Information System Design

Final Project for Advanced Database Management Systems Group 7 (Teal) | ISM6218.003F22

Team Members:

1. **Chandni Kumari**
2. **Simran Agichani**
3. **Priyam**
4. **Shuvrangshu Mukhopadhyay**

Summary of Contents

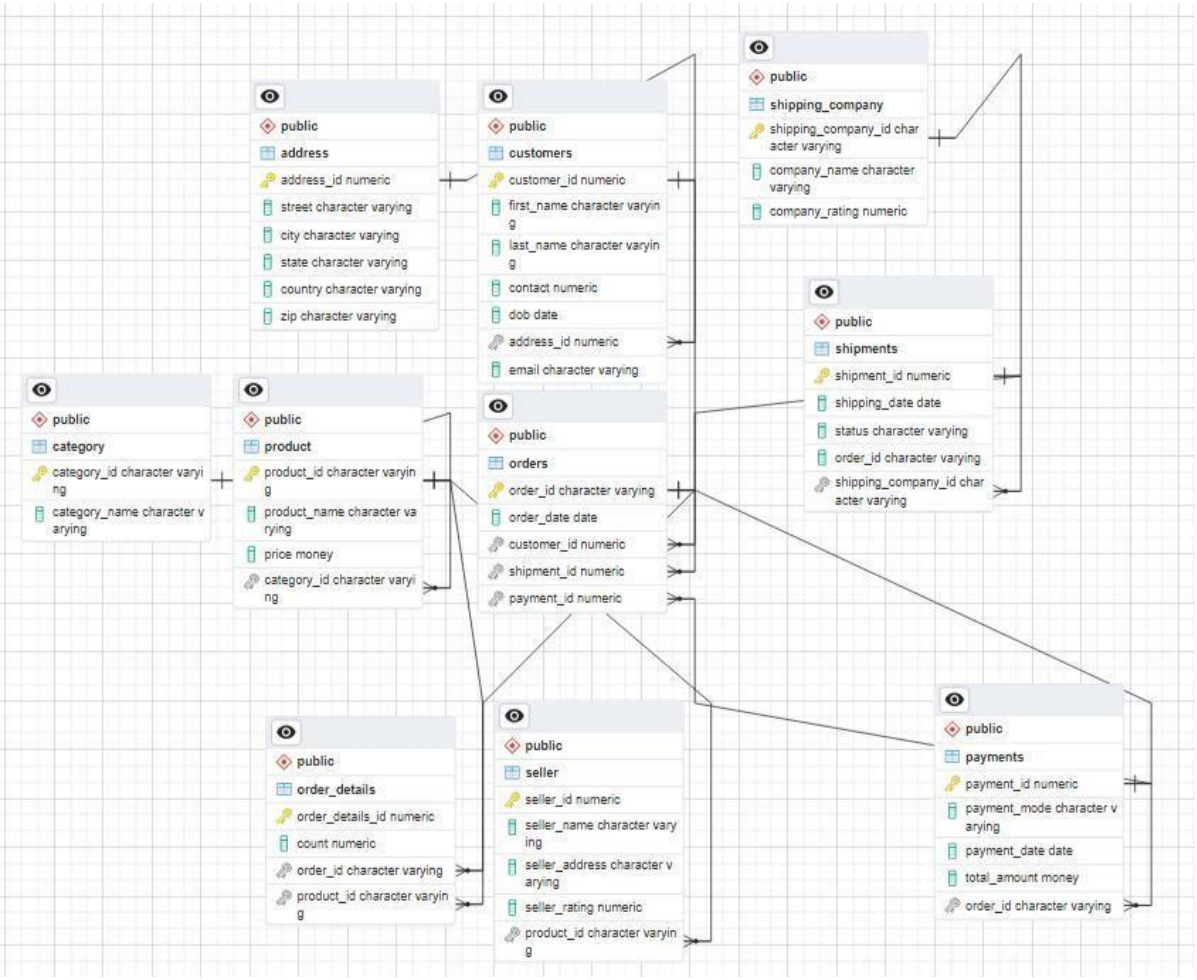
Project Contribution

Topic Area	Description	Group Member	Weight
Database Design	This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality.	Simran Agichani, Chandni Kumari	25%
Query Writing	This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures.	Priyam, Chandni Kumari	25%
Performance Tuning	In this section, you can capitalize and extend your prior experiments with indexing, optimizer modes, partitioning, parallel execution, and any other techniques you want to further explore	Shuvrangshu, Simran Agichani	25%
Data Visualization	Here you are free to explore any other topics of interest. Suggestions include DBA scripts, database security, interface design, data visualization, data mining, and NoSQL databases	Priyam, Shuvrangshu	25%

Overview:

E-Commerce is the activity of electronically buying or selling of products on online services or over the internet. eCommerce websites should maintain personal data of customers. These websites acts as an aggregator which connects sellers to customers through their website. The website should also maintain seller details and should be able to hold and place multiple order at the same time. In this project we are trying to build a database system which will encompass some of the features of an eCommerce database instances.

Entity Relationship Diagram:



Objectives:

1. The eCommerce Website's database is used to maintain data of the customers, orders, shipping, order details, and payment details provided by the customer
2. Details regarding customer, seller, shipping company and orders placed
3. Monitor and improve the value of eCommerce services
4. Contains information about sellers, shipment provided

5. Unique order ID, customer ID, category ID, seller ID, product ID, order details ID, shipment ID, payment ID, shipment company ID & address ID

Motivation:

The motive behind this database is to make interactions of customers with sellers via website owner simpler. It will store all the customer details such as their address, contact, address, billing information, there order details, it also stores the status of the orders.

Business Rules:

1. A customer can place one or more orders, but an order must be placed by one and only one customer.

2. There can be more than one customer living at a particular address, and a customer must have an address for delivery of the order.
3. An order can have multiple products and multiple orders can have a specific product.
4. If there is a product, there must be only one category associated with it, and there can be multiple products in that category.
5. A seller may must sell at least one or more products, but that product can't be sold by any other seller.
6. An order must have one payment, and that payment can't belong to any other order.
7. A shipment company may or may not process one or many orders.
8. Once an order is placed, it will be assigned a particular shipment company to deliver. It must be shipped in one shipment process.

User Requirements:

The user requirements are needed for effective use of the system, the users should completely get involved and given opportunity to participate. This can reduce the number of errors associated with the management and users. So, the user requirements that will apply to our system are given below.

1. **Customers**: Details about the customer, e.g., first_name, last_name
2. **Order Details**: Details about the ordered items and its count
3. **Orders**: Details about the order, e.g., order_id, fulfillment_status, product_id, payment_id
4. **Shipments**: Details about shipping, e.g., tracking_number, customer_id, shipping_date
5. **Shipping Company**: Details about the company responsible for shipping
6. **Category**: Details about category, e.g., category_id, category_name
7. **Address**: Details about customer's address, e.g., address_id, street_address, country
8. **Payments**: Details about payment, e.g., payment_id, payment_mode, transaction_number
9. **Product**: Details about the product ordered, e.g., product_id, product_name, price, stock
10. **Seller**: Details about sellers of products, e.g., seller_id, seller_name, product_id

Data Dictionary

Table Name	Attribute Name	Description	Key Type
Customers	customer_id	Customer's unique id	PK
	first_name	Customer's name	
	last_name	Customer's mail address	
	email	Customer's mobile number	
	contact address_id	Customer's address_id	FK (Address(address_id))
Address	address_id	Customer's unique address id	PK
	street	Street of the address	
	city	City of the address	
	state	State of the address	
	country	Country of the address	
	zipcode	Zipcode of the address	
Orders	order_id	Order's unique id	PK
	order_date	Date of order placement	
	customer_id	Customer id number	FK (Customers(customer_id)
	ship_id	Shipment id number	FK (Shipments(ship_id))
	payment_id	Payment id number	FK (Payments(payment_id))
Order_details	order_details_id	Quantity of products in order	PK
	count	Product id number	FK (Product(product_id))
	product_id order_id	Order id number	FK (Orders(order_id))
Product	product_id	Product's unique id	PK
	product_name	Name of the product	
	price	Price in dollars of the product	
	category_id	Category of the product	FK (Category(category_id)
Category	category_id	Category's unique id number	PK
	category_name	Name of the category	
Seller	seller_id	Seller's unique id	PK
	seller_name	Seller's name	
	seller_zipcode	Zipcode of the seller	
	seller_rating	Seller's rating	
	product_id	Product's id, sold by seller	FK (Product(product_id))
Payments	payment_id	Payment's transaction id	PK
	mode	Mode of the payment	
	payment_date	Date of the payment	
	total_amount	Total amount paid	
	order_id	Paid order's id number	FK (Orders(order_id))
Shipments	ship_id	Shipment's unique id	PK
	ship_date	Date of the shipment	
	status	Status of the shipment	
	order_id	Shipment's order id	FK (Orders(order_id))
	company_id	Shipping id of company	FK (Ship_company(company_id))
Shipping_company	company_id	Shipping company's unique id	PK
	company_name	Shipping company's name	
	company_rating	Shipping company's rating	

Table Views:

Dashboard Properties SQL Statistics Dependencies Dependents Processes ecomm/postgres@Leetcode*

ecomm/postgres@Leetcode

No limit

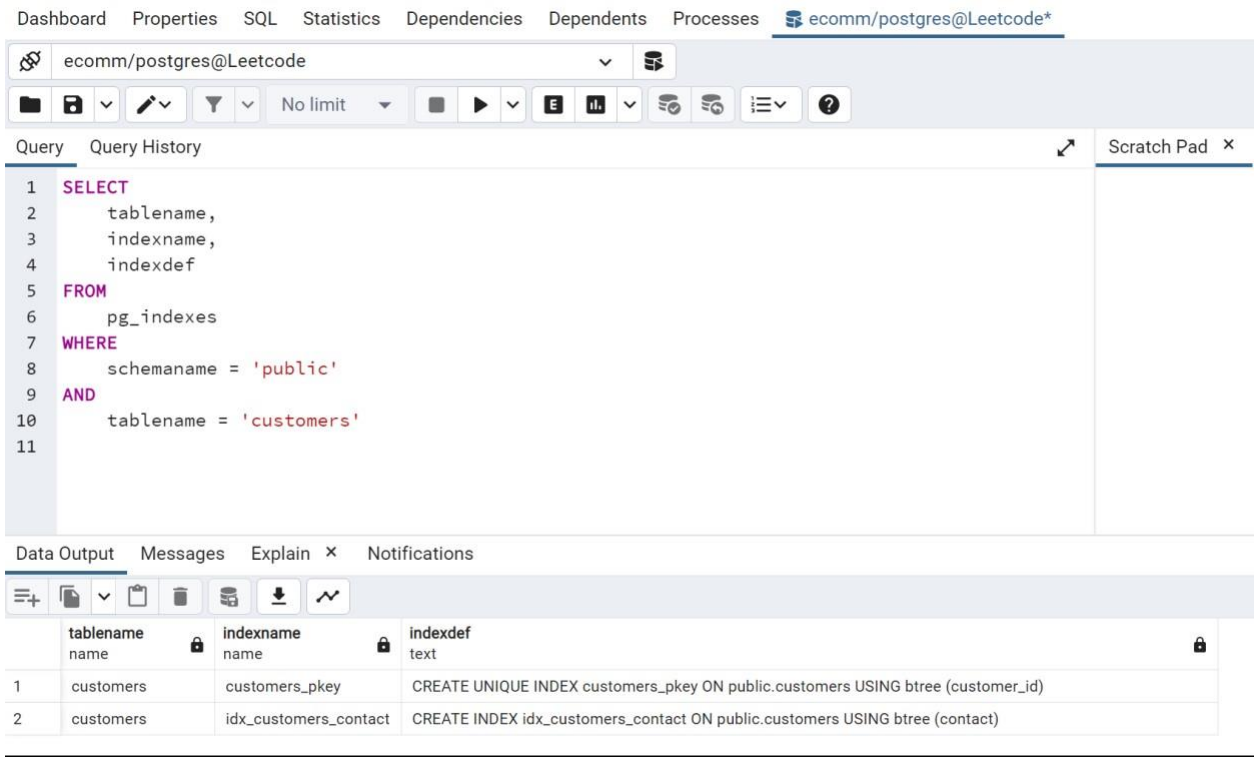
Query Query History

```
1  SELECT *
2  FROM
3      pg_catalog.pg_constraint con
4      INNER JOIN pg_catalog.pg_class rel
5                 ON rel.oid = con.conrelid
6      INNER JOIN pg_catalog.pg_namespace nsp
7                 ON nsp.oid = connamespace
8  WHERE nspname = 'public'
9
```

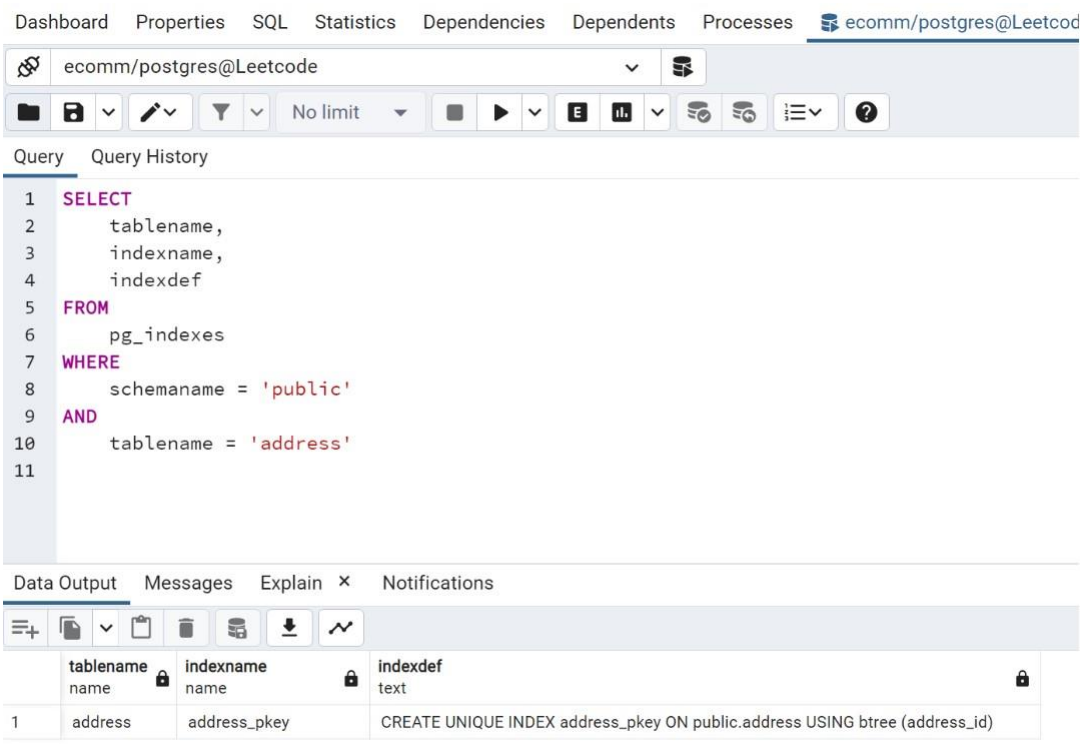
Data Output Messages Notifications

	oid oid	conname name	connamespace oid	contype "char" (1)
1	24612	product_id	2200	f
2	16617	order_id	2200	f
3	16528	order_details_pkey	2200	p
4	16514	address_pkey	2200	p
5	16552	category_pkey	2200	p
6	16674	shipping_company_pkey	2200	p
7	16575	company_rating	2200	c
8	16652	address_id	2200	f
9	16650	customers_pkey	2200	p
10	24584	payment_id	2200	f
11	16691	shipment_id	2200	f
12	16686	customer_id	2200	f
13	16591	orders_pkey	2200	p
14	16675	shipping_company	2200	f
15	16670	shipments_pkey	2200	p
16	24582	payments_pkey	2200	p
17	24622	product_id	2200	f
18	24602	seller_pkey	2200	p
19	24598	seller_rating	2200	c
20	24617	category_id	2200	f
21	24610	product_pkey	2200	p

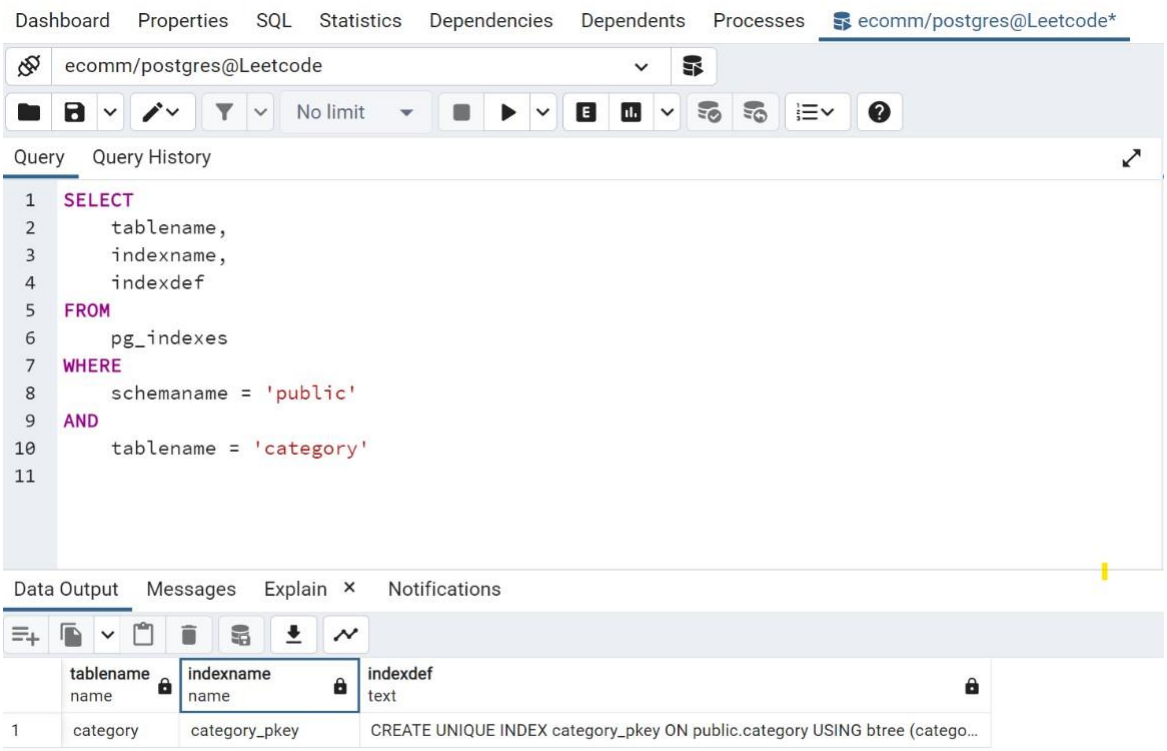
Customers:


Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
ecomm/postgres@Leetcode*
ecomm/postgres@Leetcode
No limit
Query
Query History
Scratch Pad
SELECT
tablename,
indexname,
indexdef
FROM
pg_indexes
WHERE
schemaname = 'public'
AND
tablename = 'customers'
Data Output
Messages
Explain
Notifications
tablename name
indexname name
indexdef text
1 customers customers_pkey CREATE UNIQUE INDEX customers_pkey ON public.customers USING btree (customer_id)
2 customers idx_customers_contact CREATE INDEX idx_customers_contact ON public.customers USING btree (contact)

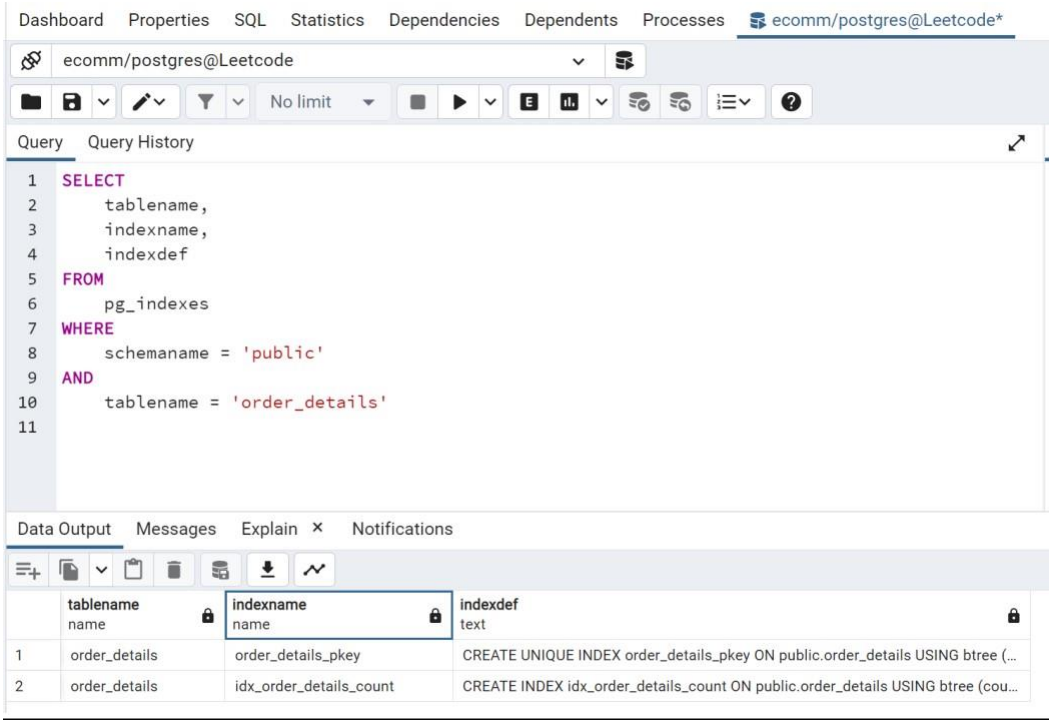
Address:



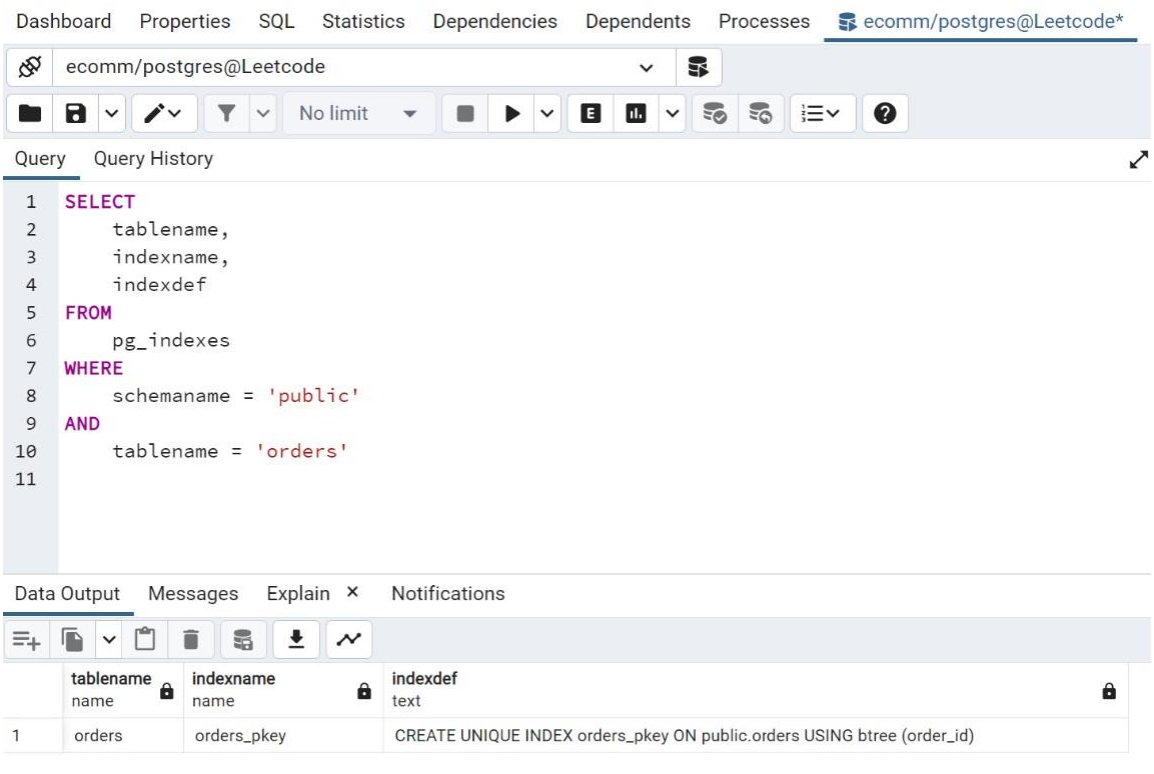
Category:



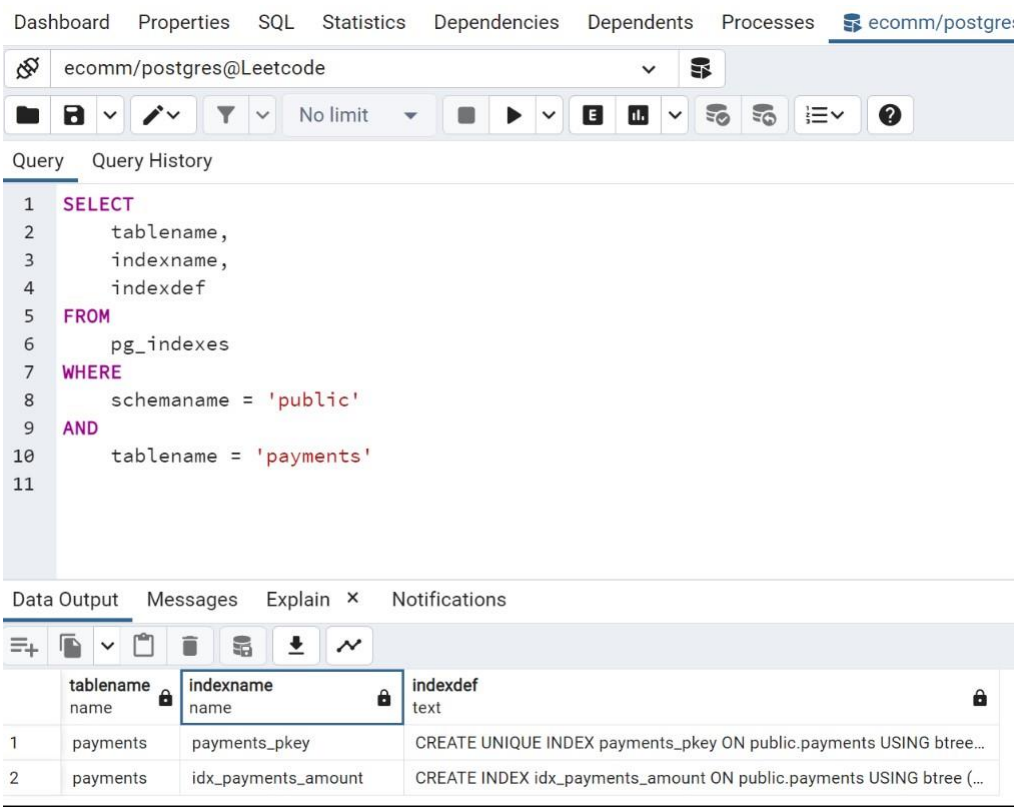
Order Details:



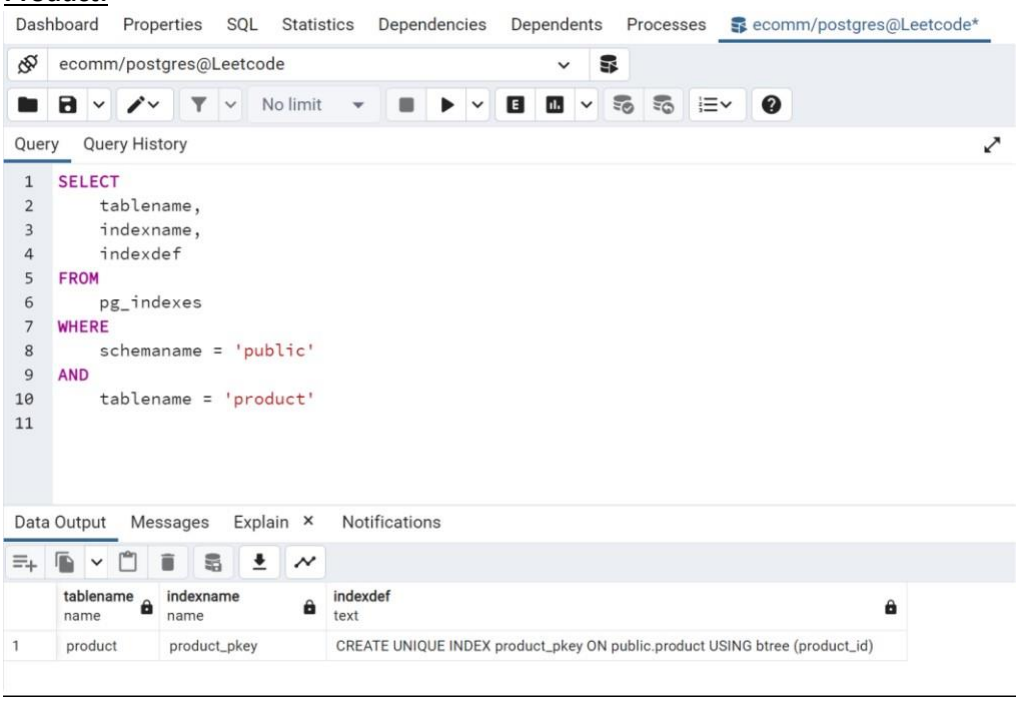
Orders:



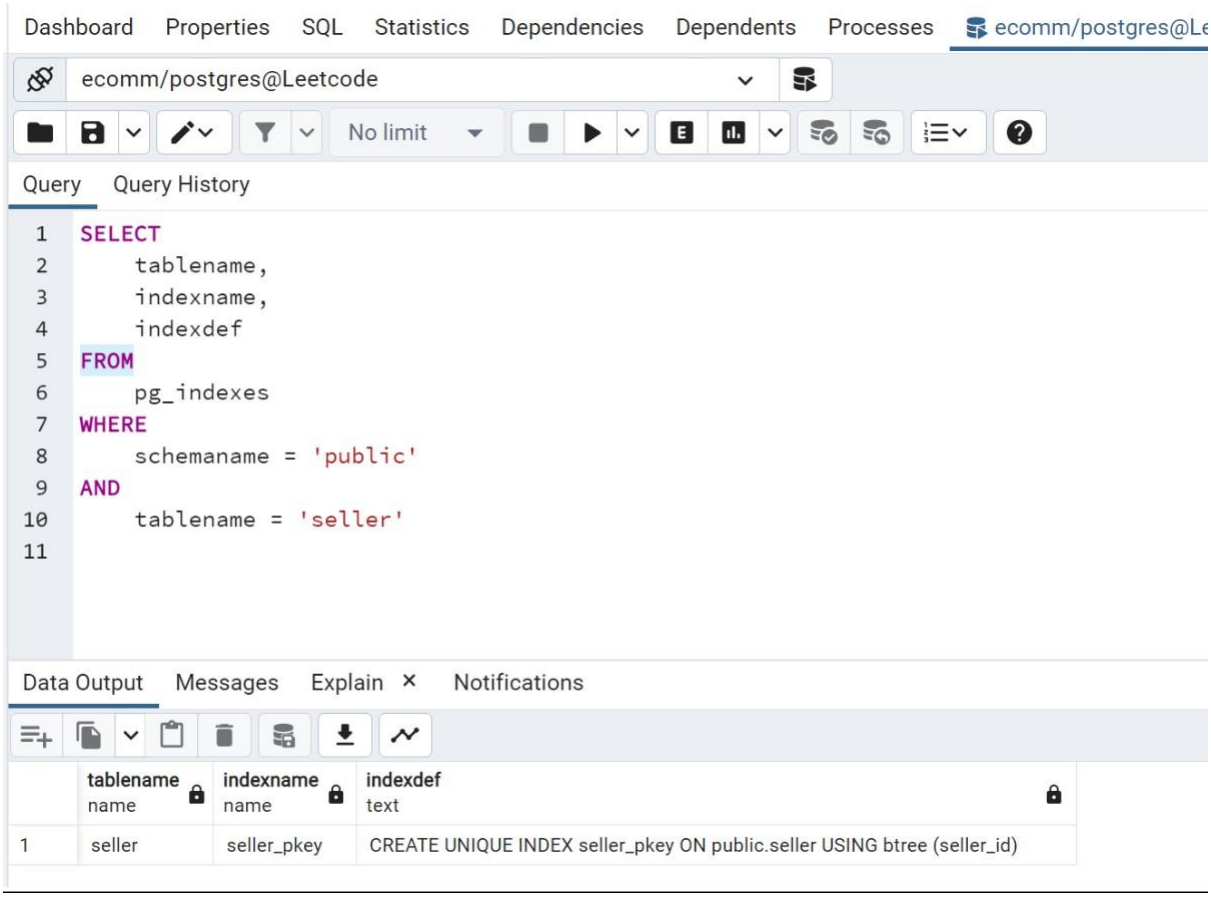
Payments:



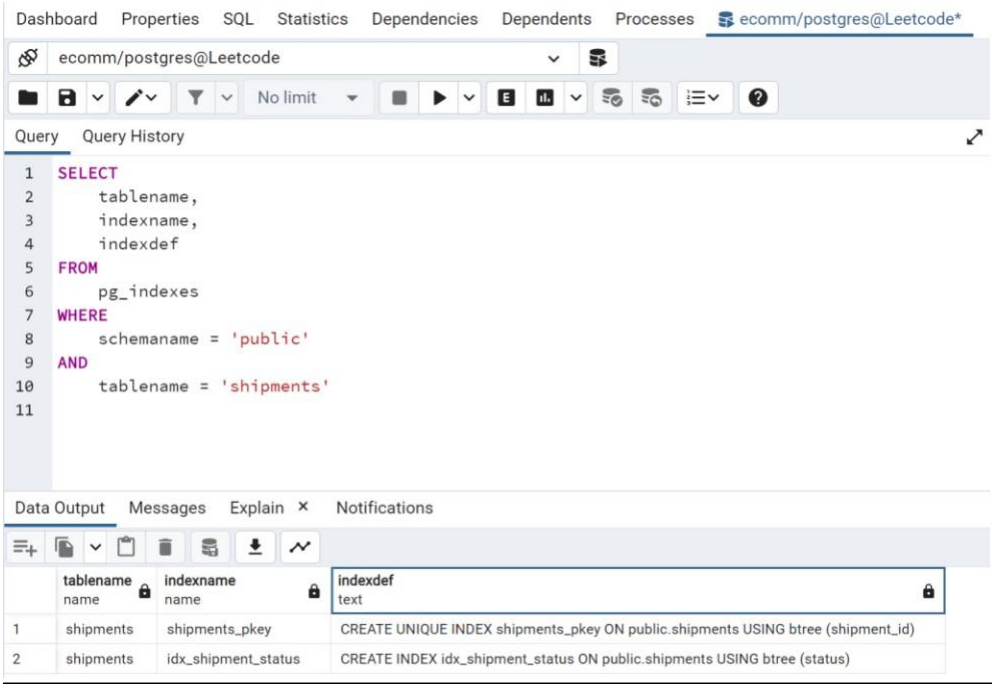
Product:



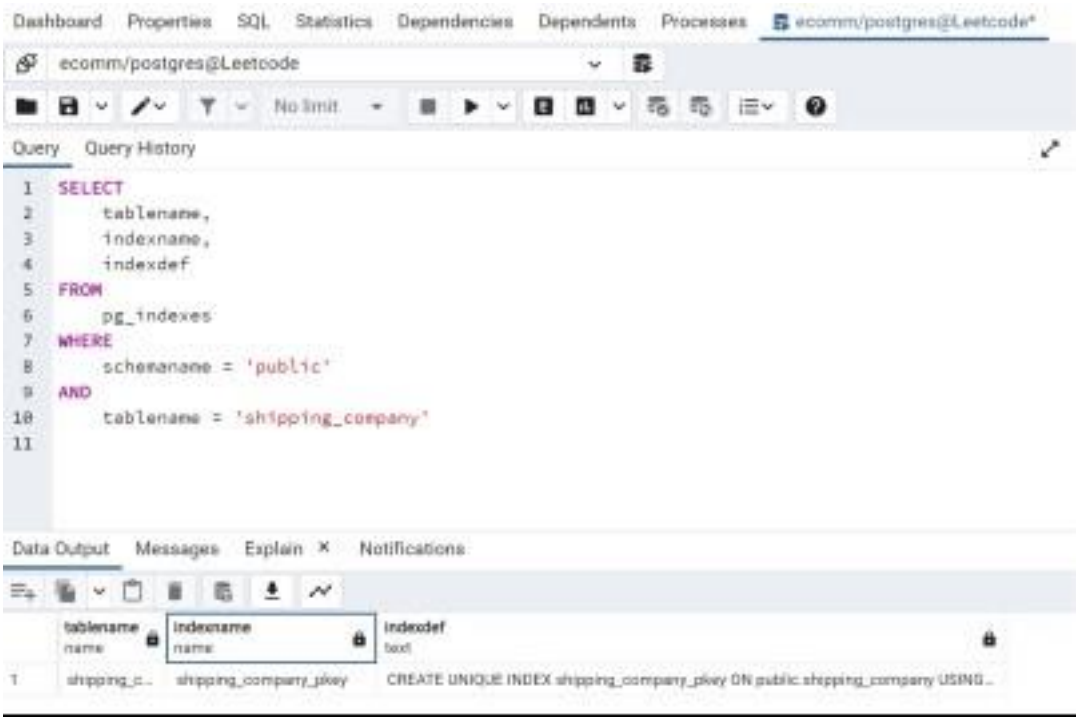
Seller:



Shipments:



Shipping Company:



Data Synthesis

The data for the project has been synthesized using a combination of an online tool named **Mockaroo** and **Microsoft Excel.** Some of the prominent functions that were used in Excel includes:

- VLOOKUP
- INDEX MATCH
- ROWS
- RAND()
- RANDBETWEEN

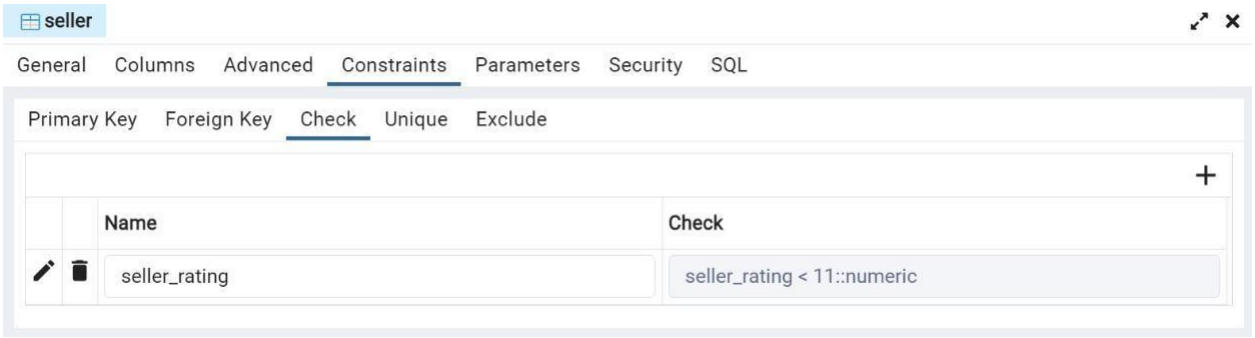
Data Integrity

Data Integrity refers to the consistency and maintenance of the data through the life cycle of the database. In a database, data integrity can be ensured through the implementation of Integrity Constraints in a table. Integrity constraints help apply business rules to the database tables. The constraints can either be at a column level or a table level. Some of the most common constraints are,

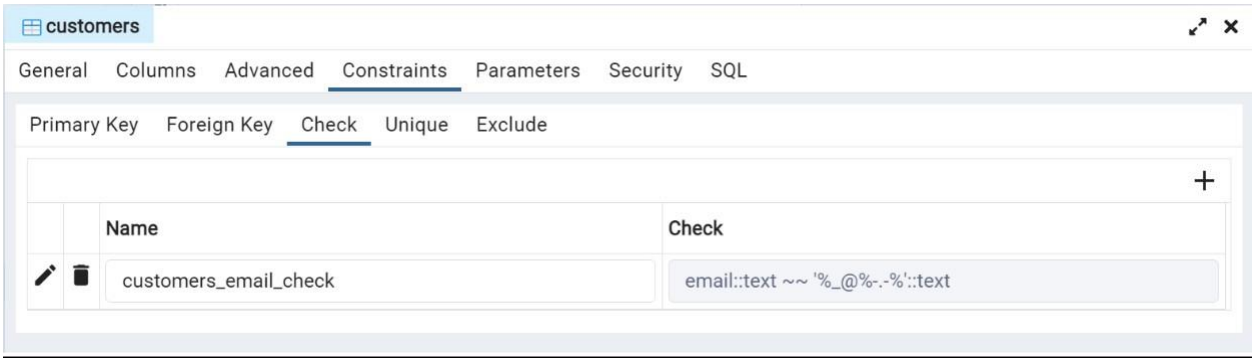
- NOT NULL – Prevents a column from having a NULL value.
- PRIMARY KEY – Uniquely identifies each row or record in table.
- FOREIGN KEY – Uniquely identifies a column that references a PRIMARY KEY in another table.
- UNIQUE – Prevents a column from having duplicate values.
- CHECK – Checks for values that satisfy a specific condition as defined by the user

Check constraints set on seller_ratings (<=10) and regex for email id check.

Seller Table Check Constraint



Customer Table Check Constraint



Performance Tuning

INDEX

An index is used to increase the overall performance of queries. Indexing does this by reducing the data pages that has to be visited or scanned every time a query is run. When we create index, by default the primary key creates a clustered index. A clustered index determines the physical order of data in a table. There can be only one clustered index per table.

```
SELECT * FROM order_details

WHERE count = 9;
```

Data Output | Messages | Explain | Notifications

	order_details_id [PK] numeric	count numeric	order_id character varying	product_id character varying
1	5	9	8bdf129e-d1e2-4...	99-4100895
2	7	9	c63f9853-ca2d-4...	08-0523481
3	10	9	742fb293-16c7-4...	50-0395478
4	27	9	b45ebf1e-ee3f-4...	15-3001991
5	43	9	91906d10-d438-...	44-1689446
6	54	9	3c562dc7-d6ca-4...	99-7013768
7	62	9	0a5c4da0-2aa8-4...	41-1891390
8	82	9	8fa42b5c-8b82-4...	21-7847905
9	88	9	97a9dee1-04a7-4...	73-8494973
10	111	9	1afdc49b-759e-4...	05-8288484

Execution Plan

Data Output | Messages | Explain × | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Bitmap Heap Scan on order_details as order_details (rows=10 loops=1) **Recheck Cond**: (count = '9'::numeric) **Heap Blocks**: exact=2	10	1
2.	→ Bitmap Index Scan using idx_order_details_count (rows=10 loops=1) **Index Cond**: (count = '9'::numeric)	10	1

Optimizer mode: Optimizer mode is used to choose better execution plans for poorly written queries. This is good for applications that routinely display partial results to users such as paging data to a customer in a web application

Query | Query History

```
1  SELECT * FROM shipments
2  WHERE status = 'Shipped'
3  LIMIT 5
```

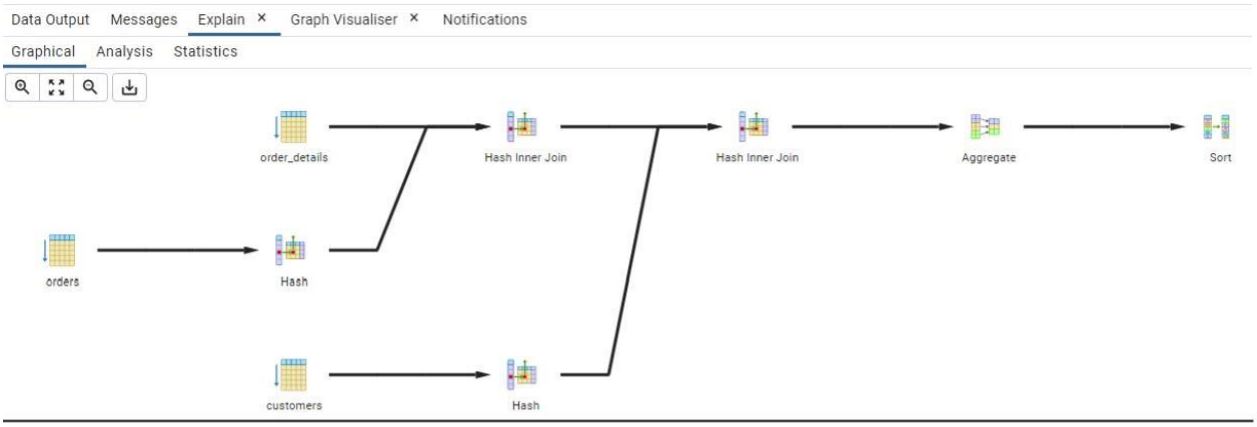
Data Output | Messages | Explain × | Graph Visualiser × | Notifications

Graphical | Analysis | Statistics

Parallelism: First, we execute the below query with the default cores. Since the minimum table size should be greater than 8 MB, we cannot add more workers per gather.

```
SELECT first_name, last_name, email, MAX(order_details.count) FROM customers
INNER JOIN orders USING(customer_id)
INNER JOIN order_details USING(order_id)
```

```
GROUP BY customer_id
HAVING MAX(order_details.count) IS NOT NULL
ORDER BY 4 DESC
```



Data Output | Messages | Explain | Graph Visualiser | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Sort (rows=61 loops=1)	61	1
2.	→ Aggregate (rows=61 loops=1) **Filter**: (max(order_details.count) IS NOT NULL) **Rows Removed by Filter**: 163 **Buckets**: **Batches**: **Memory Usage**: 73 kB	61	1
3.	→ Hash Inner Join (rows=500 loops=1) **Hash Cond**: (orders.customer_id = customers.customer_id)	500	1
4.	→ Hash Inner Join (rows=500 loops=1) **Hash Cond**: ((order_details.order_id)::text = (orders.order_id)::text)	500	1
5.	→ Seq Scan on order_details as order_details (rows=500 loops=1)	500	1
6.	→ Hash (rows=500 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 45 kB	500	1
7.	→ Seq Scan on orders as orders (rows=500 loops=1)	500	1
8.	→ Hash (rows=500 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 45 kB	500	1
9.	→ Seq Scan on customers as customers (rows=500 loops=1)	500	1

Explanation: As the first table customers gets scanned (E1), the order_details table gets scanned (E2) in the parallel. After E1 ends running, it sends the data to E2 to process. After processing of E2, it switches to perform GROUP BY operation in the parallel. This is how 2 servers run concurrently to achieve inter-operation parallelism across various operators in the query tree.

SQL Queries

1. List the top 10 customer's first_name, last_name, email and their respective count of orders. Results should be in descending order of the count.

```
SELECT
      first_name, last_name, email, MAX(order_details.count)
FROM
      customers
      INNER JOIN
      orders USING(customer_id)
      INNER JOIN order_details USING(order_id)
GROUP BY
      customer_id
HAVING
      MAX(order_details.count) IS NOT NULL
ORDER BY
      MAX(order_details.count) DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	first_name character varying	last_name character varying	email character varying	max numeric
1	Matthiew	Imms	nzamboninicx@msu.edu	15
2	Dione	Casero	juccellic8@walmart.com	15
3	Ozzie	Meneely	hmcilhattona4@discovery.com	15
4	Bonny	Richichi	amacevilly7d@dropbox.com	15
5	Coraline	Stiller	jdebell7m@ucsd.edu	15
6	Myrilla	Furley	araccio8u@cbslocal.com	14
7	Annice	Lawee	aambresin7i@netlog.com	14
8	Warren	Matityahu	dmoverley9f@vinaora.com	14
9	Teddie	Warman	scumminebj@businesswire.c...	14
10	Nerty	Boughtwood	gsissot2v@simplemachines.o...	14

2.

Print Top 10 order count split per shipping companies. Sort the results based on count, highest to lowest.

```
WITH cte AS
( SELECT
      orders.order_id, company_name
FROM
               orders
      FULL OUTER JOIN
shipments USING(shipment_id)
               FULL OUTER JOIN
    SHIPPING_COMPANY USING(shipping_company_id))

SELECT
               company_name, COUNT(cte.order_id)
FROM
               cte
WHERE
               order_id IS NOT NULL
GROUP BY
               company_name
ORDER BY 2 DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	company_name character varying	count bigint
1	Metz and Sons	7
2	Abbott, Spinka and Hermann	7
3	Rempel-Lynch	7
4	MacGyver Group	6
5	Donnelly-Fay	6
6	Skiles LLC	6
7	Schuppe and Sons	6
8	Bernier Group	6
9	Kirlin, Lowe and O'Reilly	5
10	Olson, Orn and Kautzer	5

3.

List the richest city of every country based on their spending on the website.

```
WITH cte1 AS    ( SELECT
                        COUNTRY, CITY, SUM(TOTAL_AMOUNT) AS Money_spent
FROM
                        PAYMENTS
                        JOIN
                        orders USING(payment_id)
                        JOIN
                        customers USING(customer_id)
                        JOIN
 GROUP                  address USING(address_id)
BY
                        CITY,COUNTRY),
cte2 AS
( SELECT
                        country, city, Money_spent,
DENSE_RANK() OVER(PARTITION BY COUNTRY ORDER BY Money_spent) AS rnk
 FROM           cte1)
SELECT *
FROM cte2
WHERE rnk = 1
ORDER BY money_spent DESC;
```

Data Output | Messages | Notifications

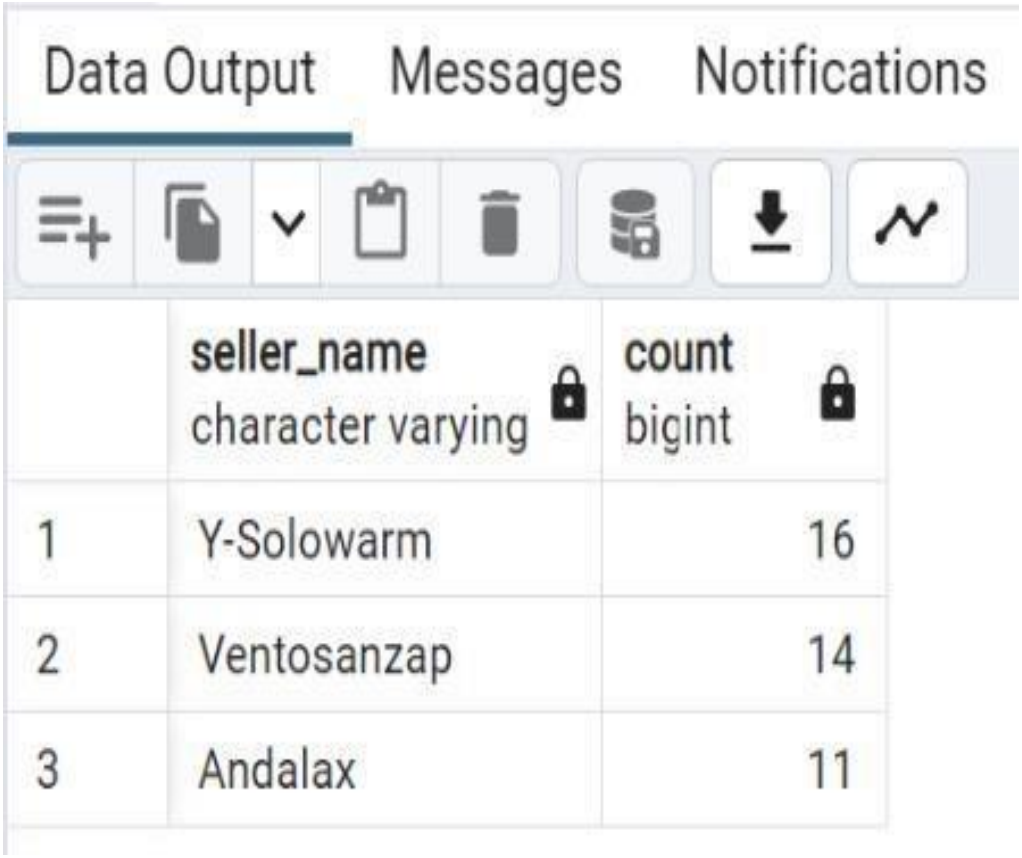
	country character varying	city character varying	money_spent numeric	rnk bigint
1	Albania	Zall-Herr	16655.05	1
2	Armenia	Bagratashen	15914.09	1
3	Haiti	Gros Morne	14117.50	1
4	Mauritius	Triolet	13756.51	1
5	Cambodia	Kampong Thom	11269.27	1
6	Spain	Vigo	10970.80	1
7	Montenegro	Rožaje	10689.75	1
8	Norway	Stavanger	10247.79	1
9	Colombia	Bagadó	9620.43	1
10	Nicaragua	Jinotepe	9200.06	1
11	South Korea	Kwangju	9048.88	1
12	Kazakhstan	Sarykemer	8690.73	1

Total rows: 73 of 73 Query complete 00:00:00.057

4.

Who are the TOP 3 vendors across the vendors based on their order size.

```
SELECT
                seller_name, COUNT(order_id)
FROM seller
                JOIN order_details USING(product_id)
                JOIN orders USING(order_id)
GROUP BY
                seller_name
ORDER BY 2 DESC
LIMIT 3;
```



	seller_name character varying	count bigint
1	Y-Solowarm	16
2	Ventosanzap	14
3	Andalax	11

5.

Classify each customer based on their age into 3 buckets, 90s Kids, Millenial and Oldi

```
SELECT
                customer_id, first_name, last_name,
                CASE
                  WHEN EXTRACT(YEAR FROM DOB) < 1981 THEN 'Gen X'
                  WHEN EXTRACT(YEAR FROM DOB) BETWEEN 1981 AND 1996
THEN 'Millenial'
                  ELSE 'Gen Z'
                END AS CustomerAgeBucket FROM
customers;
```

Data Output | Messages | Notifications

	customer_id [PK] numeric	first_name character varying	last_name character varying	customeragebucket text
1	1	Laryssa	Coggins	Gen Z
2	2	Hedwig	MacPhaden	Gen Z
3	3	Freeman	Rodliff	Gen X
4	4	Hersh	Ubsdale	Millenial
5	5	Lurline	Bengall	Gen X
6	6	Consuela	Sparey	Gen Z
7	7	Francklin	O'Noland	Millenial
8	8	Perle	Durant	Gen X
9	9	Maura	Kerrey	Millenial
10	10	Austin	Kermit	Gen Z

6. List down the top 10 products based on their selling amount, order by total amount.

```
SELECT      product.product_id, product.product_name,
SUM(total_amount)
FROM
                product JOIN order_details USING(product_id)
                        JOIN payments USING(order_id)
GROUP BY
                product_id, product_name;
```

Data Output | Messages | Notifications

	product_id [PK] character varying	product_name character varying	sum numeric
1	43-6935984	MISSHA M SIGNATURE REAL COMPLETE BB	3783.93
2	60-1194813	Sodium Polystyrene Sulfonate	7072.79
3	36-6472646	Nephrocaps	920.12
4	11-2082897	AMBROSIA TRIFIDA POLLEN	438.55
5	25-8437963	Lioresal	2168.22
6	24-9777344	Diclofenac Sodium	226.12
7	24-5898304	ACD-A	6372.75
8	73-5536954	MuSkel-S	7963.04
9	80-3207032	ZNP	1425.72
10	72-4895230	Tacrolimus	5693.77
11	75-7187408	nystatin	4322.31
12	85-2435431	Mineral oil	5253.16
13	65-5623426	Necon	877.72
14	47-0309832	Midodrine HCl	4225.79
15	95-3875616	OXACILLIN	3190.07

Total rows: 477 of 477 | Query complete 00:00:00.065

7. How many orders are delivered or shipped as of today?

```
SELECT
                status, COUNT(shipment_id)
FROM shipments
group by status;
```

Data Output | Messages | Notifications

	status character varying	count bigint
1	Delivered	491
2	Shipped	9

8. Bucket all the sellers based on their seller rating.

```
SELECT
                seller_rating,
                COUNT(seller_id)
FROM
                seller
GROUP BY
                seller_rating
ORDER BY    seller_rating
DESC, 2 DESC;
```

Data Output | Messages | Notifications

	seller_rating numeric	count bigint
1	10	27
2	9	52
3	8	52
4	7	51
5	6	59
6	5	39
7	4	53
8	3	56
9	2	50
10	1	41
11	0	20

9. How many customers did not provide State and Zip while ordering from the website?

```
SELECT
        COUNT(*) AS bad_address
FROM
        address INNER JOIN customers USING(address_id)
WHERE
        address.state IS NULL
        AND ZIP IS NULL
```

Data Output | Messages | Notifications

	bad_address bigint
1	277

10. Provide the distribution of payment_modes across all the orders created on the website.

```
Select
        payment_mode, COUNT(payment_mode)
FROM
        payments
GROUP BY
        payment_mode
```

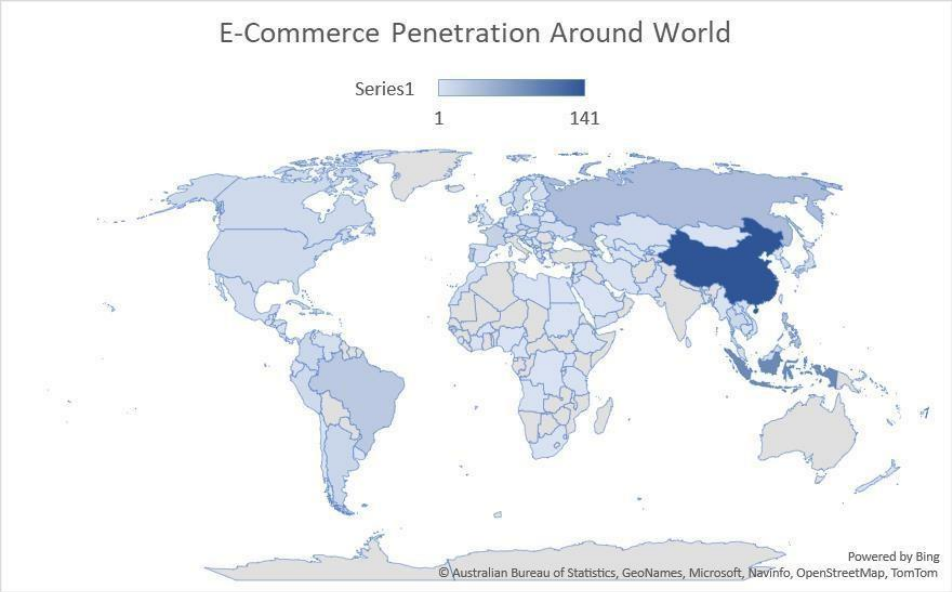
Data Visualization

Visualize number of members who have taken the policies across different countries in the world.

Data Output | Messages | Notifications

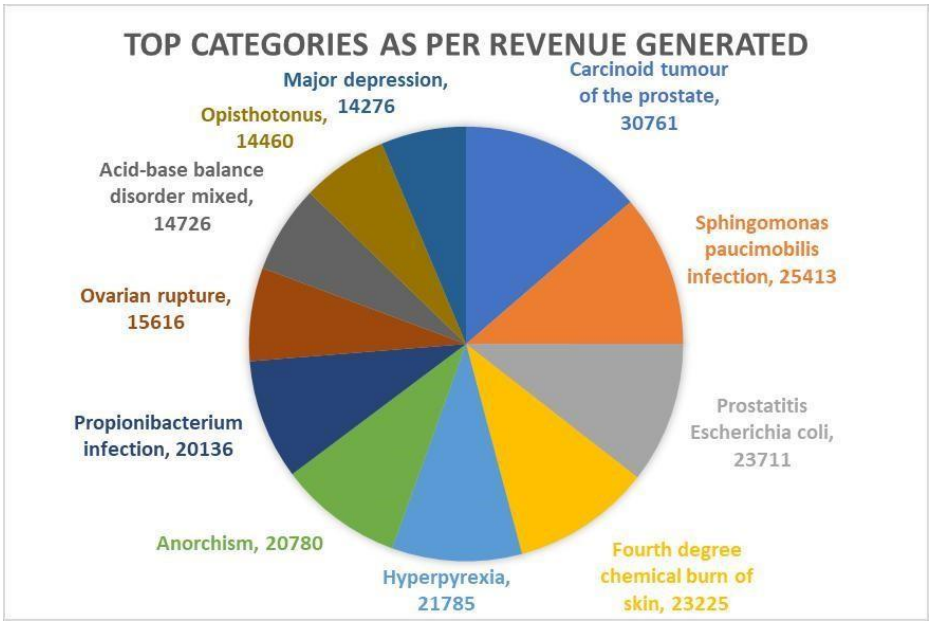
	payment_mode character varying	count bigint
1	wallet	123
2	COD	127
3	Card	125
4	ePay	125

As per the above plot, we observe that highest orders come in from China, i.e., 105, followed by Indonesia

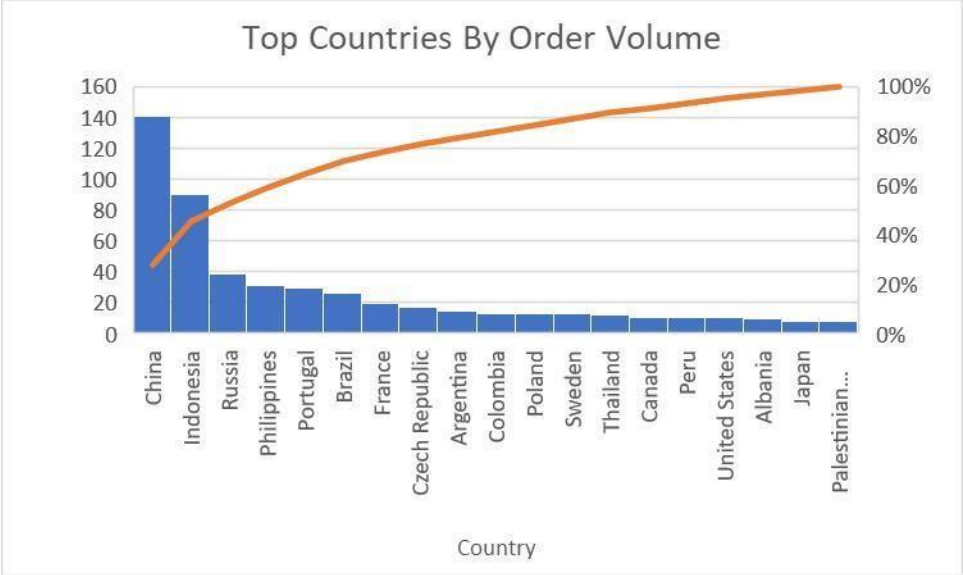


Visualize Top categories based on revenue generated.

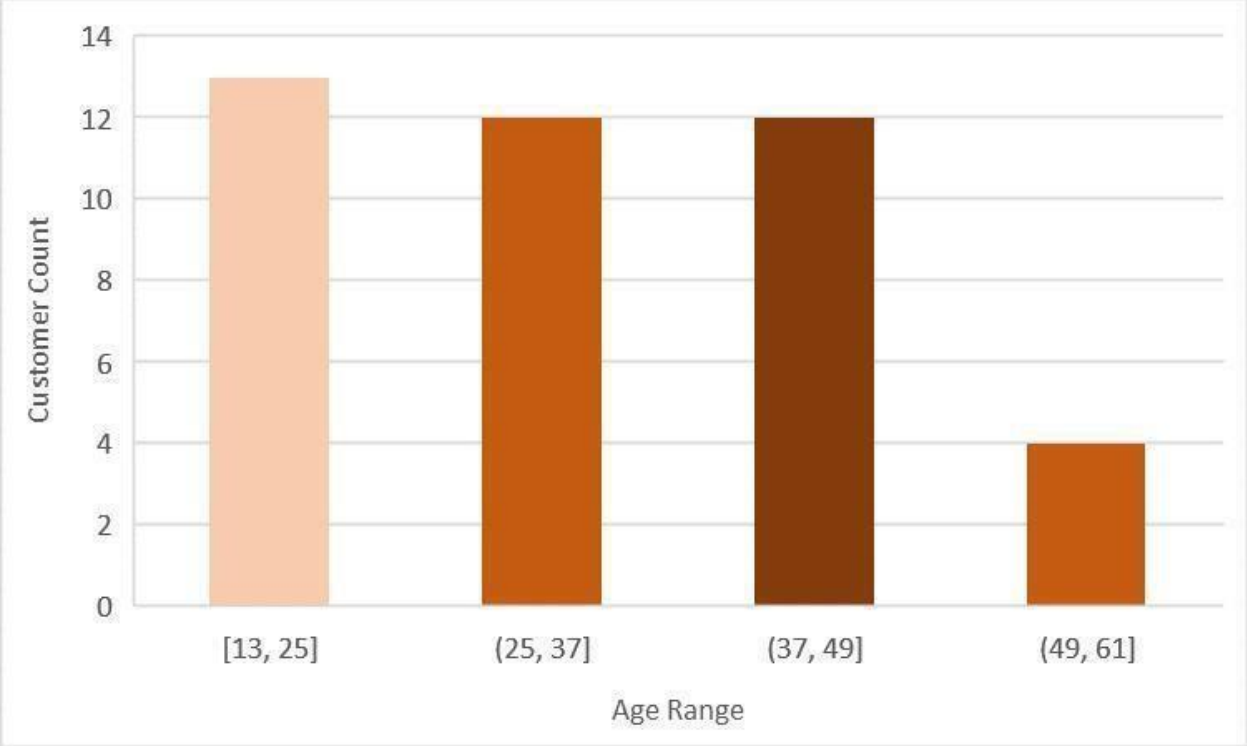
Visualize Top Countries by ordered volume



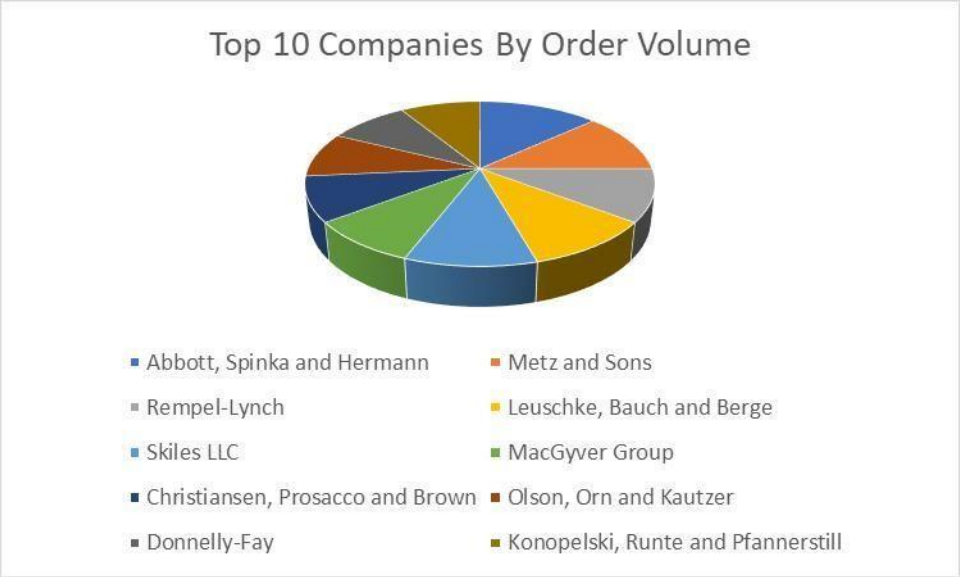
Visualize



customer's age customers based on their orders



Visualize Top 10 companies based on their volume of order processed


Top 10 Companies By Order Volume
Abbott, Spinka and Hermann
Metz and Sons
Rempel-Lynch
Leuschke, Bauch and Berge
Skiles LLC
MacGyver Group
Christiansen, Prosacco and Brown
Olson, Orn and Kautzer
Donnelly-Fay
Konopelski, Runte and Pfannerstill