Dataset: Sample - Superstore.csv

Context: Retail Analytics

Using **window functions** and **ANSI SQL,** I analyzed a retail dataset to guide business decisions regarding **discount policies, logistics,** and **regional priorities**.
The analysis includes ranking and performance metrics to identify high-impact categories and optimize operational efficiency.

Skills demonstrated: SQL window functions, ranking, analytical queries, business optimization
Language: SQL

Question 1 - Ranking and ordering by profit and sales

```
-- Question 1.1  -- In the code below, a ranking of the total profits per product was made, using the windowing function
 "Product Name", SUM(Profit) AS Total_Profit, RANK() OVER ( PARTITION BY Category ORDER BY SUM(Profit) DESC ) AS Profit_R
```

	Category object Office Su... 57.2% Technology 22.3% Furniture 20.5%	Product Name ob... Hon Deluxe... 0.1% Global Delu... 0.1% 1848 others 99.9%	Total_Profit float... -8879.9704 - 251...	Profit_Rank int64 1 - 1058
120	Furniture	Hon 4060 Series ...	96.2856	121
121	Furniture	Luxury Economy ...	94.9144	122
122	Furniture	Seth Thomas 14" ...	94.7682	123
123	Furniture	Executive Impres...	94.5152	124
124	Furniture	Executive Impres...	93.5883	125
125	Furniture	12-1/2 Diameter ...	91.1088	126
126	Furniture	DMI Eclipse Exec...	90.1764	127
127	Furniture	Executive Impres...	88.458	128
128	Furniture	Executive Impres...	87.9648	129
129	Furniture	DAX Two-Tone Sil...	84.8056	130

1,850 rows, 4 cols | 10 / page | Page 13 of 185

```
-- Question 1.2 -- The difference between the "RANK()" and the "DENSE_RANK" is that while the first, in a tie situation,
 "Product Name", SUM(Profit) AS Total_Profit, DENSE_RANK() OVER ( PARTITION BY Category ORDER BY SUM(Profit) DESC ) AS Pr
```

	Category object Office Su... 57.2% Technology 22.3% Furniture 20.5%	Product Name ob... Hon Deluxe... 0.1% Global Delu... 0.1% 1848 others 99.9%	Total_Profit float... -8879.9704 - 251...	Profit_Dense_Rank 1 - 1052
0	Furniture	Hon Deluxe Fabri...	1927.442	1
1	Furniture	Global Deluxe Hig...	1558.591	2
2	Furniture	Hon Pagoda Stac...	1540.704	3
3	Furniture	Hon 4070 Series ...	1388.6348	4
4	Furniture	Office Star - Prof...	1305.6456	5
5	Furniture	Hon Olson Stacke...	1292.1264	6
6	Furniture	GE 48" Fluoresce...	1260.221	7
7	Furniture	Hon GuestStacke...	1233.0848	8
8	Furniture	SAFCO Arc Foldin...	1179.374	9
9	Furniture	Hon 4070 Series ...	1052.8144	10

1,850 rows, 4 cols | 10 / page | Page 1 of 185

```
-- Question 1.3 -- In the code below, we use the "ROW_NUMBER" in windows defined by the "Segment" column where the orderi
 "Customer Name", SUM(Quantity) AS Total_Quantity, ROW_NUMBER() OVER( PARTITION BY Segment ORDER BY SUM(Quantity) DESC )
```

	Segment object	Customer Name o..	Total_Quantity fl...	Quantity_RowNum
	Consumer 51.6% Corporate 29.8% Home Offi... 18.7%	William Bro... 0.1% John Lee 0.1% 791 others 99.7%	2.0 - 150.0	1 - 409
0	Consumer	William Brown	146	1
1	Consumer	John Lee	143	2
2	Consumer	Steven Cartwright	133	3
3	Consumer	Emily Phan	124	4
4	Consumer	Cassandra Brand...	122	5
5	Consumer	Chloris Kastensm...	122	6
6	Consumer	Lena Cacioppo	113	7
7	Consumer	Ken Lonsdale	113	8
8	Consumer	Seth Vernon	109	9
9	Consumer	Clytie Kelty	108	10

793 rows, 4 cols 10 / page Page 1 of 80

Question 2 - Variance analysis with LAG and LEAD

```
-- Question 2.1 -- In the code below, at first the profit of all products of the same order are added "SUM(Profit)"  -- a
 "Order ID",
 "Order Date", SUM(Profit) AS Order_Profit, LAG(SUM(Profit)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "
 ) AS Previous_Order_Profit, SUM(Profit) - LAG(SUM(Profit)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "O
 ) AS Profit_Diff FROM 'Sample - Superstore.csv' GROUP BY "Customer Name", "Order ID", "Order Date" ORDER BY "Customer Na
```

	Customer Name o..	Order ID object	Order Date dateti...	Order_Profit float...	Previous_Order_...	Profit_Diff float64
	Emily Phan 0.3% Chloris Kas... 0.3% 791 others 99.4%	CA-2014-1... 0% CA-2014-1... 0% 5007 others 100%	2014-01-03 00:0...	-6892.3748 - 876...	-6892.3748 - 876...	-8759.8299 - 872...
0	Aaron Bergman	CA-2014-152905	2014-02-18 00:0...	-2.5248	Nan	Nan
1	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	15.0033	-2.5248	17.5281
2	Aaron Bergman	CA-2016-140935	2016-11-10 00:00...	116.868	15.0033	101.8647
3	Aaron Hawkins	CA-2014-122070	2014-04-22 00:0...	124.7869	Nan	Nan
4	Aaron Hawkins	CA-2014-113768	2014-05-13 00:0...	24.7992	124.7869	-99.9877
5	Aaron Hawkins	US-2014-158400	2014-10-25 00:0...	18.528	24.7992	-6.2712
6	Aaron Hawkins	CA-2014-157644	2014-12-31 00:0...	20.1681	18.528	1.6401
7	Aaron Hawkins	CA-2015-130113	2015-12-27 00:0...	136.557	20.1681	116.3889
8	Aaron Hawkins	CA-2016-162747	2016-03-20 00:0...	38.038	136.557	-98.519
9	Aaron Hawkins	CA-2017-164000	2017-12-18 00:00...	2.338	38.038	-35.7

5,009 rows, 6 cols 10 / page Page 1 of 501

```
-- Question 2.2 -- In the code below we use the "LEAD()" function to calculate the difference in the quantity of products
 "Order ID",
 "Order Date", SUM(Quantity) AS Order_Quantity, LEAD(SUM(Quantity)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order D
 ) AS Next_Order_Quantity, LEAD(SUM(Quantity)) OVER( PARTITION BY "Customer Name" ORDER BY "Order Date", "Order ID"
 ) - SUM(Quantity) AS Quantity_Diff FROM 'Sample - Superstore.csv' GROUP BY "Customer Name", "Order ID", "Order Date" ORD
```

	Customer Name o..	Order ID object	Order Date dateti...	Order_Quantity fl...	Next_Order_Qua...	Quantity_Diff floa...
	Emily Phan 0.3% Chloris Kas... 0.3% 791 others 99.4%	CA-2014-1... 0% CA-2014-1... 0% 5007 others 100%	2014-01-03 00:0...	1.0 - 52.0	1.0 - 52.0	-44.0 - 50.0
0	Aaron Bergman	CA-2014-152905	2014-02-18 00:0...	2	7	5
1	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	7	4	-3
2	Aaron Bergman	CA-2016-140935	2016-11-10 00:00...	4	Nan	Nan
3	Aaron Hawkins	CA-2014-122070	2014-04-22 00:0...	11	8	-3
4	Aaron Hawkins	CA-2014-113768	2014-05-13 00:0...	8	4	-4
5	Aaron Hawkins	US-2014-158400	2014-10-25 00:0...	4	6	2
6	Aaron Hawkins	CA-2014-157644	2014-12-31 00:0...	6	11	5
7	Aaron Hawkins	CA-2015-130113	2015-12-27 00:0...	11	7	-4
8	Aaron Hawkins	CA-2016-162747	2016-03-20 00:0...	7	7	0
9	Aaron Hawkins	CA-2017-164000	2017-12-18 00:00...	7	Nan	Nan

5,009 rows, 6 cols 10 / page Page 1 of 501

```
-- Question 2.3 -- In the code below we use the LAG() and LEAD() functions in the "Sales" column to indicate the total sa
 "Order ID",
 "Order Date", SUM(Sales) AS Order_Sales, LAG(SUM(Sales)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "Ord
 ) AS Previous_Sales, LEAD(SUM(Sales)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "Order ID"
 ) AS Next_Sales FROM 'Sample - Superstore.csv' GROUP BY "Customer Name", "Order ID", "Order Date"
) SELECT * FROM CustomerSales WHERE (Previous_Sales IS NOT NULL AND ABS(Order_Sales - Previous_Sales) > 500) OR (Next_Sal
```

	Customer Name o..	Order ID object	Order Date dateti...	Order_Sales float...	Previous_Sales fl...	Next_Sales float64
	Anna H Ber... 0.4% Laura Arm... 0.4% 583 others 99.1%	CA-2014-1... 0% CA-2015-1... 0% 2271 others 99.9%	2014-01-06 00:0...	0.556 - 23661.228	0.556 - 23661.228	0.556 - 18336.73...
0	Aaron Hawkins	CA-2014-157644	2014-12-31 00:0...	53.67	49.408	991.26
1	Aaron Hawkins	CA-2015-130113	2015-12-27 00:0...	991.26	53.67	86.45
2	Aaron Hawkins	CA-2016-162747	2016-03-20 00:0...	86.45	991.26	18.704
3	Aaron Smayling	CA-2017-113481	2017-01-02 00:0...	740.214	477.666	1476.27
4	Aaron Smayling	CA-2017-162691	2017-08-01 00:0...	1476.27	740.214	88.074
5	Aaron Smayling	US-2017-147655	2017-09-04 00:0...	88.074	1476.27	171.288
6	Adam Bellavance	CA-2016-161207	2016-08-29 00:0...	27.93	334.2	4438.686
7	Adam Bellavance	CA-2016-129714	2016-09-01 00:0...	4438.686	27.93	79.99
8	Adam Bellavance	CA-2017-159688	2017-05-07 00:0...	79.99	4438.686	20.736
9	Adam Bellavance	CA-2017-118213	2017-11-05 00:00...	240.15	20.736	2595.388

2,273 rows, 6 cols 10 / page Page 1 of 228

Question 3 - Statistics with SUM, COUNT and AVG in windows

```
-- Question 3.1 -- In the code below we use the frame "ROWS BETWEEN 2 PRECEDING AND CURRENT ROW" to select the total prof
 "Order ID",
 "Order Date", SUM(Profit) AS Order_Profit, AVG(SUM(Profit)) OVER ( PARTITION BY State ORDER BY "Order Date", "Order ID"
```

	State object	Order ID object	Order Date dateti...	Order_Profit float...	MOVING_Avg_Pr...
	California 20.4% New York 11.2% 47 others 68.4%	CA-2014-1... 0% US-2014-11... 0% 5007 others 100%	2014-01-03 00:0...	-6892.3748 - 876...	-2305.87763333...
0	Alabama	CA-2014-124023	2014-04-07 00:0...	2.7776	2.7776
1	Alabama	US-2014-118997	2014-04-08 00:0...	316.1392	159.4584
2	Alabama	CA-2014-143840	2014-05-22 00:0...	46.581	121.8326
3	Alabama	CA-2014-110408	2014-10-18 00:0...	444.6902	269.1368
4	Alabama	CA-2014-163013	2014-11-28 00:0...	3.9609	165.0773667
5	Alabama	CA-2014-120768	2014-12-19 00:0...	240.6027	229.7512667
6	Alabama	CA-2014-153087	2014-12-27 00:0...	195.9808	146.8481333
7	Alabama	CA-2015-134257	2015-03-16 00:0...	244.2543	226.9459333
8	Alabama	CA-2015-121552	2015-03-22 00:0...	5.4768	148.5706333
9	Alabama	CA-2015-104941	2015-06-13 00:0...	346.2843	198.6718

5,009 rows, 5 cols 10 / page Page 1 of 501

```
-- Question 3.2 -- In the code below we use the frame "ROWS BETWEEN 4 PRECEDING AND 1 PRECEDING" to add the total sales o
 "Order ID",
 "Order Date", SUM(Sales) AS Order_Sales, SUM(SUM(Sales)) OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "Ord
 ) AS Sales_Last4Orders FROM 'Sample - Superstore.csv' GROUP BY "Customer Name", "Order ID", "Order Date" ORDER BY "Custo
```

	Customer Name o..	Order ID object	Order Date dateti...	Order_Sales float...	Sales_Last4Orders
	Emily Phan 0.3% Chloris Kas... 0.3% 791 others 99.4%	CA-2014-1... 0% CA-2014-1... 0% 5007 others 100%	2014-01-03 00:0...	0.556 - 23661.228	0.852 - 25035.082
0	Aaron Bergman	CA-2014-152905	2014-02-18 00:0...	12.624	Nan
1	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	309.592	12.624
2	Aaron Bergman	CA-2016-140935	2016-11-10 00:00...	563.94	322.216
3	Aaron Hawkins	CA-2014-122070	2014-04-22 00:0...	257.752	Nan
4	Aaron Hawkins	CA-2014-113768	2014-05-13 00:0...	287.456	257.752
5	Aaron Hawkins	US-2014-158400	2014-10-25 00:0...	49.408	545.208
6	Aaron Hawkins	CA-2014-157644	2014-12-31 00:0...	53.67	594.616
7	Aaron Hawkins	CA-2015-130113	2015-12-27 00:0...	991.26	648.286
8	Aaron Hawkins	CA-2016-162747	2016-03-20 00:0...	86.45	1381.794
9	Aaron Hawkins	CA-2017-164000	2017-12-18 00:00...	18.704	1180.788

5,009 rows, 5 cols 10 / page Page 1 of 501

```
-- Question 3.3 -- In the code below, a CTE was created with the objective of grouping all products of the same subcatego
 "Order Date",
 "Sub-Category" FROM 'Sample - Superstore.csv' GROUP BY "Order ID", "Order Date", "Sub-Category"
) SELECT "Sub-Category",
 "Order ID",
 "Order Date", COUNT("Order ID") OVER ( PARTITION BY "Sub-Category" ORDER BY "Order Date", "Order ID" ROWS BETWEEN UNBOUN
```

	Sub-Category ob...	Order ID object	Order Date dateti...	Cumulative_Orders
	Binders 14.4% Paper 13% 15 others 72.6%	CA-2017-1... 0.1% CA-2015-1... 0.1% 5007 others 99.8%	2014-01-03 00:0...	1 - 1316
0	Accessories	CA-2014-135405	2014-01-09 00:0...	1
1	Accessories	CA-2014-162775	2014-01-13 00:0...	2
2	Accessories	CA-2014-103366	2014-01-15 00:0...	3
3	Accessories	CA-2014-140795	2014-02-01 00:0...	4
4	Accessories	CA-2014-107755	2014-02-07 00:0...	5
5	Accessories	CA-2014-127614	2014-02-11 00:0...	6
6	Accessories	CA-2014-121762	2014-02-14 00:0...	7
7	Accessories	CA-2014-109491	2014-02-20 00:0...	8
8	Accessories	CA-2014-111157	2014-03-02 00:0...	9
9	Accessories	CA-2014-169061	2014-03-05 00:0...	10

9,159 rows, 4 cols 10 / page Page 1 of 916

Question 4 - Reuse of windows with WINDOW clause

```
-- Question 4.1 -- In the code below, the functions "RANK()" and "ROW_NUMBER" were applied to the sales of each product,
Category,
"Product Name",
Sales
Discount, RANK() OVER w_region_produto AS Sales_Rank, ROW_NUMBER() OVER w_region_produto AS Sales_RowNum, -- Question 4.
```

	Region object West 32% East 28.5% 2 others 39.5%	Category object Office Su... 60.3% Furniture 21.2% Technology 18.5%	Product Name ob... Staple env... 0.5% Easy-stapl... 0.5% 1848 others 99.1%	Sales float64 0.444 - 22638.48	Discount float64 0.0 - 0.8	Sales_Rank int64 1 - 1896	Sales_RowNum i... 1 - 1897	A 0
0	Central	Furniture	HON 5400 Series...	3504.9	0	1	1	
1	Central	Furniture	Office Star - Prof...	2807.84	0	2	2	
2	Central	Furniture	Balt Solid Wood R...	2678.94	0	3	3	
3	Central	Furniture	Hon Pagoda Stac...	2567.84	0	4	4	
4	Central	Furniture	HON 5400 Series...	2453.43	0.3	5	5	
5	Central	Furniture	Riverside Palais R...	2396.2656	0.32	6	6	
6	Central	Furniture	Global Deluxe Hig...	2001.86	0	7	7	
7	Central	Furniture	Hon Deluxe Fabri...	1951.84	0	8	8	
8	Central	Furniture	Hon 4070 Series ...	1925.88	0	9	9	
9	Central	Furniture	Global Leather an...	1805.88	0	10	10	

9,994 rows, 8 cols 10 / page Page 1 of 1,000

Question 5 - Temporal patterns and segmentations

```
-- Question 5.1 -- In the code below, the DATE_TRUNC() was applied to the "Order Date" column in order to call only the f
DATE_TRUNC('month', "Order Date") AS Month, SUM(Sales) AS Monthly_Sales, RANK() OVER ( PARTITION BY DATE_TRUNC('month',
DATE_TRUNC('month', "Order Date") ORDER BY Month,
Sales_Rank;
```

	Region object South 25% West 25% 2 others 50%	Month datetime6... 2014-01-01 00:0...	Monthly_Sales fl... 199.776 - 45633....	Sales_Rank int64 1 - 4
0	South	2014-01-01 00:0...	9322.092	1
1	West	2014-01-01 00:0...	2938.723	2
2	Central	2014-01-01 00:0...	1539.906	3
3	East	2014-01-01 00:0...	436.174	4
4	South	2014-02-01 00:0...	2028.986	1
5	Central	2014-02-01 00:0...	1233.174	2
6	West	2014-02-01 00:0...	1057.956	3
7	East	2014-02-01 00:0...	199.776	4
8	South	2014-03-01 00:0...	32911.121	1
9	West	2014-03-01 00:0...	11008.898	2

192 rows, 4 cols 10 / page Page 1 of 20

```
-- Question 5.2 -- Through the "COUNT()" function and the "ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW" frame it was
 "Order ID",
 "Order Date", COUNT("Order ID") OVER ( PARTITION BY "Customer Name" ORDER BY "Order Date", "Order ID" ROWS BETWEEN UNBOU
```

	Customer Name o..	Order ID object	Order Date dateti...	Orders_Cumulative
	William Bro... 0.4% John Lee 0.3% 791 others 99.3%	CA-2017-1... 0.1% CA-2017-1... 0.1% 5007 others 99.7%	2014-01-03 00:0...	1 - 37
0	Aaron Bergman	CA-2014-152905	2014-02-18 00:0...	1
1	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	3
2	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	2
3	Aaron Bergman	CA-2014-156587	2014-03-07 00:0...	4
4	Aaron Bergman	CA-2016-140935	2016-11-10 00:00...	6
5	Aaron Bergman	CA-2016-140935	2016-11-10 00:00...	5
6	Aaron Hawkins	CA-2014-122070	2014-04-22 00:0...	2
7	Aaron Hawkins	CA-2014-122070	2014-04-22 00:0...	1
8	Aaron Hawkins	CA-2014-113768	2014-05-13 00:0...	3
9	Aaron Hawkins	CA-2014-113768	2014-05-13 00:0...	4

9,994 rows, 4 cols 10 / page Page 1 of 1,000

```
-- Question 5.3 -- In the code below two CTEs were created; in the first "monthly_profit" the sum of the profit per month
 DATE_TRUNC('month', "Order Date") AS Month, SUM(Profit) AS Monthly_Profit FROM 'Sample - Superstore.csv' GROUP BY "Custo
),
profit_diff AS ( SELECT "Customer Name", Month,
 Monthly_Profit, LAG(Monthly_Profit) OVER ( PARTITION BY "Customer Name" ORDER BY Month ) AS Prev_Month_Profit,
 Monthly_Profit - LAG(Monthly_Profit) OVER ( PARTITION BY "Customer Name" ORDER BY Month ) AS Profit_Growth FROM monthly_
) SELECT "Customer Name", Month,
 Monthly_Profit,
 Prev_Month_Profit,
 Profit_Growth FROM profit_diff WHERE Profit_Growth IS NOT NULL ORDER BY Profit_Growth DESC LIMIT 5;
```

	Customer Name o..	Month datetime6...	Monthly_Profit fl...	Prev_Month_Profit f	Profit_Growth flo...
0	Tamara Chand	2016-10-01 00:0...	8762.3891	37.7204	8724.6687
1	Cindy Stewart	2017-11-01 00:00...	43.706	-6886.5095	6930.2155
2	Raymond Buch	2017-03-01 00:0...	6734.472	72.6159	6661.8561
3	Sanjit Chand	2014-09-01 00:0...	5511.8641	2.3328	5509.5313
4	Adrian Barton	2016-12-01 00:0...	4946.37	-204.4458	5150.8158

5 rows, 5 cols 10 / page Page 1 of 1