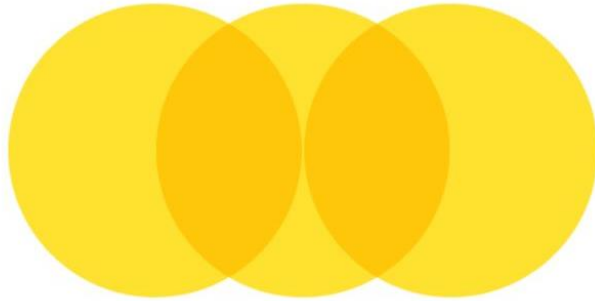


Bytewise Fellowship



Fellowship.

Powered By Bytewise

Task 5: Advanced SQL Queries on Sales Data

Name:	Muhammad Ahsan Saleem
Date:	30 th June, 2024
Submitted to:	Muhammad Bilal

Introduction:

In this task, we will delve deeper into advanced SQL queries to extract meaningful insights from the SalesManagementDB database. The objective was to write complex queries involving window functions, aggregate functions, and various types of joins to gain deeper insights into the sales data. This task included calculating the percentage contribution of each item's amount to its order's total, ranking orders within each customer, determining the average price of products by supplier, counting products in each category, and retrieving the total amount spent by each customer.

Task 5

- Using the same database you previously created perform the following tasks:

Questions:

1. Write a query to calculate the percentage contribution of each item's amount to its order's total amount, grouped by order_id.
2. Write a query to rank orders by their total amount within each customer, ordering them from highest to lowest total amount.
3. Write a query to calculate the average price of products supplied by each supplier. Exclude suppliers who have no products in the result.
4. Write a query to count the number of products in each category. Include categories with zero products in the result set.
5. Write a query to retrieve the total amount spent by each customer, along with their name and phone number. Ensure customers with no orders also appear with a total amount of 0.

Queries and Their Documentation:

Each query addresses a specific question regarding the data.

Question 1

Statement: Write a query to calculate the percentage contribution of each item's amount to its order's total amount, grouped by order_id.

Query:

```
SELECT
    item_id,
    order_id,
    amount,
    (amount/(SUM(amount) OVER (PARTITION BY order_id)))*100 AS
percentage_of_order_total
FROM
    sales.items
ORDER BY
    order_id;
```

```
SELECT
    item_id,
    order_id,
    amount,
    (amount/(SUM(amount) OVER (PARTITION BY order_id)))*100 AS percentage_of_order_total
FROM
    sales.items
ORDER BY
    order_id
```

Explanation: This query calculates the percentage contribution of each item's amount to the total amount of its order using the PARTITION BY clause.

Output:

	item_id [PK] character varying (100)	order_id character varying (100)	amount double precision	percentage_of_order_total double precision
1	caadf311-7efb-47b9-bd48-ccd4d0e9d628	0077622c-f789-402f-8f27-a8ef29b92f99	32.05	100
2	d7c40e3c-3c7c-48d1-a657-03e72e34e8...	007c6893-fc40-44f8-a794-741ad1bfd9c7	52.99	100
3	ccc44ebb-07d3-4d71-91c6-22c52bba1c...	00860b6c-6b06-4694-8708-f7aa4375d5...	99.77	46.78765709998124
4	d2d2243f-1e68-4e2d-8fb9-31c8d2862416	00860b6c-6b06-4694-8708-f7aa4375d5...	39.54	18.542487338210467
5	ed82300c-05b2-468f-8056-5a61c89a39d6	00860b6c-6b06-4694-8708-f7aa4375d5...	73.93	34.66985556180829
6	68e9bbd1-10c4-4518-8320-e7cf83dcb93a	00871b7a-6c23-4916-aeff-e22ee8e27ea9	65.86	100
7	da44883b-5b94-4107-b7e8-18dd3a6c2b...	00aa2373-5f66-4165-8ebf-7724b314028a	10.07	27.51366120218579
8	90563a9f-0c7b-4e55-b7aa-3f3d7231bf15	00aa2373-5f66-4165-8ebf-7724b314028a	26.53	72.48633879781421
9	52d92662-c37f-488b-a57c-e6ec730533d6	00ce4abc-b40a-4918-a245-5489bea007	91.49	50.558134394341295

Question 2

Statement: Write a query to rank orders by their total amount within each customer, ordering them from highest to lowest total amount.

Query:

```
SELECT
    order_id,
    total_amount,
    customer_id,
    rank() OVER (PARTITION BY customer_id ORDER BY total_amount DESC
) AS rank
FROM
    sales.orders
ORDER BY
    customer_id
```

```
SELECT
|   order_id,
|   total_amount,
|   customer_id,
|   rank() OVER (PARTITION BY customer_id ORDER BY total_amount DESC ) AS rank
FROM
|   sales.orders
ORDER BY
|   customer_id
```

Explanation: This query ranks the orders of each customer by total amount in descending order using the RANK() window function.

Output:

	order_id [PK] character varying (100)	total_amount double precision	customer_id character varying (100)	rank bigint
1	a2f7c3af-4f45-413a-951a-aaf4a8f35c95	380.06	003ee86f-4e17-4b39-bb6a-30897fe0a1...	1
2	e8e664fb-189a-48f6-8815-9075fc391b83	691.4	00d40438-5bf3-4f1b-aaa9-32130a7cd6...	1
3	e0aeaf57-5169-4300-bc73-cbd81c52b9...	903.59	01624052-d9dc-4150-bd53-255b82ce6...	1
4	df8c9be5-054e-462f-8d10-0a1f906f72d4	414.86	01624052-d9dc-4150-bd53-255b82ce6...	2
5	40d3d4b8-4f19-415a-a027-b08fd264e7...	322.84	01624052-d9dc-4150-bd53-255b82ce6...	3
6	a234d0cc-04dd-4e98-822c-bb44d5833d...	242.15	01624052-d9dc-4150-bd53-255b82ce6...	4
7	4baad587-2be9-463e-bd5c-db912ad70b...	188.72	01624052-d9dc-4150-bd53-255b82ce6...	5
8	0b346353-a7af-471c-bd6d-46a2d07f2d...	950.95	017058e3-3c0b-4f2f-a359-06cba4a8b5ff	1
9	483bc108-36ae-431d-9591-95a71189ab	700.13	017058e3-3c0b-4f2f-a359-06cba4a8b5ff	2

Question 3

Statement: Write a query to calculate the average price of products supplied by each supplier. Exclude suppliers who have no products in the result.

Query:

```
SELECT
    suppliers.supplier_id AS supplier_id,
    suppliers.name AS supplier_name,
    AVG(products.price) AS avg_price
FROM
    sales.suppliers
RIGHT JOIN
    sales.products
ON
    suppliers.supplier_id = products.supplier_id
GROUP BY
    suppliers.supplier_id,
    suppliers.name;
```

```

SELECT
    suppliers.supplier_id AS supplier_id,
    suppliers.name AS supplier_name,
    AVG(products.price) AS avg_price
FROM
    sales.suppliers
RIGHT JOIN
    sales.products
ON
    suppliers.supplier_id = products.supplier_id
GROUP BY
    suppliers.supplier_id,
    suppliers.name

```

Explanation: This query calculates the average price of products supplied by each supplier using AVG() and GROUP BY.

Output:

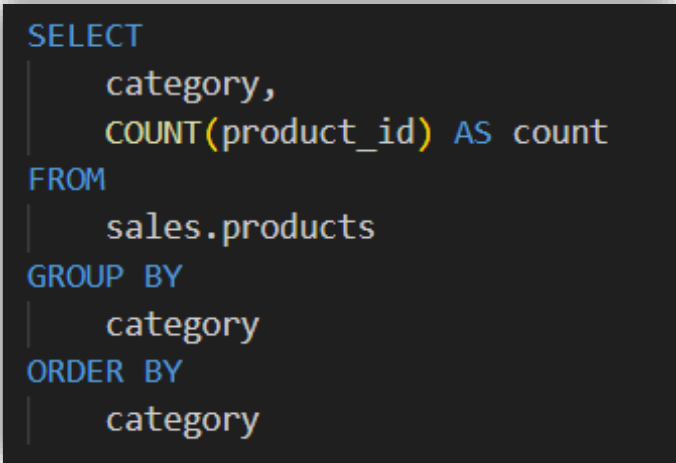
	supplier_id [PK] character varying (100)	supplier_name character varying (100)	avg_price double precision
1	08e2b063-f6fa-4ab1-90c9-e384fa1e957f	Romero Inc	135.04333333333332
2	2584a949-df20-4453-8bdf-5223c451176e	Robinson-Harris	273.07
3	72921f5a-c981-4f2a-9558-ed71d7c4502e	Romero-Holland	30.62
4	8d11deee-bb85-42f3-bf3a-cf0a7438875d	Murphy, Watts and Mcgrath	318.6
5	1ddfcc5d-bb48-454b-9ac7-a09f663a8aaa	Miller, Moore and Collins	35.58
6	54457ba9-f426-4867-bc80-f479aa1815dd	Short, Johnson and Smith	344.62
7	6c98c4d8-8056-4fff-8483-1d413a3e0b42	Olsen, Stanton and Thompson	222.53
8	469c598f-5d17-48d4-ae52-845bcaeba6...	Pearson-Kelley	263.135
9	cb02444f-0889-426e-ac49-6d20e2f381cf	Walker, Graham and Hernandez	279.5833333333333

Question 4

Statement: Write a query to count the number of products in each category. Include categories with zero products in the result set.

Query:

```
SELECT
    categories.name AS category,
    COUNT(products.product_id) AS count
FROM
    sales.categories
LEFT JOIN
    sales.products
ON
    categories.name = products.category
GROUP BY
    categories.name
ORDER BY
    category;
```



```
SELECT
|   category,
|   COUNT(product_id) AS count
FROM
|   sales.products
GROUP BY
|   category
ORDER BY
|   category
```

Explanation: This query counts the number of products in each category, including categories with zero products, GROUP BY and COUNT. Also, no JOINS or are used because all categories are available in the products table and no category_id is provided to JOIN on.

Output:

	category character varying (50)	count bigint
1	a	1
2	able	2
3	about	2
4	above	3
5	accept	1
6	according	1
7	act	1
8	actually	1
9	address	3

Question 5

Statement: Write a query to retrieve the total amount spent by each customer, along with their name and phone number. Ensure customers with no orders also appear with a total amount of 0.

Query:

```
SELECT
    c.name,
    c.phone,
    COALESCE(SUM(o.total_amount), 0) AS amount_per_customer
FROM
    sales.customers AS c
LEFT JOIN
    sales.orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.name, c.phone
ORDER BY
    amount_per_customer;
```



```

SELECT
|   c.name,
|   c.phone,
|   COALESCE(SUM(o.total_amount),0) AS amount_per_customer
FROM
|   sales.orders as o
RIGHT JOIN
|   sales.customers as c
ON
|   c.customer_id = o.customer_id
GROUP BY
|   c.name, c.phone
ORDER BY
|   amount_per_customer

```

Explanation: This query retrieves the total amount spent by each customer, including those with no orders, using COALESCE() and LEFT JOIN.

Output:

	name character varying (100) 🔒	phone character varying (50) 🔒	amount_per_customer double precision 🔒
1	Christine Malone	216-580-3605x7125	3560.20000000000003
2	Richard Frank	8414259229	3414.54
3	Robert Beck	(216)668-7359x376	3389.73
4	Joseph Shah	702.218.2910	2906.2799999999997
5	Danielle Woods	583-490-8066	2895.3
6	Shane Ramos	+1-505-576-0551x519	2767.9
7	Sherry Roberts	(590)310-6887x97988	2646.2799999999997
8	Zachary Martinez	314-757-5127x58179	2533.96
9	Alyssa Watson	+1-825-201-6131x2798	2472.71
10	Duane Walsh	297-270-8585x59708	2452.54
Total rows: 1000 of 1000		Query complete 00:00:00.081	

Conclusion:

This task involved executing complex SQL queries to derive meaningful insights from the sales data, enhancing my proficiency with window functions, aggregate functions, and joins. These skills are crucial for advanced data analysis and will significantly contribute to my growth as a data engineer.