# Interactive SQL Analysis Report: Product Performance, Time Trends, Customer Segmentation & More

**Datasets Used:** sales, customers, products  
**Business Domain:** Bike Retail

**Purpose:** This report provides a comprehensive SQL-based data analysis of a bike retail dataset. It includes product ranking, customer segmentation, time-series trends, and performance benchmarking. Each section is designed to generate business insights that drive decision-making for sales strategy, marketing, and operations.

## 🏆 Ranking Analysis

### 1. Top 5 Products by Revenue

SELECT \*  
FROM (  
 SELECT  
 p.product\_name,  
 SUM(f.sales\_amount) AS total\_revenue,  
 RANK() OVER (ORDER BY SUM(f.sales\_amount) DESC) AS rank\_products  
 FROM sales f  
 LEFT JOIN products p ON p.product\_key = f.product\_key  
 GROUP BY p.product\_name  
) AS ranked\_products  
WHERE rank\_products <= 5;

**Insight:** Identifies the most revenue-generating products.

**Impact:** Supports product prioritization in promotions and inventory planning.

### 2. Top 10 Revenue-Generating Customers

SELECT  
 c.customer\_key,  
 CONCAT(c.first\_name, ' ', c.last\_name) AS FULL\_NAME,  
 SUM(f.sales\_amount) AS total\_revenue  
FROM sales f  
LEFT JOIN customers c ON c.customer\_key = f.customer\_key  
GROUP BY c.customer\_key, FULL\_NAME  
HAVING c.customer\_key IS NOT NULL  
ORDER BY total\_revenue DESC  
LIMIT 10;

**Insight:** Highlights the most valuable customers by total revenue.

**Impact:** Useful for implementing loyalty or VIP reward programs.

### 3. Customers with the Fewest Orders

SELECT  
 c.customer\_key,  
 CONCAT(c.first\_name, ' ', c.last\_name) AS FULL\_NAME,  
 COUNT(f.order\_number) AS order\_count  
FROM sales f  
LEFT JOIN customers c ON c.customer\_key = f.customer\_key  
GROUP BY c.customer\_key, FULL\_NAME  
HAVING c.customer\_key IS NOT NULL  
ORDER BY order\_count  
LIMIT 10;

**Insight:** Identifies customers with minimal engagement.

**Impact:** Supports targeted re-engagement campaigns.

## 📈 Change Over Time Analysis

### 4. Monthly Sales Trend

SELECT  
 DATE\_FORMAT(order\_date,'%Y-%b') AS year\_order\_date,  
 MONTH(order\_date) AS month\_order\_date,  
 SUM(price) AS sum\_price  
FROM sales  
GROUP BY year\_order\_date, month\_order\_date  
HAVING year\_order\_date IS NOT NULL  
ORDER BY year\_order\_date, month\_order\_date;

**Insight:** Detects monthly seasonality and revenue trends.

**Impact:** Informs promotional and inventory timing decisions.

### 5. Cumulative Sales and Moving Average

SELECT year\_month\_order\_date,  
 sum\_price,  
 avg\_price,  
 SUM(sum\_price) OVER(PARTITION BY year\_order\_date ORDER BY year\_month\_order\_date) AS running\_total\_sales,  
 AVG(avg\_price) OVER(PARTITION BY year\_order\_date ORDER BY year\_month\_order\_date) AS moving\_average\_price  
FROM (  
 SELECT  
 DATE\_FORMAT(order\_date,'%Y-%m') AS year\_month\_order\_date,  
 YEAR(order\_date) AS year\_order\_date,  
 AVG(price) AS avg\_price,  
 SUM(price) AS sum\_price  
 FROM sales  
 GROUP BY year\_month\_order\_date, year\_order\_date  
 HAVING year\_order\_date IS NOT NULL  
) t;

**Insight:** Highlights cumulative performance and smoothed pricing trends.

**Impact:** Helps refine pricing strategies and forecast planning.

## 🔁 Yearly Performance Analysis

### 6. Year-over-Year Product Sales Comparison

WITH Yearly\_product\_sales AS (  
 SELECT YEAR(f.order\_date) AS year\_order\_date,  
 SUM(f.sales\_amount) AS sales\_sum,  
 g.product\_name AS p\_name  
 FROM sales f  
 LEFT JOIN products g ON f.product\_key = g.product\_key  
 GROUP BY year\_order\_date, product\_name  
 HAVING year\_order\_date IS NOT NULL  
)  
SELECT p\_name,  
 year\_order\_date,  
 sales\_sum,  
 AVG(sales\_sum) OVER(PARTITION BY p\_name) AS avg\_sales,  
 sales\_sum - AVG(sales\_sum) OVER(PARTITION BY p\_name) AS avg\_diff,  
 CASE  
 WHEN sales\_sum > AVG(sales\_sum) OVER(PARTITION BY p\_name) THEN 'above average'  
 WHEN sales\_sum < AVG(sales\_sum) OVER(PARTITION BY p\_name) THEN 'below average'  
 ELSE 'equal'  
 END AS avg\_change,  
 LAG(sales\_sum) OVER(PARTITION BY p\_name ORDER BY year\_order\_date) AS sales\_diff\_pyear,  
 CASE  
 WHEN LAG(sales\_sum) OVER(PARTITION BY p\_name ORDER BY year\_order\_date) > sales\_sum THEN 'decrease'  
 WHEN LAG(sales\_sum) OVER(PARTITION BY p\_name ORDER BY year\_order\_date) < sales\_sum THEN 'increase'  
 ELSE 'no change'  
 END AS py\_change  
FROM Yearly\_product\_sales  
ORDER BY p\_name, year\_order\_date;

**Insight:** Evaluates each product’s annual performance.

**Impact:** Supports decision-making on product line continuation or retirement.

## 🧩 Part-to-Whole Analysis

### 7. Subcategory Contribution to Total Sales

WITH subcategory\_analysis AS (  
 SELECT p.subcategory, SUM(f.sales\_amount) AS sales\_sum  
 FROM sales f  
 LEFT JOIN products p ON p.product\_key = f.product\_key  
 GROUP BY p.subcategory  
)  
SELECT subcategory, sales\_sum,  
 SUM(sales\_sum) OVER() AS total\_sales,  
 CONCAT(ROUND(sales\_sum / SUM(sales\_sum) OVER() \* 100, 2), '%') AS percentage\_ratio  
FROM subcategory\_analysis  
ORDER BY percentage\_ratio DESC;

**Insight:** Measures which subcategories contribute most to total revenue.

**Impact:** Focus marketing on high-contribution product lines.

### 8. Category Contribution to Total Sales

WITH category\_analysis AS (  
 SELECT p.category, SUM(f.sales\_amount) AS sales\_sum  
 FROM sales f  
 LEFT JOIN products p ON p.product\_key = f.product\_key  
 GROUP BY p.category  
)  
SELECT category,  
 CONCAT(ROUND(sales\_sum / SUM(sales\_sum) OVER() \* 100, 2), '%') AS percentage\_ratio  
FROM category\_analysis  
ORDER BY percentage\_ratio DESC;

**Insight:** Highlights revenue distribution across broad product categories.

**Impact:** Guides investment and focus in product lines.

## 🔎 Segmentation Analysis

### 9. Product Price Range Segmentation

WITH product\_cost\_analyze AS (  
 SELECT product\_key, cost,  
 CASE  
 WHEN cost < 100 THEN 'below 100'  
 WHEN cost BETWEEN 100 AND 500 THEN '100-500'  
 WHEN cost BETWEEN 500 AND 1000 THEN '500-1000'  
 ELSE 'above 1000'  
 END AS cost\_range  
 FROM products  
)  
SELECT cost\_range, COUNT(product\_key) AS no\_of\_product  
FROM product\_cost\_analyze  
GROUP BY cost\_range;

**Insight:** Classifies product count by pricing bands.

**Impact:** Useful for setting pricing tiers and stock allocation.

### 10. Customer Loyalty Segmentation

WITH customer\_category AS (  
 SELECT c.customer\_key,  
 SUM(f.sales\_amount),  
 ROUND(DATEDIFF(MAX(order\_date), MIN(order\_date)) / 30) AS lifespan,  
 CASE  
 WHEN SUM(f.sales\_amount) > 5000 AND ROUND(DATEDIFF(MAX(order\_date), MIN(order\_date)) / 30) > 12 THEN 'VIP'  
 WHEN SUM(f.sales\_amount) BETWEEN 1000 AND 5000 AND ROUND(DATEDIFF(MAX(order\_date), MIN(order\_date)) / 30) > 12 THEN 'GOLD'  
 ELSE 'NEW'  
 END AS customer\_segment  
 FROM customers c  
 LEFT JOIN sales f ON f.customer\_key = c.customer\_key  
 GROUP BY c.customer\_key  
)  
SELECT customer\_segment, COUNT(customer\_key) AS no\_of\_customer  
FROM customer\_category  
GROUP BY customer\_segment;

**Insight:** Segments customers by loyalty and lifetime value.

**Impact:** Enables tiered marketing and retention strategies.

## 📌 Final Summary

### Key Business Insights

* Top products and customers account for a large portion of overall revenue.
* Clear monthly trends indicate seasonal buying behavior.
* Specific categories dominate revenue share.
* Customer and product segments are well-defined for strategic actions.

### Strategic Recommendations

* Promote high-revenue products and categories more aggressively.
* Engage low-activity customers through targeted campaigns.
* Use customer loyalty data to create retention and upselling strategies.
* Refine pricing structure based on product distribution.