITH HOW MANY STORES

```sql
---4. SALES AGENT ASSOCIATED WITH HOW MANY STORES
SELECT SA.[Sales Agent Name], COUNT(DISTINCT ST.Stores) AS NUM_STORES
FROM Sales S
        JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
        JOIN Stores ST ON S.[Store Key] = ST.[Store Key]
GROUP BY SA.[Sales Agent Name]
ORDER BY NUM_STORES DESC;
```

	Sales Agent Name	NUM_STORES
1	Natalia Arellano Gil	149
2	Juanito Pacheco Quintero	149
3	Aureliano Cabezas Sola	149
4	Toño Prado-Arco	149
5	Ricardo Amat Casals	149
6	Teobaldo Peña Tejero	149
7	Eduardo del Azcona	149
8	Josefa Estevez Abella	149
9	Evelia Cazorla Girona	149

SALES AGENTS WITH NUMBER OF ORDERS

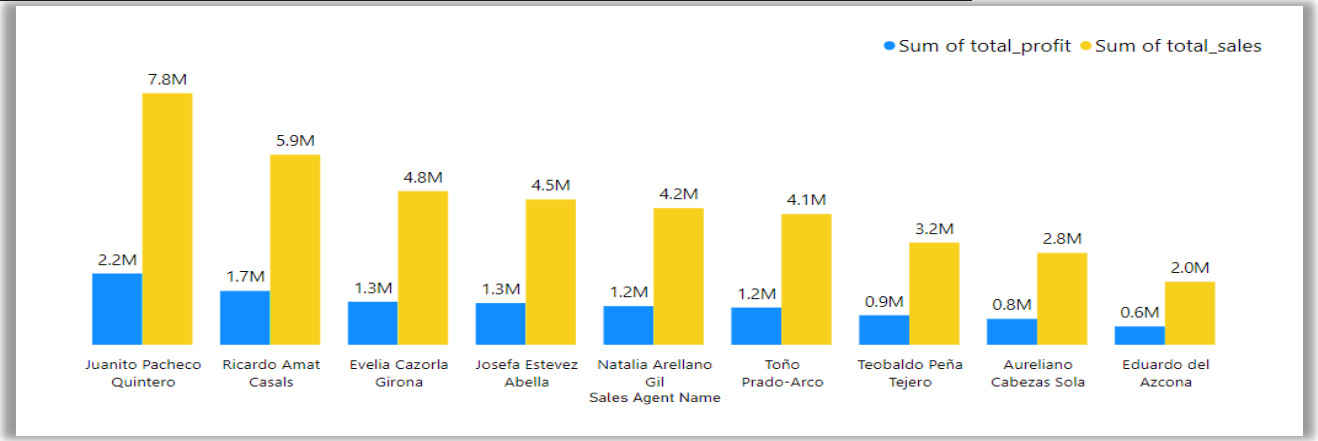
```sql
---5. SALES AGENTS WITH NUMBER OF ORDERS
SELECT SA.[Sales Agent Name], Count(*) as num_orders
FROM Sales S JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
GROUP BY SA.[Sales Agent Name]
ORDER BY num_orders DESC;
```

	Sales Agent Name	num_orders
1	Juanito Pacheco Quintero	2787
2	Ricardo Amat Casals	2095
3	Evelia Cazorla Girona	1692
4	Josefa Estevez Abella	1664
5	Natalia Arellano Gil	1552
6	Toño Prado-Arco	1447
7	Teobaldo Peña Tejero	1127
8	Aureliano Cabezas Sola	990
9	Eduardo del Azcona	705

VIEWS

VIEW TO RETRIEVE SALES REVENUE AND PROFIT EARNED BY EACH SALES AGENT ON EACH PRODUCT CATEGORY

```sql
CREATE VIEW SAgent_prod_cat_performance
AS
SELECT SA.[Sales Agent Name], P.[Products Category] , SUM(S.Sales) as total_sales, SUM(S.Profit) as total_profit
FROM Sales S
    JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
    JOIN Products P ON S.[Product Key] = P.[Product Key]
GROUP BY SA.[Sales Agent Name], P.[Products Category];

--- Retrieving data from View

SELECT * FROM SAgent_prod_cat_performance S
WHERE S.[Sales Agent Name] = 'Toño Prado-Arco'
ORDER BY total_sales DESC, total_profit DESC;
```



VIEWS

VIEW TO RETRIVE ORDER COUNT, TOTAL SALES AND TOTAL PROFIT FOR EACH STORE NAME AND ITS LOCATION(REGION)

```sql
--2. VIEW TO RETRIVE ORDER COUNT, TOTAL SALES AND TOTAL PROFIT FOR EACH STORE NAME AND ITS LOCATION(REGION)
CREATE VIEW store_loc_analysis AS
SELECT L.Region, ST.Stores, count(*) as num_orders, sum(S.sales) as total_sales, sum(S.Profit) as total_profit
FROM Sales S JOIN Stores ST ON S.[Store Key] = ST.[Store Key]
JOIN Locations L ON S.[Region Key] = L.[Region Key]
GROUP BY L.Region, ST.Stores;
```

C.) TO FIND TOP 5 STORES WITH THEIR LOCATION BY HIGHEST PROFIT EARNED

```sql
-- c.) TO FIND TOP 5 STORES WITH THEIR LOCATION BY HIGHEST PROFIT EARNED

SELECT TOP 5 sl.Stores, sl.Region, sl.total_profit
FROM store_loc_analysis sl
ORDER BY sl.total_profit DESC;
```

B.) TO FIND TOP 5 STORES WITH THEIR LOCATION BY HIGHEST SALES REVENUE

```sql
-- B.) TO FIND TOP 5 STORES WITH THEIR LOCATION BY HIGHEST SALES REVENUE

SELECT TOP 5 sl.Stores, sl.Region, sl.total_sales
FROM store_loc_analysis sl
ORDER BY sl.total_sales DESC;
```

STORED PROCEDURES

1. STORED PROCEDURE TO RETRIEVE DETAILS OF HIGH-VALUED TRANSACTIONS (> 9000 sales per transaction)

```sql
-- 1. STORED PROCEDURE TO RETRIEVE DETAILS OF HIGH-VALUED TRANSACTIONS ( > 9000 sales per transaction)

CREATE PROCEDURE high_valued_transactions AS
BEGIN
    SELECT S.[Order Date], C.Customers, P.Products, S.Quantity, S.Sales, S.Profit
    FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
                 JOIN Customers C ON S.[Customer Key] = C.[Customer Key]
                 JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
                 JOIN Stores ST ON S.[Store Key] = ST.[Store Key]
                 JOIN Locations L ON S.[Region Key] = L.[Region Key]
    WHERE S.Sales > 9000
END
-- EXECUTING STORED PROCEDURE
EXEC high_valued_transactions;
```

2. STORED PROCEDURE TO RETRIEVE DETAILS OF MEDIUM-VALUED TRANSACTIONS (BETWEEN 5000 AND 9000 sales per transaction)

```sql
-- 2. STORED PROCEDURE TO RETRIEVE DETAILS OF MEDIUM-VALUED TRANSACTIONS ( BETWEEN 5000 AND 9000 sales per transaction)

CREATE PROCEDURE medium_valued_transactions AS
BEGIN
    SELECT S.[Order Date], C.Customers, P.Products, S.Quantity, S.Sales, S.Profit
    FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
                 JOIN Customers C ON S.[Customer Key] = C.[Customer Key]
                 JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
                 JOIN Stores ST ON S.[Store Key] = ST.[Store Key]
                 JOIN Locations L ON S.[Region Key] = L.[Region Key]
    WHERE S.Sales BETWEEN 5000 AND 9000
END
-- EXECUTING STORED PROCEDURE
EXEC medium_valued_transactions;
```

STORED PROCEDURES

3. STORED PROCEDURE TO RETRIEVE DETAILS OF LOW-VALUED TRANSACTIONS (< 5000 sales per transaction)

```sql
-- 3. STORED PROCEDURE TO RETRIEVE DETAILS OF LOW-VALUED TRANSACTIONS ( < 5000 sales per transaction)

CREATE PROCEDURE low_valued_transactions AS
BEGIN
    SELECT S.[Order Date], C.Customers, P.Products, S.Quantity, S.Sales, S.Profit
    FROM Sales S JOIN Products P ON S.[Product Key] = P.[Product Key]
                 JOIN Customers C ON S.[Customer Key] = C.[Customer Key]
                 JOIN Sales_agents SA ON S.[Sales Agent Key] = SA.[Sales Agent Key]
                 JOIN Stores ST ON S.[Store Key] = ST.[Store Key]
                 JOIN Locations L ON S.[Region Key] = L.[Region Key]
    WHERE S.Sales < 5000
END
-- EXECUTING STORED PROCEDURE
EXEC low_valued_transactions;
```

4. NESTED STORED PROCEDURE CONTAINING PREVIOUS 3 PROCEDURES

```sql
--4. NESTED STORED PROCEDURE CONTAINING ABOVE 3 PROCEDURES
CREATE PROCEDURE valued_transactions(@value VARCHAR(7))
AS
BEGIN
    DECLARE @procedure AS VARCHAR(30)
    SET @procedure = CASE
    WHEN @value = 'high' THEN 'high_valued_transactions'
    WHEN @value = 'medium' THEN 'medium_valued_transactions'
    ELSE 'low_valued_transactions' END
    EXEC @procedure
END

--- EXECUTING STORED PROCEDURE
EXEC valued_transactions 'high'
```

CONCEPTS USED

1. Aggregation Functions
2. CASE STATEMENTS
3. GROUP BY & ORDER BY
4. VIEWS
6. JOINS
6. STORED PROCEDURES