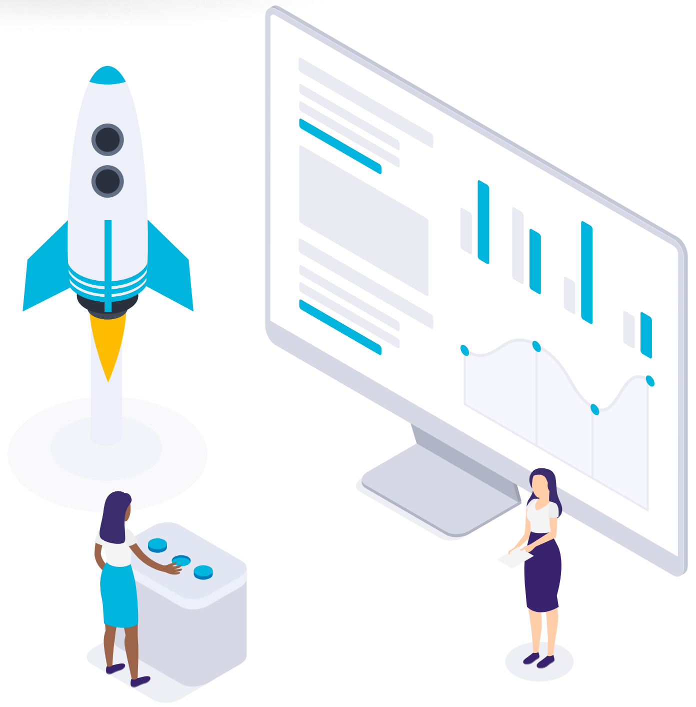
ondia

Relational DB & SQL

Sessions 10 & 11

Window Functions

Analytic Numbering Functions



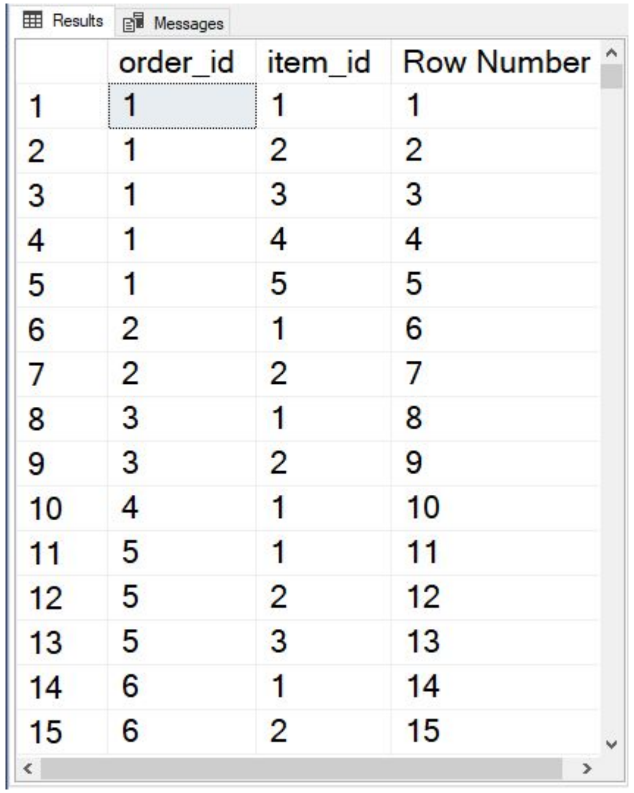
Row_Number()
Rank()
Dense_Rank() Functions

Row_Number()	Creates a column that contain assigned row numbers in order starting from 1 based on partitioning and sorting preferences.
Rank()	Unlike Row_Number; Assigns the same row number to the same values in the sorted column, based on partitioning and sorting preferences.
Dense_Rank()	Assigns not row number but ordinal number based on partitioning and sorting preferences. Assigns the same ordinal number to the same values in the sorted column

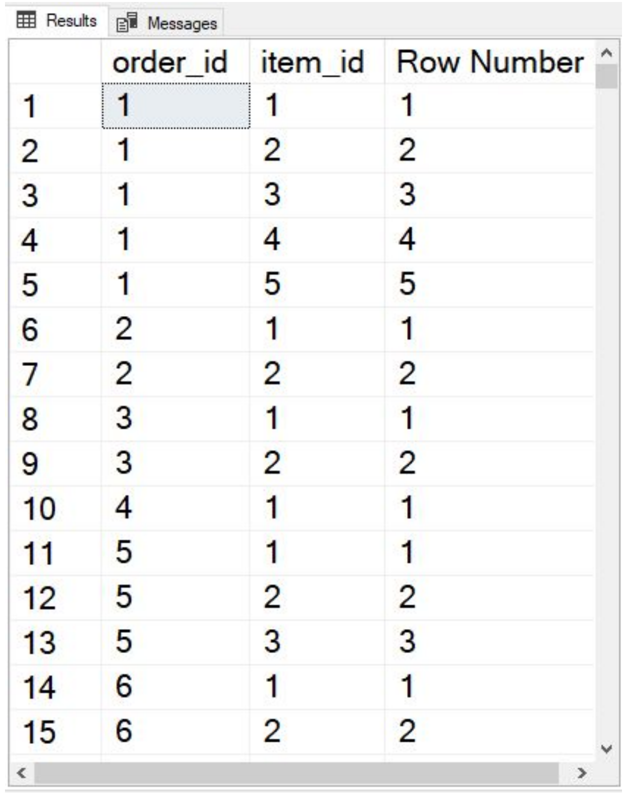
Row_Number()

```
SELECT   order_id, item_id,
         ROW_NUMBER()OVER(ORDER BY order_id) as [Row Number]
FROM     SALE.order_item
```

```
SELECT   order_id, item_id,
         ROW_NUMBER()OVER(Partition by Order_id ORDER BY order_id) as [Row Number]
FROM     SALE.order_item
```


Results Messages

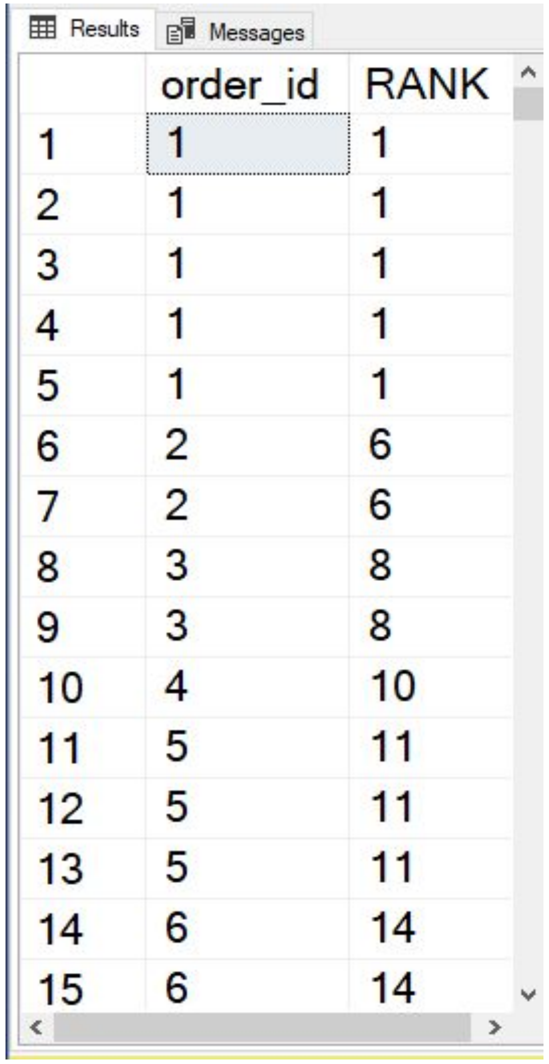
	order_id	item_id	Row Number
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	1	5	5
6	2	1	6
7	2	2	7
8	3	1	8
9	3	2	9
10	4	1	10
11	5	1	11
12	5	2	12
13	5	3	13
14	6	1	14
15	6	2	15


Results Messages

	order_id	item_id	Row Number
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	1	5	5
6	2	1	1
7	2	2	2
8	3	1	1
9	3	2	2
10	4	1	1
11	5	1	1
12	5	2	2
13	5	3	3
14	6	1	1
15	6	2	2

Rank()

```
SELECT   order_id,
         RANK() OVER (ORDER BY order_id) AS [RANK]
FROM     sale.order_item
```

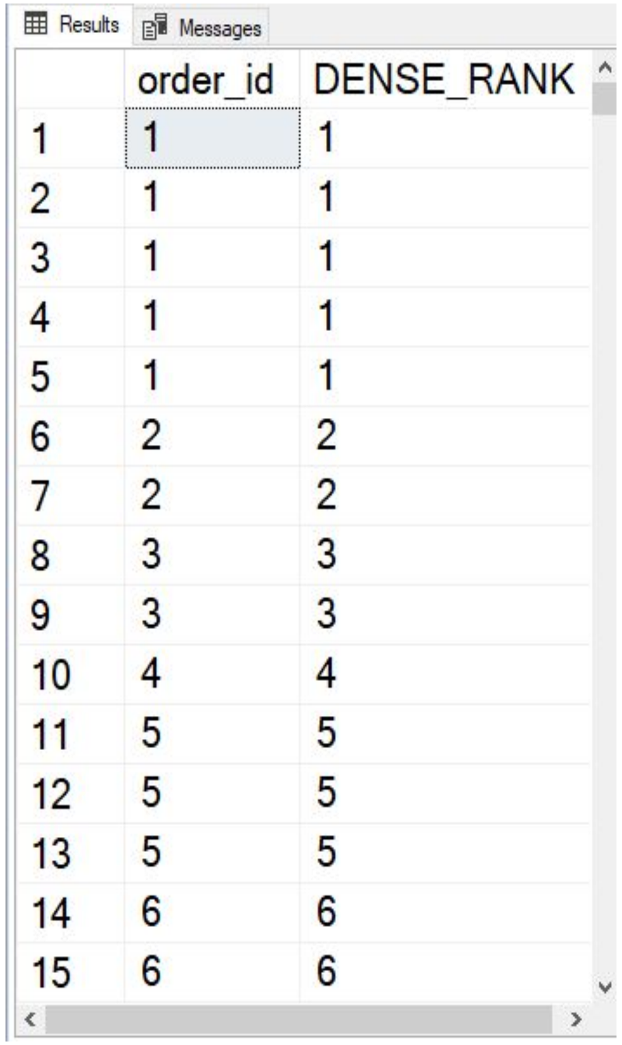


Results Messages

	order_id	RANK
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	2	6
7	2	6
8	3	8
9	3	8
10	4	10
11	5	11
12	5	11
13	5	11
14	6	14
15	6	14

Dense_Rank()

```sql
SELECT  order_id,
        DENSE_RANK() OVER (ORDER BY order_id) AS [DENSE_RANK]
FROM    sale.order_item
```

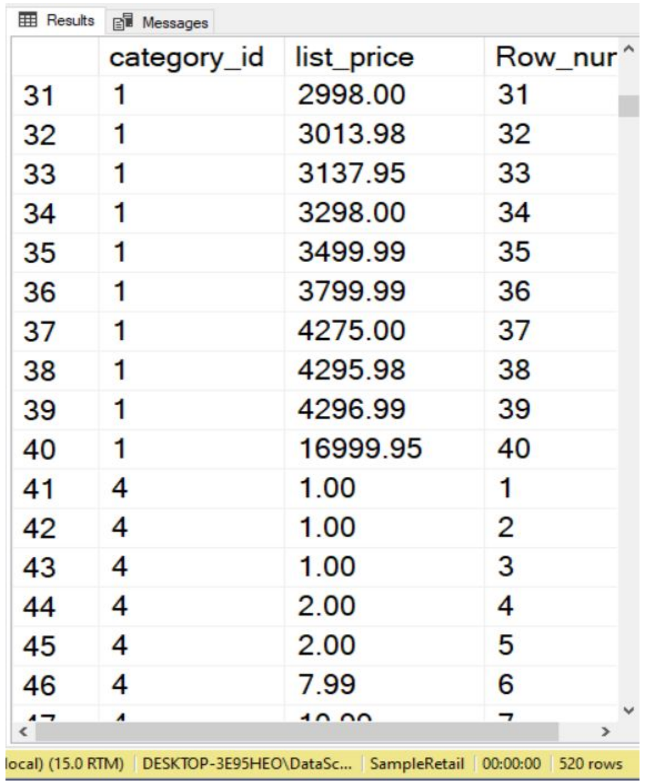


	order_id	DENSE_RANK
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	2	2
7	2	2
8	3	3
9	3	3
10	4	4
11	5	5
12	5	5
13	5	5
14	6	6
15	6	6

Query Time For You

Assign an ordinal number to the product prices for each category in ascending order

Expected Output:



Results Messages

	category_id	list_price	Row_nur
31	1	2998.00	31
32	1	3013.98	32
33	1	3137.95	33
34	1	3298.00	34
35	1	3499.99	35
36	1	3799.99	36
37	1	4275.00	37
38	1	4295.98	38
39	1	4296.99	39
40	1	16999.95	40
41	4	1.00	1
42	4	1.00	2
43	4	1.00	3
44	4	2.00	4
45	4	2.00	5
46	4	7.99	6

```
local) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows
```

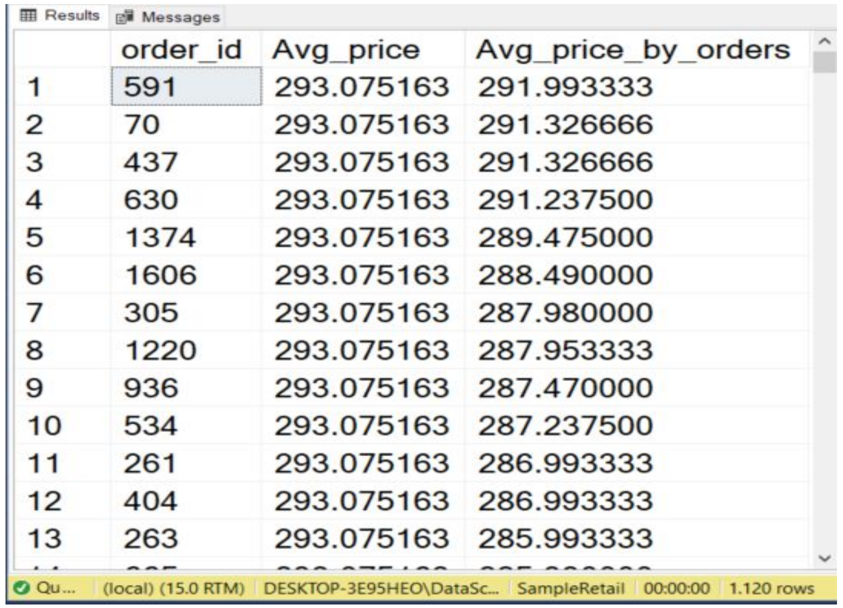
Query Time

Lets try previous query again using RANK() and DENSE_RANK() functions and discuss differences among all of them..

Query Time For You

Which orders' average product price is lower than the overall average price?

Expected Output:



Results Messages

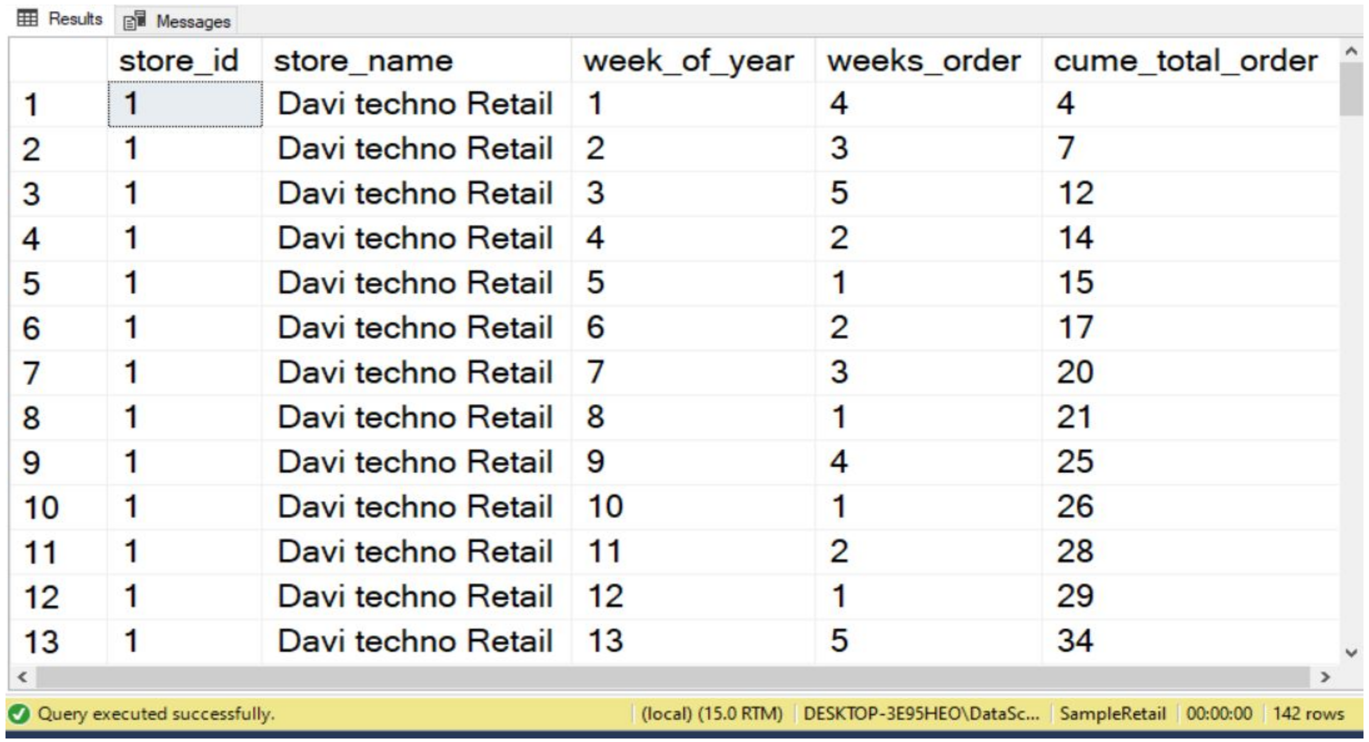
	order_id	Avg_price	Avg_price_by_orders
1	591	293.075163	291.993333
2	70	293.075163	291.326666
3	437	293.075163	291.326666
4	630	293.075163	291.237500
5	1374	293.075163	289.475000
6	1606	293.075163	288.490000
7	305	293.075163	287.980000
8	1220	293.075163	287.953333
9	936	293.075163	287.470000
10	534	293.075163	287.237500
11	261	293.075163	286.993333
12	404	293.075163	286.993333
13	263	293.075163	285.993333

```
Qu... | (local) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 1.120 rows
```

Query Time For You

Question: Calculate the stores' weekly cumulative count of orders for 2018

Expected Output:



Results | Messages

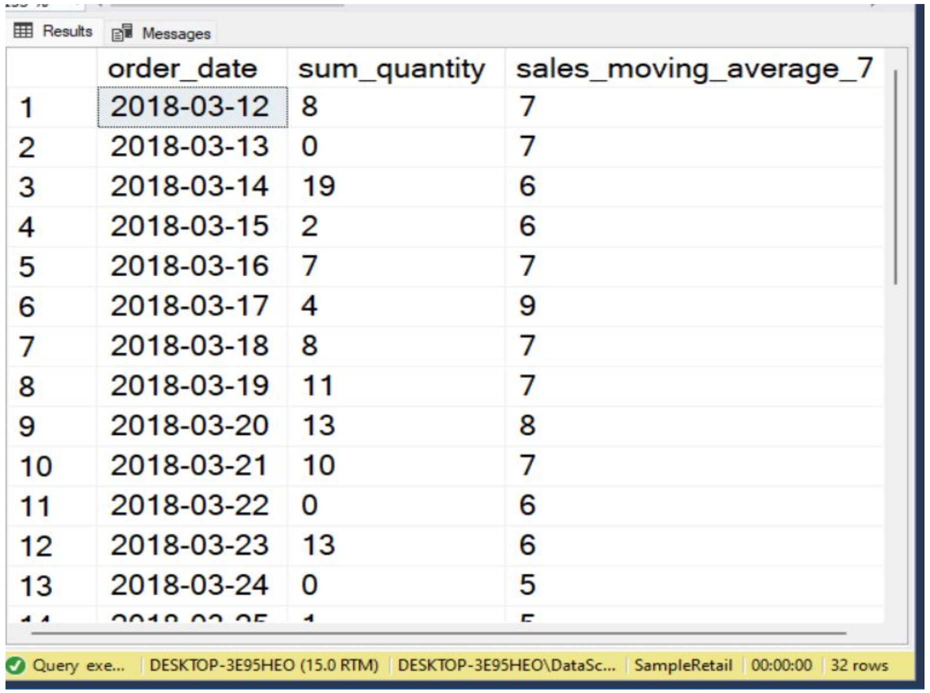
	store_id	store_name	week_of_year	weeks_order	cume_total_order
1	1	Davi techno Retail	1	4	4
2	1	Davi techno Retail	2	3	7
3	1	Davi techno Retail	3	5	12
4	1	Davi techno Retail	4	2	14
5	1	Davi techno Retail	5	1	15
6	1	Davi techno Retail	6	2	17
7	1	Davi techno Retail	7	3	20
8	1	Davi techno Retail	8	1	21
9	1	Davi techno Retail	9	4	25
10	1	Davi techno Retail	10	1	26
11	1	Davi techno Retail	11	2	28
12	1	Davi techno Retail	12	1	29
13	1	Davi techno Retail	13	5	34

```
Query executed successfully.   (local) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 142 rows
```

Query Time

Question: Calculate 7-day moving average of the number of products sold between '2018-03-12' and '2018-04-12'.

Expected Output:



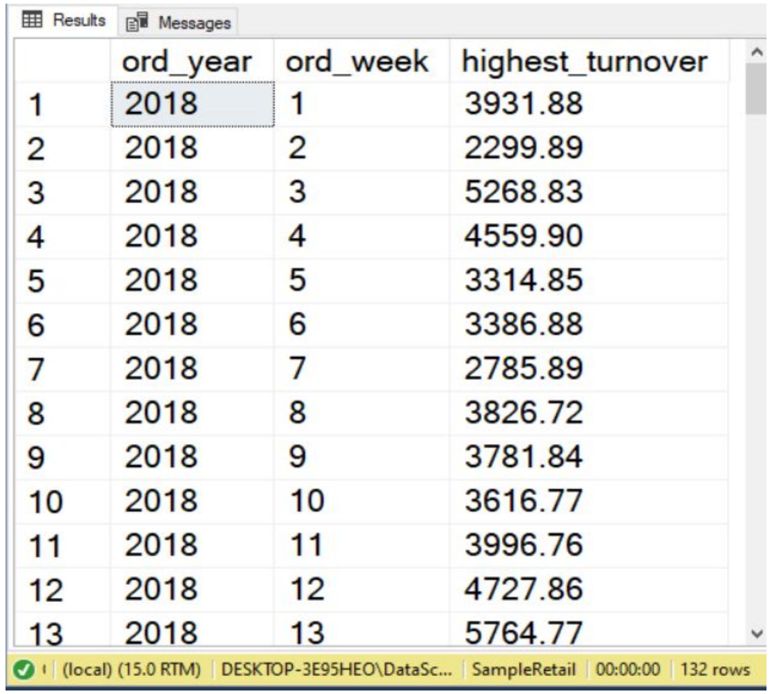
	order_date	sum_quantity	sales_moving_average_7
1	2018-03-12	8	7
2	2018-03-13	0	7
3	2018-03-14	19	6
4	2018-03-15	2	6
5	2018-03-16	7	7
6	2018-03-17	4	9
7	2018-03-18	8	7
8	2018-03-19	11	7
9	2018-03-20	13	8
10	2018-03-21	10	7
11	2018-03-22	0	6
12	2018-03-23	13	6
13	2018-03-24	0	5

Query exe... | DESKTOP-3E95HEO (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 32 rows

Query Time

Question: Write a query that returns the highest daily turnover amount for each week on a yearly basis.

Expected Output:



Results | Messages

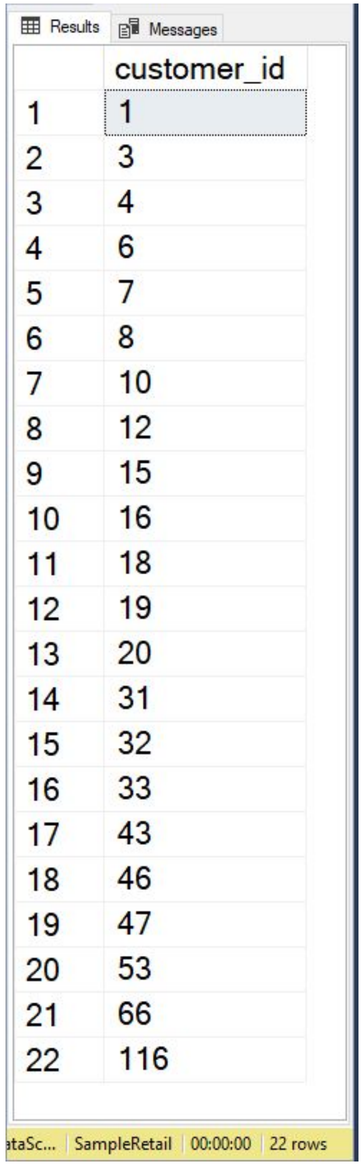
	ord_year	ord_week	highest_turnover
1	2018	1	3931.88
2	2018	2	2299.89
3	2018	3	5268.83
4	2018	4	4559.90
5	2018	5	3314.85
6	2018	6	3386.88
7	2018	7	2785.89
8	2018	8	3826.72
9	2018	9	3781.84
10	2018	10	3616.77
11	2018	11	3996.76
12	2018	12	4727.86
13	2018	13	5764.77

(local) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 132 rows

Query Time

Question: List customers whose have at least 2 consecutive orders are not shipped.

Expected Output:



Results | Messages

	customer_id
1	1
2	3
3	4
4	6
5	7
6	8
7	10
8	12
9	15
10	16
11	18
12	19
13	20
14	31
15	32
16	33
17	43
18	46
19	47
20	53
21	66
22	116

ataSc... | SampleRetail | 00:00:00 | 22 rows

Cume_Dist()
Percent_Rank()
Ntile(N) Functions

Cume_Dist()

Creates a column that contain cumulative distribution of the sorted column values. **CUME_DIST = ROW NUMBER / TOTAL ROWS**

Percent_Rank()

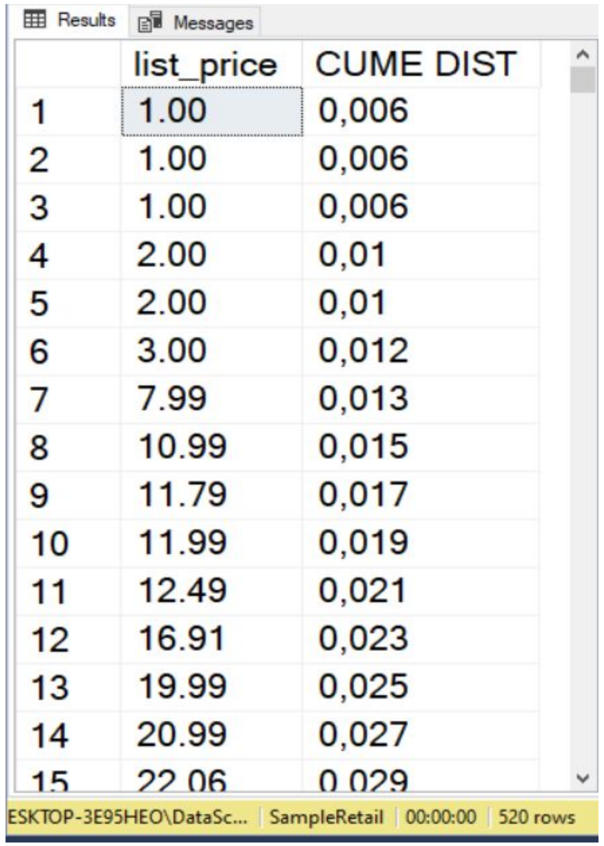
Creates a column that contain relative standing of a value in the sorted column values. **PERCENT_RANK = (ROW NUMBER -1) / (TOTAL ROWS -1)**

Ntile(N)

Divides the sorted column into equal groups according to the given parameter (N) value and returns which group the each values are in.

Cume_Dist()

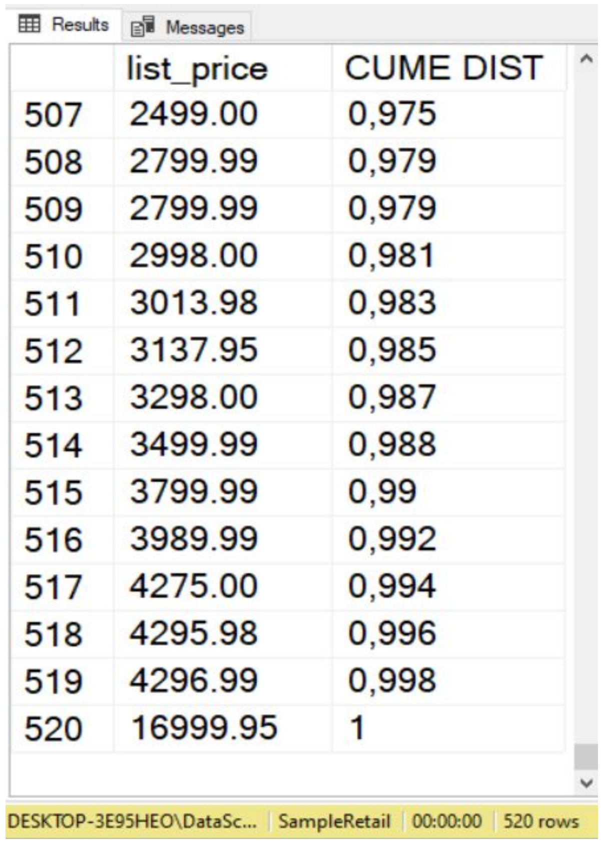
```
SELECT  list_price,
        ROUND(CUME_DIST() OVER (ORDER BY list_price) , 3) AS [CUME_DIST]
FROM    product.product
```



Results | Messages

	list_price	CUME DIST
1	1.00	0,006
2	1.00	0,006
3	1.00	0,006
4	2.00	0,01
5	2.00	0,01
6	3.00	0,012
7	7.99	0,013
8	10.99	0,015
9	11.79	0,017
10	11.99	0,019
11	12.49	0,021
12	16.91	0,023
13	19.99	0,025
14	20.99	0,027
15	22.06	0,029

ESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows



Results | Messages

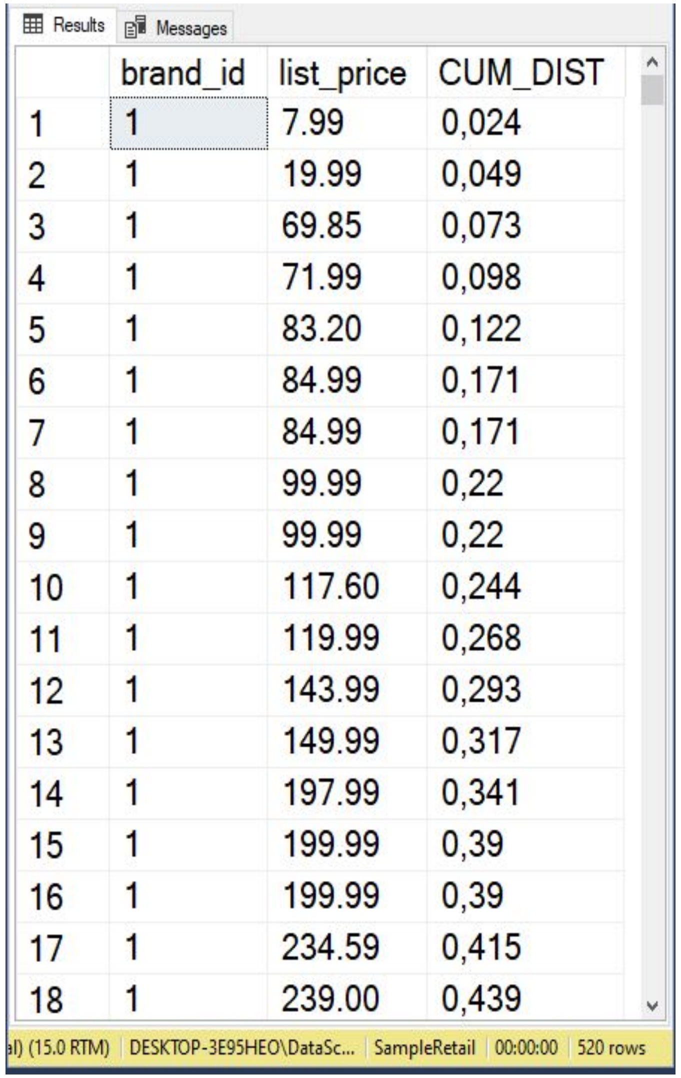
	list_price	CUME DIST
507	2499.00	0,975
508	2799.99	0,979
509	2799.99	0,979
510	2998.00	0,981
511	3013.98	0,983
512	3137.95	0,985
513	3298.00	0,987
514	3499.99	0,988
515	3799.99	0,99
516	3989.99	0,992
517	4275.00	0,994
518	4295.98	0,996
519	4296.99	0,998
520	16999.95	1

DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows

Query Time For You

Write a query that returns the cumulative distribution of the list price in product table by brand.

Expected Output:

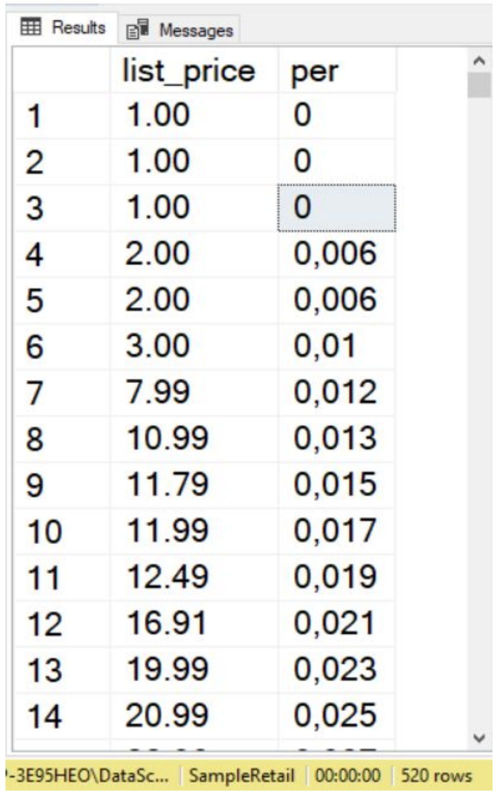


	brand_id	list_price	CUM_DIST
1	1	7.99	0,024
2	1	19.99	0,049
3	1	69.85	0,073
4	1	71.99	0,098
5	1	83.20	0,122
6	1	84.99	0,171
7	1	84.99	0,171
8	1	99.99	0,22
9	1	99.99	0,22
10	1	117.60	0,244
11	1	119.99	0,268
12	1	143.99	0,293
13	1	149.99	0,317
14	1	197.99	0,341
15	1	199.99	0,39
16	1	199.99	0,39
17	1	234.59	0,415
18	1	239.00	0,439

```
al) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows
```

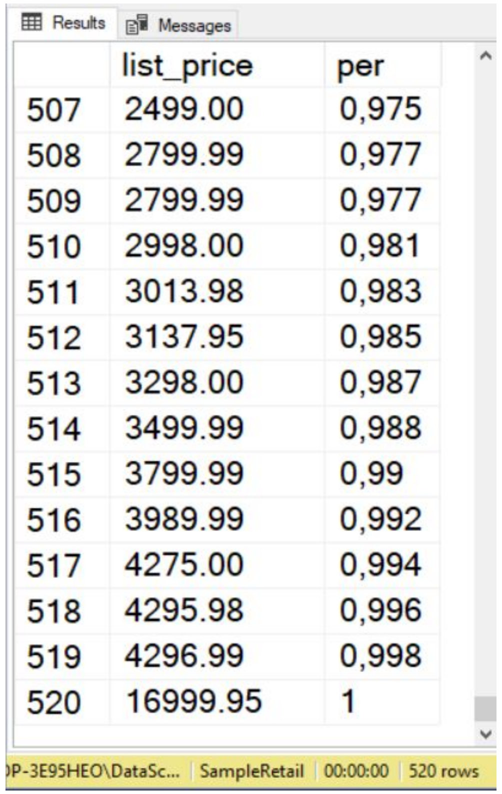
Percent_Rank()

```sql
SELECT  list_price,
        ROUND(PERCENT_RANK() OVER (ORDER BY list_price) , 3) AS [PERCENT_RANK]
FROM    product.product
```



	list_price	per
1	1.00	0
2	1.00	0
3	1.00	0
4	2.00	0,006
5	2.00	0,006
6	3.00	0,01
7	7.99	0,012
8	10.99	0,013
9	11.79	0,015
10	11.99	0,017
11	12.49	0,019
12	16.91	0,021
13	19.99	0,023
14	20.99	0,025

-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows

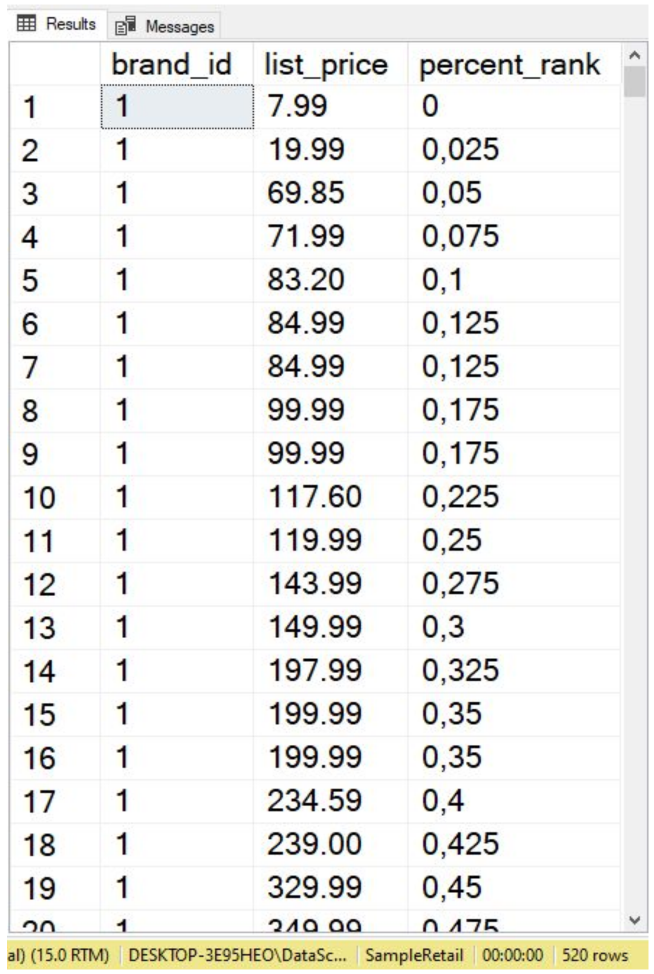


	list_price	per
507	2499.00	0,975
508	2799.99	0,977
509	2799.99	0,977
510	2998.00	0,981
511	3013.98	0,983
512	3137.95	0,985
513	3298.00	0,987
514	3499.99	0,988
515	3799.99	0,99
516	3989.99	0,992
517	4275.00	0,994
518	4295.98	0,996
519	4296.99	0,998
520	16999.95	1

)P-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows

Query Time

Write a query that returns the relative standing of the list price in product table by brand.

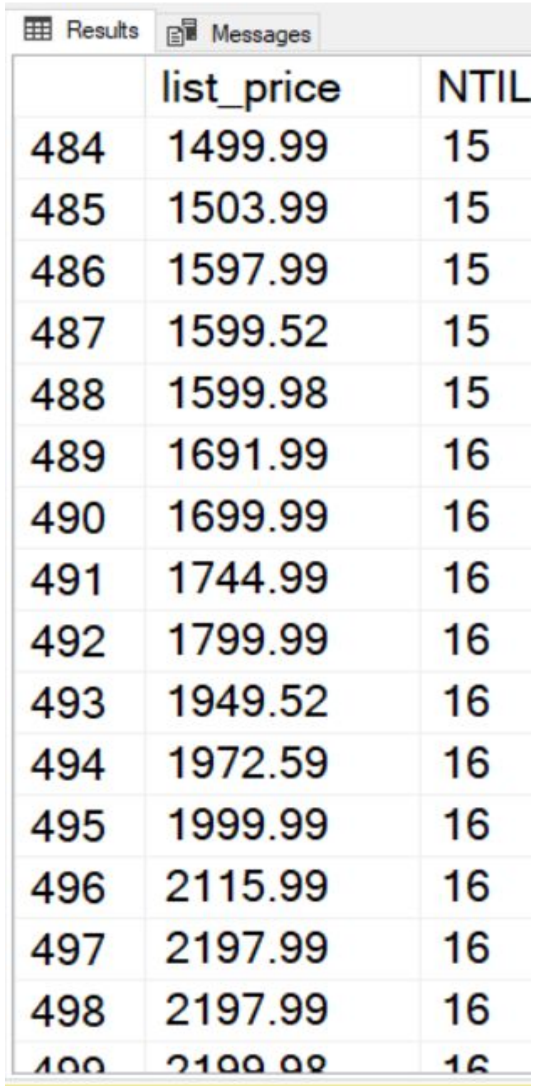

Results | Messages

	brand_id	list_price	percent_rank
1	1	7.99	0
2	1	19.99	0,025
3	1	69.85	0,05
4	1	71.99	0,075
5	1	83.20	0,1
6	1	84.99	0,125
7	1	84.99	0,125
8	1	99.99	0,175
9	1	99.99	0,175
10	1	117.60	0,225
11	1	119.99	0,25
12	1	143.99	0,275
13	1	149.99	0,3
14	1	197.99	0,325
15	1	199.99	0,35
16	1	199.99	0,35
17	1	234.59	0,4
18	1	239.00	0,425
19	1	329.99	0,45
20	1	349.99	0,475

al) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 520 rows

Ntile(N)

```
SELECT  list_price,
        NTILE(16) OVER (ORDER BY list_price) AS [NTILE]
FROM    product.product
```



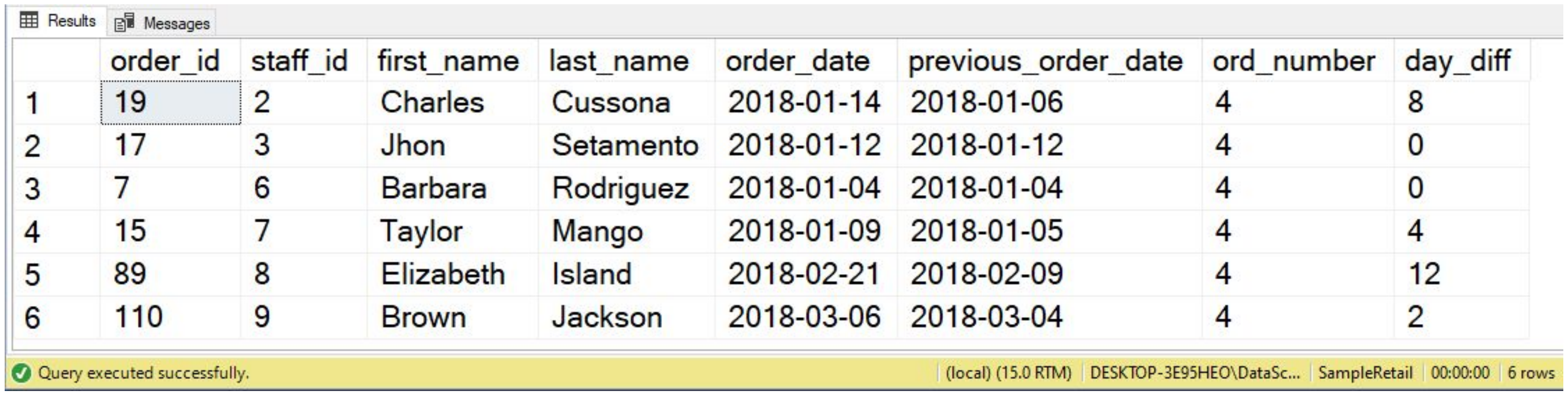
Results | Messages

	list_price	NTIL
484	1499.99	15
485	1503.99	15
486	1597.99	15
487	1599.52	15
488	1599.98	15
489	1691.99	16
490	1699.99	16
491	1744.99	16
492	1799.99	16
493	1949.52	16
494	1972.59	16
495	1999.99	16
496	2115.99	16
497	2197.99	16
498	2197.99	16
499	2199.98	16

Query Time For You

Write a query that returns how many days are between the third and fourth order dates of each staff.

Expected Output:



Results | Messages

	order_id	staff_id	first_name	last_name	order_date	previous_order_date	ord_number	day_diff
1	19	2	Charles	Cussona	2018-01-14	2018-01-06	4	8
2	17	3	Jhon	Setamento	2018-01-12	2018-01-12	4	0
3	7	6	Barbara	Rodriguez	2018-01-04	2018-01-04	4	0
4	15	7	Taylor	Mango	2018-01-09	2018-01-05	4	4
5	89	8	Elizabeth	Island	2018-02-21	2018-02-09	4	12
6	110	9	Brown	Jackson	2018-03-06	2018-03-04	4	2

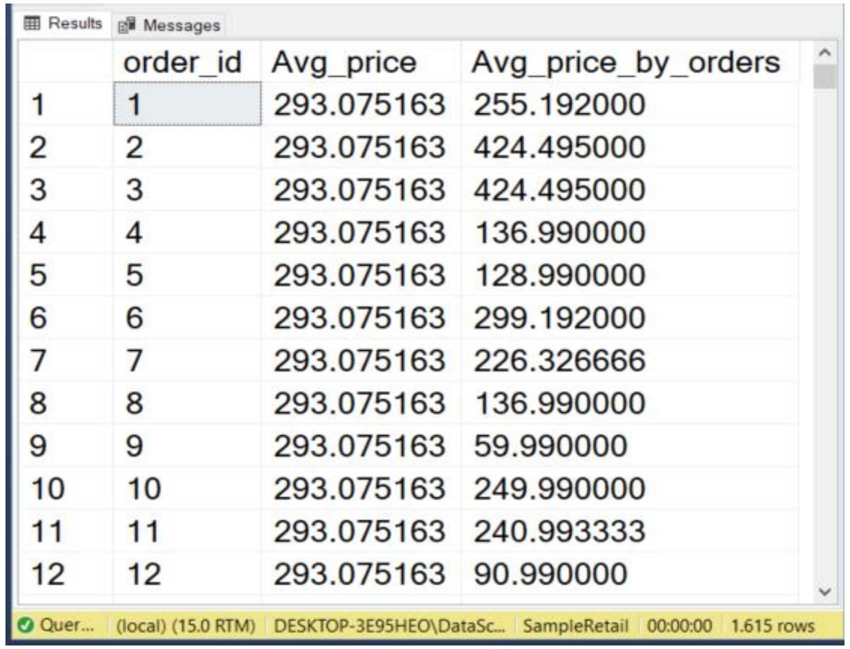
Query executed successfully. | (local) (15.0 RTM) | DESKTOP-3E95HEO\DataSc... | SampleRetail | 00:00:00 | 6 rows

Query Time For You

Write a query that returns both of the followings:

- The average product price
- The average product price by orders.

Expected Output:



Results | Messages

	order_id	Avg_price	Avg_price_by_orders
1	1	293.075163	255.192000
2	2	293.075163	424.495000
3	3	293.075163	424.495000
4	4	293.075163	136.990000
5	5	293.075163	128.990000
6	6	293.075163	299.192000
7	7	293.075163	226.326666
8	8	293.075163	136.990000
9	9	293.075163	59.990000
10	10	293.075163	249.990000
11	11	293.075163	240.993333
12	12	293.075163	90.990000

Quer... (local) (15.0 RTM) DESKTOP-3E95HEO\DataSc... SampleRetail 00:00:00 1.615 rows ondia

THANKS!

Any questions?