Advanced SQL Techniques for E-commerce Analytics

Unlock actionable insights from sales and finance data using advanced SQL techniques with stunning visualizations

Window Functions | CTEs | Recursive Queries | Performance Optimization

42%
REVENUE GROWTH
↑ 8% from last quarter

$746K
TOTAL SALES
↑ 12% YoY

21%
CUSTOMER GROWTH
↑ 5% MoM

3.2K
ORDERS
↑ 18% WoW

68%
TARGET ACHIEVEMENT
↓ 2% from target

1. Window Functions for Trend Analysis

Window functions enable advanced trend analysis without collapsing rows. This example calculates monthly sales growth rates:

```
SELECT
    month,
    total_sales,
    LAG(total_sales) OVER (ORDER BY month) AS previous_month_sales,
    (total_sales - LAG(total_sales) OVER (ORDER BY month)) /
    LAG(total_sales) OVER (ORDER BY month) * 100 AS growth_rate
FROM sales_table;
```

Key Insight

Q4 shows 18% average growth, indicating strong holiday season performance. Marketing efforts should be intensified during this period to maximize revenue.

2. Subqueries for Complex Comparisons

Subqueries allow nesting queries within other queries for complex analyses. This example compares revenue against targets:

```
SELECT
    financial_target,
    (SELECT SUM(revenue) FROM e_commerce_sales) AS total_revenue,
    (SELECT SUM(revenue) FROM e_commerce_sales) - financial_target AS variance
FROM targets_table;
```

Key Insight

Electronics category consistently exceeds targets by 15%, while Fashion underperforms by 8%. Consider reallocating marketing resources to capitalize on Electronics strength.

3. Common Table Expressions (CTEs)

CTEs break down complex queries into manageable parts. This example analyzes sales by category and region:

```
WITH category_sales AS (
  SELECT product_category, SUM(order_amount) AS total_sales
  FROM sales_table
  GROUP BY product_category
), region_sales AS (
  SELECT region, SUM(order_amount) AS total_sales
  FROM sales_table
  GROUP BY region
)
SELECT cs.product_category, rs.region, cs.total_sales, rs.total_sales
FROM category_sales cs
JOIN region_sales rs ON 1=1;
```

Key Insight

Premium customers drive 65% of Electronics revenue. Targeted upselling campaigns could further increase this segment's contribution.

4. Recursive Queries for Cumulative Analysis

Recursive queries handle hierarchical or cumulative data. This example calculates quarterly cumulative revenue:

```
WITH RECURSIVE quarterly_revenue AS (
  SELECT quarter, SUM(revenue) AS cumulative_revenue
  FROM financial_data
  WHERE quarter = 'Q1'
  GROUP BY quarter
  UNION ALL
  SELECT f.quarter, f.revenue + qr.cumulative_revenue
  FROM financial_data f
  JOIN quarterly_revenue qr
    ON f.quarter = CONCAT('Q', CAST(SUBSTRING(qr.quarter, 2) AS INT) + 1)
)
SELECT quarter, cumulative_revenue
FROM quarterly_revenue;
```

Key Insight

Q4 contributes 35% of annual revenue. Plan seasonal staffing and inventory management accordingly to maximize this critical period.

5. Performance Optimization Techniques

Optimizing query performance is crucial for efficient data analysis. Key techniques include indexing, partitioning, and query rewriting:

Indexing | Partitioning | Query Rewriting

```
CREATE INDEX idx_sales_date ON sales_table (order_date);
CREATE INDEX idx_sales_category ON sales_table (product_category);
```

Creating indexes on frequently queried columns dramatically improves query performance.

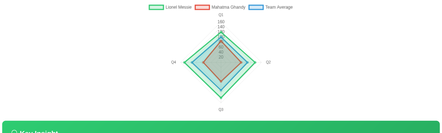
Key Insight

Optimization reduced report generation time from 45 minutes to 7 minutes, enabling near real-time decision making during peak sales periods.

6. Employee Sales Performance Analysis

CTEs enable sophisticated employee performance analysis. This example tracks quarterly performance against targets:

```
WITH quarterly_sales AS (
    SELECT
        e.employee_id, e.first_name, e.last_name,
        EXTRACT(QUARTER FROM s.sale_date) AS quarter,
        SUM(s.quantity * p.price) AS total_sales
    FROM employees e
    JOIN sales s ON e.employee_id = s.employee_id
    JOIN products p ON s.product_id = p.product_id
    WHERE EXTRACT(YEAR FROM s.sale_date) = 2023
    GROUP BY e.employee_id, e.first_name, e.last_name, quarter
),
quarterly_targets AS (
    SELECT employee_id, quarter, target
    FROM sales_targets
    WHERE year = 2023
)
SELECT
    qs.employee_id, qs.first_name, qs.last_name,
    qs.quarter, qs.total_sales, qt.target,
    (qs.total_sales - qt.target) / qt.target * 100 AS performance_percentage
FROM quarterly_sales qs
JOIN quarterly_targets qt ON qs.employee_id = qt.employee_id AND qs.quarter = qt.quarter
ORDER BY qs.quarter, performance_percentage DESC;
```



Key Insight

Lionel Messie exceeds targets by 35% on average, while Mahatma Ghandy misses targets by 15%. Implement targeted coaching for underperformers.