Product Reviews and Customer Segmentation Using SQL

Solving Real-World Business Problems through Data-Driven Insights

Bishesh Pokharel

January 8, 2026

Abstract

This report presents a comprehensive data analytics project utilizing advanced SQL techniques to solve two critical business problems: Customer Segmentation and Product Performance Analysis. The project demonstrates proficiency in query optimization, metric calculation, and actionable insight generation for strategic decision-making.

Contents

1 Executive Summary

This project showcases advanced SQL analytics capabilities applied to real-world business scenarios. Using a structured data warehouse approach (gold layer tables), the analysis addresses two fundamental business challenges:

- **Customer Segmentation:** Identifying and categorizing customers based on behavioral patterns, lifetime value, and engagement metrics to enable targeted marketing strategies.
- **Product Performance Analysis:** Evaluating product-level metrics to optimize inventory, pricing, and product development decisions.

The project employs SQL views, Common Table Expressions (CTEs), window functions, and complex aggregations to transform raw transactional data into actionable business intelligence.

2 Dataset Overview

The analysis leverages a multi-dimensional data warehouse following the star schema design pattern with gold-layer tables representing cleaned, business-ready data.

2.1 Data Scale

- **Transactions:** 60,398 sales records
- **Customers:** 18,484 unique customers
- **Products:** 295 distinct products

2.2 Table Schemas

2.2.1 gold.fact_sales (Fact Table)

Transactional fact table containing 60,398 records with 9 columns:

- `order_number`: Unique order identifier
- `product_key, customer_key`: Foreign keys to dimensions
- `order_date, shipping_date, due_date`: Temporal dimensions
- `sales_amount, quantity, price`: Numeric measures

2.2.2 gold.dim_customers (Customer Dimension)

Customer master data containing 18,484 records with 10 columns:

- `customer_key, customer_id, customer_number`: Identifiers
- `first_name, last_name`: Personal information
- `country, marital_status, gender`: Demographic attributes
- `birthdate, create_date`: Temporal attributes for age and tenure calculations

2.2.3 gold.dim_products (Product Dimension)

Product catalog containing 295 records with 11 columns:

- `product_key, product_id, product_number`: Identifiers
- `product_name`: Product description
- `category_id, category, subcategory`: Hierarchical categorization
- `maintenance, product_line`: Product attributes
- `cost`: Cost basis for profitability analysis
- `start_date`: Product launch date

This dimensional model supports efficient analytical queries while maintaining data integrity through proper foreign key relationships.

3 Data Analysis Techniques

This section outlines the core analytical approaches implemented throughout the project. Each technique addresses specific business questions and demonstrates different SQL capabilities.

3.1 Change Over Time (Trends Analysis)

Objective: Identify temporal patterns in customer behavior and product performance to understand growth trajectories, seasonal variations, and long-term trends.

Key Metrics Analyzed:

- Monthly and yearly sales trends
- Customer acquisition and retention rates over time
- Product sales velocity changes

SQL Techniques Used: Date functions (DATEDIFF, DATEPART, FORMAT), GROUP BY with time dimensions, aggregations across time periods.

Implementation:

This analysis uses date formatting and aggregation to track sales patterns over time:

Listing 1: Monthly Sales Trends Query

```
SELECT
    FORMAT(order_date, 'yyyy-MMM') AS order_date,
    SUM(sales_amount) AS total_sales,
    SUM(DISTINCT customer_key) AS total_customer,
    SUM(quantity) AS total_quantity
FROM gold.fact_sales
WHERE order_date IS NOT NULL
GROUP BY FORMAT(order_date, 'yyyy-MMM')
ORDER BY FORMAT(order_date, 'yyyy-MMM');
```

Key Insights:

- Monthly aggregation reveals seasonality patterns in sales
- Customer acquisition trends visible through unique customer counts

- Quantity trends indicate demand fluctuations
- Data spans from December 2010 through November 2011, showing consistent monthly tracking

Results Messages

	order_date	total_sales	total_customer	total_quantity
1	2010-Dec	43419	121970	14
2	2011-Apr	502042	1421607	157
3	2011-Aug	614516	1736095	193
4	2011-Dec	669395	1279679	222
5	2011-Feb	466307	1157065	144
6	2011-Jan	469795	1059291	144
7	2011-Jul	596710	1854477	188
8	2011-Jun	737793	2367828	230
9	2011-Mar	485165	1321315	150
10	2011-May	561647	1725908	174
11	2011-Nov	660507	1222822	208
12	2011-Oct	708164	1108577	221
13	2011-Sep	603047	967744	185
14	2012-Apr	400324	1789249	219
15	2012-Aug	523887	2343911	294
16	2012-Dec	624454	2838052	483
17	2012-Feb	506992	2001623	260
18	2012-Jan	495363	1576191	252
19	2012-Jul	444533	2034532	246
20	2012-Jun	555142	2670674	318
21	2012-Mar	373478	1734134	212
22	2012-May	358866	1782574	207
23	2012-Nov	537918	2894779	324
24	2012-Oct	535125	2561902	313

Figure 1: Monthly sales trends showing order date, total sales, customer count, and quantity over time

Business Application: This trend analysis enables forecasting, inventory planning, and identification of peak sales periods for resource allocation.

3.2 Cumulative Analysis

Objective: Calculate running totals and cumulative metrics to understand overall growth patterns and identify inflection points in business performance.

Key Metrics Analyzed:

- Cumulative revenue by customer and product
- Running total of sales over time
- Moving averages for smoothing trends

SQL Techniques Used: Window functions (SUM OVER, AVG OVER), ORDER BY within window frames, date truncation functions.

Implementation:

This analysis leverages advanced window functions to calculate running totals and moving averages:

Listing 2: Cumulative Sales with Moving Average

```
SELECT
    order_date,
    total_sales,
    average_sales,
    SUM(total_sales) OVER(ORDER BY order_date) AS running_total_sales,
    SUM(average_sales) OVER(ORDER BY order_date) AS moving_average
FROM (
    SELECT
        DATETRUNC(YEAR, order_date) AS order_date,
        SUM(sales_amount) AS total_sales,
        AVG(sales_amount) AS average_sales
    FROM gold.fact_sales
    WHERE order_date IS NOT NULL
    GROUP BY DATETRUNC(YEAR, order_date)
) t;
```

Key Insights:

- Running total shows cumulative revenue growth from 2010 to 2014
- Moving average smooths out year-to-year fluctuations
- Cumulative sales reached nearly 30M by 2014
- Moving average provides trend direction independent of volatility

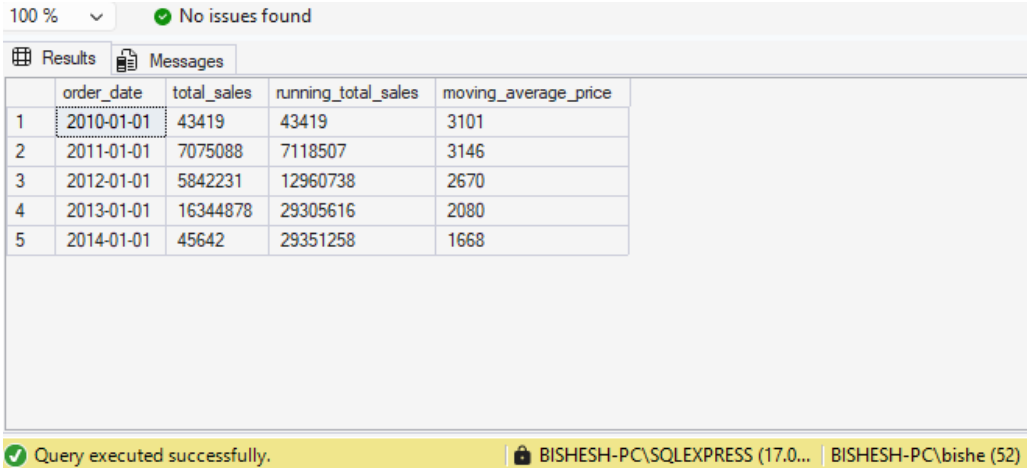


Figure 2: Cumulative sales analysis with running totals and moving averages by year

Business Application: Cumulative metrics enable long-term growth assessment, milestone tracking, and identification of acceleration or deceleration in business performance.

3.3 Performance Analysis

Objective: Benchmark and compare performance across different entities (customers, products, time periods) to identify top performers and underperformers.

Key Metrics Analyzed:

- Product year-over-year performance comparison
- Sales performance relative to product average
- Performance change classification (Above avg, Below avg, Avg)

SQL Techniques Used: Window functions (AVG OVER with PARTITION BY), LAG function for previous period comparison, CASE statements for performance classification.

Implementation:

This analysis uses window functions to calculate performance metrics and year-over-year comparisons:

Listing 3: Year-over-Year Product Performance Analysis

```
WITH yearly_sales AS (
    SELECT
        YEAR(s.order_date) AS order_year,
        p.product_name,
        SUM(s.sales_amount) AS total_sales
    FROM gold.fact_sales AS s
    LEFT JOIN gold.dim_products AS p
        ON s.product_key = p.product_key
    WHERE order_date IS NOT NULL
    GROUP BY YEAR(s.order_date), p.product_name
)
SELECT
    order_year,
    product_name,
    total_sales,
    AVG(total_sales) OVER(PARTITION BY product_name) AS sales_avg,
    total_sales - AVG(total_sales) OVER(PARTITION BY product_name)
        AS average_sales_performance,
```

```
    CASE WHEN total_sales - AVG(total_sales) OVER(PARTITION BY
        product_name) > 0
          THEN 'Above avg'
          WHEN total_sales - AVG(total_sales) OVER(PARTITION BY
              product_name) < 0
          THEN 'Below avg'
          ELSE 'Avg' END AS avg_change,
    LAG(total_sales) OVER(PARTITION BY product_name ORDER BY order_year
        )
         AS previous_year_sales,
    total_sales - LAG(total_sales) OVER(PARTITION BY product_name ORDER
         BY order_year)
         AS previous_year_performance,
    CASE WHEN total_sales - LAG(total_sales) OVER(PARTITION BY
        product_name ORDER BY order_year) > 0
          THEN 'Above avg'
          WHEN total_sales - LAG(total_sales) OVER(PARTITION BY
              product_name ORDER BY order_year) < 0
          THEN 'Below avg'
          ELSE 'Avg' END AS previous_year_diff
FROM yearly_sales
ORDER BY product_name;
```

Key Insights:

- Products show varying performance patterns across years
- LAG function enables direct year-over-year comparison
- Performance classification provides actionable categories
- Some products consistently perform above average, while others fluctuate

91 % No issues found Ln: 16, Ch: 50 TABS CRLF Windows 12

Results Messages

	order_year	product_name	total_sales	sales_avg	average_sales_performance	avg_change	previous_year_Sales	previous_year_performance	previous_year_diff
1	2012	All-Purpose Bike Stand	159	13197	-13038	Below avg	NULL	NULL	Avg
2	2013	All-Purpose Bike Stand	37683	13197	24486	Above avg	159	37524	Above avg
3	2014	All-Purpose Bike Stand	1749	13197	-11448	Below avg	37683	-35934	Below avg
4	2012	AWC Logo Cap	72	6570	-6498	Below avg	NULL	NULL	Avg
5	2013	AWC Logo Cap	18891	6570	12321	Above avg	72	18819	Above avg
6	2014	AWC Logo Cap	747	6570	-5823	Below avg	18891	-18144	Below avg
7	2013	Bike Wash - Dissolver	6960	3636	3324	Above avg	NULL	NULL	Avg
8	2014	Bike Wash - Dissolver	312	3636	-3324	Below avg	6960	-6648	Below avg
9	2013	Classic Vest- L	11968	6240	5728	Above avg	NULL	NULL	Avg
10	2014	Classic Vest- L	512	6240	-5728	Below avg	11968	-11456	Below avg
11	2013	Classic Vest- M	11840	6368	5472	Above avg	NULL	NULL	Avg
12	2014	Classic Vest- M	896	6368	-5472	Below avg	11840	-10944	Below avg
13	2012	Classic Vest- S	64	3648	-3584	Below avg	NULL	NULL	Avg
14	2013	Classic Vest- S	10368	3648	6720	Above avg	64	10304	Above avg
15	2014	Classic Vest- S	512	3648	-3136	Below avg	10368	-9856	Below avg
16	2012	Fender Set - Mountain	110	15554	-15444	Below avg	NULL	NULL	Avg
17	2013	Fender Set - Mountain	44484	15554	28930	Above avg	110	44374	Above avg
18	2014	Fender Set - Mountain	2068	15554	-13486	Below avg	44484	-42416	Below avg
19	2012	Half-Finger Gloves- L	24	3544	-3520	Below avg	NULL	NULL	Avg

Figure 3: Product performance analysis showing yearly sales, averages, and YoY changes

Business Application: Performance analysis identifies products requiring promotional support, inventory adjustments, or potential discontinuation based on historical trends.

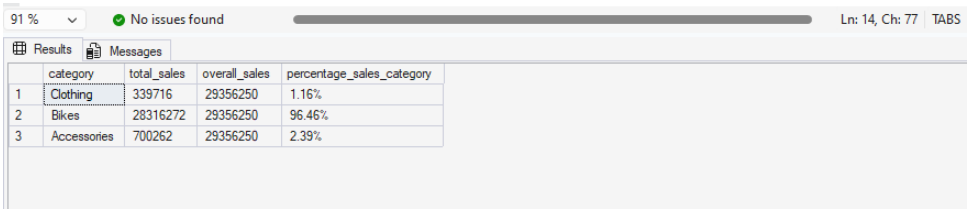
3.4 Part-to-Whole Analysis

Objective: Understand proportional contributions of different segments to overall business metrics, enabling resource allocation decisions.

Key Metrics Analyzed:

- Revenue contribution by customer segment
- Product category sales distribution
- Age group purchasing patterns

SQL Techniques Used: Percentage calculations, GROUP BY with ROLLUP, subqueries for total calculations, CASE statements for categorization.



	category	total_sales	overall_sales	percentage_sales_category
1	Clothing	339716	29356250	1.16%
2	Bikes	28316272	29356250	96.46%
3	Accessories	700262	29356250	2.39%

Figure 4: Part-to-Whole Analysis Solution

3.5 Data Segmentation

Objective: Categorize entities into meaningful groups based on behavioral and demographic characteristics to enable targeted strategies.

Key Segmentation Dimensions:

- Customer segments: VIP, Regular, New
- Age groups: <20, 20-29, 30-39, 40-49, 50+
- Product segments: High-Performer, Mid-Range, Low-Performer

SQL Techniques Used: CASE statements, conditional logic, multi-criteria segmentation, business rule implementation.

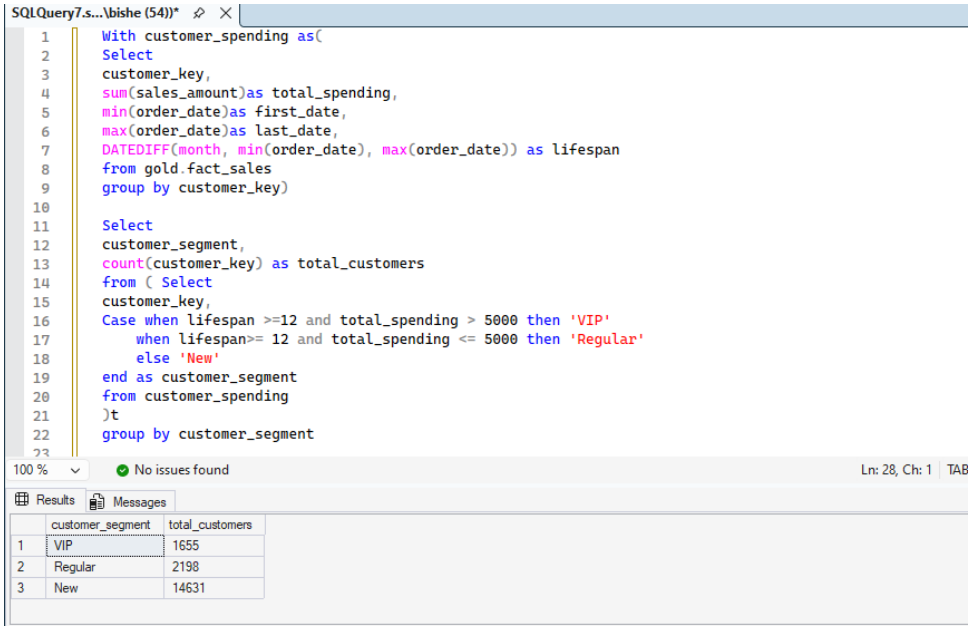


Figure 5: Customer Segmentation – SQL Query Results

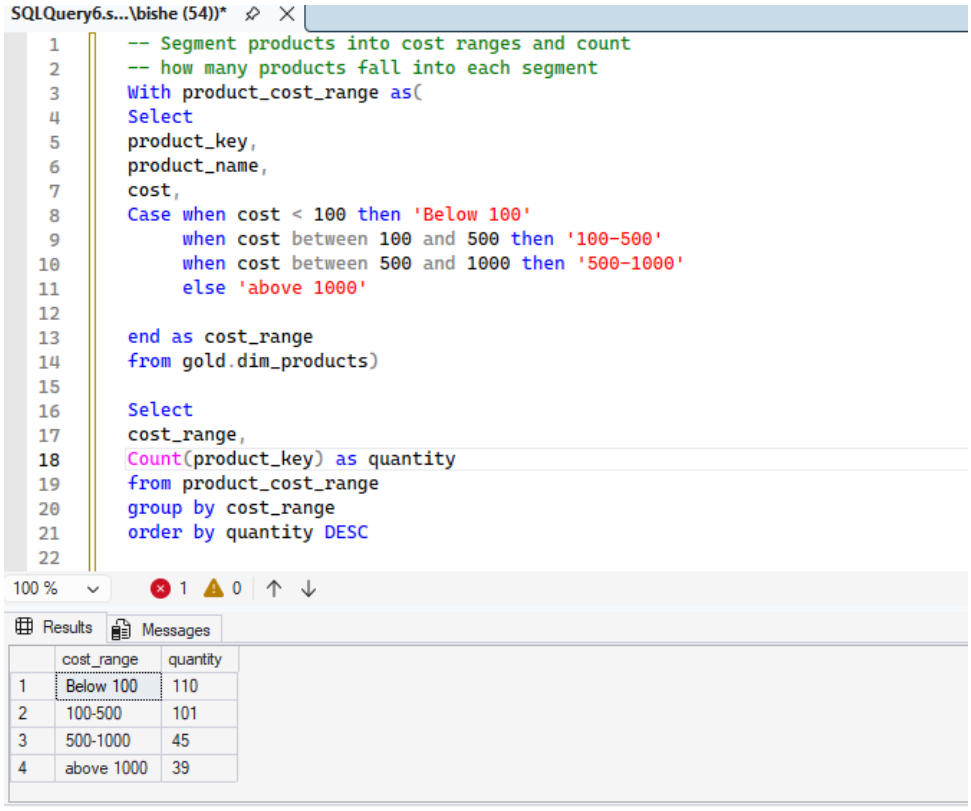


Figure 6: Product Segmentation – SQL Query Results

4 Real-World Business Problem 1: Customer Segmentation

4.1 Problem Statement

Understanding customer behavior and value is critical for optimizing marketing spend, improving retention, and maximizing customer lifetime value. The business needs to segment customers based on their purchasing patterns, engagement history, and demographic characteristics.

4.2 Business Objectives

- Identify high-value customers (VIP segment) for personalized engagement
- Recognize regular customers for loyalty programs
- Detect new customers requiring onboarding strategies
- Analyze age group preferences for targeted product recommendations
- Calculate customer recency to identify at-risk customers

4.3 Data Model

The analysis utilizes a star schema with the following key tables:

- `gold.fact_sales`: Transactional fact table
- `gold.dim_customers`: Customer dimension table

4.4 Methodology

The customer segmentation solution implements a multi-stage SQL view (`gold_report_customer`) using CTEs:

Stage 1 - Base Query:

- Joins sales facts with customer dimensions
- Calculates customer age from birthdate
- Filters valid orders

Stage 2 - Customer Aggregation:

- Aggregates transaction-level data to customer level
- Calculates total orders, sales, products purchased
- Determines customer lifespan (months between first and last order)
- Identifies last order date for recency calculation

Stage 3 - Segmentation & KPIs:

- Segments customers into VIP, Regular, and New categories
- Groups customers by age brackets
- Calculates Average Order Value (AOV)
- Computes Average Monthly Spend
- Determines recency (months since last purchase)

4.5 Segmentation Logic

Customer Segments:

- **VIP:** Lifespan $\geq$ 12 months AND Total Sales $>$ \$5,000
- **Regular:** Lifespan $\geq$ 12 months AND Total Sales $\leq$ \$5,000
- **New:** Lifespan $<$ 12 months

Age Groups:

- Less than 20
- 20-29
- 30-39
- 40-49
- 50 and above

4.6 Key SQL Implementation

Listing 4: Customer Segmentation View (Excerpt)

```sql
CREATE VIEW gold_report_customer AS
WITH base_query AS (
    SELECT s.sales_amount, s.product_key, s.order_date,
           c.customer_key, c.customer_number,
           CONCAT(c.first_name,' ',c.last_name) AS customer_name,
           DATEDIFF(YEAR, c.birthdate, GETDATE()) AS age
    FROM gold.fact_sales AS s
    LEFT JOIN gold.dim_customers AS c
        ON s.customer_key = c.customer_key
    WHERE order_date IS NOT NULL
),
customer_aggregation AS (
    SELECT customer_key, customer_name, age,
           MAX(order_date) AS last_order_date,
           SUM(sales_amount) AS total_sales,
           COUNT(DISTINCT product_key) AS total_product,
           DATEDIFF(MONTH, MIN(order_date), MAX(order_date)) AS
              lifespan
    FROM base_query
    GROUP BY customer_key, customer_name, age
)
SELECT *,
       CASE WHEN lifespan >= 12 AND total_sales > 5000 THEN 'VIP'
            WHEN lifespan >= 12 AND total_sales <= 5000 THEN 'Regular'
            ELSE 'New' END AS customer_segment
FROM customer_aggregation;
```

4.7 Query Results

	age	customer_number	customer_key	customer_name	total_product	total_orders	total_sales	recency_months	total_quantity	lifespan	age_group	customer_details
1	55	AW00011000	1	Jon Yang	8	3	8249	152	8	28	50 and above	VIP
2	50	AW00011001	2	Eugene Huang	10	3	6384	145	11	35	50 and above	VIP
3	55	AW00011002	3	Ruben Torres	4	3	8114	155	4	25	50 and above	VIP
4	53	AW00011003	4	Christy Zhu	9	3	8139	152	9	29	50 and above	VIP
5	47	AW00011004	5	Elizabeth Joh...	6	3	8196	152	6	28	40-49	VIP
6	50	AW00011005	6	Julio Ruiz	6	3	8121	152	6	29	50 and above	VIP
7	50	AW00011006	7	Janet Alvarez	5	3	8119	152	5	28	50 and above	VIP
8	57	AW00011007	8	Marco Mehta	8	3	8211	154	8	26	50 and above	VIP
9	51	AW00011008	9	Rob Verhoff	7	3	8106	154	7	26	50 and above	VIP
10	57	AW00011009	10	Shannon Carl...	5	3	8091	152	5	28	50 and above	VIP
11	57	AW00011010	11	Jacquelyn Su...	4	3	8088	152	4	28	50 and above	VIP
12	57	AW00011011	12	Curtis Lu	4	3	8133	154	4	27	50 and above	VIP
13	47	AW00011012	13	Lauren Walker	5	2	81	147	5	7	40-49	New
14	47	AW00011013	14	Ian Jenkins	5	2	114	144	5	9	40-49	New
15	53	AW00011014	15	Sydney Bennett	5	2	138	153	6	1	50 and above	New

Figure 7: Customer Segmentation Results (SQL Query Output)

4.8 Business Value & Insights

This customer segmentation enables:

- Targeted marketing campaigns for each segment
- Identification of at-risk customers based on recency
- Resource allocation for high-value customer retention
- Age-specific product recommendations
- Predictive modeling for customer lifetime value

5 Real-World Business Problem 2: Product Performance Analysis

5.1 Problem Statement

Product portfolio management requires understanding which products drive revenue, which are underperforming, and how to optimize inventory and marketing investments. The business needs granular product-level metrics to make data-driven decisions.

5.2 Business Objectives

- Identify high-performing products for promotion and inventory prioritization
- Detect low-performing products for potential discontinuation or re-pricing
- Analyze product lifecycle metrics (lifespan, recency)
- Calculate profitability indicators (cost vs. selling price)
- Understand customer reach per product

5.3 Data Model

The analysis utilizes the following tables:

- `gold.fact_sales`: Transactional fact table
- `gold.dim_products`: Product dimension table

5.4 Methodology

The product analysis solution implements a SQL view (`gold_report_products`) with a three-stage CTE structure:

Stage 1 - Base Query:

- Joins sales facts with product dimensions
- Retrieves core fields: product details, sales amounts, order dates
- Filters valid sales records

Stage 2 - Product Aggregations:

- Aggregates transaction data to product level
- Calculates total orders, sales, quantity sold
- Determines unique customer count per product
- Computes product lifespan and last sale date
- Derives average selling price

Stage 3 - Segmentation & KPIs:

- Segments products into performance tiers
- Calculates Average Order Revenue (AOR)
- Computes Average Monthly Revenue
- Determines recency (months since last sale)

5.5 Product Segmentation Logic

Performance Tiers:

- **High-Performer:** Total Sales > $50,000
- **Mid-Range:** Total Sales between $10,000 and $50,000
- **Low-Performer:** Total Sales < $10,000

5.6 Key SQL Implementation

Listing 5: Product Performance View (Excerpt)

```sql
CREATE VIEW gold_report_products AS
WITH base_query AS (
    SELECT f.order_number, f.order_date, f.sales_amount,
           f.quantity, p.product_key, p.product_name,
           p.category, p.subcategory, p.cost
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p
        ON f.product_key = p.product_key
    WHERE order_date IS NOT NULL
),
product_aggregations AS (
    SELECT product_key, product_name, category,
           DATEDIFF(MONTH, MIN(order_date), MAX(order_date)) AS
              lifespan,
           COUNT(DISTINCT customer_key) AS total_customers,
           SUM(sales_amount) AS total_sales
    FROM base_query
    GROUP BY product_key, product_name, category
)
SELECT *,
       CASE WHEN total_sales > 50000 THEN 'High-Performer'
            WHEN total_sales >= 10000 THEN 'Mid-Range'
            ELSE 'Low-Performer' END AS product_segment
FROM product_aggregations;
```

5.7 Query Results

	product_key	product_name	category	subcategory	cost	last_sale_date	recency_in_months	product_segment	lifespan	total_orders	total_sales	total_quantity
1	3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	2011-12-27	169	High-Performer	11	49	165375	49
2	4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	2011-12-27	169	High-Performer	11	45	151875	45
3	5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	2011-12-21	169	High-Performer	11	60	202500	60
4	6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	2011-12-26	169	High-Performer	12	57	192375	57
5	7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	2011-12-22	169	High-Performer	12	58	197200	58
6	8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	2011-12-28	169	High-Performer	11	42	142800	42
7	9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	2011-12-12	169	High-Performer	12	49	166600	49
8	10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	2011-12-23	169	High-Performer	11	36	122400	36
9	16	Road-150 Red- 44	Bikes	Road Bikes	2171	2011-12-28	169	High-Performer	12	281	1005418	281
10	17	Road-150 Red- 48	Bikes	Road Bikes	2171	2011-12-28	169	High-Performer	12	337	1205786	337
11	18	Road-150 Red- 52	Bikes	Road Bikes	2171	2011-12-27	169	High-Performer	12	302	1080556	302
12	19	Road-150 Red- 56	Bikes	Road Bikes	2171	2011-12-27	169	High-Performer	12	295	1055510	295
13	20	Road-150 Red- 62	Bikes	Road Bikes	2171	2011-12-28	169	High-Performer	12	336	1202208	336
14	36	Road-650 Black- 44	Bikes	Road Bikes	487	2012-12-26	157	Mid-Range	23	63	47565	63
15	37	Road-650 Black- 48	Bikes	Road Bikes	487	2012-12-25	157	Mid-Range	21	60	45552	60
16	38	Road-650 Black- 52	Bikes	Road Bikes	487	2012-12-19	157	High-Performer	23	89	66915	89

Figure 8: Product Performance Results (SQL Query Output)

5.8 Key Performance Indicators (KPIs)

- **Total Sales:** Revenue generated by each product

- **Total Orders:** Number of transactions per product
- **Total Customers:** Unique customer reach
- **Average Selling Price:** Price point analysis
- **Average Order Revenue:** Revenue per transaction
- **Average Monthly Revenue:** Normalized revenue performance
- **Recency:** Months since last sale (staleness indicator)
- **Lifespan:** Product age in the market

5.9 Business Value & Insights

This product performance analysis enables:

- Strategic inventory management based on performance tiers
- Identification of products requiring promotional support
- Detection of stale inventory (high recency)
- Pricing optimization using cost vs. selling price analysis
- Cross-selling opportunities based on customer reach metrics

6 Technical Implementation Highlights

6.1 SQL Techniques Demonstrated

- **Common Table Expressions (CTEs):** Multi-stage query organization for readability and performance
- **Window Functions:** Advanced aggregations and rankings
- **Date Functions:** DATEDIFF, GETDATE() for temporal calculations
- **Conditional Logic:** Complex CASE statements for segmentation
- **NULL Handling:** NULLIF, COALESCE for robust calculations
- **String Functions:** CONCAT for data formatting
- **View Creation:** Reusable, performant data structures

6.2 Data Quality Considerations

- Filtering NULL order dates to ensure data integrity
- Division-by-zero protection in average calculations
- Consistent date handling across queries
- Proper handling of edge cases (zero lifespan, zero orders)

7 Conclusion

This project demonstrates comprehensive SQL analytics capabilities applied to real-world business challenges. The customer segmentation and product performance analyses provide actionable insights that directly support strategic decision-making in marketing, inventory management, and customer relationship management.

The implementation showcases best practices in SQL development, including modular query design, efficient aggregations, and scalable view architectures suitable for enterprise data warehousing environments.

8 Repository Access

Full SQL code, documentation, and additional analyses are available in the project repository:

Github Link of Project