EXPLORING E-COMMERCE WEBSITE

INTERNSHIP_MAJOR_PROJECT AT ACADEMOR

AUTHOR/ANALYST: SAGNIK ADAK

JANUARY 23,2024

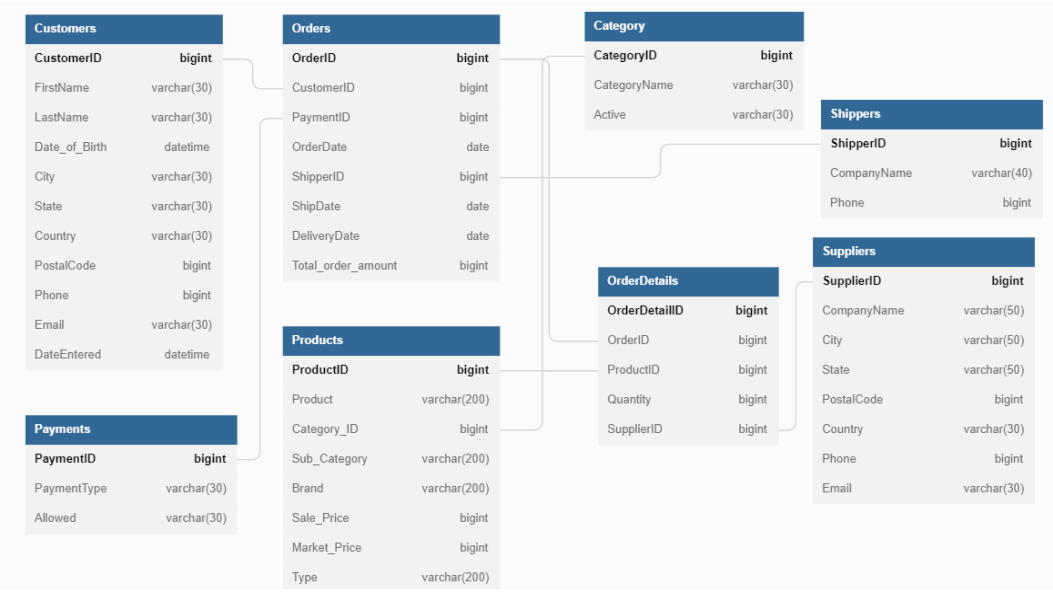
DATA IMPORTATION

- First I applied data processing and data cleaning operation on the given data-sheets through Excel.
- Saved the corrected files in CSV format.
- Then I created the eight tables in PostgreSQL and imported the CSV files.

Query | Query History

```
1  create table shippers(ShipperID int,CompanyName varchar,Phone varchar(15));
2
3  create table customers(CustomerID int,FirstName varchar,LastName varchar,Date_of_Birth date,City varchar,State varchar,
4                         country varchar, PostalCode int,Phone varchar(15), Email varchar,DateEntered  date);
5  create table payments(PaymentID int,PaymentType varchar,Allowed varchar);
6  create table orders(OrderID int, CustomerID int,PaymentID int,OrderDate date,
7                      ShipperID int, ShipDate date, DeliveryDate date,Total_order_amount float);
8  create table products(ProductID int,Product varchar,CategoryID int,Sub_Category varchar, Brand varchar,
9                        Sale_Price int, Market_Price int,Type varchar);
10 create table category(CategoryID int,CategoryName varchar,Activr varchar);
11 create table orderdetails(OrderDetailID int, OrderID int, ProductID int,Quantity int,SupplierID int);
12 create table suppliers(SupplierID int,CompanyName varchar, City varchar, State varchar,
13                        PostalCode int, Country varchar,Phone varchar(15),Email varchar);
```

(Table names and their columns are written)



(Relations of the tables are shown)

SET-1

1. Write a Query to Print the position of the character ‘a’ in the First Names and Last Names of Customers.

```
18  select firstname, position ('a' in firstname) as firstname_position,
19  lastname, position ('a' in lastname)as lastname_position
20  from customers
```

Data Output | Messages | Notifications

	firstname character varying	firstname_position integer	lastname character varying	lastname_position integer
1	James	2	Smith	0
2	Robert	0	Downey Jr	0
3	John	0	Williams	6
4	Michael	5	Johnson	0
5	Steve	0	Williams	6
6	David	2	Beckham	6
7	Richard	5	Brown	0
8	Joseph	0	James	2
9	Thomas	5	Jones	0
10	Charles	3	King	0

Total rows: 525 of 525 | Query complete 00:00:00.086

2. Write a Query to Print the Position of the substring ‘ch’ in the First Names of all the Customers.

```
24  select firstname, position ('ch' in firstname) as firstname_ch_position
25  from customers
26
```

Data Output | Messages | Notifications

	firstname character varying	firstname_ch_position integer
1	James	0
2	Robert	0
3	John	0
4	Michael	3
5	Steve	0
6	David	0
7	Richard	3
8	Joseph	0
9	Thomas	0
10	Charles	0
11	Christopher	0
12	Daniel	0
13	Matthew	0
14	Anthony	0
15	Mark	0
16	Donald	0
17	Steven	0
18	Paul	0
19	Andrew	0
20	Joshua	0

Total rows: 525 of 525 | Query complete 00:00:00.096

3. Write a Query to print the Position of the character ‘e’ in the First Names of all customers,if ‘e’ does not exist in the first name then use CASE WHEN to print NULL instead of 0.

```
31  select firstname ,
32      case when
33          position('e' in firstname) > 0
34    then
35          position('e' in firstname)
36    else null end as position_e_firstname
37  from customers
38
```

Data Output | Messages | Notifications

	firstname character varying	position_e_firstname integer
1	James	4
2	Robert	4
3	John	[null]
4	Michael	6
5	Steve	3
6	David	[null]
7	Richard	[null]
8	Joseph	4
9	Thomas	[null]
10	Charles	6
11	Christopher	10
12	Daniel	5
13	Matthew	6
14	Anthony	[null]
15	Mark	[null]
16	Donald	[null]
17	Steven	3

Successfully r

Total rows: 525 of 525 Query complete 00:00:00.084

4. Write a Query to print the time taken in days to deliver orders from the shipping date for each order in the Orders table.

```
41  select orderid, shipdate, deliverydate, deliverydate - shipdate as days
42      from orders
43
```

Data Output | Messages | Notifications

	orderid integer	shipdate date	deliverydate date	days integer
1	7655500	2020-01-13	2020-01-19	6
2	7655501	2020-01-24	2020-01-27	3
3	7655502	2020-02-11	2020-02-21	10
4	7655503	2020-02-13	2020-02-26	13
5	7655504	2020-02-15	2020-02-20	5
6	7655505	2020-02-24	2020-02-27	3
7	7655506	2020-02-23	2020-02-27	4
8	7655507	2020-03-04	2020-03-11	7
9	7655508	2020-02-28	2020-03-08	9
10	7655509	2020-02-27	2020-03-08	10
11	7655510	2020-03-07	2020-03-14	7
12	7655511	2020-03-07	2020-03-17	10
13	7655512	2020-03-14	2020-03-27	13
14	7655513	2020-03-12	2020-03-26	14
15	7655514	2020-03-14	2020-03-22	8
16	7655515	2020-03-20	2020-03-28	8
17	7655516	2020-03-14	2020-03-19	5

Successfully run. Total query runtim

Total rows: 1000 of 4999 Query complete 00:00:00.181

5. Write a query to print all customer details. Order your output in ascending order of the first 3 characters of FirstName.

```
select *
from customers
order by left(firstname, 3) asc;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying
1	57132	Aaron	Paul	1976-10-13	Namur	Namur	Belgium	322042	8668707337	Aaron.Paul@gmail.com
2	57316	Aaron	Axel	1994-06-09	Galway	Galway	Ireland	771952	6015357565	Aaron.Axel@gmail.com
3	57569	Abdullah	Lochlan	1965-03-06	Belfast	Belfast	Northern Ireland	523272	7792025369	Abdullah.Lochlan@gmail.
4	57558	Abel	Archie	1990-04-21	Villach	Carinthia	Austria	750733	8732427068	Abel.Archie@gmail.com
5	57261	Abigail	Richards	1965-01-13	Brussels	Brussels-Capital	Belgium	829257	9617315057	Abigail.Richards@gmail.c
6	57134	Adam	Levin	1994-07-06	San Diego	California	United States	529879	7460297856	Adam.Levin@gmail.com
7	57313	Adam	Federer	1973-05-12	Melbourne	Victoria	Australia	261795	7788434637	Adam.Federer@gmail.cor
8	57576	Adrian	Enzo	1956-06-10	Zurich	Zurich	Switzerland	657311	9739029706	Adrian.Enzo@gmail.com
9	57369	Aidan	Maximilian	1959-07-03	NvÆmes	Occitanie	France	343973	7083741130	Aidan.Maximilian@gmail.
10	57368	Aiden	Sanchez	1971-09-24	Lisbon	Lisbon	Portugal	978786	6297648653	Aiden.Sanchez@gmail.co
11	57505	Alan	Anthony	1957-12-30	Townsville	Queensland	Australia	777883	8727734228	Alan.Anthony@gmail.com
12	57484	Alasdair	Odin	1999-01-05	Hobart	Tasmania	Australia	386859	7433996128	Alasdair.Odin@gmail.com
13	57167	Alan	Walker	1999-05-01	Houston	Texas	United States	425321	7519015025	Alan.Walker@gmail.com
14	57163	Albert	Black	1975-03-12	Houston	Texas	United States	531839	8269461310	Albert.Black@gmail.com
15	57425	Albie	Peter	1999-04-07	Vienna	Vienna	Austria	615221	6015427642	Albie.Peter@gmail.com
16	57291	Alexander	Elijah	1956-08-26	Athens	Attica	Gre			

Successfully run. Total query runtime: 158 msec. 525 rows affected.

Total rows: 525 of 525 | Query complete 00:00:00.158 | Ln 51, Col 34

6. Write a query to print pairs of customers who belong to the same state.

```
select c1.customerid as customerid,
       c1.firstname as firstname,
       c1.lastname as lastname,
       c2.customerid as customerid2,
       c1.firstname as firstname2,
       c1.lastname as lastname2,
       c1.state as common_state
       from customers c1, customers c2
       where c1.customerid < c2.customerid and c1.state= c2.state;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	customerid2 integer	firstname2 character varying	lastname2 character varying	common_state character varying
1	57081	James	Smith	57301	James	Smith	New York
2	57081	James	Smith	57283	James	Smith	New York
3	57081	James	Smith	57264	James	Smith	New York
4	57081	James	Smith	57248	James	Smith	New York
5	57081	James	Smith	57230	James	Smith	New York
6	57081	James	Smith	57215	James	Smith	New York
7	57081	James	Smith	57196	James	Smith	New York
8	57081	James	Smith	57169	James	Smith	New York
9	57081	James	Smith	57157	James	Smith	New York
10	57081	James	Smith	57154	James	Smith	New York
11	57081	James	Smith	57082	James	Smith	New York
12	57082	Robert	Downey Jr	57301	Robert	Downey Jr	New York
13	57082	Robert	Downey Jr	57283	Robert	Downey Jr	New York
14	57082	Robert	Downey Jr	57264	Robert	Downey Jr	New York
15	57082	Robert	Downey Jr	57248	Robert	Downey Jr	New York
16	57082	Robert	Downey Jr	57230	Robert	Downey Jr	New York
17	57082	Robert	Downey Jr	57215	Robert	Downey Jr	New York

Total rows: 1000 of 2645 | Query complete 00:00:00.142 | Ln 64, Col 1

7. Get the total number of records in the table.

```
69  select count(*) as total_records from customers
70
```

Data Output | Messages | Notifications

	total_records bigint
1	525

8. Get the Total Number of Records for Brand Harpic.

```
73  select count(*) as total_records from products where brand= 'Harpic';
74
```

Data Output | Messages | Notifications

	total_records bigint
1	28

9.Get the total Number of DISTINCT records for the Brand Harpic. What is the difference from the output of the above query- and why?

```
77  select count(distinct productid) as total_records from products where brand= 'Harpic';
```

Data Output | Messages | Notifications

	total_records bigint
1	28

1. Total Number of Records for the Brand 'Harpic': (qn no. 8)

-This query (`SELECT COUNT(*) AS total_records FROM products WHERE brand = 'Harpic';`) **counts all records in the 'products' table where the brand is 'Harpic', including duplicate records. If there are multiple records with the same product details and brand, each of them is counted.**

2. Total Number of DISTINCT Records for the Brand 'Harpic'(qn no. 9)

- In this query (`SELECT COUNT(DISTINCT productID) AS total_distinct_records FROM products WHERE brand = 'Harpic';`), the `DISTINCT` keyword is used with the `COUNT` function. It **counts only the distinct (unique) product IDs for the brand 'Harpic'. If there are duplicate records with the same product details, only one of them is counted.**

In summary, the **first query counts all records, including duplicates**, while the **second query counts only distinct records** based on the specified column (`productID` in this case).

10.Get the Total Number of Records for Brand Lizol.

```
81  select count(*) as total_records from products where brand= 'Lizol';
82
```

Data Output | Messages | Notifications

	total_records bigint
1	20

11.Get the total number of records for each Brand in a single output.

```
85  select brand, count(*) as total_records from products group by brand;
86
```

Data Output | Messages | Notifications

	brand character varying	total_records bigint
1	C&S Electric	3
2	Fini	13
3	Yakult	2
4	Spigadoro	2
5	Zorabian	7
6	Meixiang	3
7	SterloMax	1
8	Ujala	3
9	LaOpala	56
10	Toska Chocolates	14
11	Sri Sri Tattva	84
12	4700BC	7
13	Clenz U	3
14	Raju Candles	1
15	Tabac	2
16	Shakira	6
17	Farmers Family	3

Total rows: 1000 of 2245 | Query complete 00:00:00.155

12.Calculate the AVERAGE market Price for all products belonging to Type "Nachos & Chips".

```
89  select round(avg(market_price),2)as avg_market_price
90  from products where type = 'Nachos & Chips';
91
```

Data Output | Messages | Notifications

	avg_market_price numeric
1	140.55

13.Get the SUM of Market Price of all Products belonging to Type "Dry Fruits & Berries".

```
94  select sum(market_price ) as total
95  from products
96  where type = 'Dry Fruits & Berries';
97
```

Data Output | Messages | Notifications

	total bigint
1	81559

14.Get the MAX Sale Price of Products across each Sub-Category.

```
100 select sub_category, max(sale_price) as max_sale_price
101 from products
102 group by sub_category;
103
```

Data Output | Messages | Notifications

	sub_category character varying	max_sale_price integer
1	Dairy & Cheese	1350
2	Cookware & Non Stick	7999
3	Party & Festive Needs	1149
4	Hair Care	3795
5	Cereals & Breakfast	1796
6	Atta, Flours & Sooji	850
7	Car & Shoe Care	700
8	Tinned & Processed Food	800
9	Flask & Casserole	1029
10	Breakfast Cereals	784
11	Steel Utensils	3999
12	Fruit Juices & Drinks	650
13	Dairy	1108
14	Spreads, Sauces, Ketchup	1950
15	Skin Care	3105
16	Sausages, Bacon & Salami	875
17	Salt, Sugar & Jaggery	799

Total rows: 90 of 90 | Query complete 00:00:00.147

15.Get the DISTINCT Count of Categories.

```
106  select count(distinct categoryid)as distinct_category_count
107  from category
108
```

Data Output | Messages | Notifications

	distinct_category_count bigint
1	11

16.Get the DISTINCT Count of Products of the Type "Canned Seafood".

```
111  select count(distinct productid)as distinct_product_count
112  from products
113  where type = 'Canned Seafood'
114
115
```

Data Output | Messages | Notifications

	distinct_product_count bigint
1	31

SET-2

1) Identify the count of distinct products that the company sells within each category

```
5  select category.categoryname, count(distinct products.productid) as distinct_product_count
6  from category join products on category.categoryid = products.productid
7  group by category.categoryname
```

Data Output | Messages | Notifications

	categoryname character varying	distinct_product_count bigint
1	Baby Care	1
2	Bakery, Cakes & Dairy	1
3	Beauty & Hygiene	1
4	Beverages	1
5	Cleaning & Household	1
6	Eggs, Meat & Fish	1
7	Foodgrains, Oil & Masala	1
8	Fruits & Vegetables	1
9	Gourmet & World Food	1
10	Kitchen, Garden & Pets	1
11	Snacks & Branded Foods	1

2) Identify the average order amount by each CustomerID in each month of Year 2020

```
11 select customerid, extract (month from orderdate) as order_month,
12   avg(total_order_amount) as avg_order_amount
13   from orders
14 where extract (year from orderdate) = 2020
15 group by customerid, extract (month from orderdate);
```

Data Output | Messages | Notifications

	customerid integer	order_month numeric	avg_order_amount double precision
1	57081	5	8034
2	57081	6	9703.8
3	57081	7	5698
4	57081	9	44193
5	57081	10	17184.15
6	57082	5	5401
7	57082	7	68614
8	57082	12	7314
9	57083	1	25112
10	57083	3	19049.55
11	57083	9	19222.2
12	57084	3	21730
13	57084	5	17685
14	57084	10	7069.5
15	57084	12	76
16	57085	5	11122
17	57085	9	4571.1
18	57086	1	22453
19	57086	2	13293

Total rows: 558 of 558 | Query complete 00:00:00.107

3) Identify the Month-Year combinations which had the highest customer acquisition.

```
20  select
21  extract ( month from dateentered ) as acquisition_month,
22  extract (year from dateentered ) as acquisition_year,
23  count(distinct customerid) as customer_count
24  from customers
25  group by acquisition_month, acquisition_year
26  order by  customer_count desc
27  limit 1;
28
```

Data Output | Messages | Notifications

	acquisition_month numeric	acquisition_year numeric	customer_count bigint
1	2	2021	41

4) Identify the most selling ProductID in 2021

```
31  select  productid , sum(quantity) as total_quantity_sold
32  from orderdetails join orders on orderdetails.orderid = orders.orderid
33  where extract(year from orderdate)=2021
34  group by productid
35  order by total_quantity_sold desc
36  limit 1;
37
```

Data Output | Messages | Notifications

	productid integer	total_quantity_sold bigint
1	19591	81

5) Identify which Supplier ID supplied the least number of products

```
39
40  select s.supplierid, count (od.productid) as total_supplied_product
41  from suppliers s left join orderdetails od on s.supplierid=od.supplierid
42  group by s.supplierid
43  order by total_supplied_product asc
44  limit 1;
45
```

Data Output | Messages | Notifications

	supplierid integer	total_supplied_product bigint
1	2	4566

6) Get details of those customers who have ordered for a total amount of more than 7000 during last quarter of Year 21.

```
49
50 select c.customerid, c.firstname, c.lastname, o.orderid, o.orderdate, o.total_order_amount
51 from customers c join orders o on c.customerid= o.customerid
52  where extract(year from o.orderdate)= 2021
53  and extract (quarter from o.orderdate)=4
54  and  o.total_order_amount > 7000
55  order by  o.total_order_amount;
56
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	orderid integer	orderdate date	total_order_amount double precision
1	57605	Malcolm	Julian	7660494	2021-12-28	7013
2	57516	Ewan	Phoenix	7659378	2021-11-09	7026
3	57479	Ibrahim	Taylor	7660383	2021-12-25	7030
4	57451	Ciaran	Arthur	7659334	2021-11-07	7036
5	57443	Felix	Brodie	7659269	2021-11-03	7044
6	57334	Luca	Patricia	7659122	2021-10-27	7087
7	57500	Campbell	Charles	7659078	2021-10-25	7090
8	57580	Cailean	Colby	7660255	2021-12-20	7139
9	57123	Gregory	Perez	7658742	2021-10-04	7152
10	57304	Joshua	Chan	7658964	2021-10-19	7160.56
11	57254	Kathryn	Clark	7659056	2021-10-24	7164
12	57580	Cailean	Colby	7659354	2021-11-08	7180
13	57325	Nathan	Nikodem	7658823	2021-10-10	7184
14	57561	Franciszek	Finn	7659014	2021-10-21	7224.8
15	57408	Austin	Johnny	7659866	2021-12-03	7238

Total rows: 1000 of 1333 | Query complete 00:00:00.085

7) Find the no. of orders fulfilled by Suppliers residing in the same Country as the customer.

```
58 select c.customerid, s.supplierid, count(o.orderid) as fulfilled_orders
59 from customers c join orders o on c.customerid= o.customerid
60 join orderdetails od on o.orderid = od.orderid
61 join suppliers s on od.supplierid=s.supplierid
62 where c.country = s.country
63 group by c.customerid,s.supplierid;
64
```

Data Output | Messages | Notifications

	customerid integer	supplierid integer	fulfilled_orders bigint
1	57204	4	17
2	57204	5	17
3	57213	4	16
4	57213	5	15
5	57217	4	8
6	57217	5	8
7	57218	4	24
8	57218	5	16
9	57219	4	14
10	57219	5	13
11	57228	4	7
12	57228	5	10
13	57235	4	13
14	57235	5	10
15	57246	4	4

Total rows: 80 of 80 | Query complete 00:00:00.108

8) Find out the top 4 best-selling products in each of the categories that are currently active on the Website.

```sql
SELECT
    productID,
    product,
    categoryID,
    sale_price,
    categoryname
FROM (
    SELECT
        p.productID,
        p.product,
        p.categoryID,
        p.sale_price,
        c.categoryname,
        RANK() OVER (PARTITION BY p.categoryID ORDER BY od.quantity DESC) AS ranking
    FROM
        Products p
    JOIN
        OrderDetails od ON p.productID = od.productID
    JOIN
        Category c ON p.categoryID = c.categoryID
    WHERE
        c.activr = 'Yes'
) AS RankedProducts
WHERE
    ranking <= 4;
```

Data Output | Messages | Notifications

	productid integer	product character varying	categoryid integer	sale_price integer	categoryname character varying
1	20672	Paper Napkins - Oval Grey	5001	49	Cleaning & Household
2	9145	Smart Broom - Regular, 39 Sticks	5001	75	Cleaning & Household
3	16662	Fevicreate Make Your Own Solar System	5001	196	Cleaning & Household
4	136	Clean-Up All-Purpose Cleaner with Bleach	5001	425	Cleaning & Household
5	53	Phenyl - Organic Floor Cleaner	5001	320	Cleaning & Household

Total rows: 1000 of 1287 | Query complete 00:00:00.133

9) Find the out the least selling products in each of the categories that are currently active on the website

```sql
SELECT
    productID,
    product,
    categoryID,
    sale_price,
    categoryname,
    ranking
FROM (
    SELECT
        p.productID,
        p.product,
        p.categoryID,
        p.sale_price,
        c.categoryname,
        RANK() OVER (PARTITION BY p.categoryID ORDER BY od.quantity) AS ranking
    FROM Products p JOIN OrderDetails od ON p.productID = od.productID
    JOIN Category c ON p.categoryID = c.categoryID
    WHERE c.activr = 'Yes'
) AS RankedProducts
WHERE ranking = 1;
```

Data Output | Messages | Notifications

	productid integer	product character varying	categoryid integer	sale_price integer	categoryname character varying	ranking bigint
1	11349	Designer Jyot - Green	5001	239	Cleaning & Household	1
2	60	Disinfectant Floor Cleaner Liquid - Pine, Kills 99.9% Germs	5001	169	Cleaning & Household	1
3	10484	Ball Hanging Rice Paper Decorative Lantern - Kandal White, 14 Inch	5001	119	Cleaning & Household	1
4	20613	Facial Tissue - 2 Ply, Soft Pack	5001	135	Cleaning & Household	1
5	5085	Air Effects Air Freshener - Sandalwood	5001	279	Cleaning & Household	1
6	11365	Silver Plated Kalash - Size 1	5001	95	Cleaning & Household	1
7	11209	Tortoise Plate	5001	82	Cleaning & Household	1
8	9208	Grease/Stain/Dirt Cleaning Sponge - Multicolour, H130037MC	5001	135	Cleaning & Household	1

Total rows: 1000 of 1330 | Query complete 00:00:00.177 | Ln 167, Col 19

10) Find the cumulative sum of total orders placed for the year 2020 (solve using both Self Join & Window Function).

```sql
-- using self join

SELECT
    a.orderdate,
    a.total_order_amount,
    SUM(b.total_order_amount) AS cumulative_sum
FROM Orders a
JOIN Orders b ON a.orderdate >= b.orderdate
WHERE extract(YEAR from a.orderdate) = 2020
GROUP BY a.orderdate, a.total_order_amount
ORDER BY a.orderdate;
```

Data Output | Messages | Notifications

	orderdate date	total_order_amount double precision	cumulative_sum double precision
1	2020-01-12	25112	25112
2	2020-01-20	22453	47565
3	2020-02-06	13293	60858
4	2020-02-09	16063	76921
5	2020-02-11	15193	92114
6	2020-02-15	13581	105695
7	2020-02-20	74120	179815
8	2020-02-23	12092	191907

Total rows: 725 of 725 | Query complete 00:00:00.543

```sql
--using  window function

SELECT
    orderdate,
    total_order_amount,SUM(total_order_amount) OVER (ORDER BY orderdate) AS cumulative_sum
FROM Orders
WHERE extract(YEAR from orderdate) = 2020
ORDER BY orderdate;
```

Data Output | Messages | Notifications

	orderdate date	total_order_amount double precision	cumulative_sum double precision
1	2020-01-12	25112	25112
2	2020-01-20	22453	47565
3	2020-02-06	13293	60858
4	2020-02-09	16063	76921
5	2020-02-11	15193	92114
6	2020-02-15	13581	105695
7	2020-02-20	74120	179815
8	2020-02-23	12092	191907
9	2020-02-26	48258	247674
10	2020-02-26	7509	247674
11	2020-03-02	11803.1	259477.1
12	2020-03-05	3851.35	283463.45
13	2020-03-05	20135	283463.45
14	2020-03-06	4837.35	288300.8
15	2020-03-07	10991	299291.8
16	2020-03-12	39769	339060.8

Total rows: 725 of 725 | Query complete 00:00:00.132

11) Find the top 3 Shipper companies in terms of a) Average delivery time for each category for the latest year b) Volume for latest year

```
247  --QN 11 (a)
248
249  SELECT
250      s.companyname,
251      c.categoryname,
252  AVG(o.deliverydate - o.shipdate) AS avg_delivery_time
253  FROM Shippers s JOIN Orders o ON s.shipperid = o.shipperid
254  JOIN OrderDetails od ON o.orderID = od.orderID
255  JOIN Products p ON od.productID = p.productID
256  JOIN Category c ON p.categoryID = c.categoryID
257  WHERE extract(YEAR from o.deliverydate) = 2022
258  -- 2022 is the latest date, we could use current_date() but it will
259  -- be 2024 which is not available in the data sheet.
260  GROUP BY s.companyname, c.categoryname
261  ORDER BY c.categoryname, avg_delivery_time
262  LIMIT 3;  -- To get the top 3 for each category
```

Data Output | Messages | Notifications

	companyname character varying	categoryname character varying	avg_delivery_time numeric
1	Lufthansa Cargo	Baby Care	10.5000000000000000
2	Hapag-Lloyd	Baby Care	10.5000000000000000
3	COSCO(China Ocean Shipping Company)	Baby Care	12.2500000000000000

```
264  --QN 11 (b)
265
266
267  SELECT
268      s.companyname,SUM(od.quantity) AS total_volume
269  FROM Shippers s JOIN Orders o ON s.shipperid = o.shipperid
270  JOIN OrderDetails od ON o.orderID = od.orderID
271  WHERE extract(YEAR from o.deliverydate) = 2022
272  -- 2022 is the latest date, we could use current_date() but it will
273  -- be 2024 which is not available in the data sheet.
274  GROUP BY s.companyname
275  ORDER BY total_volume DESC
276  LIMIT 3;  -- To get the top 3 shippers by volume
277
278
```

Data Output | Messages | Notifications

	companyname character varying	total_volume bigint
1	DTDC	3215
2	ONE(Ocean Network Express)	3169
3	Blue Dart	3048

12) Find the top 25 customers in terms of a) Total no. of orders placed for Year 2021 b) Total Purchase Amount for the Year 2021

```
--QN 12 (a)

SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    COUNT(o.orderID) AS total_orders
FROM Customers c
JOIN Orders o ON c.customerID = o.customerID
WHERE extract(YEAR from o.deliverydate) = 2021
GROUP BY c.customerID, c.firstname, c.lastname
ORDER BY total_orders DESC
LIMIT 25;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	total_orders bigint
1	57334	Luca	Patricia	19
2	57543	Rex	Ellis	17
3	57363	Carson	Jose	16
4	57388	Evan	Robert	16
5	57396	Calvin	Michael	16
6	57486	Cruz	Duncan	16
7	57383	Roman	Hamish	16
8	57456	Cohen	Jayden	16
9	57282	Oliver	Alan	15
10	57330	Luke	Bobby	15
11	57459	Jason	Calvin	15
12	57470	Colton	Ashton	15
13	57431	Zachary	Steven	15
14	57343	Arran	Dorothy	15
15	57455	Caelan	Lochlan	15

Total rows: 25 of 25 | Query complete 00:00:00.111

```
--QN 12  (b)

SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    SUM(p.sale_price * od.quantity) AS total_purchase_amount
FROM Customers c JOIN Orders o ON c.customerID = o.customerID
JOIN OrderDetails od ON o.orderID = od.orderID
JOIN Products p ON od.productID = p.productID
WHERE extract(YEAR from o.deliverydate) = 2021
GROUP BY c.customerID, c.firstname, c.lastname
ORDER BY total_purchase_amount DESC
LIMIT 25;  -- To get the top 25 customers by total purchase amount
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	total_purchase_amount bigint
1	57249	Jacqueline	Fernandez	418230
2	57486	Cruz	Duncan	400891
3	57363	Carson	Jose	381911
4	57396	Calvin	Michael	366397
5	57373	Henry	Rohan	364467
6	57339	Ryan	Lyon	356230
7	57433	Charles	Kade	353917
8	57102	Kevin	Hart	352154
9	57470	Colton	Ashton	343854
10	57334	Luca	Patricia	341316
11	57451	Ciaran	Arthur	338318
12	57392	Lochlan	Ruaridh	328342
13	57347	Dylan	Deborah	324888
14	57427	Mohammad	Ciaran	318426
15	57393	Jayden	Connor	314349

Total rows: 25 of 25 | Query complete 00:00:00.091

13) Find the cumulative average order amount at a monthly level for year 2021 a) Each category b) Each customer

```sql
--QN 13

SELECT
    c.categoryname,
    extract(MONTH from o.orderdate) AS month,
    AVG(SUM(p.sale_price * od.quantity)) OVER (PARTITION BY c.categoryname
    ORDER BY extract(MONTH from o.orderdate)) AS cumulative_avg_order_amount
FROM Category c
JOIN Products p ON c.categoryID = p.categoryID
JOIN OrderDetails od ON p.productID = od.productID
JOIN Orders o ON od.orderID = o.orderID
WHERE extract(YEAR from o.deliverydate) = 2021
Group by c.categoryname, MONTH
ORDER BY c.categoryname, month;
```

Data Output | Messages | Notifications

	categoryname character varying	month numeric	cumulative_avg_order_amount numeric
1	Baby Care	1	97062.000000000000
2	Baby Care	2	127810.000000000000
3	Baby Care	3	120199.333333333333
4	Baby Care	4	139753.500000000000
5	Baby Care	5	143074.600000000000
6	Baby Care	6	149162.333333333333
7	Baby Care	7	170988.142857142857
8	Baby Care	8	175379.500000000000
9	Baby Care	9	195606.000000000000
10	Baby Care	10	230246.900000000000
11	Baby Care	11	244233.454545454545
12	Baby Care	12	238109.416666666667
13	Bakery, Cakes & Dairy	1	32698.000000000000
14	Bakery, Cakes & Dairy	2	29993.500000000000
15	Bakery, Cakes & Dairy	3	42272.000000000000

Total rows: 132 of 132 | Query complete 00:00:00.166

```sql
--QN 13 (b)

SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    extract(MONTH from o.orderdate) AS month,
    AVG(SUM(p.sale_price * od.quantity)) OVER (PARTITION BY c.customerID
 ORDER BY extract(MONTH from o.orderdate)) AS cumulative_avg_order_amount
FROM Customers c JOIN Orders o ON c.customerID = o.customerID
JOIN OrderDetails od ON o.orderID = od.orderID
JOIN Products p ON od.productID = p.productID
WHERE extract(YEAR from o.deliverydate) = 2021
GROUP BY c.customerID, c.firstname, c.lastname, MONTH
ORDER BY c.customerID, month;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	month numeric	cumulative_avg_order_amount numeric
1	57081	James	Smith	3	51979.000000000000
2	57081	James	Smith	7	29830.500000000000
3	57081	James	Smith	11	23082.000000000000
4	57082	Robert	Downey Jr	4	25356.000000000000
5	57082	Robert	Downey Jr	7	20617.000000000000
6	57082	Robert	Downey Jr	9	16474.333333333333
7	57082	Robert	Downey Jr	10	15654.750000000000
8	57082	Robert	Downey Jr	11	14635.600000000000
9	57082	Robert	Downey Jr	12	13415.333333333333
10	57083	John	Williams	1	29820.000000000000
11	57083	John	Williams	3	41924.000000000000
12	57083	John	Williams	4	31029.666666666667
13	57083	John	Williams	8	34013.000000000000
14	57083	John	Williams	9	30673.000000000000

Total rows: 1000 of 2540 | Query complete 00:00:00.210

14) Find the 3-day rolling average for the total purchase amount by each customer.

```sql
SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    o.orderdate,
    AVG(sale_price * quantity) OVER (PARTITION BY c.customerID ORDER
    BY orderdate ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS rolling_avg_purchase_amount
FROM Customers c
JOIN Orders o ON c.customerID = o.customerID
JOIN OrderDetails od ON o.orderID = od.orderID
JOIN Products p ON od.productID = p.productID
ORDER BY customerID, orderdate;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	orderdate date	rolling_avg_purchase_amount numeric
1	57081	James	Smith	2020-05-05	1280.0000000000000000
2	57081	James	Smith	2020-05-05	1245.5000000000000000
3	57081	James	Smith	2020-05-05	888.6666666666666667
4	57081	James	Smith	2020-05-05	628.6666666666666667
5	57081	James	Smith	2020-05-05	858.3333333333333333
6	57081	James	Smith	2020-05-05	1233.3333333333333333
7	57081	James	Smith	2020-05-05	1622.6666666666666667
8	57081	James	Smith	2020-06-10	4583.3333333333333333
9	57081	James	Smith	2020-07-29	4200.0000000000000000
10	57081	James	Smith	2020-07-29	4587.6666666666666667
11	57081	James	Smith	2020-07-29	1899.3333333333333333
12	57081	James	Smith	2020-09-24	7345.6666666666666667
13	57081	James	Smith	2020-09-24	9610.3333333333333333

Total rows: 1000 of 27527 | Query complete 00:00:00.204

15) Get the cumulative sum of total_order_amount for orders placed by each customer ordered by the orderID.

```sql
SELECT
    o.orderID,
    o.customerID,
    c.firstname,
    c.lastname,
    o.orderdate,
    o.total_order_amount,
    SUM(o.total_order_amount) OVER (PARTITION BY o.customerID ORDER BY o.orderID) AS cumulative_sum_total_amount
FROM
    Orders o
JOIN
    Customers c ON o.customerID = c.customerID
ORDER BY
    o.customerID, o.orderID;
```

Data Output | Messages | Notifications

	orderid integer	customerid integer	firstname character varying	lastname character varying	orderdate date	total_order_amount double precision	cumulative_sum_total_amount double precision
1	7655564	57081	James	Smith	2020-05-05	8034	8034
2	7655626	57081	James	Smith	2020-06-10	9703.8	17737.8
3	7655725	57081	James	Smith	2020-07-29	5698	23435.8
4	7655876	57081	James	Smith	2020-09-24	44193	67628.8
5	7655923	57081	James	Smith	2020-10-06	23278.5	90907.3
6	7655984	57081	James	Smith	2020-10-22	11089.8	101997.1
7	7656675	57081	James	Smith	2021-03-27	49380.05	151377.15000000002
8	7657586	57081	James	Smith	2021-07-07	7682	159059.15000000002
9	7659816	57081	James	Smith	2021-11-30	9585	168644.15000000002
10	7655569	57082	Robert	Downey Jr	2020-05-09	5401	5401

Total rows: 1000 of 4999 | Query complete 00:00:00.125

16) Print the cumulative sum of Total_Transaction_Value for each of the months of the year 2020.

```
SELECT
    EXTRACT(MONTH FROM orderdate) AS month,
    SUM(total_order_amount) OVER (ORDER BY EXTRACT(MONTH FROM orderdate)) AS cumulative_sum
FROM Orders
WHERE EXTRACT(YEAR FROM orderdate) = 2020;
```

Data Output | Messages | Notifications

	month numeric	cumulative_sum double precision
1	1	47565
2	1	47565
3	2	247674
4	2	247674
5	2	247674
6	2	247674
7	2	247674
8	2	247674
9	2	247674
10	2	247674
11	3	566645.3
12	3	566645.3
13	3	566645.3
14	3	566645.3
15	3	566645.3
16	3	566645.3
17	3	566645.3
18	3	566645.3
19	3	566645.3
20	3	566645.3

Total rows: 725 of 725 | Query complete 00:00:00.242

17) Print the cumulative average of Total_Transaction_value for each of the quarters of the year 2021.

```
SELECT
    EXTRACT(QUARTER FROM orderdate) AS quarter,
    AVG(total_order_amount) OVER (ORDER BY EXTRACT(QUARTER FROM orderdate))
    AS cumulative_average
FROM Orders
WHERE EXTRACT(YEAR FROM orderdate) = 2021;
```

Data Output | Messages | Notifications

	quarter numeric	cumulative_average double precision
1	1	19000.134031936126
2	1	19000.134031936126
3	1	19000.134031936126
4	1	19000.134031936126
5	1	19000.134031936126
6	1	19000.134031936126
7	1	19000.134031936126
8	1	19000.134031936126
9	1	19000.134031936126
10	1	19000.134031936126
11	1	19000.134031936126
12	1	19000.134031936126
13	1	19000.134031936126

Total rows: 1000 of 4274 | Query complete 00:00:00.176

18) Print the cumulative average of the quantity of products ordered in each quarter of the years 2020 and 2021. (Think about the condition on which you shall be ordering the records before you start calculating the cumulative average).

```
SELECT
    EXTRACT(QUARTER FROM o.orderdate) AS quarter,
    AVG(od.quantity) OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
    EXTRACT(QUARTER FROM o.orderdate) ORDER BY o.orderdate)
    AS cumulative_average
FROM Orders o
JOIN OrderDetails od ON o.orderID = od.orderID
WHERE EXTRACT(YEAR FROM o.orderdate) IN (2020, 2021);
```

Data Output | Messages | Notifications

	quarter numeric	cumulative_average numeric
1	1	8.3750000000000000
2	1	8.3750000000000000
3	1	8.3750000000000000
4	1	8.3750000000000000
5	1	8.3750000000000000
6	1	8.3750000000000000
7	1	8.3750000000000000
8	1	8.3750000000000000
9	1	10.0000000000000000
10	1	10.0000000000000000
11	1	10.0000000000000000
12	1	10.0000000000000000
13	1	10.0000000000000000
14	1	10.3888888888888889

Total rows: 1000 of 27527 | Query complete 00:00:00.266

19) Identify and print the details of products that were the second most ordered in terms of total quantity for each month of each year.

```
SELECT
    productID,
    product,
    EXTRACT(YEAR FROM orderdate) AS order_year,
    EXTRACT(MONTH FROM orderdate) AS order_month,
    total_quantity
FROM (
    SELECT
        p.productID,
        p.product,
        o.orderdate,
        SUM(od.quantity) AS total_quantity,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate), p.productID
        ORDER BY SUM(od.quantity) DESC) AS ranking
    FROM Products p
    JOIN OrderDetails od ON p.productID = od.productID
    JOIN Orders o ON od.orderID = o.orderID
    GROUP BY p.productID, p.product, o.orderdate
) AS RankedProducts
WHERE ranking = 2;
```

Data Output | Messages | Notifications

	productid integer	product character varying	order_year numeric	order_month numeric	total_quantity bigint
1	6011	Anti-Dandruff Shampoo	2020	5	3
2	6787	Nirgundi Oil - Joint Guard	2020	5	4
3	1516	Hand Wash White Tea & Ginger - Anti Bacterial	2020	6	10
4	6070	Red Onion Hair Conditioner	2020	6	2
5	11124	Achar - Sweet & Sour Berry	2020	6	3
6	9043	Pro-Styler	2020	7	2
7	10454	Organic Coriander Powder	2020	7	10

Total rows: 1000 of 1121 | Query complete 00:00:00.851 | Ln 51, Col 19

20) Identify and print the details of products that were the that generated the 5th most revenue for each quarter of each year.

```sql
SELECT
    productID,
    product,
    EXTRACT(YEAR FROM orderdate) AS order_year,
    EXTRACT(QUARTER FROM orderdate) AS order_quarter,
    total_revenue
FROM (
    SELECT
        p.productID,
        p.product,
        o.orderdate,
        SUM(od.quantity * p.sale_price) AS total_revenue,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(QUARTER FROM o.orderdate)
        ORDER BY SUM(od.quantity * p.sale_price) DESC) AS ranking
    FROM Products p
    JOIN OrderDetails od ON p.productID = od.productID
    JOIN Orders o ON od.orderID = o.orderID
    GROUP BY p.productID, p.product, o.orderdate
) AS RankedProducts
WHERE
    ranking = 5;
```

Data Output | Messages | Notifications

	productid integer	product character varying	order_year numeric	order_quarter numeric	total_revenue bigint
1	1209	Proactive Protection Gift Set	2020	1	14136
2	4855	1981 Eau De Toilette Pour Homme	2020	2	46800
3	9426	Smart Spin Mop - With Double Bucket	2020	3	44910
4	18568	Stainless Steel Contura Pressure Cooker SSC50	2020	4	58548
5	4767	Perfume - EDP for Woman	2021	1	63000
6	10792	Pro Expert Nutrition Large Breed Puppy (3-18 Months) Dry Dog Food	2021	2	71680
7	18796	Stainless Steel Induction Compatible Pressure Cooker - Silver HSS	2021	3	67186

Total rows: 8 of 8 Query complete 00:00:00.240

21) Print the details of total transactions value made on each day whose info is available in thedatabase.

```sql
SELECT
    orderdate,
    SUM(od.quantity * p.sale_price) AS total_transaction_value
FROM
    Orders o
JOIN
    OrderDetails od ON o.orderID = od.orderID
JOIN
    Products p ON od.productID = p.productID
GROUP BY
    orderdate
ORDER BY
    orderdate;
```

Data Output | Messages | Notifications

	orderdate date	total_transaction_value bigint
1	2020-01-12	25112
2	2020-01-20	22453
3	2020-02-06	13293
4	2020-02-09	16063
5	2020-02-11	15193
6	2020-02-15	13581
7	2020-02-20	74120
8	2020-02-23	12092
9	2020-02-26	55767
10	2020-03-02	13886

Total rows: 634 of 634 Query complete 00:00:00.139

22) Along with the output of the above question, print a new column which provides a 5 day rolling average of the total transaction value made each day. (Meaning consider current row and 4 preceding rows).

```
97  SELECT
98      orderdate,
99      SUM(od.quantity * p.sale_price) AS total_transaction_value,
100     AVG(SUM(od.quantity * p.sale_price)) OVER
101     (ORDER BY orderdate ROWS BETWEEN 4 PRECEDING AND CURRENT ROW)
102     AS rolling_avg
103 FROM Orders o
104 JOIN OrderDetails od ON o.orderID = od.orderID
105 JOIN Products p ON od.productID = p.productID
106 GROUP BY
107     orderdate
108 ORDER BY
109     orderdate;
110
111
```

Data Output | Messages | Notifications

	orderdate date	total_transaction_value bigint	rolling_avg numeric
1	2020-01-12	25112	25112.000000000000
2	2020-01-20	22453	23782.500000000000
3	2020-02-06	13293	20286.000000000000
4	2020-02-09	16063	19230.250000000000
5	2020-02-11	15193	18422.800000000000
6	2020-02-15	13581	16116.600000000000
7	2020-02-20	74120	26450.000000000000
8	2020-02-23	12092	26209.800000000000
9	2020-02-26	55767	34150.600000000000
10	2020-03-02	13886	33889.200000000000

Total rows: 634 of 634 Query complete 00:00:00.162

23) Print the year, month, productID, Product_Name, Total_Quantity, Total_Revenue for those products that were ordered the most number of times in terms of total_quantity for each year and quarter combination.

```
SELECT
    order_year,order_quarter,productID,
    product_name,total_quantity,total_revenue
FROM (SELECT p.productID,
        p.product AS product_name,
        EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(QUARTER FROM o.orderdate) AS order_quarter,
        SUM(od.quantity) AS total_quantity,
        SUM(od.quantity * p.sale_price) AS total_revenue,
        ROW_NUMBER() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(QUARTER FROM o.orderdate)
        ORDER BY SUM(od.quantity) DESC) AS row_num
    FROM Products p
    JOIN OrderDetails od ON p.productID = od.productID
    JOIN Orders o ON od.orderID = o.orderID
    GROUP BY p.productID, product_name, EXTRACT(YEAR FROM o.orderdate),
    EXTRACT(QUARTER FROM o.orderdate)
    ) AS RankedProducts
WHERE row_num = 1;
```

Data Output | Messages | Notifications

	order_year numeric	order_quarter numeric	productid integer	product_name character varying	total_quantity bigint	total_revenue bigint
1	2020	1	2668	Rolled Oats - Gluten Free	22	3300
2	2020	2	1516	Hand Wash White Tea & Ginger - Anti Bacterial	29	10121
3	2020	3	9782	Deep Action Lemon Mint Anti Germ Gel Toothpaste	40	3800
4	2020	4	18291	Himalayan Pink Rock Salt Premium - Fine Grain	36	2700
5	2021	1	17188	Small Picture Hanging Strips	41	7339
6	2021	2	19061	100% Melamine Dinner Set - Red	51	33609
7	2021	3	3558	Dark Chocolate - Nutella Hazelnut	53	15635
8	2021	4	19591	Cheese - Pepper Jack, Block	81	40986

Total rows: 8 of 8 | Query complete 00:00:00.217

24) Using the result set of the above question, compare and find the maximum Total_Quantity for each row, comparing each row values with 2 Previous months and 1 following month.

```sql
SELECT
    order_year,order_quarter,productID,product_name,
    total_quantity,total_revenue,total_quantity AS max_quantity,
    LAG(total_quantity, 1) OVER (PARTITION BY productID
    ORDER BY order_year, order_quarter) AS prev_month_quantity,
    LAG(total_quantity, 2) OVER (PARTITION BY productID
    ORDER BY order_year, order_quarter) AS prev_2_months_quantity,
    LEAD(total_quantity, 1) OVER (PARTITION BY productID
    ORDER BY order_year, order_quarter) AS next_month_quantity
FROM (  SELECT p.productID,p.product AS product_name,
        EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(QUARTER FROM o.orderdate) AS order_quarter,
        SUM(od.quantity) AS total_quantity,
        SUM(od.quantity * p.sale_price) AS total_revenue,
        ROW_NUMBER() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(QUARTER FROM o.orderdate) ORDER BY SUM(od.quantity) DESC)
      AS row_num
    FROM Products p
    JOIN OrderDetails od ON p.productID = od.productID
    JOIN Orders o ON od.orderID = o.orderID
    GROUP BY p.productID, product_name, EXTRACT(YEAR FROM o.orderdate),
    EXTRACT(QUARTER FROM o.orderdate)
) AS RankedProducts;
```

Data Output | Messages | Notifications

	order_year numeric	order_quarter numeric	productid integer	product_name character varying	total_quantity bigint	total_revenue bigint	max_quantity bigint
1	2020	2	1	Original Disinfectant Toilet Cleaner Liquid	1	489	1
2	2020	3	1	Original Disinfectant Toilet Cleaner Liquid	16	7824	16
3	2021	4	1	Original Disinfectant Toilet Cleaner Liquid	10	4890	10
4	2021	3	4	Harpic Toilet Cleaner Liquid - Original 1 L + Lizol Floor Cleaner, Citrus - 2L	6	2772	6
5	2021	3	5	Disinfectant Bathroom Cleaner Liquid - Lemon	18	5382	18
6	2021	1	6	Super Saver Pack Toilet Cleaner Original, 500ml + Bathroom Cleaner Lemon, 500ml	12	1944	12

Successfully run. Total query runtime: 253 msec. 24082 rows affected.

Total rows: 1000 of 24082 | Query complete 00:00:00.253 | Ln 157, Col 21

25) Print the Year, Month, PaymentID, PaymentType, Total_Transaction_Value for those PaymentTypes that had the highest transaction values for for each year and month combination.

```sql
SELECT order_year,order_month,PaymentID,PaymentType,Total_Transaction_Value
FROM (SELECT EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(MONTH FROM o.orderdate) AS order_month, p.PaymentID,p.PaymentType,
        SUM(od.quantity * pro.sale_price) AS Total_Transaction_Value,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate), EXTRACT(MONTH FROM o.orderdate)
        ORDER BY SUM(od.quantity * pro.sale_price) DESC) AS rnk
    FROM Payments p JOIN Orders o ON p.PaymentID = o.PaymentID
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    JOIN Products pro ON od.ProductID = pro.ProductID
    GROUP BY EXTRACT(YEAR FROM o.orderdate),EXTRACT(MONTH FROM o.orderdate),
        p.PaymentID,p.PaymentType
) AS RankedPayments
WHERE rnk = 1
ORDER BY order_year,order_month,PaymentID;
```

Data Output | Messages | Notifications

	order_year numeric	order_month numeric	paymentid integer	paymenttype character varying	total_transaction_value bigint
1	2020	1	2	POD	25112
2	2020	2	4	Credit Card	180508
3	2020	3	4	Credit Card	133022
4	2020	4	4	Credit Card	248048
5	2020	5	4	Credit Card	388640
6	2020	6	4	Credit Card	736537
7	2020	7	4	Credit Card	472484
8	2020	8	4	Credit Card	563409
9	2020	9	4	Credit Card	1142230
10	2020	10	4	Credit Card	908549
11	2020	11	4	Credit Card	1171625
12	2020	12	4	Credit Card	1152974
13	2021	1	4	Credit Card	1352078

Total rows: 24 of 24 Query complete 00:00:00.156

26) Using the result set of the above question, calculate the average total_transaction_value for the previous month, current month and 1 following month.

```sql
SELECT
order_year,order_month,AVG(Total_Transaction_Value) AS avg_transaction_value
FROM (SELECT EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(MONTH FROM o.orderdate) AS order_month,
        p.PaymentID,p.PaymentType,
        SUM(od.quantity * pro.sale_price) AS Total_Transaction_Value,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate)
        ORDER BY SUM(od.quantity * pro.sale_price) DESC) AS rnk
    FROM Payments p JOIN Orders o ON p.PaymentID = o.PaymentID
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    JOIN Products pro ON od.ProductID = pro.ProductID
    GROUP BY EXTRACT(YEAR FROM o.orderdate),
             EXTRACT(MONTH FROM o.orderdate),p.PaymentID,p.PaymentType
) AS RankedPayments
WHERE rnk = 1
AND (
    (order_year, order_month) = (EXTRACT(YEAR FROM CURRENT_DATE), EXTRACT(MONTH FROM CURRENT_DATE)) OR
    (order_year = EXTRACT(YEAR FROM CURRENT_DATE) AND order_month = EXTRACT(MONTH FROM CURRENT_DATE) - 1) OR
    (order_year = EXTRACT(YEAR FROM CURRENT_DATE) AND order_month = EXTRACT(MONTH FROM CURRENT_DATE) + 1)
-- as we know that current date belongs to 2024 , so we will not find any data of 2024
-- but the query will be this if we use current_date()
) GROUP BY order_year,order_month;
```

Data Output | Messages | Notifications

order_year numeric	order_month numeric	avg_transaction_value numeric

(current_date belongs to 2024, but we don't have any data of 2024)

```sql
SELECT
order_year,order_month,AVG(Total_Transaction_Value) AS avg_transaction_value
FROM (SELECT EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(MONTH FROM o.orderdate) AS order_month,
        p.PaymentID,p.PaymentType,
        SUM(od.quantity * pro.sale_price) AS Total_Transaction_Value,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate)
        ORDER BY SUM(od.quantity * pro.sale_price) DESC) AS rnk
    FROM Payments p JOIN Orders o ON p.PaymentID = o.PaymentID
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    JOIN Products pro ON od.ProductID = pro.ProductID
    GROUP BY EXTRACT(YEAR FROM o.orderdate),
             EXTRACT(MONTH FROM o.orderdate),p.PaymentID,p.PaymentType
) AS RankedPayments
WHERE rnk = 1
AND (
    (order_year, order_month) = (EXTRACT(YEAR FROM CURRENT_DATE), EXTRACT(MONTH FROM CURRENT_DATE)) OR
    (order_year = EXTRACT(YEAR FROM CURRENT_DATE) AND order_month = EXTRACT(MONTH FROM CURRENT_DATE) - 1) OR
    (order_year = EXTRACT(YEAR FROM CURRENT_DATE) AND order_month = EXTRACT(MONTH FROM CURRENT_DATE) + 1)
-- as we know that current date belongs to 2024 , so we will not find any data of 2024
-- but the query will be this if we use current_date()
) GROUP BY order_year,order_month;
```

Data Output | Messages | Notifications

	order_year numeric	order_month numeric	paymentid integer	paymenttype character varying	total_transaction_value bigint	rnk bigint
1	2020	1	2	POD	25112	1
2	2020	1	3	PayPal	22453	2
3	2020	2	4	Credit Card	180508	1
4	2020	2	6	Net banking	12092	2
5	2020	2	5	Wallet	7509	3
6	2020	3	4	Credit Card	133022	1
7	2020	3	3	PayPal	90798	2

Total rows: 131 of 131 | Query complete 00:00:00.182

(just for checking a part of query)

```sql
SELECT order_year,order_month,
    AVG(Total_Transaction_Value) AS avg_transaction_value
FROM ( SELECT
        EXTRACT(YEAR FROM o.orderdate) AS order_year,
        EXTRACT(MONTH FROM o.orderdate) AS order_month,
        p.PaymentID,p.PaymentType,
        SUM(od.quantity * pro.sale_price) AS Total_Transaction_Value,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate) ORDER BY
        SUM(od.quantity * pro.sale_price) DESC) AS rnk
    FROM Payments p JOIN Orders o ON p.PaymentID = o.PaymentID
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    JOIN Products pro ON od.ProductID = pro.ProductID
    WHERE o.orderdate >= '2021-01-01' AND o.orderdate < '2022-01-01'
    GROUP BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate),p.PaymentID, p.PaymentType
) AS RankedPayments
WHERE rnk = 1  AND (
    (order_year, order_month) = (2021, 1) OR
    (order_year = 2021 AND order_month = 12) OR
    (order_year = 2022 AND order_month = 1)
                    )   GROUP BY order_year,order_month;
-- if we use 2021 as our current year it may look like this
-- (as an example)
```

Data Output | Messages | Notifications

	order_year numeric	order_month numeric	avg_transaction_value numeric
1	2021	1	1352078.000000000000
2	2021	12	6654555.000000000000

(if we use 2021 instead of using current year , it will give this result)

27) Print the details of the immediately previous order along with the current order details from the orders table.

```
SELECT
    current_order.orderID AS current_order_id,
    current_order.customerID AS current_customer_id,
    current_order.orderdate AS current_order_date,
    previous_order.orderID AS previous_order_id,
    previous_order.customerID AS previous_customer_id,
    previous_order.orderdate AS previous_order_date
FROM
    Orders AS current_order
LEFT JOIN
    Orders AS previous_order ON current_order.orderID = previous_order.orderID + 1
ORDER BY
    current_order.orderdate;
```

Data Output | Messages | Notifications

	current_order_id integer	current_customer_id integer	current_order_date date	previous_order_id integer	previous_customer_id integer	previous_order_date date
1	7655500	57083	2020-01-12	[null]	[null]	[null]
2	7655501	57086	2020-01-20	7655500	57083	2020-01-12
3	7655502	57086	2020-02-06	7655501	57086	2020-01-20
4	7655503	57088	2020-02-09	7655502	57086	2020-02-06
5	7655504	57090	2020-02-11	7655503	57088	2020-02-09
6	7655505	57094	2020-02-15	7655504	57090	2020-02-11
7	7655506	57092	2020-02-20	7655505	57094	2020-02-15
8	7655507	57095	2020-02-23	7655506	57092	2020-02-20
9	7655508	57105	2020-02-26	7655507	57095	2020-02-23
10	7655509	57095	2020-02-26	7655508	57105	2020-02-26

Total rows: 1000 of 4999 Query complete 00:00:00.125

28) Print CustomerID, FirstName, LastName, OrderId, OrderDate, Previous_Order_Id, Previous_Order_date.

```
SELECT
    C.CustomerID,
    C.FirstName,
    C.LastName,
    O1.OrderID,
    O1.OrderDate,
    O2.OrderID AS Previous_Order_Id,
    O2.OrderDate AS Previous_Order_date
FROM
    Customers C
JOIN
    Orders O1 ON C.CustomerID = O1.CustomerID
LEFT JOIN
    Orders O2 ON C.CustomerID = O2.CustomerID AND O1.OrderDate > O2.OrderDate
ORDER BY
    C.CustomerID, O1.OrderDate;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	orderid integer	orderdate date	previous_order_id integer	previous_order_date date
1	57081	James	Smith	7655564	2020-05-05	[null]	[null]
2	57081	James	Smith	7655626	2020-06-10	7655564	2020-05-05
3	57081	James	Smith	7655725	2020-07-29	7655564	2020-05-05
4	57081	James	Smith	7655725	2020-07-29	7655626	2020-06-10
5	57081	James	Smith	7655876	2020-09-24	7655626	2020-06-10
6	57081	James	Smith	7655876	2020-09-24	7655564	2020-05-05
7	57081	James	Smith	7655876	2020-09-24	7655725	2020-07-29
8	57081	James	Smith	7655923	2020-10-06	7655626	2020-06-10
9	57081	James	Smith	7655923	2020-10-06	7655564	2020-05-05
10	57081	James	Smith	7655923	2020-10-06	7655725	2020-07-29

Total rows: 1000 of 24198 Query complete 00:00:00.156

29) Identify the top 20 products sold in terms of total revenue (Total Revenue = Quantity * Sale_Price).

```
265  SELECT P.productID,P.product,SUM(OD.quantity * P.sale_price) AS Total_Revenue
266  FROM Products P JOIN OrderDetails OD ON P.productID = OD.productID
267  GROUP BY P.productID, P.product
268  ORDER BY Total_Revenue DESC
269  LIMIT 20;
```

Data Output | Messages | Notifications

	productid integer	product character varying	total_revenue bigint
1	18568	Stainless Steel Contura Pressure Cooker SSC50	192864
2	12576	Raw Manuka Honey K Factor - 16+	190800
3	4711	AQVA Divina Body Mist	182000
4	4817	Voyage Sport Eau De Toilette	164736
5	4259	Man In Black Eau De Parfum	164700
6	18730	Futura Hard Anodised Induction Compatible Pressure Cooker - Silver, IFP...	156354
7	10989	U-Clip Clipper	156000
8	9616	Olive Oil - Refined Pomace Mild	149566
9	7187	Whey Protein Powder	147400
10	19320	Lily Ville Square Medium Plate Sets	139359
11	11012	Bravura Clipper	137500
12	20295	Quantum Max Dishwasher Tablets Regular	134139
13	22002	Battery Operator Lady Shaver	130910
14	4624	1 Million Prive EDP	127600
15	833	Baby Bottle & Food Warmer Set	125362
16	4842	Infinite Rush Eau De Toilette	122625
17	16823	Stainless Steel Dinner Set - Apple Shape, Laser + Hammered	119970
18	4730	Black Edition Eau De Parfum For Men	112860
19	4855	1981 Eau De Toilette Pour Homme	112320
20	4768	Deodorant Body Spray - Cologne	111000

Total rows: 20 of 20 | Query complete 00:00:00.152

30) Identify the top 12 products in terms of Total Quantity

```
273  SELECT
274      P.productID,
275      P.product,
276      SUM(OD.quantity) AS Total_Quantity
277  FROM Products P
278  JOIN OrderDetails OD ON P.productID = OD.productID
279  GROUP BY
280      P.productID, P.product
281  ORDER BY
282      Total_Quantity DESC
283  LIMIT 12;
```

Data Output | Messages | Notifications

	productid integer	product character varying	total_quantity bigint
1	17639	Plain Container Jumbo Combo Set - Pink	88
2	10872	Chicken & Egg Adult Dog Food	85
3	16921	Copper Steel Water Jug With Glass Set	83
4	19591	Cheese - Pepper Jack, Block	81
5	21056	Tea Bags - Lemon	81
6	10113	Organic - Imli Powder Dehydrated	80
7	12218	Classic Sulphurless Sugar/Sakkare	77
8	3838	Butterscotch Nuts	77
9	6411	Antiseptic - Disinfectant Liquid	76
10	8656	Organic Black Pepper/Kari Menasu	76
11	10745	DentastixÃ Dog Treat Oral Care For Adult Large Breed (25 kg+), 7 Stic...	76
12	19417	Bonechina Milk Mug - Blue, Pisces Print	76

Total rows: 12 of 12 | Query complete 00:00:00.126

31) Create a Year on Year Analysis in which you print the total transaction amount for each quarter of 2020, then print the total transaction amount for each quarter of 2021 in 2 new columns (Columns to be printed: Year, Quarter, Total Transaction Amount, Next_Year, Next_Year_Quarter, Next_Year_Total_Transaction_Amount).

```
--QN 31

SELECT
    EXTRACT(YEAR FROM orderdate) AS Year,
    EXTRACT(QUARTER FROM orderdate) AS Quarter,
    SUM(quantity * sale_price) AS Total_Transaction_Amount,
    LEAD(EXTRACT(YEAR FROM orderdate)) OVER (ORDER BY EXTRACT(YEAR FROM orderdate),
                                EXTRACT(QUARTER FROM orderdate))AS Next_Year,
    LEAD(EXTRACT(QUARTER FROM orderdate)) OVER (ORDER BY
EXTRACT(YEAR FROM orderdate), EXTRACT(QUARTER FROM orderdate)) AS Next_Year_Quarter,
    LEAD(SUM(quantity * sale_price)) OVER (ORDER BY EXTRACT(YEAR FROM orderdate),
            EXTRACT(QUARTER FROM orderdate)) AS Next_Year_Total_Transaction_Amount
FROM Orders O JOIN OrderDetails OD ON O.orderID = OD.orderID
JOIN Products P ON OD.productID = P.productID
WHERE EXTRACT(YEAR FROM orderdate) IN (2020, 2021)
GROUP BY EXTRACT(YEAR FROM orderdate), EXTRACT(QUARTER FROM orderdate)
ORDER BY EXTRACT(YEAR FROM orderdate), EXTRACT(QUARTER FROM orderdate);
```

Data Output | Messages | Notifications

	year numeric	quarter numeric	total_transaction_amount bigint	next_year numeric	next_year_quarter numeric	next_year_total_transaction_amount bigint
1	2020	1	580265	2020	2	2745049
2	2020	2	2745049	2020	3	4177774
3	2020	3	4177774	2020	4	6323558
4	2020	4	6323558	2021	1	9767160
5	2021	1	9767160	2021	2	14707363
6	2021	2	14707363	2021	3	21456615
7	2021	3	21456615	2021	4	33737832
8	2021	4	33737832	[null]	[null]	[null]

Total rows: 8 of 8 Query complete 00:00:00.108

32) Find the top 3 Shipper companies in terms of Average delivery time for each category for the latest year.

```
SELECT categoryname, companyname, avg_delivery_time
FROM ( SELECT c.categoryname, s.companyname,
        AVG(o.deliverydate - o.shipdate) AS avg_delivery_time,
        ROW_NUMBER() OVER (PARTITION BY c.categoryname
        ORDER BY AVG(o.deliverydate - o.shipdate) ASC) AS rank_within_category
    FROM Orders o
    JOIN Shippers s ON o.shipperid = s.shipperid
    JOIN OrderDetails od ON o.orderID = od.orderID
    JOIN Products p ON od.productID = p.productID
    JOIN Category c ON p.categoryID = c.categoryID
    WHERE  extract (year from o.deliverydate ) = 2022
-- 2022 is taken as the last year, data of 2024 is not in data sheet
    GROUP BY  c.categoryname, s.companyname
) AS RankedShipperDelays
WHERE rank_within_category <= 3;
```

Data Output | Messages | Notifications

	categoryname character varying	companyname character varying	avg_delivery_time numeric
1	Baby Care	Hapag-Lloyd	10.5000000000000000
2	Baby Care	Lufthansa Cargo	10.5000000000000000
3	Baby Care	COSCO(China Ocean Shipping Company)	12.2500000000000000
4	Bakery, Cakes & Dairy	Lufthansa Cargo	11.5000000000000000
5	Bakery, Cakes & Dairy	ONE(Ocean Network Express)	11.6666666666666667
6	Bakery, Cakes & Dairy	Delhivery	12.0000000000000000
7	Beauty & Hygiene	Delhivery	11.7419354838709677
8	Beauty & Hygiene	Hapag-Lloyd	11.7948717948717949
9	Beauty & Hygiene	Fed Ex	12.0000000000000000
10	Beverages	Delhivery	11.2000000000000000
11	Beverages	ONE(Ocean Network Express)	11.8888888888888889

Total rows: 33 of 33 | Query complete 00:00:00.123

12	Beverages	Lufthansa Cargo	12.5000000000000000
13	Cleaning & Household	Fed Ex	11.3157894736842105
14	Cleaning & Household	Hapag-Lloyd	12.6666666666666667
15	Cleaning & Household	DTDC	12.8000000000000000
16	Eggs, Meat & Fish	Fed Ex	4.0000000000000000
17	Eggs, Meat & Fish	Delhivery	9.5000000000000000
18	Eggs, Meat & Fish	DTDC	10.8000000000000000
19	Foodgrains, Oil & Masala	Fed Ex	9.6363636363636364
20	Foodgrains, Oil & Masala	ONE(Ocean Network Express)	10.9393939393939394
21	Foodgrains, Oil & Masala	Delhivery	11.2272727272727273
22	Fruits & Vegetables	DTDC	7.6666666666666667
23	Fruits & Vegetables	Hapag-Lloyd	10.0000000000000000
24	Fruits & Vegetables	COSCO(China Ocean Shipping Company)	14.0000000000000000
25	Gourmet & World Food	Fed Ex	12.1538461538461538
26	Gourmet & World Food	Hapag-Lloyd	12.5588235294117647
27	Gourmet & World Food	Delhivery	12.6750000000000000
28	Kitchen, Garden & Pets	Fed Ex	11.2903225806451613
29	Kitchen, Garden & Pets	Lufthansa Cargo	11.6666666666666667
30	Kitchen, Garden & Pets	Delhivery	12.0312500000000000
31	Snacks & Branded Foods	Fed Ex	10.5294117647058824
32	Snacks & Branded Foods	ONE(Ocean Network Express)	11.2000000000000000
33	Snacks & Branded Foods	Lufthansa Cargo	12.1666666666666667

Total rows: 33 of 33 | Query complete 00:00:00.123

33) Find the top 3 Shipper companies in terms of Volume for latest year.

```sql
SELECT
    s.companyname,
    COUNT(o.orderID) AS order_volume
FROM
    Shippers s
JOIN
    Orders o ON s.shipperid = o.shipperid
WHERE
    extract (year from o.deliverydate )=2022
GROUP BY
    s.companyname
ORDER BY
    order_volume desc
LIMIT 3;
-- 2022 is taken as the last year
```

Data Output | Messages | Notifications

	companyname character varying	order_volume bigint
1	ONE(Ocean Network Express)	57
2	DTDC	52
3	Blue Dart	49

34) Identify the number of orders spent by each customer, then divide them into 3 buckets, give the 3 buckets the tags: Shopping Freak for bucket-1, Regular Customer for bucket-2, Occasional Customer for bucket-3.

```sql
SELECT
    customerID,
    COUNT(DISTINCT orderID) AS order_count,
    CASE
        WHEN COUNT(DISTINCT orderID) >= 10 THEN 'Shopping Freak'
        WHEN COUNT(DISTINCT orderID) >= 5 THEN 'Regular Customer'
        ELSE 'Occasional Customer'
    END AS customer_category
FROM
    Orders
GROUP BY
    customerID;
```

Data Output | Messages | Notifications

	customerid integer	order_count bigint	customer_category text
1	57081	9	Regular Customer
2	57082	9	Regular Customer
3	57083	13	Shopping Freak
4	57084	13	Shopping Freak
5	57085	6	Regular Customer
6	57086	10	Shopping Freak
7	57087	10	Shopping Freak
8	57088	12	Shopping Freak
9	57089	12	Shopping Freak
10	57090	12	Shopping Freak
11	57091	11	Shopping Freak
12	57092	14	Shopping Freak
13	57093	10	Shopping Freak
14	57094	10	Shopping Freak

Total rows: 525 of 525 Query complete 00:00:00.151

35) Find out the least selling products in each of the categories (in the Categories that are currently active on the website).

```
SELECT
    c.categoryID,
    c.categoryname,
    p.productID,
    p.product,
    COALESCE(SUM(od.quantity), 0) AS total_quantity_sold
FROM Category c
JOIN Products p ON c.categoryID = p.categoryID
LEFT JOIN OrderDetails od ON p.productID = od.productID
LEFT JOIN Orders o ON od.orderID = o.orderID
WHERE c.activr = 'Yes'
GROUP BY c.categoryID,c.categoryname, p.product,p.productID
ORDER BY c.categoryID, total_quantity_sold;
```

Data Output | Messages | Notifications

	categoryid integer	categoryname character varying	productid integer	product character varying
1	5001	Cleaning & Household	16430	Notebook - Ruled, Small, Single Line, Soft Bound
2	5001	Cleaning & Household	173	Disinfectant Surface & Floor Cleaner Liquid - Floral
3	5001	Cleaning & Household	9409	Mop - Round With Pvs Handle
4	5001	Cleaning & Household	67	Black Phenyl
5	5001	Cleaning & Household	9137	T - Mop Refill
6	5001	Cleaning & Household	2625	Car Freshener Gel, Refreshing Lemon
7	5001	Cleaning & Household	1790	Royce Premium High Quality Pedal Plastic Dustbin / Garbage Bin - Green
8	5001	Cleaning & Household	1912	Combs Derby Deluxe
9	5001	Cleaning & Household	1892	Washing Net Bag - For Under Garments/Lingerie, BB085
10	5001	Cleaning & Household	1770	Colour Metal Dustbin / Garbage Bin - Mesh
11	5001	Cleaning & Household	4983	aer Home Air Freshener Spray - Fresh Lush Green
12	5001	Cleaning & Household	16408	Multi Purpose Hand Tool Set

Total rows: 1000 of 21249 | Query complete 00:00:00.215

36) Print the details of the 10th most ordered products in each month of each year.

```
SELECT
    year, month, productID, product, total_quantity_ordered
FROM (
    SELECT
        EXTRACT(YEAR FROM o.orderdate) AS year,
        EXTRACT(MONTH FROM o.orderdate) AS month,
        p.productID,
        p.product,
        SUM(od.quantity) AS total_quantity_ordered,
        DENSE_RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
        EXTRACT(MONTH FROM o.orderdate) ORDER BY SUM(od.quantity) DESC) AS rank
    FROM  Orders o JOIN OrderDetails od ON o.orderID = od.orderID
    JOIN Products p ON od.productID = p.productID
    GROUP BY year, month, p.productID, p.product
) AS ranked_products
WHERE  rank = 10
ORDER BY year, month, productID;
```

Data Output | Messages | Notifications

	year numeric	month numeric	productid integer	product character varying	total_quantity_ordered bigint
1	2020	1	12743	Sauce - Sweet & Sour	1
2	2020	2	7689	Bamboo Wood Square Basket - For Bread,Fruit or Flowers, 8x8 Inch	9
3	2020	2	10252	Organic - Rasam Powder	9
4	2020	2	19288	Chai/Opalware Coffee Cup & Saucer Set - Regular, Radiant Curves	9
5	2020	3	4512	ShotÂ Body Spray - Maxx, Trend	11
6	2020	3	5292	House Party - Non Veg	11
7	2020	3	6612	Cucumber Water - Cleanser & Toner, For Men & Women	11
8	2020	3	7949	Tamarind/Hunisehannu/Imli - Seedless	11
9	2020	3	11733	Organic Rice Dosa Ready Mix	11

Total rows: 365 of 365 | Query complete 00:00:00.190

37) Rank the customers based on the date on which their details were entered. The oldest entered customer will get rank 1 and so on.

```sql
SELECT
    customerID,
    firstname,
    lastname,
    date_of_birth,
    city,
    state,
    country,
    postalcode,
    phone,
    email,
    dateentered,
    RANK() OVER (ORDER BY dateentered) AS customer_rank
FROM
    Customers
ORDER BY
    customer_rank;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character vary
1	57081	James	Smith	1987-03-26	New York	New York	United States	280862	9638483934
2	57082	Robert	Downey Jr	1973-05-24	New York	New York	United States	376573	6588282115
3	57083	John	Williams	1990-04-14	Chicago	Illinois	United States	485629	7641021429
4	57084	Michael	Johnson	1953-03-25	Brisbane	Queensland	Australia	260866	7354232181
5	57085	Steve	Williams	1971-04-24	Bremen	Bremen	Germany	740338	6285552036
6	57086	David	Beckham	1965-10-18	Villach	Carinthia	Austria	621492	7756781677
7	57087	Richard	Brown	1958-12-16	San Antonio	Texas	United States	110945	8420993031
8	57088	Joseph	James	1967-08-27	Amsterdam	North Holland	Netherlands	186896	6691466381
9	57089	Thomas	Jones	1950-04-30	Dallas	Texas	United States	174080	8024463594
10	57090	Charles	King	1993-06-24	Warsaw	Masovian	Poland	589838	8498546902
11	57091	Christopher	Garcia	1988-03-21	Austin	Texas	United States	772795	9610580408

Total rows: 525 of 525 | Query complete 00:00:00.155

	r varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date	customer_rank bigint
1		1987-03-26	New York	New York	United States	280862	9638483934	James.Smith@gmail.com	2020-01-02	1
2	Jr	1973-05-24	New York	New York	United States	376573	6588282115	Robert.Downey Jr@gmail.com	2020-01-06	2
3		1990-04-14	Chicago	Illinois	United States	485629	7641021429	John.Williams@gmail.com	2020-01-11	3
4		1953-03-25	Brisbane	Queensland	Australia	260866	7354232181	Michael.Johnson@gmail.com	2020-01-16	4
5		1971-04-24	Bremen	Bremen	Germany	740338	6285552036	Steve.Williams@gmail.com	2020-01-17	5
6	n	1965-10-18	Villach	Carinthia	Austria	621492	7756781677	David.Beckham@gmail.com	2020-01-20	6
7		1958-12-16	San Antonio	Texas	United States	110945	8420993031	Richard.Brown@gmail.com	2020-01-20	6
8		1967-08-27	Amsterdam	North Holland	Netherlands	186896	6691466381	Joseph.James@gmail.com	2020-01-25	8
9		1950-04-30	Dallas	Texas	United States	174080	8024463594	Thomas.Jones@gmail.com	2020-01-26	9
10		1993-06-24	Warsaw	Masovian	Poland	589838	8498546902	Charles.King@gmail.com	2020-01-27	10
11		1988-03-21	Austin	Texas	United States	772795	9610580408	Christopher.Garcia@gmail.com	2020-02-03	11
12		1966-02-10	Dublin	Dublin	Ireland	189723	7947018222	Daniel.Rensch@gmail.com	2020-02-03	11
13		1971-06-03	Brisbane	Queensland	Australia	901170	7764122382	Matthew.Miller@gmail.com	2020-02-04	13
14		1957-04-12	Brussels	Brussels-Capital	Belgium	322713	9371469569	Anthony.James@gmail.com	2020-02-07	14
15		1960-03-13	Brussels	Brussels-Capital	Belgium	943158	8396048181	Mark.Davis@gmail.com	2020-02-09	15
16	ez	1952-01-29	Patras	Western Greece	Greece	127461	9919606399	Donald.Rodriguez@gmail.com	2020-02-09	15
17		1995-07-13	Braga	Braga	Portugal	531783	7287587361	Steven.Smith@gmail.com	2020-02-10	17
18		1960-09-21	Seattle	Washington	United States	427696	8892185286	Paul.Walker@gmail.com	2020-02-11	18
19		1998-03-04	Bruges	West Flanders	Belgium	717216	9289999420	Andrew.Martinez@gmail.com	2020-02-11	18
20	ez	1994-05-07	Braga	Braga	Portugal	700077	7810204569	Joshua.Fernandez@gmail.com	2020-02-13	20
21	ez	1987-08-07	Belfast	Belfast	Northern Ireland	462012	6036253941	Kenneth.Hernandez@gmail.com	2020-02-14	21
22		1967-04-02	Galway	Galway	Ireland	343088	6130341879	Kevin.Hart@gmail.com	2020-02-16	22
23		1952-01-26	Bremen	Bremen	Germany	627019	8605500276	Brian.Lopez@gmail.com	2020-02-17	23
24		1970-07-05	Amsterdam	North Holland	Netherlands	811771	9515240360	George.Clooney@gmail.com	2020-02-18	24
25	z	1972-07-08	Patras	Western Greece	Greece	588452	9234402314	Edward.Gonzalez@gmail.com	2020-02-20	25
26		1976-11-09	Bucharest	Bucharest	Romania	335722	9379225699	Ronald.Reagan@gmail.com	2020-02-21	26
27		1970-05-29	Zurich	Zurich	Switzerland	884138	8426394400	Timothy.Wilson@gmail.com	2020-02-22	27

Total rows: 525 of 525 | Query complete 00:00:00.155 | Ln 413, Col 1

(other columns from result)

38) Rank the customers on the basis of their ages. Give the oldest customer the rank 1.

```
413 SELECT
414     customerID,
415     firstname,
416     lastname,
417     date_of_birth,
418     city,
419     state,
420     country,
421     postalcode,
422     phone,
423     email,
424     dateentered,
425     RANK() OVER (ORDER BY date_of_birth DESC) AS customer_age_rank
426 FROM
427     Customers
428 ORDER BY
429     customer_age_rank;
430
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character vary
1	57167	Alan	Walker	1999-05-01	Houston	Texas	United States	425321	7519015025
2	57219	Nicole	Williams	1999-04-29	Amritsar	Punjab	India	188604	7736783821
3	57327	Carter	Rohan	1999-04-10	Galway	Galway	Ireland	976949	7057989867
4	57425	Albie	Peter	1999-04-07	Vienna	Vienna	Austria	615221	6015427642
5	57116	Stephen	Cole	1999-03-16	Warsaw	Masovian	Poland	930079	7949241495
6	57412	Zack	Linda	1999-02-16	Athens	Attica	Greece	772376	9160812590
7	57580	Cailean	Colby	1999-01-09	Wroclaw	Lower Silesian	Poland	400725	8782854284
8	57484	Alasdair	Odin	1999-01-05	Hobart	Tasmania	Australia	386859	7433996128
9	57504	Travis	Zak	1998-10-23	Hobart	Tasmania	Australia	990877	8896670704
10	57332	Angus	Johnny	1998-09-29	Derry	Derry City and Strabane	Northern Ireland	344263	7688895230

Total rows: 525 of 525 | Query complete 00:00:00.145

	ying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date	customer_age_rank bigint
1		1999-05-01	Houston	Texas	United States	425321	7519015025	Alan.Walker@gmail.com	2020-04-20	1
2		1999-04-29	Amritsar	Punjab	India	188604	7736783821	Nicole.Williams@gmail.com	2020-07-09	2
3		1999-04-10	Galway	Galway	Ireland	976949	7057989867	Carter.Rohan@gmail.com	2021-01-19	3
4		1999-04-07	Vienna	Vienna	Austria	615221	6015427642	Albie.Peter@gmail.com	2021-04-09	4
5		1999-03-16	Warsaw	Masovian	Poland	930079	7949241495	Stephen.Cole@gmail.com	2020-03-02	5
6		1999-02-16	Athens	Attica	Greece	772376	9160812590	Zack.Linda@gmail.com	2021-03-25	6
7		1999-01-09	Wroclaw	Lower Silesian	Poland	400725	8782854284	Cailean.Colby@gmail.com	2021-11-06	7
8		1999-01-05	Hobart	Tasmania	Australia	386859	7433996128	Alasdair.Odin@gmail.com	2021-06-19	8
9		1998-10-23	Hobart	Tasmania	Australia	990877	8896670704	Travis.Zak@gmail.com	2021-07-14	9
10		1998-09-29	Derry	Derry City and Strabane	Northern Ireland	344263	7688895230	Angus.Johnny@gmail.com	2021-01-23	10
11		1998-09-01	Faridabad	Haryana	India	176968	8994343494	Joyce.Ryder@gmail.com	2020-08-20	11
12		1998-07-12	Salzburg	Salzburg	Austria	547157	9096515210	Musa.Henry@gmail.com	2021-09-09	12
13		1998-06-17	Vienna	Vienna	Austria	844788	7858796446	Erik.Elliott@gmail.com	2021-07-23	13
14		1998-04-23	Vila Nova de Gaia	Porto	Portugal	265945	6128815387	Eric.Finlay@gmail.com	2021-10-14	14
15		1998-03-12	Brussels	Brussels-Capital	Belgium	494735	8117434671	Christina.Aguilera@gmail.com	2020-08-26	15
16		1998-03-04	Bruges	West Flanders	Belgium	717216	9289999420	Andrew.Martinez@gmail.com	2020-02-11	16
17		1998-02-10	New York	New York	United States	803691	9837229493	Ashley.White@gmail.com	2020-06-03	17
18		1998-01-26	Lisbon	Lisbon	Portugal	195374	7789921151	Edward.Lewis@gmail.com	2021-03-27	18
19		1998-01-05	Moscow	Moscow	Russia	272100	6269994683	Beverly.Hills@gmail.com	2020-10-06	19
20		1997-12-15	Patras	Western Greece	Greece	279516	7368939317	Emma.Beckham@gmail.com	2020-07-10	20
21		1997-12-04	Vienna	Vienna	Austria	673350	7814242880	Riley.Mary@gmail.com	2021-01-24	21
22		1997-11-25	Belfast	Belfast	Northern Ireland	155218	7908414060	Ollie.Joseph@gmail.com	2021-01-12	22
23		1997-11-23	Los Angeles	California	United States	323529	7849411743	Austin.Emerson@gmail.com	2020-04-10	23
24		1997-11-11	Berlin	Berlin	Germany	683623	9981480800	Keegan.Harley@gmail.com	2021-09-03	24
25		1997-10-05	Nice	Provence-Alpes-Cv¥te d'A...	France	990563	7088534361	Michael.Kathleen@gmail.com	2021-02-10	25
26		1997-07-08	Galway	Galway	Ireland	585217	7674643172	Danielle.Barrett@gmail.com	2020-10-09	26
27		1997-07-07	Milan	Lombardy	Italy	367391	6157137435	Ruairi.Jaxson@gmail.com	2021-05-19	27

Total rows: 525 of 525 | Query complete 00:00:00.145 | Ln 429, Col 23

(other columns from result)

39) Print the count of customers from each city against the record of each customer.

```
433 SELECT
434     customerID,
435     firstname,
436     lastname,
437     city,
438     state,
439     country,
440     postalcode,
441     phone,
442     email,
443     dateentered,
444     (SELECT COUNT(*) FROM Customers c2 WHERE c1.city = c2.city) AS customer_count_in_city
445 FROM
446     Customers c1
447 ORDER BY
448     customerID;
449
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email chara
1	57081	James	Smith	New York	New York	United States	280862	9638483934	Jame
2	57082	Robert	Downey Jr	New York	New York	United States	376573	6588282115	Robe
3	57083	John	Williams	Chicago	Illinois	United States	485629	7641021429	John
4	57084	Michael	Johnson	Brisbane	Queensland	Australia	260866	7354232181	Mich
5	57085	Steve	Williams	Bremen	Bremen	Germany	740338	6285552036	Steve
6	57086	David	Beckham	Villach	Carinthia	Austria	621492	7756781677	Davic
7	57087	Richard	Brown	San Antonio	Texas	United States	110945	8420993031	Richa
8	57088	Joseph	James	Amsterdam	North Holland	Netherlands	186896	6691466381	Josej
9	57089	Thomas	Jones	Dallas	Texas	United States	174080	8024463594	Thor
10	57090	Charles	King	Warsaw	Masovian	Poland	589838	8498546902	Charl

Total rows: 525 of 525 | Query complete 00:00:00.194

	e r varying	lastname character varying	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date	customer_count_in_city bigint
1		Smith	New York	New York	United States	280862	9638483934	James.Smith@gmail.com	2020-01-02	12
2		Downey Jr	New York	New York	United States	376573	6588282115	Robert.Downey Jr@gmail.com	2020-01-06	12
3		Williams	Chicago	Illinois	United States	485629	7641021429	John.Williams@gmail.com	2020-01-11	1
4		Johnson	Brisbane	Queensland	Australia	260866	7354232181	Michael.Johnson@gmail.com	2020-01-16	6
5		Williams	Bremen	Bremen	Germany	740338	6285552036	Steve.Williams@gmail.com	2020-01-17	4
6		Beckham	Villach	Carinthia	Austria	621492	7756781677	David.Beckham@gmail.com	2020-01-20	10
7		Brown	San Antonio	Texas	United States	110945	8420993031	Richard.Brown@gmail.com	2020-01-20	1
8		James	Amsterdam	North Holland	Netherlands	186896	6691466381	Joseph.James@gmail.com	2020-01-25	19
9		Jones	Dallas	Texas	United States	174080	8024463594	Thomas.Jones@gmail.com	2020-01-26	1
10		King	Warsaw	Masovian	Poland	589838	8498546902	Charles.King@gmail.com	2020-01-27	10
11	her	Garcia	Austin	Texas	United States	772795	9610580408	Christopher.Garcia@gmail.com	2020-02-03	1
12		Rensch	Dublin	Dublin	Ireland	189723	7947018222	Daniel.Rensch@gmail.com	2020-02-03	11
13		Miller	Brisbane	Queensland	Australia	901170	7764122382	Matthew.Miller@gmail.com	2020-02-04	6
14		James	Brussels	Brussels-Capital	Belgium	322713	9371469569	Anthony.James@gmail.com	2020-02-07	24
15		Davis	Brussels	Brussels-Capital	Belgium	943158	8396048181	Mark.Davis@gmail.com	2020-02-09	24
16		Rodriguez	Patras	Western Greece	Greece	127461	9919606399	Donald.Rodriguez@gmail.com	2020-02-09	9
17		Smith	Braga	Braga	Portugal	531783	7287587361	Steven.Smith@gmail.com	2020-02-10	7
18		Walker	Seattle	Washington	United States	427696	8892185286	Paul.Walker@gmail.com	2020-02-11	1
19		Martinez	Bruges	West Flanders	Belgium	717216	9289999420	Andrew.Martinez@gmail.com	2020-02-11	1
20		Fernandez	Braga	Braga	Portugal	700077	7810204569	Joshua.Fernandez@gmail.com	2020-02-13	7
21		Hernandez	Belfast	Belfast	Northern Ireland	462012	6036253941	Kenneth.Hernandez@gmail.com	2020-02-14	23
22		Hart	Galway	Galway	Ireland	343088	6130341879	Kevin.Hart@gmail.com	2020-02-16	12
23		Lopez	Bremen	Bremen	Germany	627019	8605500276	Brian.Lopez@gmail.com	2020-02-17	4
24		Clooney	Amsterdam	North Holland	Netherlands	811771	9515240360	George.Clooney@gmail.com	2020-02-18	19
25		Gonzalez	Patras	Western Greece	Greece	588452	9234402314	Edward.Gonzalez@gmail.com	2020-02-20	9
26		Reagan	Bucharest	Bucharest	Romania	335722	9379225699	Ronald.Reagan@gmail.com	2020-02-21	13
27		Wilson	Zurich	Zurich	Switzerland	884138	8426394400	Timothy.Wilson@gmail.com	2020-02-22	24

Total rows: 525 of 525 | Query complete 00:00:00.194 | Ln 449, Col 1

(other columns from result)

40) Get details of the customers whose details were entered very first amongst their respective countries.

```
SELECT
    customerID,firstname,lastname,date_of_birth,city,
    state,country,postalcode,phone,email,dateentered
FROM (
    SELECT
        customerID,firstname,lastname,date_of_birth,city,
        state,country,postalcode,phone,email,dateentered,
        RANK() OVER (PARTITION BY country ORDER BY dateentered) AS entry_rank
    FROM Customers
) AS RankedCustomers
WHERE entry_rank = 1;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date
1	57084	Michael	Johnson	1953-03-25	Brisbane	Queensland	Australia	260866	7354232181	Michael.Johnson@gmail.com	2020-01-16
2	57086	David	Beckham	1965-10-18	Villach	Carinthia	Austria	621492	7756781677	David.Beckham@gmail.com	2020-01-20
3	57094	Anthony	James	1957-04-12	Brussels	Brussels-Capital	Belgium	322713	9371469569	Anthony.James@gmail.com	2020-02-07
4	57240	Lauren	Gottileb	1993-01-27	Paris	vÉle-de-France	France	361087	8366844510	Lauren.Gottileb@gmail.com	2020-09-04
5	57085	Steve	Williams	1971-04-24	Bremen	Bremen	Germany	740338	6285552036	Steve.Williams@gmail.com	2020-01-17
6	57096	Donald	Rodriguez	1952-01-29	Patras	Western Greece	Greece	127461	9919606399	Donald.Rodriguez@gmail.com	2020-02-09
7	57204	Melissa	Clooney	1991-12-31	Gurgaon	Haryana	India	128305	8287186662	Melissa.Clooney@gmail.com	2020-06-10
8	57092	Daniel	Rensch	1966-02-10	Dublin	Dublin	Ireland	189723	7947018222	Daniel.Rensch@gmail.com	2020-02-03
9	57314	William	King	1996-09-24	Turin	Piedmont	Italy	983366	6253310855	William.King@gmail.com	2021-01-13
10	57088	Joseph	James	1967-08-27	Amsterdam	North Holland	Netherlands	186896	6691466381	Joseph.James@gmail.com	2020-01-25
11	57383	Roman	Hamish	1953-04-03	Wellington	Wellington	New Zealand	958930	6234098979	Roman.Hamish@gmail.com	2021-03-01
12	57101	Kenneth	Hernandez	1987-08-07	Belfast	Belfast	Northern Ireland	462012	6036253941	Kenneth.Hernandez@gmail.com	2020-02-14
13	57090	Charles	King	1993-06-24	Warsaw	Masovian	Poland	589838	8498546902	Charles.King@gmail.com	2020-01-27
14	57097	Steven	Smith	1995-07-13	Braga	Braga	Portugal	531783	7287587361	Steven.Smith@gmail.com	2020-02-10
15	57106	Ronald	Reagan	1976-11-09	Bucharest	Bucharest	Romania	335722	9379225699	Ronald.Reagan@gmail.com	2020-02-21

Total rows: 20 of 20 | Query complete 00:00:00.125 | Ln 462, Col 22

41) Get details of the customer whose First Name comes at the end when you order the customers in alphabetical order.

```
SELECT
    customerID,
    firstname,
    lastname,
    date_of_birth,
    city,
    state,
    country,
    postalcode,
    phone,
    email,
    dateentered
FROM
    Customers
ORDER BY
    firstname DESC
LIMIT 1;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date
1	57575	Zayn	Bailey	1974-05-25	Warsaw	Masovian	Poland	184986	7628741723	Zayn.Bailey@gmail.com	2021-10-26

42) Get details of customers whose First Name comes at the end when you order the customers in alphabetical order amongst their respective countries.

```sql
WITH RankedCustomers AS (
    SELECT customerID,firstname,lastname,date_of_birth,city,state,
        country,postalcode,phone, email,dateentered,
        RANK() OVER (PARTITION BY country ORDER BY firstname DESC) AS name_rank
    FROM Customers
)
SELECT
    customerID,firstname,lastname,date_of_birth,city,state,
        country,postalcode,phone, email,dateentered
FROM RankedCustomers
WHERE name_rank = 1;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	date_of_birth date	city character varying	state character varying	country character varying	postalcode integer	phone character varying (15)	email character varying	dateentered date
1	57483	Zander	Luka	1954-06-18	Geelong	Victoria	Australia	954551	7442257638	Zander.Luka@gmail.com	2021-06-18
2	57358	Theodore	Lee	1959-10-06	Vienna	Vienna	Austria	789084	9235126073	Theodore.Lee@gmail.com	2021-02-15
3	57523	Vincent	Jonathan	1960-06-02	Brussels	Brussels-Capital	Belgium	787603	8084706989	Vincent.Jonathan@gmail.com	2021-08-17
4	57404	Tyler	Elijah	1956-02-03	Lille	Hauts-de-France	France	909456	9665092780	Tyler.Elijah@gmail.com	2021-03-18
5	57326	Tommy	Rocco	1972-12-27	Brandenburg	Potsdam	Germany	512833	9223470662	Tommy.Rocco@gmail.com	2021-01-18
6	57412	Zack	Linda	1999-02-16	Athens	Attica	Greece	772376	9160812590	Zack.Linda@gmail.com	2021-03-25
7	57297	Thomas	Patricia	1984-08-16	Jaipur	Rajasthan	India	948553	6466821987	Thomas.Patricia@gmail.com	2020-12-24
8	57236	Victoria	Queen	1971-12-03	Galway	Galway	Ireland	151232	8417363453	Victoria.Queen@gmail.com	2020-08-22
9	57314	William	King	1996-09-24	Turin	Piedmont	Italy	983366	6253310855	William.King@gmail.com	2021-01-13
10	57437	Zak	Peter	1975-01-17	Amsterdam	North Holland	Netherlands	542924	9643370720	Zak.Peter@gmail.com	2021-04-23
11	57409	Zac	Mary	1969-05-30	Wellington	Wellington	New Zealand	264951	7614541479	Zac.Mary@gmail.com	2021-03-23
12	57512	Yusuf	Felix	1957-02-04	Belfast	Belfast	Northern Ireland	427874	9932962637	Yusuf.Felix@gmail.com	2021-08-02
13	57575	Zayn	Bailey	1974-05-25	Warsaw	Masovian	Poland	184986	7628741723	Zayn.Bailey@gmail.com	2021-10-26

Total rows: 19 of 19 | Query complete 00:00:00.131 | Ln 496, Col 2

43) Get the most ordered product's details in each quarter of each year.

```sql
WITH QuarterlyOrderCounts AS (
    SELECT EXTRACT(YEAR FROM o1.orderdate) AS year,
        EXTRACT(QUARTER FROM o1.orderdate) AS quarter,
        COUNT(od1.orderdetailID) AS order_count
    FROM Orders o1
    JOIN  OrderDetails od1 ON o1.orderID = od1.orderID
    GROUP BY EXTRACT(YEAR FROM o1.orderdate), EXTRACT(QUARTER FROM o1.orderdate)
)   SELECT p.productID,p.product,p.categoryID,p.sub_category,
    p.brand,p.sale_price,p.market_price,p.type,
    EXTRACT(YEAR FROM o.orderdate) AS year,
    EXTRACT(QUARTER FROM o.orderdate) AS quarter
FROM Products p
JOIN OrderDetails od ON p.productID = od.productID
JOIN Orders o ON od.orderID = o.orderID
JOIN  QuarterlyOrderCounts qoc ON EXTRACT(YEAR FROM o.orderdate) = qoc.year
      AND EXTRACT(QUARTER FROM o.orderdate) = qoc.quarter
WHERE qoc.order_count = (
        SELECT MAX(order_count)
        FROM QuarterlyOrderCounts
        WHERE year = EXTRACT(YEAR FROM o.orderdate) AND quarter = EXTRACT(QUARTER FROM o.orderdate)
    );
```

Data Output | Messages | Notifications

	productid integer	product character varying	categoryid integer	sub_category character varying	brand character varying	sale_price integer	market_price integer	type character va
1	1	Original Disinfectant Toilet Cleaner Liquid	5001	All Purpose Cleaners	Harpic	489	534	Toilet Cleane
2	1	Original Disinfectant Toilet Cleaner Liquid	5001	All Purpose Cleaners	Harpic	489	534	Toilet Cleane
3	1	Original Disinfectant Toilet Cleaner Liquid	5001	All Purpose Cleaners	Harpic	489	534	Toilet Cleane
4	4	Harpic Toilet Cleaner Liquid - Original 1 L + Lizol Floor Cleaner, Citrus - 2L	5001	All Purpose Cleaners	bb Combo	462	558	Toilet Cleane
5	5	Disinfectant Bathroom Cleaner Liquid - Lemon	5001	All Purpose Cleaners	Harpic	299	362	Toilet Cleane
6	6	Super Saver Pack Toilet Cleaner Original, 500ml + Bathroom Cleaner Lemon, 500ml	5001	All Purpose Cleaners	Harpic	162	184	Toilet Cleane
7	7	Original 1 ltr + Bathroom Cleaner Lemon 500 ml	5001	All Purpose Cleaners	Harpic	247	273	Toilet Cleane

Total rows: 1000 of 27527 | Query complete 00:01:15.097 | Ln 523, Col 7

44) Get the details of category whose products where ordered the most in each month of each year.

```
527 SELECT categoryID,categoryname,year,month,total_orders
528 FROM (  SELECT c.categoryID, c.categoryname,
529         EXTRACT(YEAR FROM o.orderdate) AS year,
530         EXTRACT(MONTH FROM o.orderdate) AS month,
531         COUNT(od.orderdetailID) AS total_orders,
532         RANK() OVER (PARTITION BY EXTRACT(YEAR FROM o.orderdate),
533         EXTRACT(MONTH FROM o.orderdate) ORDER BY COUNT(od.orderdetailID) DESC) AS rank_order
534     FROM Category c
535     JOIN Products p ON c.categoryID = p.categoryID
536     JOIN  OrderDetails od ON p.productID = od.productID
537     JOIN  Orders o ON od.orderID = o.orderID
538     GROUP BY  c.categoryID, c.categoryname, EXTRACT(YEAR FROM o.orderdate),
539       EXTRACT(MONTH FROM o.orderdate)
540 ) ranked
541 WHERE  rank_order = 1;
```

Data Output | Messages | Notifications

	categoryid integer	categoryname character varying	year numeric	month numeric	total_orders bigint
1	5008	Beauty & Hygiene	2020	1	5
2	5008	Beauty & Hygiene	2020	2	19
3	5008	Beauty & Hygiene	2020	3	35
4	5008	Beauty & Hygiene	2020	4	48
5	5008	Beauty & Hygiene	2020	5	75
6	5008	Beauty & Hygiene	2020	6	108
7	5008	Beauty & Hygiene	2020	7	99
8	5008	Beauty & Hygiene	2020	8	116
9	5008	Beauty & Hygiene	2020	9	164
10	5008	Beauty & Hygiene	2020	10	178
11	5008	Beauty & Hygiene	2020	11	150
12	5008	Beauty & Hygiene	2020	12	182

Total rows: 24 of 24 Query complete 00:00:00.212

45) Get the details of the orders placed by customers placed by them the 5th time.

```
--QN 45

SELECT
    o.orderID,
    o.customerID,
    c.firstname,
    c.lastname,
    o.orderdate,
    o.shipperid,
    o.shipdate,
    o.deliverydate,
    o.total_order_amount
FROM  Orders o JOIN Customers c ON o.customerID = c.customerID
WHERE o.customerID IN (
        SELECT customerID
        FROM  Orders
        GROUP BY customerID
        HAVING COUNT(DISTINCT orderID) = 5
);
```

Data Output | Messages | Notifications

	orderid integer	customerid integer	firstname character varying	lastname character varying	orderdate date	shipperid integer	shipdate date	deliverydate date	total_order_amount double precision
1	7655512	57107	Timothy	Wilson	2020-03-05	8	2020-03-14	2020-03-27	20135
2	7655538	57107	Timothy	Wilson	2020-04-10	3	2020-04-14	2020-05-03	15779
3	7655543	57149	Gerald	Gun	2020-04-15	7	2020-04-20	2020-05-03	54861
4	7655559	57157	Bob	Dylan	2020-05-01	6	2020-05-06	2020-05-25	27427
5	7655581	57157	Bob	Dylan	2020-05-17	1	2020-05-18	2020-06-07	30666
6	7655610	57164	Willie	Adams	2020-06-03	7	2020-06-13	2020-06-28	26028
7	7655634	57107	Timothy	Wilson	2020-06-14	8	2020-06-17	2020-07-01	13528.8
8	7655642	57157	Bob	Dylan	2020-06-19	3	2020-06-24	2020-06-25	1

Succe

Total rows: 115 of 115 Query complete 00:00:00.191

46) Create a Quarter-wise ranking in terms of revenue generated in each product category in each year.

```
SELECT
    p.categoryID,
    EXTRACT(YEAR FROM o.orderdate) AS year,
    EXTRACT(QUARTER FROM o.orderdate) AS quarter,
    RANK() OVER (PARTITION BY p.categoryID, EXTRACT(YEAR FROM o.orderdate),
    EXTRACT(QUARTER FROM o.orderdate) ORDER BY SUM(o.total_order_amount) DESC) AS revenue_rank
FROM
    Orders o
JOIN
    OrderDetails od ON o.orderID = od.orderID
JOIN
    Products p ON od.productID = p.productID
GROUP BY
    p.categoryID, EXTRACT(YEAR FROM o.orderdate), EXTRACT(QUARTER FROM o.orderdate);
```

Data Output | Messages | Notifications

	categoryid integer	year numeric	quarter numeric	revenue_rank bigint
1	5001	2020	1	1
2	5001	2020	2	1
3	5001	2020	3	1
4	5001	2020	4	1
5	5001	2021	1	1
6	5001	2021	2	1
7	5001	2021	3	1
8	5001	2021	4	1

Total rows: 88 of 88 | Query complete 00:00:00.150

47) Identify the age of top 10 customers who spent the most.

```
SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM c.date_of_birth) AS age,
    SUM(o.total_order_amount) AS total_spent
FROM Customers c
JOIN Orders o ON c.customerID = o.customerID
GROUP BY c.customerID, c.firstname, c.lastname, c.date_of_birth
ORDER BY total_spent DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	age numeric	total_spent double precision
1	57249	Jacqueline	Fernandez	37	442544.35
2	57495	Toby	Richard	70	421414.88
3	57574	Richard	Corey	69	408974.44
4	57455	Caelan	Lochlan	46	405883
5	57486	Cruz	Duncan	28	401144.87
6	57232	Diane	Morgan	71	393376.82
7	57213	Shirley	Taylor	31	389205.68
8	57200	Michelle	Allen	42	388967.43
9	57339	Ryan	Lyon	55	384503.2
10	57102	Kevin	Hart	57	376974.07999999996

48) Identify the count of brands whose products the company sells within each category.

```
597  SELECT
598      c.categoryname,
599      p.brand,
600      COUNT(DISTINCT p.brand) AS brand_count
601  FROM
602      Products p
603  JOIN
604      Category c ON p.categoryID = c.categoryID
605  GROUP BY
606      c.categoryname, p.brand
607  ORDER BY
608      c.categoryname, brand_count DESC;
609
```

Data Output | Messages | Notifications

	categoryname character varying	brand character varying	brand_count bigint
1	Baby Care	1st Bites	1
2	Baby Care	Amul	1
3	Baby Care	Aptamil	1
4	Baby Care	Aveeno	1
5	Baby Care	Baby Dove	1
6	Baby Care	Bebe	1
7	Baby Care	Bella	1
8	Baby Care	Bella baby Happy	1
9	Baby Care	bigbasket	1
10	Baby Care	Bionova	1
11	Baby Care	BIOTIQUE	1
12	Baby Care	Bodyguard	1
13	Baby Care	BTN Sports	1
14	Baby Care	Canopus	1

Total rows: 1000 of 2690 | Query complete 00:00:00.243

49) Print details of customers along with details of total orders and total spend. For those customers who ordered more than thrice and received those orders in less than 7 days.

```
613  SELECT
614      c.customerID,
615      c.firstname,
616      c.lastname,
617      COUNT(DISTINCT o.orderID) AS total_orders,
618      SUM(o.total_order_amount) AS total_spend, MAX(o.deliverydate - o.orderdate)as days
619  FROM Customers c
620  JOIN  Orders o ON c.customerID = o.customerID
621  WHERE o.orderdate IS NOT NULL
622  GROUP BY c.customerID, c.firstname, c.lastname
623  HAVING COUNT(DISTINCT o.orderID) > 3
624      AND MAX(o.deliverydate - o.orderdate) < 7;
625
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	total_orders bigint	total_spend double precision	days integer

```sql
--QN 49
--REASON FOR NOT HAVING DATA WHEN DAYS < 7
SELECT
    c.customerID,
    c.firstname,
    c.lastname,
    COUNT(DISTINCT o.orderID) AS total_orders,
    SUM(o.total_order_amount) AS total_spend, MAX(o.deliverydate - o.orderdate)as days
-- if we observe days, we can see that no 'days' is less than 7
-- IN result we can see that minimum value of 'days' is 11
FROM Customers c
JOIN  Orders o ON c.customerID = o.customerID
WHERE o.orderdate IS NOT NULL
GROUP BY c.customerID, c.firstname, c.lastname
HAVING COUNT(DISTINCT o.orderID) > 3
order by days asc;
    --AND MAX(o.deliverydate - o.orderdate) < 7;
```

Data Output | Messages | Notifications

	customerid integer	firstname character varying	lastname character varying	total_orders bigint	total_spend double precision	days integer
1	57382	Lachlan	Ryan	4	80382	11
2	57321	Callum	Lennon	4	74575	15
3	57576	Adrian	Enzo	5	125363	15
4	57452	Jasper	Hunter	4	98388	16
5	57254	Kathryn	Clark	7	100989.35	17
6	57492	Rudi	Ian	8	175919.96	18
7	57336	Joseph	Linda	4	68664	18
8	57191	Nancy	Lyon	5	77550.3	19
9	57364	Innes	White	7	126232.9	19
10	57493	Spencer	Jensen	6	63145	19
11	57226	Catherine	Green	6	101444.75	19
12	57552	Maximilian	Reggie	4	58215.32	19

Total rows: 516 of 516 Query complete 00:00:00.121

(Reason: from the SQL query you will find that minimum days = 11

That’s why we will find no data for days<7)

50) Print details of the 5 most ordered products.

```
631 SELECT
632     p.productID,
633     p.product,
634     p.categoryID,
635     p.sub_category,
636     p.brand,
637     SUM(od.quantity) AS total_order_quantity
638 FROM  Products p
639 JOIN  OrderDetails od ON p.productID = od.productID
640 GROUP BY  p.productID, p.product, p.categoryID, p.sub_category, p.brand
641 ORDER BY  total_order_quantity DESC
642 LIMIT 5;
643
```

Data Output | Messages | Notifications

	productid integer	product character varying	categoryid integer	sub_category character varying	brand character varying	total_order_quantity bigint
1	17639	Plain Container Jumbo Combo Set - Pink	5002	Storage & Accessories	Suvidha	88
2	10872	Chicken & Egg Adult Dog Food	5002	Pet Food & Accessories	Drools	85
3	16921	Copper Steel Water Jug With Glass Set	5002	Steel Utensils	Frestol	83
4	19591	Cheese - Pepper Jack, Block	5004	Dairy & Cheese	Fresho Signature	81
5	21056	Tea Bags - Lemon	5004	Drinks & Beverages	Mlesna	81

51) Print the details of the payment method from each quarter of each year which had the highest transaction value.

```
647 SELECT
648     paymenttype,
649     year,
650     quarter,
651     total_transaction_value
652 FROM (  SELECT
653         paymenttype,
654         EXTRACT(YEAR FROM orderdate) AS year,
655         EXTRACT(QUARTER FROM orderdate) AS quarter,
656         SUM(total_order_amount) AS total_transaction_value,
657         RANK() OVER (PARTITION BY EXTRACT(YEAR FROM orderdate),
658         EXTRACT(QUARTER FROM orderdate) ORDER BY SUM(total_order_amount) DESC) AS ranking
659     FROM  Orders
660     JOIN Payments ON Orders.paymentID = Payments.paymentID
661     GROUP BY  paymenttype, EXTRACT(YEAR FROM orderdate), EXTRACT(QUARTER FROM orderdate)
662 ) RankedPayments
663 WHERE  ranking = 1;
664
```

Data Output | Messages | Notifications

	paymenttype character varying	year numeric	quarter numeric	total_transaction_value double precision
1	Credit Card	2020	1	313530
2	Credit Card	2020	2	1299571.3000000003
3	Credit Card	2020	3	1960310.6999999997
4	Credit Card	2020	4	2967481.9000000004
5	Credit Card	2021	1	4713764.150000001
6	Credit Card	2021	2	7202443
7	Credit Card	2021	3	10913244
8	Credit Card	2021	4	17477608