ADVANCED SQL CONCEPTS

WINDOW FUNCTIONS

ROW_NUMBER()

- DEFINITION: ASSIGNS A UNIQUE SEQUENTIAL INTEGER TO ROWS WITHIN A PARTITION OF A RESULT SET. THE NUMBERING STARTS AT 1 FOR THE FIRST ROW IN EACH PARTITION.

- USE CASE: USEFUL WHEN YOU NEED A UNIQUE IDENTIFIER FOR EACH ROW WITHIN A PARTITION, OFTEN USED FOR PAGINATION OR RANKING RESULTS IN A SPECIFIC ORDER.

- SYNTAX EXAMPLE:

ROW_NUMBER() OVER (PARTITION BY COLUMN_NAME ORDER BY COLUMN_NAME)

RANK()

- DEFINITION: ASSIGNS A RANK TO ROWS WITHIN A PARTITION OF A RESULT SET. THE RANK VALUES ARE CONSECUTIVE INTEGERS, WITH TIES RECEIVING THE SAME RANK, AND SUBSEQUENT RANKS BEING SKIPPED BASED ON THE NUMBER OF TIED ROWS.

- USE CASE: USEFUL WHEN YOU WANT TO HANDLE TIES BY ASSIGNING THE SAME RANK TO IDENTICAL VALUES, BUT WITH GAPS IN THE RANKING FOR TIED VALUES.

- SYNTAX EXAMPLE:

RANK() OVER (PARTITION BY COLUMN_NAME ORDER BY COLUMN_NAME)

DENSE_RANK()

- DEFINITION: ASSIGNS A RANK TO ROWS WITHIN A PARTITION OF A RESULT SET. LIKE RANK(), IT HANDLES TIES BY ASSIGNING THE SAME RANK TO IDENTICAL VALUES, BUT WITHOUT GAPS IN THE RANKING SEQUENCE.
- USE CASE: USEFUL WHEN YOU WANT TO HANDLE TIES BY ASSIGNING THE SAME RANK TO IDENTICAL VALUES, BUT WANT A CONTINUOUS RANKING SEQUENCE WITHOUT GAPS.
- SYNTAX EXAMPLE:

```
DENSE_RANK() OVER (PARTITION BY COLUMN_NAME ORDER BY COLUMN_NAME)
```

SELECT * FROM COFFEE_SALES;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ï»¿transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type	product_detail
1	01-01-2023	07:06:11	2	5	Lower Manhattan	32	3	Coffee	Gourmet brewed coffee	Ethiopia Rg
2	01-01-2023	07:08:56	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea	Spicy Eye Open
3	01-01-2023	07:14:04	2	5	Lower Manhattan	59	4.5	Drinking Chocolate	Hot chocolate	Dark chocolate I
4	01-01-2023	07:20:24	1	5	Lower Manhattan	22	2	Coffee	Drip coffee	Our Old Time Di
5	01-01-2023	07:22:41	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea	Spicy Eye Open
6	01-01-2023	07:22:41	1	5	Lower Manhattan	77	3	Bakery	Scone	Oatmeal Scone
7	01-01-2023	07:25:49	1	5	Lower Manhattan	22	2	Coffee	Drip coffee	Our Old Time Di
8	01-01-2023	07:33:34	2	5	Lower Manhattan	28	2	Coffee	Gourmet brewed coffee	Columbian Medi
9	01-01-2023	07:39:13	1	5	Lower Manhattan	39	4.25	Coffee	Barista Espresso	Latte Rg
10	01-01-2023	07:39:34	2	5	Lower Manhattan	58	3.5	Drinking Chocolate	Hot chocolate	Dark chocolate I
11	01-01-2023	07:43:05	1	5	Lower Manhattan	56	2.55	Tea	Brewed Chai tea	Spicy Eye Open

I WILL BE USING THIS DATA TO PRACTICE THE CONCEPTS

1) RANK TRANSACTIONS BY UNIT PRICE WITHIN EACH PRODUCT CATEGORY.

```
SELECT
UNIT_PRICE,PRODUCT_CATEGOR
Y, DENSE_RANK()
OVER(PARTITION BY
PRODUCT_CATEGORY ORDER BY
UNIT_PRICE) AS RNK FROM
COFFEE_SALES;
```

Result Grid | Filter Rows:

unit_price	product_category	rnk
2.65	Bakery	1
2.65	Bakery	1
2.65	Bakery	1
2.65	Bakery	1
2.65	Bakery	1
2.65	Bakery	1
3	Bakery	2
3	Bakery	2
3	Bakery	2
3	Bakery	2
3	Bakery	2
3	Bakery	2

2) DETERMINE THE RANK OF EACH PRODUCT TYPE BASED ON THE TOTAL QUANTITY SOLD ACROSS ALL TRANSACTIONS.

```
SELECT SUM(TRANSACTION_QTY)
AS QTY,PRODUCT_TYPE ,
DENSE_RANK() OVER ( ORDER BY
SUM(TRANSACTION_QTY) ) AS RNK
FROM COFFEE_SALES
GROUP BY PRODUCT_TYPE;
```

Result Grid Filter Rows:

qty	product_type	rnk
18	Green tea	1
21	Green beans	2
23	House blend Beans	3
26	Organic Chocolate	4
27	Drinking Chocolate	5
29	Gourmet Beans	6
30	Clothing	7

3) ASSIGN A RANK TO EACH TRANSACTION BASED ON THE TRANSACTION DATE AND TIME, WITHIN EACH STORE LOCATION.

```
SELECT
TRANSACTION_DATE,TRANSACTION_TIME,STORE_LOCATION, DENSE_RANK()
OVER(PARTITION BY STORE_LOCATION
ORDER BY TRANSACTION_DATE,
TRANSACTION_TIME ) AS RNK FROM
COFFEE_SALES;
```

Result Grid | Filter Rows: | Export:

transaction_date	transaction_time	store_location	rnk
01-01-2023	11:01:48	Astoria	1
01-01-2023	11:01:58	Astoria	2
01-01-2023	11:01:58	Astoria	2
01-01-2023	11:08:11	Astoria	3
01-01-2023	11:09:01	Astoria	4
01-01-2023	11:10:21	Astoria	5
01-01-2023	11:10:21	Astoria	5

4) FIND THE DENSE RANK OF EACH PRODUCT DETAIL BASED ON THE TRANSACTION QUANTITY, WITHIN EACH PRODUCT CATEGORY.

```
SELECT
PRODUCT_DETAIL,PRODUCT_CATEGORY,
SUM(TRANSACTION_QTY) AS
QTY,DENSE_RANK() OVER (PARTITION BY
PRODUCT_CATEGORY ORDER BY
SUM(TRANSACTION_QTY) DESC) AS RNK
FROM COFFEE_SALES GROUP BY
PRODUCT_CATEGORY,PRODUCT_DETAIL ;
```

Result Grid | Filter Rows: | Export:

product_detail	product_category	qty	rnk
Chocolate Croissant	Bakery	300	1
Ginger Scone	Bakery	239	2
Cranberry Scone	Bakery	199	3
Hazelnut Biscotti	Bakery	197	4
Jumbo Savory Scone	Bakery	196	5
Scottish Cream Scone	Bakery	186	6
Ginger Biscotti	Bakery	182	7
Croissant	Bakery	179	8
Chocolate Chip Biscotti	Bakery	177	9
Almond Croissant	Bakery	175	10

4) LIST THE ROW NUMBER FOR EACH TRANSACTION BASED ON THE TRANSACTION DATE AND TIME, WITHIN EACH STORE LOCATION.

```
SELECT
TRANSACTION_DATE,TRANSACTION_TIM
E,STORE_LOCATION, ROW_NUMBER()
OVER(PARTITION BY STORE_LOCATION
ORDER BY TRANSACTION_DATE,
TRANSACTION_TIME ) AS RNK FROM
COFFEE_SALES;
```

Result Grid | Filter Rows: | Export:

transaction_date	transaction_time	store_location	rnk
01-01-2023	11:01:48	Astoria	1
01-01-2023	11:01:58	Astoria	2
01-01-2023	11:01:58	Astoria	3
01-01-2023	11:08:11	Astoria	4
01-01-2023	11:09:01	Astoria	5
01-01-2023	11:10:21	Astoria	6
01-01-2023	11:10:21	Astoria	7
01-01-2023	11:10:58	Astoria	8
01-01-2023	11:12:29	Astoria	9
01-01-2023	11:16:02	Astoria	10

5) CALCULATE THE DENSE RANK OF EACH STORE BASED ON THE TOTAL UNIT PRICE OF PRODUCTS SOLD, WITHIN EACH PRODUCT TYPE.

```
WITH SALESSUMMARY AS (
SELECT
STORE_LOCATION,
PRODUCT_TYPE,
SUM(UNIT_PRICE) AS TOTAL
FROM COFFEE_SALES
GROUP BY STORE_LOCATION,
PRODUCT_TYPE
)
SELECT
STORE_LOCATION,
PRODUCT_TYPE,
TOTAL,
DENSE_RANK() OVER(PARTITION BY
STORE_LOCATION ORDER BY TOTAL DESC)
AS RNK
FROM SALESSUMMARY;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

store_location	product_type	total	rnk
Astoria	Brewed Chai tea	1759.8499999999933	1
Astoria	Barista Espresso	1746	2
Astoria	Hot chocolate	1709.75	3
Astoria	Gourmet brewed coffee	1480.6000000000029	4
Astoria	Scone	1077.5	5
Astoria	Brewed Black tea	1036	6
Astoria	Brewed herbal tea	991	7
Astoria	Premium brewed coffee	860.4500000000025	8
Astoria	Pastry	777.25	9
Astoria	Organic brewed coffee	760.1000000000008	10
Astoria	Drip coffee	697	11
Astoria	Biscotti	620.5	12

6) DETERMINE THE RANK OF EACH PRODUCT ID BASED ON THE AVERAGE UNIT PRICE WITHIN EACH PRODUCT CATEGORY.

```
SELECT
PRODUCT_ID,ROUND(AVG(UNIT_PRIC
E),2) AS AVGG,PRODUCT_CATEGORY ,
DENSE_RANK() OVER(PARTITION BY
PRODUCT_CATEGORY ORDER BY
AVG(UNIT_PRICE)) AS RNK FROM
COFFEE_SALES GROUP BY
PRODUCT_CATEGORY,PRODUCT_ID;
```

product_id	avgg	product_category	rnk
77	3	Bakery	1
72	3.2	Bakery	2
70	3.26	Bakery	3
69	3.26	Bakery	4
74	3.5	Bakery	5
75	3.51	Bakery	6
76	3.52	Bakery	7
73	3.75	Bakery	8
71	3.75	Bakery	9
79	3.77	Bakery	10
78	4.51	Bakery	11

7) DETERMINE THE DENSE RANK OF EACH PRODUCT TYPE BASED ON THE TOTAL TRANSACTION QUANTITY, WITHIN EACH PRODUCT CATEGORY.

```
SELECT
DENSE_RANK() OVER (
PARTITION BY PRODUCT_CATEGORY
ORDER BY SM DESC
) AS RNK,
SM,
PRODUCT_TYPE,
PRODUCT_CATEGORY
FROM (
SELECT
SUM(TRANSACTION_QTY) AS SM,
PRODUCT_TYPE,
PRODUCT_CATEGORY
FROM COFFEE_SALES
GROUP BY PRODUCT_TYPE, PRODUCT_CATEGORY
) AS A;
```

Result Grid | Filter Rows: | Export:

rnk	sm	product_type	product_category
1	993	Scone	Bakery
2	654	Pastry	Bakery
3	556	Biscotti	Bakery
1	51	Housewares	Branded
2	30	Clothing	Branded
1	2444	Gourmet brewed coffee	Coffee
2	2339	Barista Espresso	Coffee
3	1260	Organic brewed coffee	Coffee
4	1185	Drip coffee	Coffee
5	1177	Premium brewed coffee	Coffee

8) ASSIGN A ROW NUMBER TO EACH TRANSACTION BASED ON TRANSACTION TIME, WITHIN EACH PRODUCT CATEGORY.

```
SELECT
PRODUCT_CATEGORY,TRANSACTION_
TIME ,`TRANSACTION_ID`,
ROW_NUMBER() OVER(PARTITION BY
PRODUCT_CATEGORY) AS ROW_NUM
FROM COFFEE_SALES;
```

Result Grid | Filter Rows: | Export:

product_category	transaction_time	ï»¿transaction_id	row_num
Bakery	08:24:29	13837	1
Bakery	16:31:22	14210	2
Bakery	16:04:34	14191	3
Bakery	07:12:13	13764	4
Bakery	16:00:26	14188	5
Bakery	10:03:52	13949	6
Bakery	11:52:49	14039	7
Bakery	07:15:48	13768	8
Bakery	13:38:53	14110	9
Bakery	07:15:51	13770	10
Bakery	08:32:28	13844	11
Bakery	15:36:38	14177	12

9) FIND THE DENSE RANK OF EACH PRODUCT BASED ON TOTAL REVENUE (UNIT PRICE MULTIPLIED BY TRANSACTION QUANTITY) WITHIN EACH STORE LOCATION.

```
SELECT STORE_LOCATION,
PRODUCT_ID,SUM(UNIT_PRICE*TRANSA
CTION_QTY) AS TOTAL_REV ,
DENSE_RANK() OVER(PARTITION BY
STORE_LOCATION ORDER BY
SUM(UNIT_PRICE*TRANSACTION_QTY))
AS RNK FROM COFFEE_SALES GROUP BY
PRODUCT_ID,STORE_LOCATION;
```

Result Grid | Filter Rows: | Export:

store_location	product_id	total_rev	rnk
Astoria	14	17.9	1
Astoria	16	26.849999999999998	2
Astoria	18	32.849999999999994	3
Astoria	64	37.60000000000001	4
Astoria	10	40	5
Astoria	13	44.75	6
Astoria	11	44.75	6
Astoria	5	45	7
Astoria	15	46.25	8
Astoria	63	49.60000000000001	9
Astoria	65	51.20000000000001	10
Astoria	12	53.7	11

10)CALCULATE THE RANK OF EACH STORE LOCATION BASED ON THE TOTAL QUANTITY SOLD, WITHIN EACH PRODUCT CATEGORY.

```
SELECT STORE_LOCATION
,PRODUCT_CATEGORY,SUM(TRANSAC
TION_QTY),RANK() OVER(PARTITION
BY PRODUCT_CATEGORY ORDER BY
SUM(TRANSACTION_QTY)) AS RNK
FROM COFFEE_SALES GROUP BY
STORE_LOCATION,PRODUCT_CATEG
ORY;
```

Result Grid | Filter Rows: | Export: | Wrap C

store_location	product_category	sum(transaction_qty)	rnk
Astoria	Bakery	717	1
Hell's Kitchen	Bakery	720	2
Lower Manhattan	Bakery	766	3
Hell's Kitchen	Branded	11	1
Lower Manhattan	Branded	32	2
Astoria	Branded	38	3
Astoria	Coffee	2722	1
Lower Manhattan	Coffee	2798	2
Hell's Kitchen	Coffee	2885	3
Astoria	Coffee beans	50	1
Lower Manhattan	Coffee beans	58	2

THANK YOU