Name: Krushnakumar Patle

Email: krishnapatle128@gmail.com

Batch: Data Engineering Batch-1

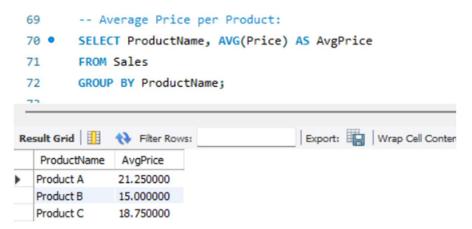
## 1. Total Aggregation Functions

#### **Total Sales Amount:**

The SUM() aggregate function returns the sum of all the numeric values from the column.

## **Average Price per Product:**

The AVG() aggregate function returns the mean average of the numeric values in the specified column.



# **Maximum Quantity Sold:**

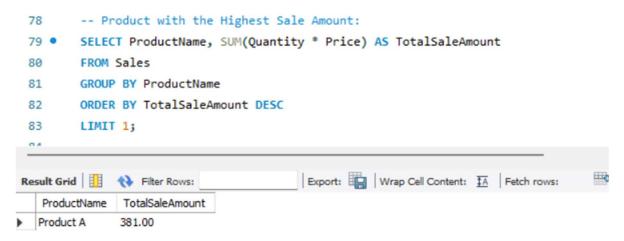
The MAX() aggregate function returns the largest value from the column.

```
74 -- Maximum Quantity Sold:
75 • SELECT MAX(Quantity) AS MaxQuantity
76 FROM Sales;

77

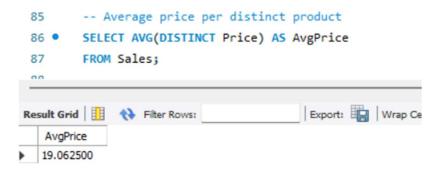
Result Grid  Filter Rows: Export: Wrap Cell Cor
```

## **Product with the Highest Sale Amount:**



## Average price per distinct product:

The DISTINCT() aggregate function returns the set of unique values in the column, discarding multiples.

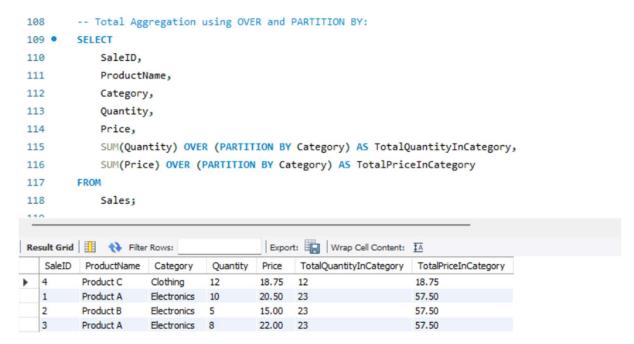


# **Count of distinct products:**

The COUNT() aggregate function returns the number of items in the column.

# 2. OVER and PARTITION BY Clause in SQL Queries& Total Aggregation using OVER and PARTITION BY in SQL Queries

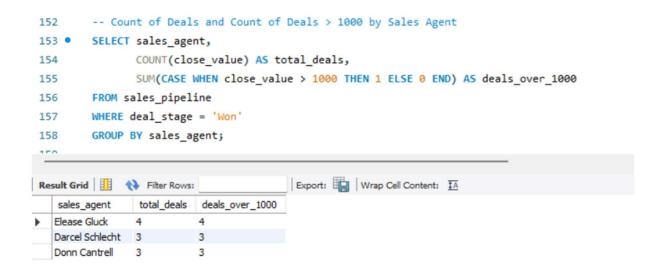
The OVER and PARTITION BY clauses in SQL are used in conjunction with window functions to perform calculations across a specific subset of rows within a result set. These clauses are commonly used for analytical functions that require operations on groups of rows.



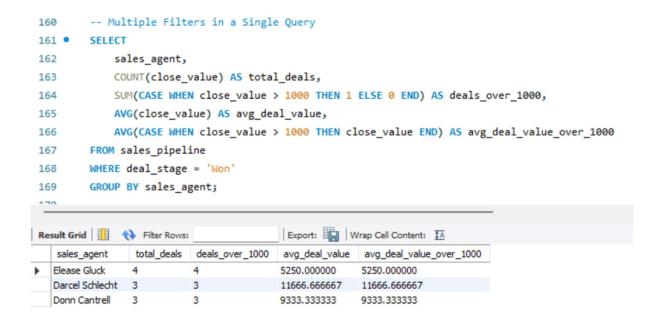
3. Rules and Restrictions to Group and Filter Data in SQL queries

```
-- Average Deal Value by Sales Agent
146
        SELECT sales_agent, AVG(close_value) AS avg_deal_value
        FROM sales_pipeline
147
        WHERE deal_stage = 'Won'
148
        GROUP BY sales_agent
149
        ORDER BY avg deal value DESC;
150
Export: Wrap Cell Content: IA
  sales_agent
              avg_deal_value
  Darcel Schlecht 11666.666667
  Donn Cantrell 9333.333333
  Elease Gluck
               5250.000000
```

## Count of Deals and Count of Deals > 1000 by Sales Agent



Multiple Filters in a Single Query



#### 4. What is SQL Order of Execution?

SQL order of execution refers to the sequence in which different clauses and operations within a SQL query are processed by the database management system.

Each SQL query consists of various components such as SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY clauses, along with functions and operators. Understanding the order in which these components are executed is vital for producing accurate and efficient query results.

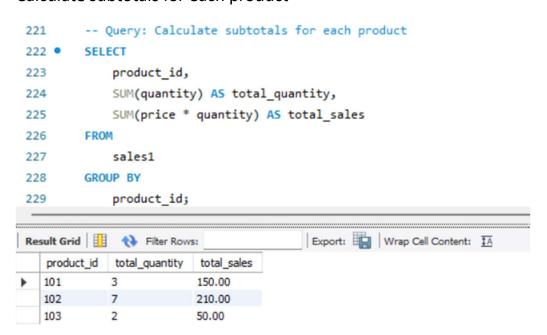
```
-- Query: Get the average salary for each department with more than two employees
189
190 •
       SELECT
191
          department,
         AVG(salary) AS avg_salary
193
       FROM
194
           employees
195
       WHERE
196
           salary > 50000
197
       GROUP BY
198
          department
       HAVING
199
200
          COUNT(employee_id) > 2
       ORDER BY
202
           avg_salary DESC;
Export: Wrap Cell Content: IA
  department avg_salary
```

#### 5. How to calculate Subtotals in SQL Queries:

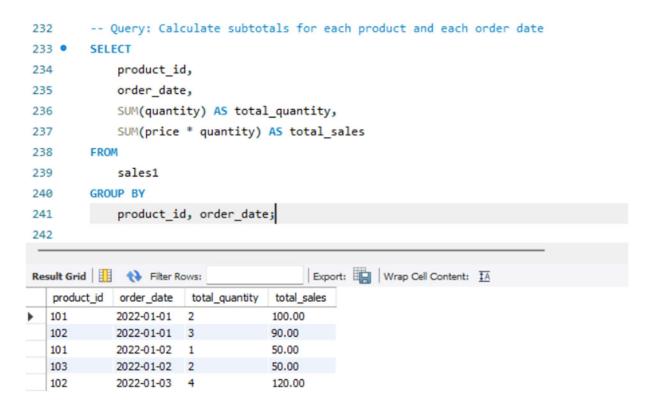
Calculating subtotals in SQL queries involves using the **GROUP BY** clause to group rows based on a specific column or set of columns. Subtotals are then computed for each group using aggregate functions such as **SUM**, **AVG**, **COUNT**, etc. Here's a basic example to illustrate how you can calculate subtotals:

```
205
        -- How to calculate Subtotals in SQL Queries
206 • ⊖ CREATE TABLE sales1 (
            order id INT PRIMARY KEY,
207
            product id INT,
208
            quantity INT,
209
            price DECIMAL(10, 2),
210
            order_date DATE
211
212
        );
213
214 •
        INSERT INTO sales1 VALUES
            (1, 101, 2, 50.00, '2022-01-01'),
215
            (2, 102, 3, 30.00, '2022-01-01'),
216
            (3, 101, 1, 50.00, '2022-01-02'),
217
            (4, 103, 2, 25.00, '2022-01-02'),
218
            (5, 102, 4, 30.00, '2022-01-03');
219
```

## Calculate subtotals for each product



Calculate subtotals for each product and each order date



The **UNION**, **EXCEPT**, and **INTERSECT** operators in SQL Server are used to combine or compare the results of two or more SELECT queries. Here are the key differences between these operators:

#### 1. UNION Operator:

- **Purpose:** Combines the result sets of two or more SELECT statements into a single result set.
- Syntax:

SELECT column1, column2, ... FROM table1 UNION SELECT column1, column2, ... FROM table2;

• **Result Set:** Includes all unique rows from the combined result sets. Duplicate rows are automatically eliminated.

#### 2. EXCEPT Operator (MINUS in some databases):

- Purpose: Returns the rows that are present in the result set of the first SELECT statement but not in the result set of the second SELECT statement.
- Syntax:

SELECT column1, column2, ... FROM table1 EXCEPT SELECT column1, column2, ... FROM table2;

• **Result Set:** Contains only the rows that exist in the first result set but not in the second result set.

#### 3. **INTERSECT Operator:**

- **Purpose:** Returns the common rows that are present in both the result sets of the two SELECT statements.
- Syntax:

SELECT column1, column2, ... FROM table1 INTERSECT SELECT column1, column2, ... FROM table2;

• **Result Set:** Contains only the rows that are common to both result sets.

#### 4. Number of Columns:

- **UNION:** The number and data types of columns must match in both SELECT statements.
- **EXCEPT and INTERSECT:** The number and data types of columns must match in both SELECT statements.

#### 5. **Sorting:**

- **UNION:** Automatically orders the result set and removes duplicates.
- **EXCEPT and INTERSECT:** Do not automatically order the result set, and you may need to use additional sorting if desired.

#### 6. Usage Considerations:

- **UNION:** Typically used when you want to combine the results of two queries with similar structures.
- **EXCEPT and INTERSECT:** Used when you want to compare or find differences between two result sets.