Final Project

Part 2:

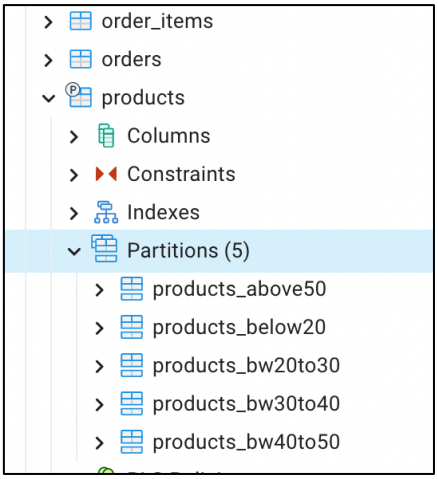
Implementing fragmentation and replication techniques for optimizing the performance of the distributed database system in your chosen topic.

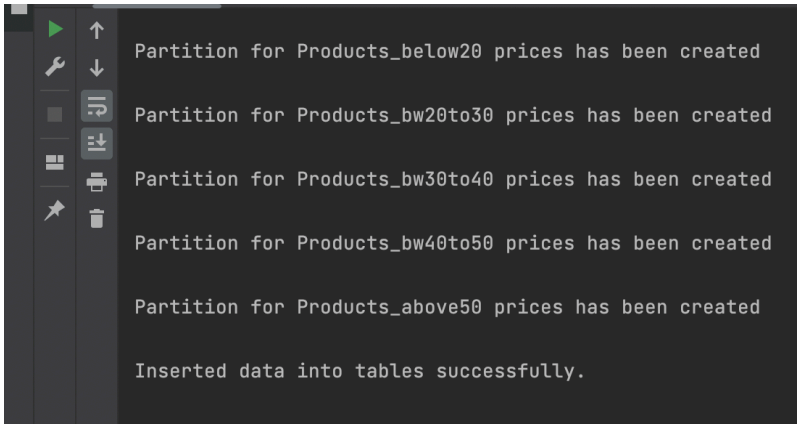
For our E-commerce database we have implemented Horizontal fragmentation on the products table for the 'Price' field. We have partitioned the table into five partitions to insert data based on various price ranges of the products.

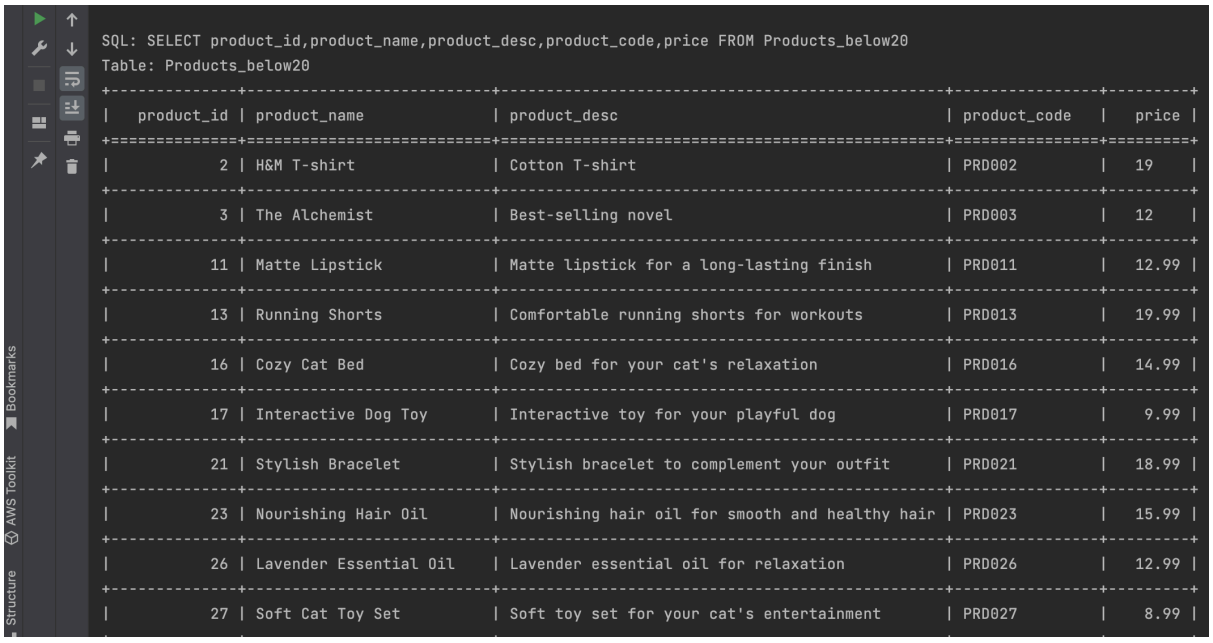
The script Task2.py will implement this functionality by first creating the partition tables and then importing more data into the products and its related tables to check if the data is split up properly.

Partition Table	**Price Range (Min, Max)**	**SQL Statement**	**Explanation**
Products_below20	0, 20	**CREATE TABLE Products_below20 PARTITION OF products FOR VALUES FROM (0) TO (20);**	Partition for products with prices below 20
Products_bw20to30	20, 30	**CREATE TABLE Products_bw20to30 PARTITION OF products FOR VALUES FROM (20) TO (30);**	Partition for products with prices between 20 and 30
Products_bw30to40	30, 40	**CREATE TABLE Products_bw30to40 PARTITION OF products FOR VALUES FROM (30) TO (40);**	Partition for products with prices between 30 and 40
Products_bw40to50	40, 50	**CREATE TABLE Products_bw40to50 PARTITION OF products FOR VALUES FROM (40) TO (50);**	Partition for products with prices between 40 and 50
Products_above50	50, 1000 (Infinity)	**CREATE TABLE Products_above50 PARTITION OF products FOR VALUES FROM (50) TO (1000);**	Partition for products with prices above 50 (using **Infinity** as the upper bound)

The screenshots of Horizontal fragmentation are as below:

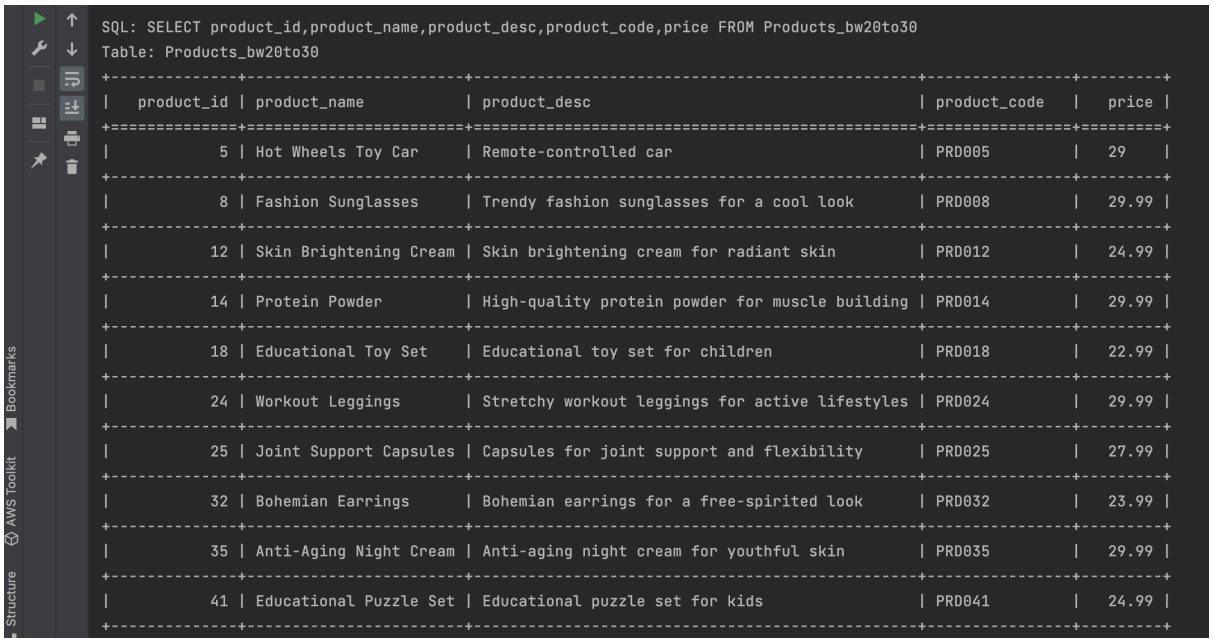






SQL: SELECT product_id,product_name,product_desc,product_code,price FROM Products_below20
Table: Products_below20

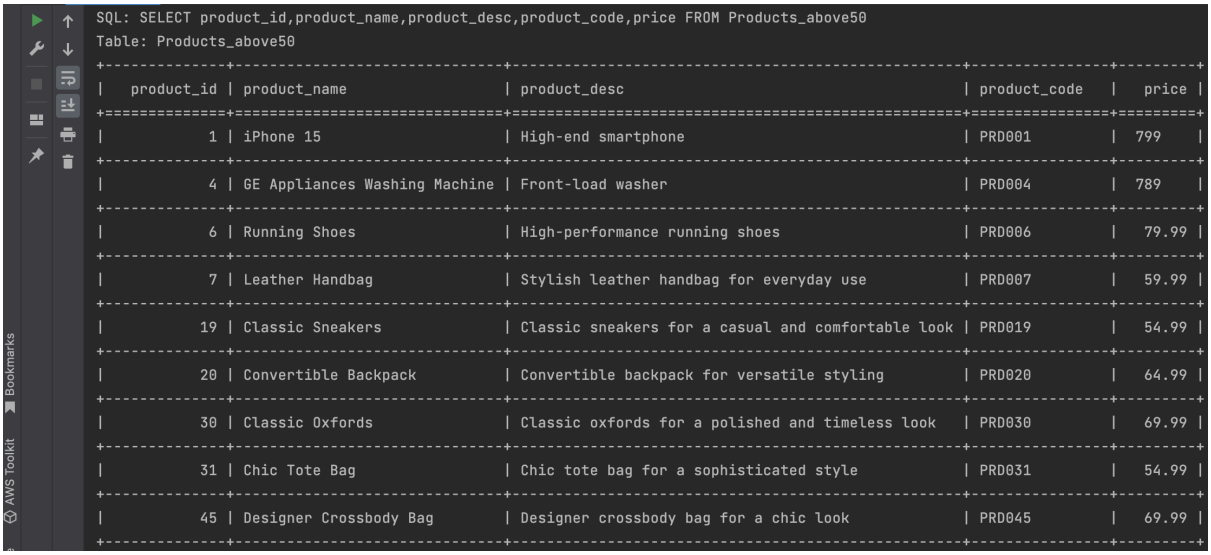
product_id	product_name	product_desc	product_code	price
2	H&M T-shirt	Cotton T-shirt	PRD002	19
3	The Alchemist	Best-selling novel	PRD003	12
11	Matte Lipstick	Matte lipstick for a long-lasting finish	PRD011	12.99
13	Running Shorts	Comfortable running shorts for workouts	PRD013	19.99
16	Cozy Cat Bed	Cozy bed for your cat's relaxation	PRD016	14.99
17	Interactive Dog Toy	Interactive toy for your playful dog	PRD017	9.99
21	Stylish Bracelet	Stylish bracelet to complement your outfit	PRD021	18.99
23	Nourishing Hair Oil	Nourishing hair oil for smooth and healthy hair	PRD023	15.99
26	Lavender Essential Oil	Lavender essential oil for relaxation	PRD026	12.99
27	Soft Cat Toy Set	Soft toy set for your cat's entertainment	PRD027	8.99



SQL: SELECT product_id,product_name,product_desc,product_code,price FROM Products_bw20to30
Table: Products_bw20to30

product_id	product_name	product_desc	product_code	price
5	Hot Wheels Toy Car	Remote-controlled car	PRD005	29
8	Fashion Sunglasses	Trendy fashion sunglasses for a cool look	PRD008	29.99
12	Skin Brightening Cream	Skin brightening cream for radiant skin	PRD012	24.99
14	Protein Powder	High-quality protein powder for muscle building	PRD014	29.99
18	Educational Toy Set	Educational toy set for children	PRD018	22.99
24	Workout Leggings	Stretchy workout leggings for active lifestyles	PRD024	29.99
25	Joint Support Capsules	Capsules for joint support and flexibility	PRD025	27.99
32	Bohemian Earrings	Bohemian earrings for a free-spirited look	PRD032	23.99
35	Anti-Aging Night Cream	Anti-aging night cream for youthful skin	PRD035	29.99
41	Educational Puzzle Set	Educational puzzle set for kids	PRD041	24.99

```
SQL: SELECT product_id,product_name,product_desc,product_code,price FROM Products_bw40to50
Table: Products_bw40to50
+-------------+--------------------+-------------------------------------------------+---------------+---------+
|   product_id | product_name        | product_desc                                    | product_code  |   price |
+=============+====================+=================================================+===============+=========+
|           10 | Travel Backpack     | Durable travel backpack for on-the-go adventures | PRD010        |   49.99 |
+-------------+--------------------+-------------------------------------------------+---------------+---------+
|           22 | Aviator Sunglasses  | Classic aviator sunglasses for a timeless look  | PRD022        |   44.99 |
+-------------+--------------------+-------------------------------------------------+---------------+---------+
|           36 | Badminton Racket Set | Badminton racket set for recreational play     | PRD036        |   49.99 |
+-------------+--------------------+-------------------------------------------------+---------------+---------+
|           44 | Classic Loafers     | Classic loafers for a timeless style            | PRD044        |   49.99 |
+-------------+--------------------+-------------------------------------------------+---------------+---------+
```



```
SQL: SELECT product_id,product_name,product_desc,product_code,price FROM Products_above50
Table: Products_above50
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|   product_id | product_name                | product_desc                                    | product_code  |   price |
+=============+=============================+=================================================+===============+=========+
|            1 | iPhone 15                   | High-end smartphone                             | PRD001        |  799    |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|            4 | GE Appliances Washing Machine | Front-load washer                             | PRD004        |  789    |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|            6 | Running Shoes               | High-performance running shoes                  | PRD006        |   79.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|            7 | Leather Handbag             | Stylish leather handbag for everyday use        | PRD007        |   59.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           19 | Classic Sneakers            | Classic sneakers for a casual and comfortable look | PRD019        |   54.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           20 | Convertible Backpack        | Convertible backpack for versatile styling      | PRD020        |   64.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           30 | Classic Oxfords             | Classic oxfords for a polished and timeless look | PRD030        |   69.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           31 | Chic Tote Bag               | Chic tote bag for a sophisticated style         | PRD031        |   54.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           45 | Designer Crossbody Bag      | Designer crossbody bag for a chic look          | PRD045        |   69.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
```

```
SQL: SELECT product_id,product_name,product_desc,product_code,price FROM Products_bw30to40
Table: Products_bw30to40
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|   product_id | product_name                | product_desc                                    | product_code  |   price |
+=============+=============================+=================================================+===============+=========+
|            9 | Statement Necklace          | Statement necklace for a bold fashion statement | PRD009        |   39.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           15 | Aromatherapy Diffuser       | Aromatherapy diffuser for a calming atmosphere  | PRD015        |   34.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           33 | Classic Wayfarer Sunglasses | Classic wayfarer sunglasses for a timeless appeal | PRD033        |   39.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           39 | Cozy Cat Tower              | Cozy tower for your cat's comfort               | PRD039        |   39.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
|           42 | Stylish Kids' Jacket        | Stylish jacket for fashionable kids             | PRD042        |   34.99 |
+-------------+-----------------------------+-------------------------------------------------+---------------+---------+
```

Vertical Fragmentation:

We've performed vertical fragmentation on the "ADDRESS" table, considering that certain information, such as address_line1, and postal_code, is frequently accessed, while details like address_line2, city, country, and address_type are rarely needed. Consequently, we've created two fragments of the table:

1. **Address_Basic_Info Fragment:**
    - This fragment contains the essential information that is frequently accessed, including address_id, address_line1, postal_code, etc.
2. **Address_Rare_Info Fragment:**

- This fragment includes details that are less frequently needed, such as address_line2, city, country, and address_type.

This strategic vertical fragmentation aims to enhance the performance of our distributed database, optimizing data retrieval for commonly accessed elements and minimizing the overhead associated with less frequently used information.

Results of vertical fragmentation:

Fragmented Table: Address Basic info Fragment)

address_id	address_line1	address_type	postal_code
1	123 Main St	Billing	12345
2	456 Elm St	Shipping	54321
3	789 Oak St	Billing	98765
4	101 Pine Rd	Shipping	45678
5	321 Cedar Ave	Billing	65432
6	611 Rural Rd	Billing	12345
7	654 Forest Ave	Shipping	54321
8	987 Farmer Ave	Billing	98765
9	201 Salado Rd	Shipping	45678
10	712 Jentily Dr	Billing	65432

Fragmented Table: Address rare info fragment

address_id	address_line2	city	country
1		Cityville	USA
2	Apt 23H	Townsville	USA
3		Villagetown	Canada
4	Apt 5J	Hamlet City	USA
5		Forestville	USA
6		Cityville	USA
7	Apt 2B	Townsville	USA
8		Villagetown	Canada
9	Suite 100	Hamlet City	USA
10		Forestville	USA

Replication:

As part of replication setup we have configured a master-slave replication model to increase data availability. We have implemented the setup by running both master and slave servers as docker containers.

Steps implemented:

1) Created a master docker container named master_db with a primary dev database called mydatabase and a user password for the same.

2) Set the master's postgresql.conf configuration file as below, making the slave to accept only read only connections from users.

```
wal_level = hot_standby
archive_mode = on
archive_command = 'cd .'
max_replication_slots=2
max_wal_senders = 3
wal_keep_size = 16MB
hot_standby = on
hot_standby_feedback = on
```

- **wal_level = hot_standby**: This setting determines the amount of information written to the Write-Ahead Logging (WAL) system. The **hot_standby** level enables the recording of enough information to support read-only queries on a standby server.
- **archive_mode = on**: When set to **on**, this enables archiving of WAL segments. This is useful for creating a backup of the Write-Ahead Log to a different location, allowing for point-in-time recovery on a standby server.
- **archive_command = 'cd .'**:This is the command that PostgreSQL will execute to archive a WAL segment. In this case, it is set to **cd .**, essentially doing nothing. It's a placeholder setting indicating that archiving is turned on, but no specific archiving action is taken. In a production environment, you would replace this command with a script or command that archives WAL segments to a designated location.
- **max_replication_slots=2**: Sets the maximum number of replication slots available for standby servers. Replication slots are used to track which WAL segments have been sent to and received by standby servers.
- **max_wal_senders = 3**:Sets the maximum number of simultaneously defined replication connections. This includes both physical replication (streaming replication) and logical replication connections. In our case, it allows up to 3 replication connections.
- **wal_keep_size = 16MB**: Specifies the minimum amount of WAL data to retain in the pg_wal directory before recycling old WAL files. This setting ensures that enough WAL data is kept to support standby servers that might fall behind due to temporary disconnections.
- **hot_standby = on**: This setting enables the hot standby feature, allowing a standby server to be used for read-only queries. It enables the server to start in standby mode and apply changes from the primary server.
- **hot_standby_feedback = on**:When set to **on**, this setting allows a standby server to send information back to the primary about the oldest transaction it is still actively using. This information is used to avoid removing WAL segments that might still be needed by a standby server.

These settings collectively configure PostgreSQL to operate in a streaming replication mode where a standby server can connect to a primary server, receive changes, and remain in sync for read-only queries

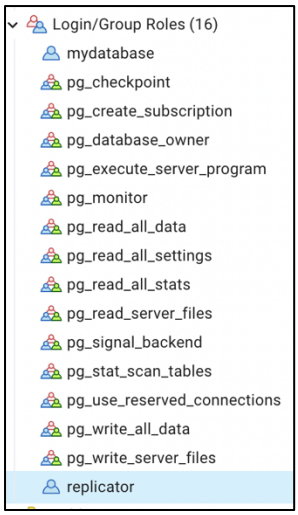
3) Using **pg_basebackup** command created a base backup of a PostgreSQL database cluster for setting up streaming replication, where the master_db replicates its data to the standby instance which would be our slave DB.
pg_basebackup -h 127.0.0.1 -p5432 -D /tmp/postgresslave -U replicator -P -v
This created a postgresslave folder in the tmp directory.

```
(base) priyadarshiniramakrishnan@Priyadarshinis-MacBook-Pro ~ % ls /tmp
Acrobat-2bc500aeed0f77655f6a21fcd53fc852        com.adobe.reader.rna.0.1f5.DC               com.apple.launchd.xE6aZefpAG                postgresslave
Acrobat-4b95be924477a6260e73c53df89e7078        com.adobe.reader.rna.115f2.1f5              com.google.Keystone                         powerlog
com.adobe.acrobat.rna.RdrCefBrowserLock.DC      com.adobe.reader.rna.ee7.1f5                postgres
```

4) creating a PostgreSQL user named "replicator" with the ability to log in, the **REPLICATION** privilege for replication purposes, and the password "replpass." This

user is typically used for setting up replication, where our slave server connects to the master server to replicate its data changes.

```
CREATE USER replicator REPLICATION LOGIN ENCRYPTED PASSWORD 'replpass';
```



5) Configured the postgresslave's conf file appropriately to setup replication.

```
primary_conninfo = 'host=db port=5432 user=replicator password=replpass'
wal_level = hot_standby
archive_mode = on
archive_command = 'cd .'
primary_slot_name = 'standby_replication_slot'
max_wal_senders = 3
hot_standby = on
hot_standby_feedback = on
#restore_command = 'cd .'
```

- **primary_conninfo = 'host=db port=5432 user=replicator password=replpass'**: Specifies the connection information needed for the standby server to connect to the primary server.
- **wal_level = hot_standby**: Similar to the explanation for the master server, this setting determines the amount of information written to the Write-Ahead Logging (WAL) system.
- The **hot_standby** level enables the recording of enough information to support read-only queries on the standby server.
- **archive_mode = on**: Enables archiving of WAL segments on the standby server. Although the standby server primarily uses streaming replication, enabling archiving allows for additional backup options and point-in-time recovery.
- **archive_command = 'cd .'**: The command that PostgreSQL will execute on the standby server to archive a received WAL segment. In this case, it is set to **cd .**, which essentially does nothing. This is similar to the primary server's configuration, where it is a placeholder indicating that archiving is turned on but no specific archiving action is taken.
- **primary_slot_name = 'standby_replication_slot'**: Specifies the name of the replication slot on the primary server to be used by the standby server. This setting ensures that the standby server connects to the correct replication slot on the primary.

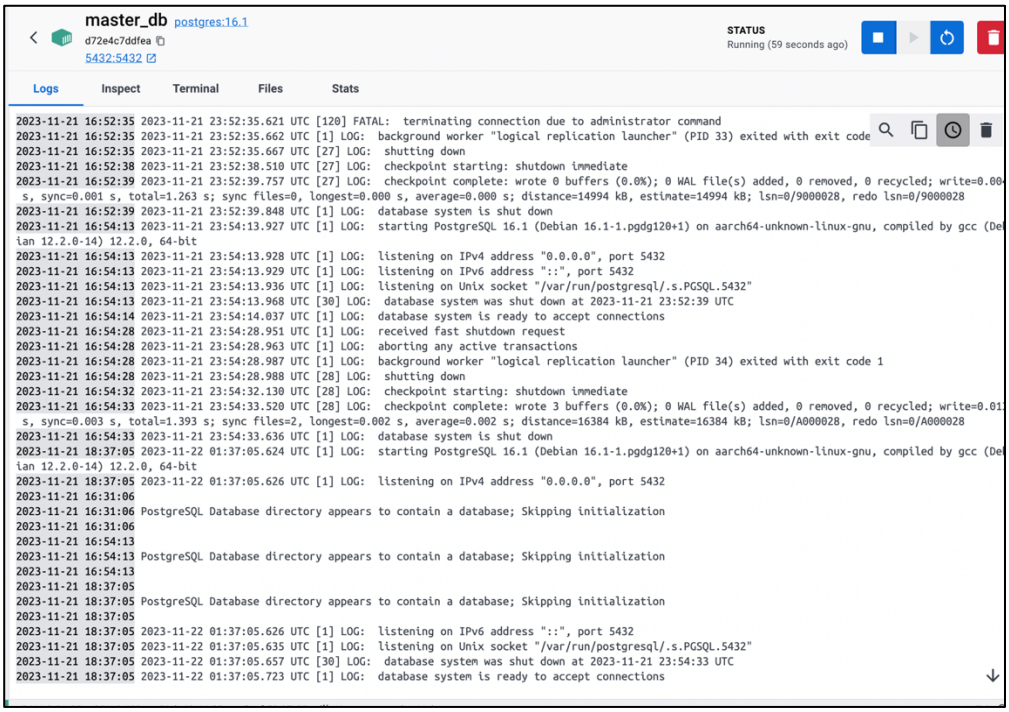
- **max_wal_senders = 3**: Sets the maximum number of simultaneously defined replication connections on the standby server. In this case, it allows up to 3 replication connections to the primary server.
- **hot_standby = on**: Enables the hot standby feature on the standby server, allowing it to be used for read-only queries. It enables the server to start in standby mode and apply changes from the primary server.
- **hot_standby_feedback = on**: When set to **on**, allows the standby server to send information back to the primary about the oldest transaction it is still actively using. This information is used to avoid removing WAL segments on the primary server that might still be needed by the standby server.

These settings, when applied to our standby server, ensure that it can connect to the primary server, receive changes through streaming replication, and operate in a read-only mode for queries.

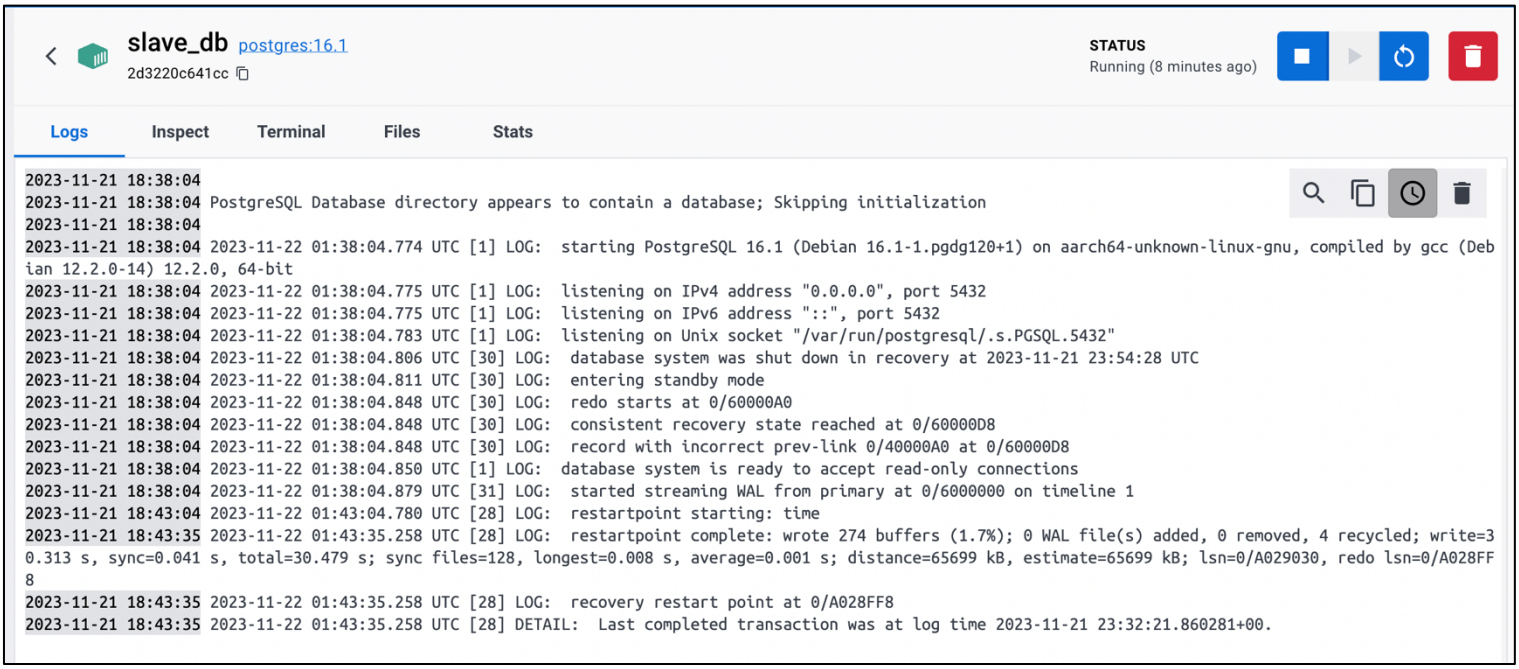
6) Created a slave_db in docker that accepts read-only connections from the users. It Links the container "master_db" to the “slave_db” being created and assigns an alias "db" to it. This allows the "slave_db" container to communicate with the "master_db" container.
docker run --name slave_db -e POSTGRES_USER=mydatabase -e POSTGRES_PASSWORD=mydatabase -v /tmp/postgresslave:/var/lib/postgresql/data --link master_db:db -it postgres:16.1

```
(base) priyadarshiniramakrishnan@Priyadarshinis-MacBook-Pro ~ % docker run --name slave_db -e POSTGRES_USER=m
 master_db:db -it postgres:16.1


PostgreSQL Database directory appears to contain a database; Skipping initialization

2023-11-22 01:38:04.774 UTC [1] LOG:  starting PostgreSQL 16.1 (Debian 16.1-1.pgdg120+1) on aarch64-unknown-l
2023-11-22 01:38:04.775 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2023-11-22 01:38:04.775 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2023-11-22 01:38:04.783 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2023-11-22 01:38:04.806 UTC [30] LOG:  database system was shut down in recovery at 2023-11-21 23:54:28 UTC
2023-11-22 01:38:04.811 UTC [30] LOG:  entering standby mode
2023-11-22 01:38:04.848 UTC [30] LOG:  redo starts at 0/60000A0
2023-11-22 01:38:04.848 UTC [30] LOG:  consistent recovery state reached at 0/60000D8
2023-11-22 01:38:04.848 UTC [30] LOG:  record with incorrect prev-link 0/40000A0 at 0/60000D8
2023-11-22 01:38:04.850 UTC [1] LOG:  database system is ready to accept read-only connections
2023-11-22 01:38:04.879 UTC [31] LOG:  started streaming WAL from primary at 0/6000000 on timeline 1
```

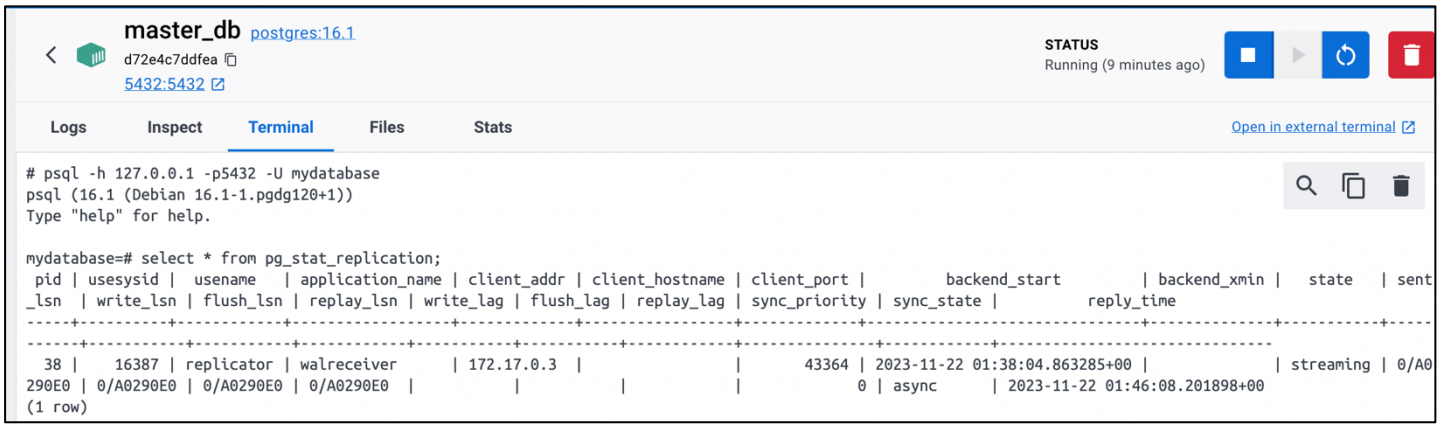
7) Master_db logs:



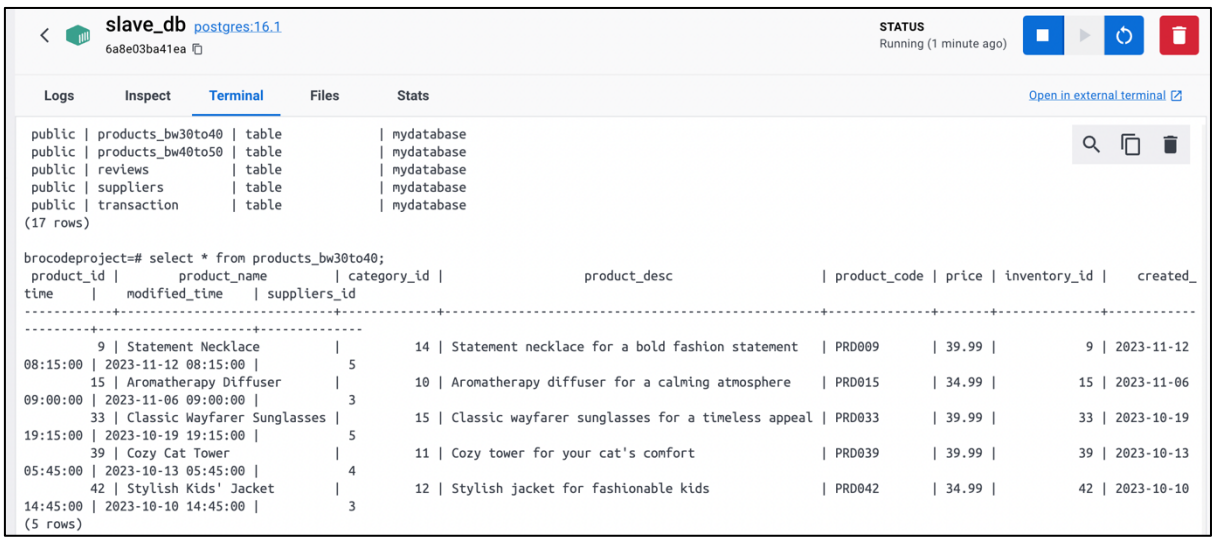
8) Slave_db logs:



9) Pg_stat_replication logs of master_db:



10) Querying the slave_db:



```
mydatabase=# \c brocodeproject
You are now connected to database "brocodeproject" as user "mydatabase".
brocodeproject=# \dt
                    List of relations
 Schema |       Name         |       Type         |   Owner
--------+--------------------+--------------------+------------
 public | address            | table              | mydatabase
 public | categories         | table              | mydatabase
 public | contact_details    | table              | mydatabase
 public | customer           | table              | mydatabase
 public | delivery           | table              | mydatabase
 public | inventory          | table              | mydatabase
 public | order_items        | table              | mydatabase
 public | orders             | table              | mydatabase
 public | products           | partitioned table  | mydatabase
 public | products_above50   | table              | mydatabase
 public | products_below20   | table              | mydatabase
 public | products_bw20to30  | table              | mydatabase
 public | products_bw30to40  | table              | mydatabase
 public | products_bw40to50  | table              | mydatabase
 public | reviews            | table              | mydatabase
 public | suppliers          | table              | mydatabase
 public | transaction        | table              | mydatabase
(17 rows)

brocodeproject=#
```