Exercise1: Ranking and Window Functions

Goal: Use ROW\_NUMBER(), RANK(), DENSE\_RANK(), OVER(), and PARTITION BY.

Scenario:

Find the top 3 most expensive products in each category using different ranking functions.

Steps:

1. Use ROW\_NUMBER() to assign a unique rank within each category.

2. Use RANK() and DENSE\_RANK() to compare how ties are handled.

3. Use PARTITION BY Category and ORDER BY Price DESC.

CREATE TABLE dbo.products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

category VARCHAR(50) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

stock\_quantity INT

);

-- Insert sample data with some price ties within categories

INSERT INTO dbo.products (product\_id, product\_name, category, price, stock\_quantity)

VALUES

(1, 'Ultra HD TV', 'Electronics', 1999.99, 15),

(2, 'Gaming Laptop', 'Electronics', 1899.99, 8),

(3, 'Premium Sound System', 'Electronics', 1999.99, 12),

(4, 'Wireless Earbuds', 'Electronics', 299.99, 30),

(5, 'Organic Cotton T-Shirt', 'Clothing', 49.99, 50),

(6, 'Designer Jeans', 'Clothing', 129.99, 25),

(7, 'Premium Leather Jacket', 'Clothing', 399.99, 10),

(8, 'Cashmere Sweater', 'Clothing', 199.99, 15),

(9, 'Luxury Mattress', 'Home', 2499.99, 5),

(10, 'Smart Thermostat', 'Home', 299.99, 20),

(11, 'Premium Blender', 'Home', 499.99, 12),

(12, 'Memory Foam Pillow', 'Home', 99.99, 40),

(13, 'Professional Camera', 'Electronics', 3499.99, 7),

(14, 'High-End DSLR Lens', 'Electronics', 1299.99, 4),

(15, 'Executive Office Chair', 'Furniture', 599.99, 9),

(16, 'Standing Desk', 'Furniture', 499.99, 6),

(17, 'Modern Sofa', 'Furniture', 1299.99, 3),

(18, 'Coffee Table', 'Furniture', 299.99, 12);

-- Query to compare all three ranking functions in one result

WITH ranked\_products AS (

SELECT

category,

product\_name,

price,

stock\_quantity,

ROW\_NUMBER() OVER (PARTITION BY category ORDER BY price DESC) AS row\_num,

RANK() OVER (PARTITION BY category ORDER BY price DESC) AS rank\_val,

DENSE\_RANK() OVER (PARTITION BY category ORDER BY price DESC) AS dense\_rank\_val

FROM

dbo.products

)

SELECT

category,

product\_name,

price,

stock\_quantity,

row\_num,

rank\_val,

dense\_rank\_val

FROM

ranked\_products

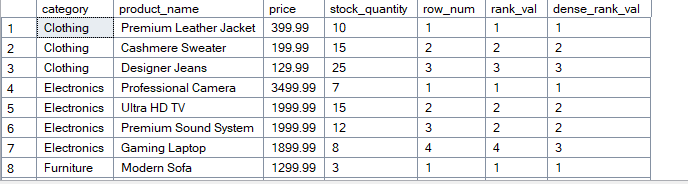
WHERE

row\_num <= 3 -- For ROW\_NUMBER(), ensures top 3 regardless of ties

OR rank\_val <= 3 -- For RANK(), may include more than 3 if ties exist

OR dense\_rank\_val <= 3; -- For DENSE\_RANK(), ensures top 3 ranking groups

-- Alternatively, show separate results for each ranking function



-- 1. Using ROW\_NUMBER()

SELECT \* FROM (

SELECT

category,

product\_name,

price,

stock\_quantity,

ROW\_NUMBER() OVER (PARTITION BY category ORDER BY price DESC) AS rank

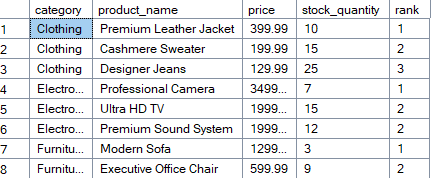
FROM

dbo.products

) AS rows

WHERE

rank <= 3;



-- 2. Using RANK()

SELECT \* FROM (

SELECT

category,

product\_name,

price,

stock\_quantity,

RANK() OVER (PARTITION BY category ORDER BY price DESC) AS rank

FROM

dbo.products

) AS ranks

WHERE

rank <= 3;

-- 3. Using DENSE\_RANK()

SELECT \* FROM (

SELECT

category,

product\_name,

price,

stock\_quantity,

DENSE\_RANK() OVER (PARTITION BY category ORDER BY price DESC) AS rank

FROM

dbo.products

) AS dense\_ranks

WHERE

rank <= 3;

-- Advanced example: Filter products with ranking and additional conditions

SELECT \* FROM (

SELECT

category,

product\_name,

price,

stock\_quantity,

AVG(price) OVER (PARTITION BY category) AS avg\_category\_price,

RANK() OVER (PARTITION BY category ORDER BY price DESC) AS rank,

DENSE\_RANK() OVER (ORDER BY price DESC) AS global\_rank,

price - LAG(price, 1, 0) OVER (PARTITION BY category ORDER BY price DESC) AS price\_diff\_to\_next

FROM

dbo.products

) AS advanced\_analysis

WHERE

rank <= 3

AND stock\_quantity > 5;

