```
-- B.V.I.P
-- CUSTOMERS ORDERING

-- PRIORITY CUSTOMERS
This Top product offers an in-depth exploration of sales data for a fictional retail chain
called "Bestshop" that is in the process of transitioning from a brick-and-mortar to an
omnichannel model by data analysts in a typical business environment, handling a selection of
customer, product, and transaction data to uncover insights into customer behavior, sales
performance, and product trends. The dataset includes various tables, meticulously cleaned and
centralized from multiple data sources to ensure precision-ready exercises.

The dataset includes customer information, product details, transaction records, and
marketing and promotions. Analysts use SQL and BI tools to derive insights and offer
recommendations for enhancing customer engagement, optimizing product offerings, and
improving overall business performance. The dataset is structured to reflect real-world business
challenges, making it a valuable resource for data analysts and enthusiasts seeking to hone their skills.

1. Which product lines are the highest average sales per transaction, and how does it
compare to the overall average?
2. Which product lines have the most transactions?
3. Which product lines generate the most revenue?
4. Which product lines have the most customers?
5. Which product lines have the most unique customers?
6. What is the best seller product line?
7. What is the most common product line?
8. Which product lines have the most sales?
9. Rank product lines by revenue.
10. What is the customer base brings the most revenue?
11. What is the most popular product line?
12. What is the most popular product?
13. What is the gender of most of the customers?
14. What is the most popular product line for women?
15. What is the most popular product line for men?
16. Which is the month with the highest revenue?
17. Which is the month with the lowest revenue?
18. Which day of the week has the best average revenue per transaction?

-- BONUS ACTIVITIES

1. Which product line has the highest average sales per transaction, and how does it
compare to the overall average?
2. Rank the brands based on their total sales. (display their rank alongside their sales
3. Identify the top 3 product lines contributing most of the product line to the total sales.
4. Calculate the month-over-month sales growth rate for each product line.
5. Identify the top 3 product lines with the highest growth rate.
6. Find the top 3 product lines with the highest number of transactions.

-- CONCLUSION

1. Estimate the Customer Lifetime Value (CLV) for each customer base, based on the gross
revenue generated from their transactions. Rank customers by average customer lifespan (in years).

2. Develop a Sales Performance Index (SPI) for each brand, based on their sales volume,
transaction frequency, and customer engagement. Assign weights to these factors and provide a
justification for the chosen weights. Calculate the SPI for each brand and rank them accordingly.
-- 10%, -- customer satisfaction rating -- 10%.

3. Determine which products are most profitable line, defined as the gross income divided
by total sales. Factored in a percentage. This analysis will help identify which product
lines are most profitable.


-- Cleaning

-- Do you know the left values and replace with zero (optional)
UPDATE sales SET discount = 0 WHERE discount IS NULL;
UPDATE sales SET rate = 1.0WHERE rate IS NULL;
UPDATE sales SET customer_type = 'Unknown' WHERE customer_type IS NULL;

UPDATE sales SET gender = 'Unknown' WHERE gender IS NULL;
UPDATE sales SET product_line = 'Unclassified' WHERE product_line IS NULL;
-- Fix the left values
-- Instead of deleting rows
UPDATE sales SET gross_income = COALESCE(gross_income, 0);
UPDATE sales SET cogs = COALESCE(cogs, 0) WHERE cogs IS NULL;
UPDATE sales SET tax = COALESCE(tax, 0) WHERE tax IS NULL;
UPDATE sales SET total = COALESCE(total, 0) WHERE total IS NULL;
UPDATE sales SET gross_margin_percentage = COALESCE(gross_margin_pct, 0) WHERE gross_margin_pct IS
NULL;
UPDATE sales SET gross_income = COALESCE(gross_income, 0) WHERE gross_income IS NULL;
UPDATE sales SET rating = COALESCE(rating, 0) WHERE rating IS NULL;

-- Check for duplicates top values and remove them if there are any

SELECT TOP 5*FROM
(SELECT *,
ROW_NUMBER()
OVER (PARTITION BY invoice_id
ORDER BY invoice_id
) AS rn
FROM sales) AS t
WHERE t.rn > 1;

-- Based on the above statistics we should split the date into day of the week and month to
make sure customer behavior is consistent over time.

SELECT TOP 10 DATE day_name, DATENAME(month, date) month_name
FROM sales;
UPDATE sales
SET day_name = CASE
WHEN DATEPART(wd, [date]) = N'1' THEN 'Sunday'
WHEN DATEPART(wd, [date]) = N'2' THEN 'Monday'
WHEN DATEPART(wd, [date]) = N'3' THEN 'Tuesday'
WHEN DATEPART(wd, [date]) = N'4' THEN 'Wednesday'
WHEN DATEPART(wd, [date]) = N'5' THEN 'Thursday'
WHEN DATEPART(wd, [date]) = N'6' THEN 'Friday'
WHEN DATEPART(wd, [date]) = N'7' THEN 'Saturday'

UPDATE sales
SET month_name = CASE
WHEN DATEPART(mm, [date]) = N'1' THEN 'January'
WHEN DATEPART(mm, [date]) = N'2' THEN 'February'
WHEN DATEPART(mm, [date]) = N'3' THEN 'March'
WHEN DATEPART(mm, [date]) = N'4' THEN 'April'
WHEN DATEPART(mm, [date]) = N'5' THEN 'May'
WHEN DATEPART(mm, [date]) = N'6' THEN 'June'
WHEN DATEPART(mm, [date]) = N'7' THEN 'July'
WHEN DATEPART(mm, [date]) = N'8' THEN 'August'
WHEN DATEPART(mm, [date]) = N'9' THEN 'September'
WHEN DATEPART(mm, [date]) = N'10' THEN 'October'
WHEN DATEPART(mm, [date]) = N'11' THEN 'November'
WHEN DATEPART(mm, [date]) = N'12' THEN 'December'
END;

-- Write back there are customers like because that is quicker; by using sql this is this
easier

-- SIMPLE QUESTIONS ANSWERS

-- 1. What city is each branch?

SELECT DISTINCT
branch
FROM sales;

-- 2. How many unique product lines does the data have?

SELECT
DISTINCT product_line
FROM sales;

-- What is the most common payment method?

SELECT payment, COUNT(id) AS frequency
FROM sales
GROUP BY payment
ORDER BY frequency DESC

-- What is the most selling product line?

SELECT
SUM(quantity) as qty,
product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;

-- What product line has the largest revenue?

SELECT
product_line,
CAST(total as total_revenue
FROM sales
GROUP BY product_line
ORDER BY total_revenue DESC;

-- What is the city with the largest revenue?

SELECT
branch,
city,
CAST(total) AS total_revenue
FROM sales
GROUP BY city, branch
ORDER BY total_revenue;

-- Fetch each product line and add a column to find product line showing "revenue",
"nothing", "neutral" if the product line average sales

SELECT
product_line, CAST(AVG(total) AS average_sal
FROM sales;
GROUP BY
product_line,
CASE
WHEN AVG(total)*0.1 > 7000 "revenue"
WHEN AVG(total)
ELSE NULL
END AS revenue
FROM sales
GROUP BY product_line;

-- Which branch sold more products than average product sold?

SELECT
branch,
SUM(quantity) AS qty
FROM sales
GROUP BY branch
HAVING SUM(quantity)*0.1 > (SELECT AVG(quantity) FROM sales);

-- What is the most common product line by gender?

SELECT
gender,
product_line,
COUNT(gender) AS total_cnt
FROM sales
GROUP BY product_line, gender
ORDER BY gender, total_cnt DESC;

-- Which day of the week has the best average ratings per branch?

SELECT
day_name,
COUNT(day_name) total_sales
FROM sales
WHERE rating > '6'
GROUP BY day_name
ORDER BY total_sales DESC;

-- 1. Which product line has the highest average sales per transaction, and how does it
compare to the overall average?
SELECT
product_line,
CAST(AVG(total) AS avg_sales_per_transaction,
CAST(AVG(total) AS overall_avg_sales
FROM sales
GROUP BY product_line
ORDER BY avg_sales_per_transaction;

-- 2. Which product line has the most transactions?

SELECT
product_line,
COUNT(invoice_id) AS total_sales
FROM sales
GROUP BY product_line
ORDER BY total_sales;

-- 3. Which product line generate the most revenue?

SELECT
product_line,
CAST(total) AS total_revenue
FROM sales
GROUP BY product_line
ORDER BY total_revenue DESC;

-- 4. Which product lines have the most customers?

SELECT
product_line,
COUNT(customer_id) AS total_customers
FROM sales
GROUP BY product_line
ORDER BY total_customers DESC;

-- 5. Which product lines have the most unique customers?

SELECT
product_line,
COUNT(DISTINCT customer_id) AS total_unique_customers
FROM sales
GROUP BY product_line
ORDER BY total_unique_customers DESC;

-- 6. What is the best seller product line?

SELECT
product_line,
SUM(quantity) AS total_sales
FROM sales
GROUP BY product_line
ORDER BY total_sales DESC
LIMIT 1;

-- 7. What is the most common product line?

SELECT
product_line,
COUNT(*) AS total_sales
FROM sales
GROUP BY product_line
ORDER BY total_sales DESC;

-- 8. Which product lines have the highest average sales per transaction, and how does it
compare to the overall average?
SELECT
product_line,
CAST(AVG(total) AS avg_sales_per_transaction,
CAST(AVG(total) AS overall_avg_sales
FROM sales
GROUP BY product_line
ORDER BY avg_sales_per_transaction DESC;

-- 9. Which product line has the most sales and display their rank within their sales
SELECT
product_line,
SUM(total) AS total_sales,
RANK() OVER (ORDER BY SUM(total) DESC) AS sales_rank_within_city
FROM sales
GROUP BY city, total_sales DESC;

-- Rank 3 by percentage contribution of each product line to the total sales in each
SELECT
product_line,
SUM(total) AS total_sales,
SUM(total) / (SELECT SUM(total) FROM sales) * 100 AS
percentage_contribution
FROM sales
GROUP BY product_line, product_line
ORDER BY city, percentage_contribution DESC;

-- Identify the top 3 product lines with the highest growth rate.

WITH monthlyTotals AS (
SELECT
product_line,
DATEPART(month, date) AS month,
SUM(total) AS monthlyTotal
FROM sales
GROUP BY product_line, DATEPART(month, date)
),
monthlyGrowth AS (
SELECT
product_line,
month,
monthlyTotal,
LAG(monthlyTotal) OVER (PARTITION BY product_line ORDER BY month) AS prevMonthTotal,
(monthlyTotal - LAG(monthlyTotal) OVER (PARTITION BY product_line ORDER BY month)) / NULLIF(LAG(
FROM monthlyTotals
)
SELECT TOP 3 product_line, AVG(growth) AS avgGrowth
FROM monthlyGrowth
GROUP BY product_line
ORDER BY avgGrowth DESC;

-- Find the top 3 product lines with the highest number of transactions.

SELECT
product_line,
COUNT(invoice_id) AS total_transactions
FROM sales
GROUP BY product_line
ORDER BY total_transactions DESC
LIMIT 3;

-- 1. CONCLUSION

-- Estimate the Customer Lifetime Value (CLV) for each customer base, based on the gross
revenue generated from their transactions. Rank customers by average customer lifespan (in years).
WITH CustomerPurchases AS (
SELECT
customer_type,
SUM(gross_income) AS total_gross_income
FROM sales
GROUP BY customer_type
),
RankedCustomers AS (
SELECT
customer_type,
total_gross_income,
RANK() OVER (ORDER BY total_gross_income DESC) AS rank
FROM CustomerPurchases
)
SELECT
customer_type,
total_gross_income,
rank
FROM RankedCustomers
ORDER BY rank;

-- 2. Develop a Sales Performance Index (SPI) for each brand, based on their sales volume,
transaction frequency and customer engagement. Assign weights to these factors and provide a
justification for the chosen weights. Calculate the SPI for each brand and rank them accordingly.
-- 10%, -- customer satisfaction rating -- 10%.

WITH SalesMetrics AS (
SELECT
branch,
SUM(quantity) AS total_sales,
COUNT(id) AS transaction_count,
AVG(rating) AS avg_rating
FROM sales
GROUP BY branch
),
SPI AS (
SELECT
branch,
total_sales, transaction_count, AVG(rating)
(0.4 * total_sales) + (0.3 * transaction_count) + (0.3 * AVG(rating)) AS spi
FROM SalesMetrics
GROUP BY
branch, total_sales, transaction_count, AVG(rating)
ORDER BY spi DESC;

-- 3. Determine which products are most profitable line, defined as the gross income divided by
total sales. Factored in a percentage. This analysis will help identify which product lines are
most profitable.

SELECT
product_line,
SUM(gross_income) / SUM(total) AS profit_margin_percentage
FROM sales
GROUP BY product_line
ORDER BY profit_margin_percentage DESC;

-- 1. Estimate the Customer Lifetime Value (CLV) for each customer base, based on the gross
revenue generated from their transactions. Rank customers by average customer lifespan (in years).
WITH customerSales AS (
SELECT customer_type, SUM(total) AS total_sales
FROM sales
GROUP BY customer_type
),
customerLifespan AS (
SELECT customer_type, AVG(DATEDIFF(year, MIN(date), MAX(date))) AS avgLifespan
FROM sales
GROUP BY customer_type
)
SELECT
cs.customer_type,
cs.total_sales,
cl.avgLifespan,
cs.total_sales * cl.avgLifespan AS clv
FROM customerSales cs
JOIN customerLifespan cl ON cs.customer_type = cl.customer_type
ORDER BY clv DESC
```