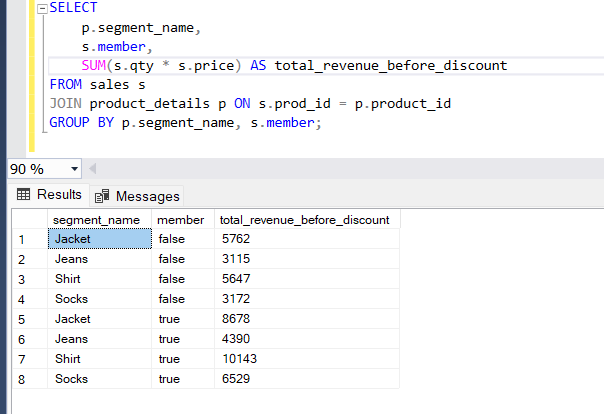
**Hexaware Technologies**

SQL-Coding Challenge

**Joins:**

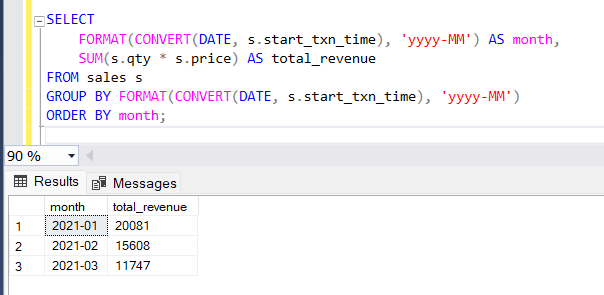
A Join combines data from multiple tables based on related columns, allowing you to retrieve comprehensive datasets. Common types include Inner Join (returns matching records), Left Join (returns all records from the left table), Right Join (returns all from the right table), and Full Join (combines both left and right). Joins are used to consolidate information spread across different tables, like linking customer data with their orders.

**1. Segmented Revenue Analysis by Membership and Product Segment**



**Explanation:** In this query we are using **JOIN** to combine product segment information and groups data by segment and membership status. This provides a detailed view of revenue generated by each segment, showing the impact of membership on different segments.

**2. Monthly Revenue Trend Analysis**

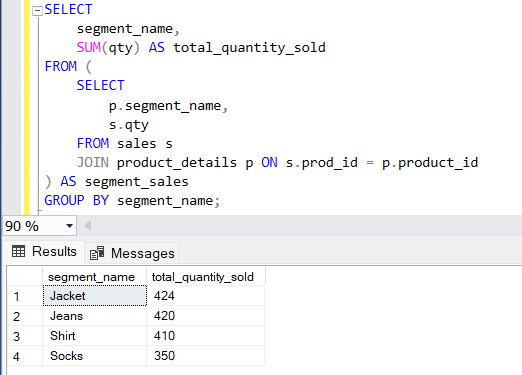


**Explanation:** This query extracts the month from the transaction time, groups data by month, and calculates monthly revenue totals. The output will show monthly revenue trends, useful for identifying seasonal demand patterns.

**Subqueries:**

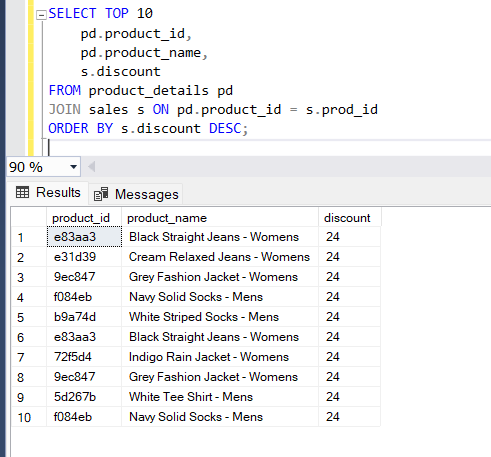
A **Subquery** is a query embedded within another query to perform operations that depend on the results of the first query. They are useful for filtering or calculating data that can't be easily done in a single query, such as identifying products with sales above average.

**3. Total Quantity Sold by Category**

****

**Explanation:** This query uses a subquery to retrieve the segment name and quantity for each sale. The main query then aggregates the quantity sold for each segment. The INNER JOIN ensures that each sale is matched with the correct segment in the product\_details table.

**4. Find the Top 10 Products with the Highest Discount**

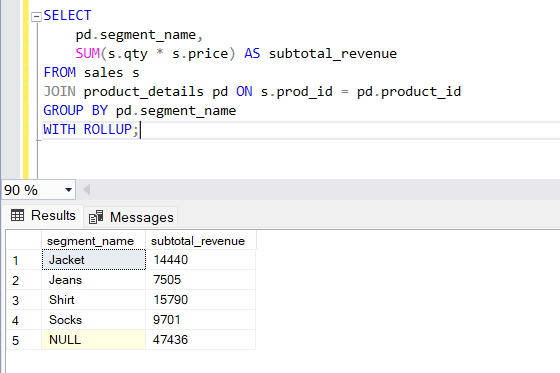
****

**Explanation:** This query retrieves the top 10 products with the highest discount values by joining the product\_details and sales tables on the product ID. It orders the results in descending order based on the discount and limits the output to the top 10 records using the TOP keyword.

**Subtotal (ROLLUP and GROUPING):**

Subtotaling aggregates data at different levels using ROLLUP or GROUPING, providing hierarchical summaries. This is useful for reports, like showing sales totals by category and segment, and gives a clear overview of grouped data at various levels of detail. Subtotals help in summarizing large datasets efficiently.

**5. Query to Calculate Subtotal of Revenue for Each Product Category**

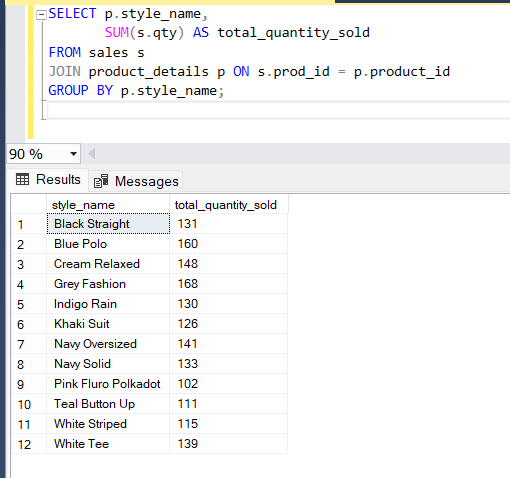
****

**Explanation:** This query calculates the revenue for each product segment and includes a grand total using ROLLUP. The SUM(s.qty \* s.price) computes the revenue, and GROUP BY pd.segment\_name groups the results by segment. The ROLLUP adds a final row for the grand total revenue across all segments.

**Group By Clause**

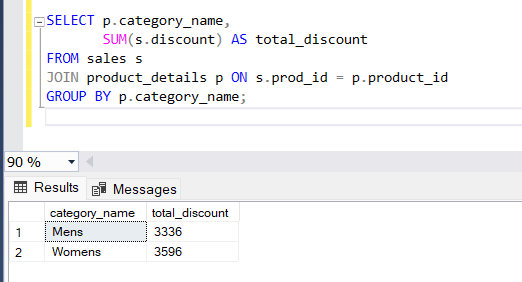
The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like calculating totals or averages. It helps in applying aggregate functions, such as COUNT(), SUM(), or AVG(), to grouped data.

**1. Total Quantity Sold by Product Style**



**Explanation:** Here, the query groups data by product style and calculates the total quantity sold for each style using the SUM function.

**2. Total Discount by Product Category**

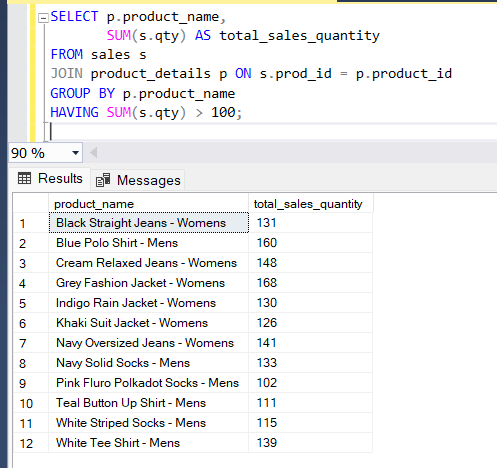


**Explanation:**  
The GROUP BY clause groups the data by product category, and the SUM function is used to calculate the total discount for each category.

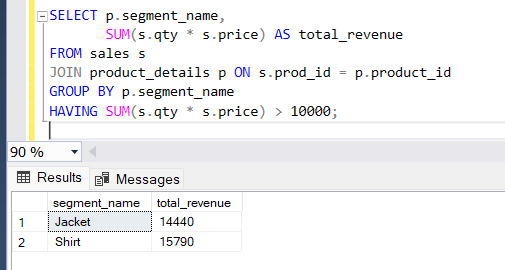
**Having Clause**

The HAVING clause is used to filter the results after the GROUP BY operation. It allows you to apply conditions to the grouped data, unlike WHERE, which filters rows before grouping. HAVING is typically used with aggregate functions to filter groups based on conditions like total sales exceeding a threshold.

**3. Products with Sales Above a Specific Threshold**



**4. Segments with More than a Specific Revenue**



**Explanation:** This query groups the data by product segment and uses the HAVING clause to filter only those segments where total revenue exceeds 10,000.