PostgreSQL

Customer Orders Database on AWS RDS

This project involves creating a cloud-hosted customer order database on AWS RDS with PostgreSQL. The database stores customer and order information, with key queries to analyze data. It showcases skills in database design, SQL query execution, and cloud deployment, and is documented on GitHub and Notion for documentation and publication.

Hi, I'm Chris, and thank you for checking out my work! I'm hopeful this project exemplifies my knowledge of SQL and my passion for leveraging this language to optimize both professional and personal endeavors.

"Let's take a look at how I built this out. This was my first time creating my own database, and it turned out to be an incredibly fun and rewarding hands-on learning experience. I enjoyed the process of designing, setting up, and working with SQL in the cloud, and it gave me valuable insights into database management and query optimization."

AWS RDS Configuration

Initializing Aurora PostgreSQL Engine

For this project, I set up an AWS RDS instance using the PostgreSQL engine, selecting the most practical configuration for short-term use. I configured the necessary settings, including but not limited to the database name, username, and password, to establish a cloud-hosted environment for the customer order database.

Engine: Aurora PostgreSQL (*v 16.6*)

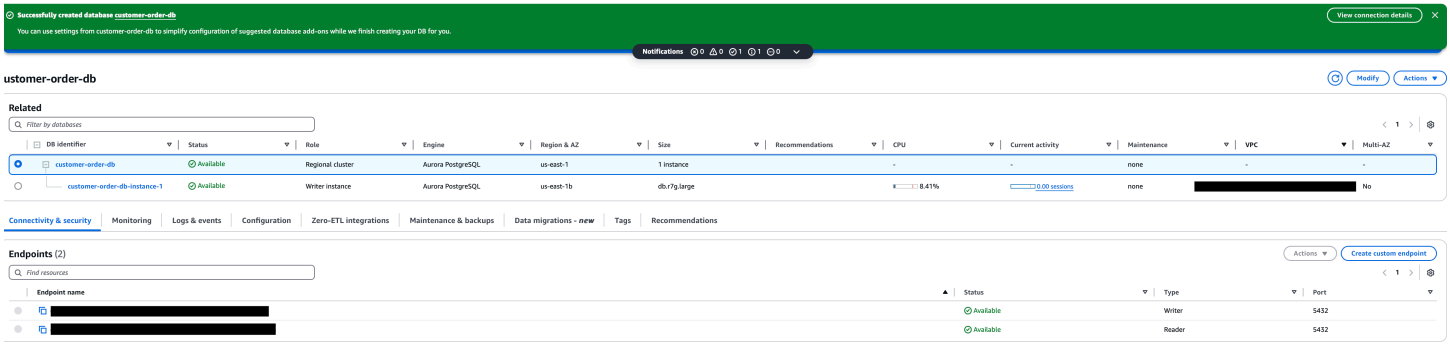
Template: DEV/Test

Cluster Configuration: Aurora Standard (*Cost Effective; No Output*)

Instance Configuration: db.r7g.large

Public Access: Yes (*Allowing me to connect to RDS Instance via pgAdmin*)

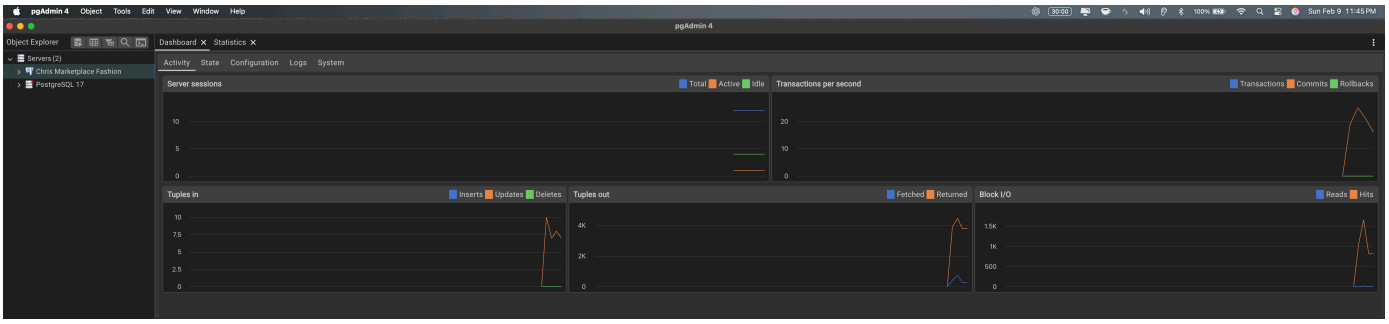
VPC (firewall**):** Allows inbound traffic on (Port ***5432***)



Caption

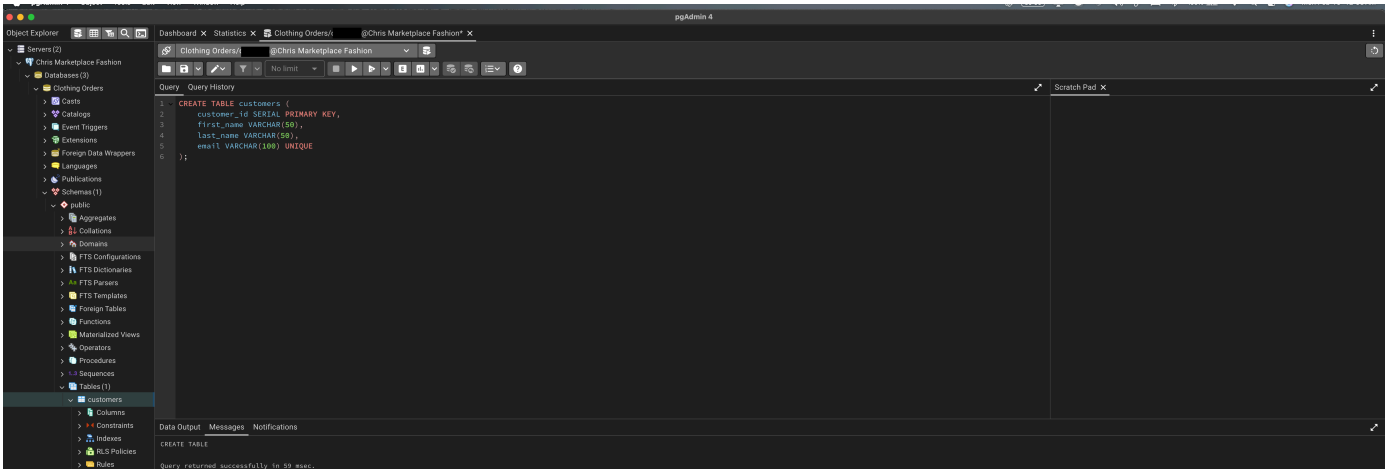
Connecting RDS Instance via pgAdmin

During the setup of my AWS RDS PostgreSQL instance, I faced a “connection timeout expired” error in pgAdmin. After confirming that the instance and public access were set up correctly, I realized my IP wasn’t added to the security group’s inbound rules. To troubleshoot, I used Hyper (*terminal application*) to install Homebrew and Telnet. By running Telnet to test the connection to the RDS endpoint, I identified the issue. Once I added my public IP (verified via curl ***ifconfig.me***) to the security group, I successfully connected to the database through pgAdmin and proceeded with my work.

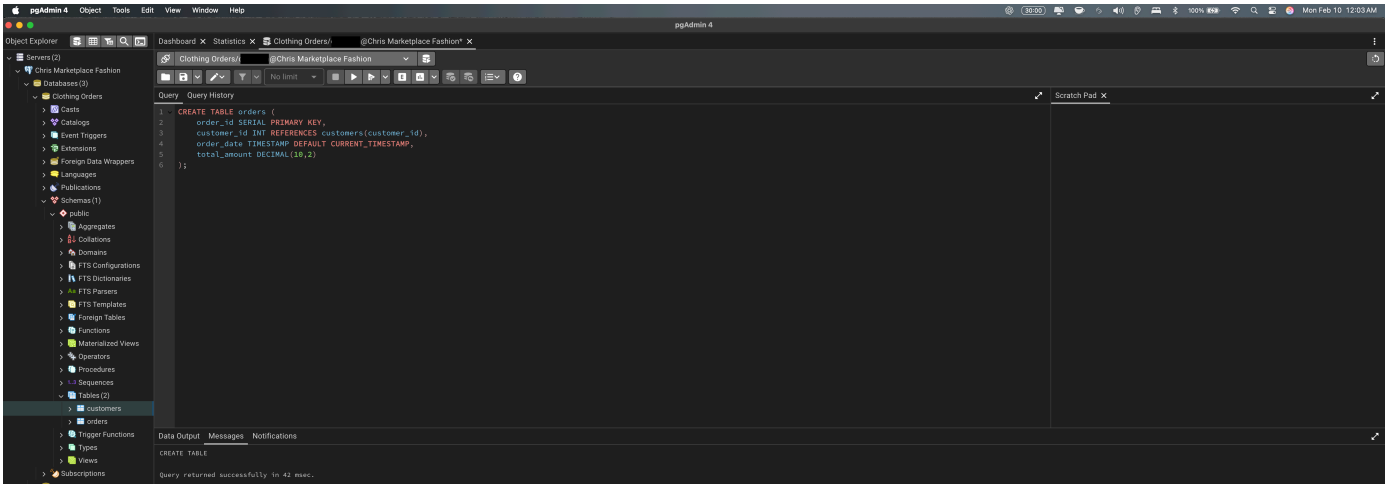


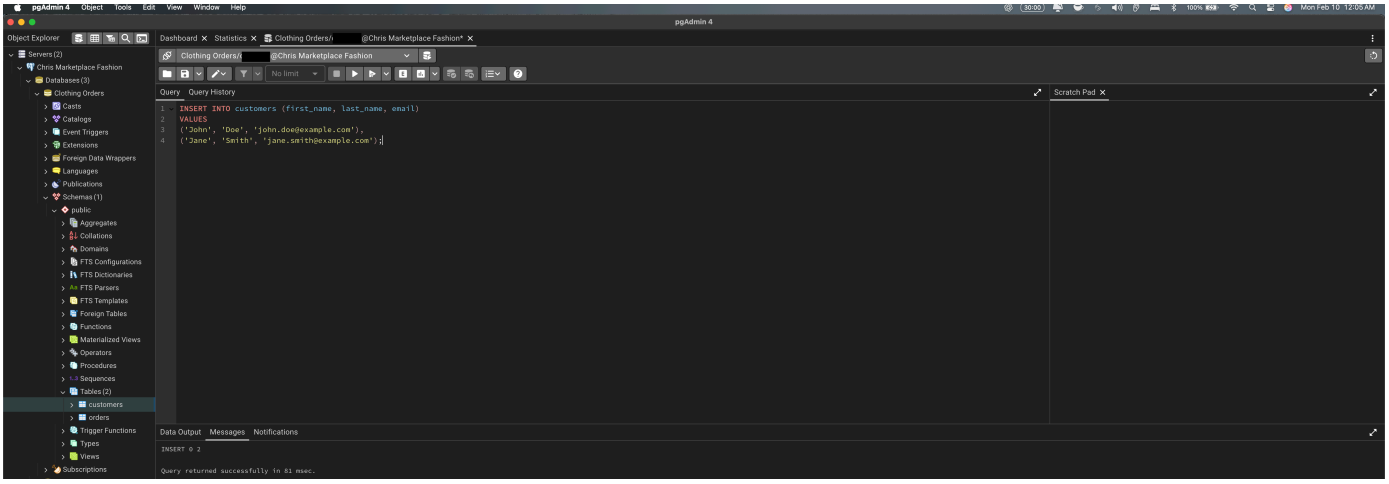
Configuring Database Schema

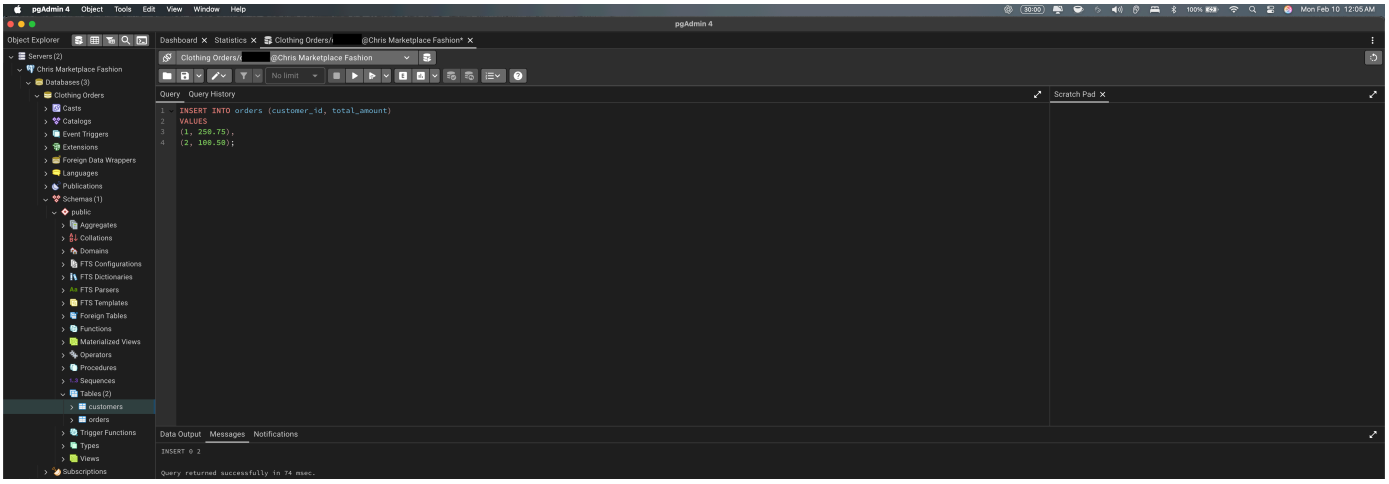
Once connected, I created the database schema with two tables: customers and orders. The customers table tracks customer information, while the orders table links customer orders to their respective details. After setting up the schema, I inserted sample data into both tables, representing two customers and their corresponding orders. Finally, I used an INNER JOIN query to retrieve and display related data from both tables, ensuring everything was correctly linked and functional.



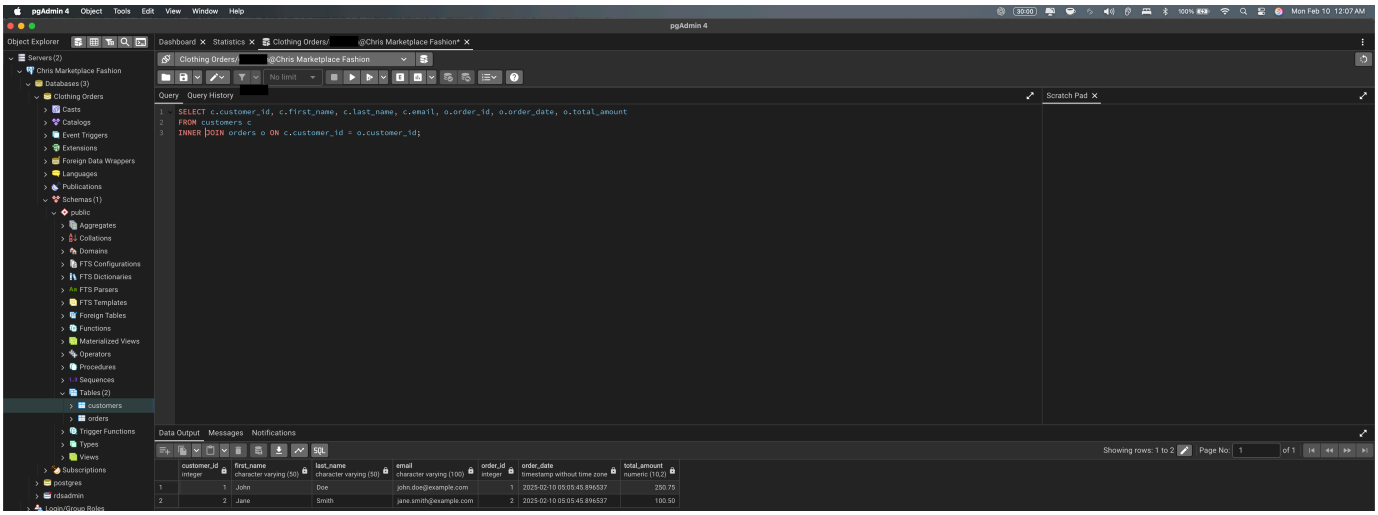
Customer Orders Database on AWS RDS (PostgreSQL)

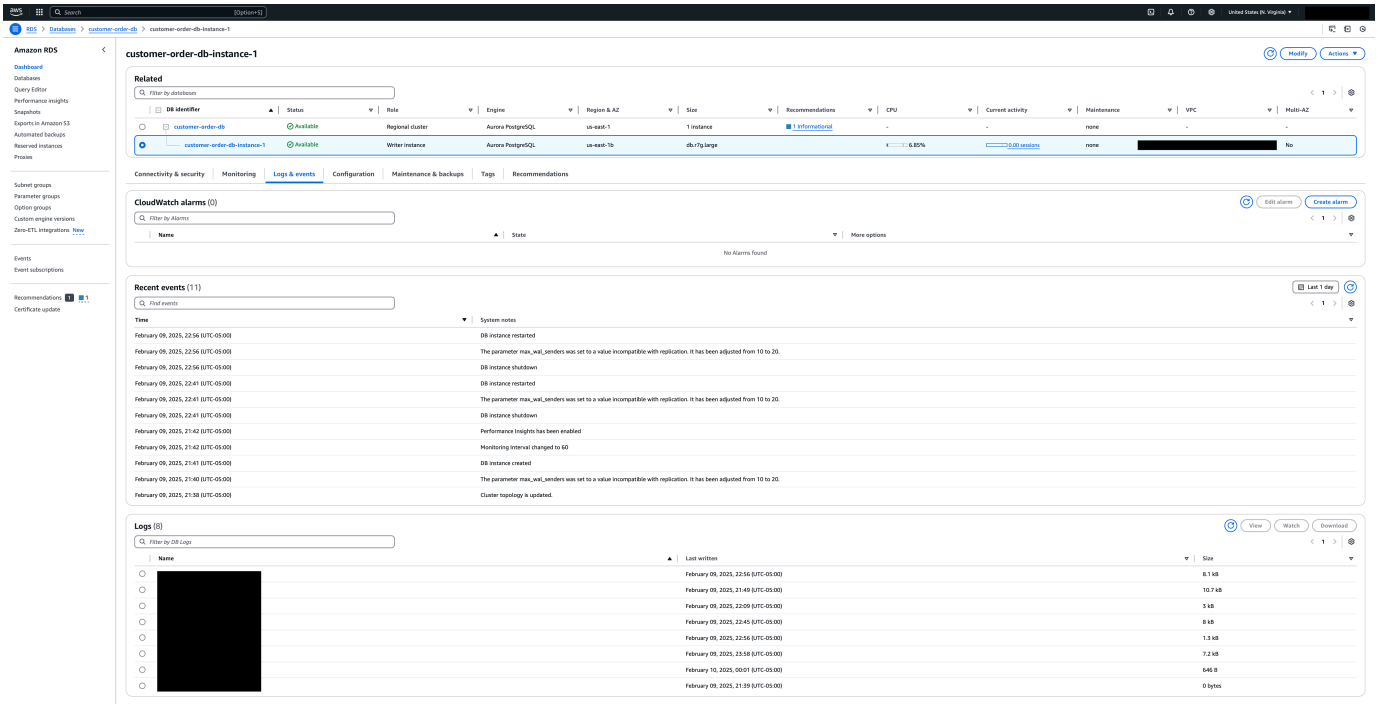






Customer Orders Database on AWS RDS (PostgreSQL)





Caption

Harnessing the Power of PostgreSQL in AWS

This project has not only solidified my foundational understanding of cloud databases and AWS, but it has also sharpened my problem-solving and technical troubleshooting abilities, which are critical in support and operations roles. The experience gained will be invaluable as I pursue more complex challenges, both in AWS cloud architecture and technical support. I am eager to apply these skills in real-world environments, where I can optimize cloud infrastructures and provide comprehensive support solutions to organizations. By advancing my knowledge further with more intricate database projects, testing new features, and exploring integrations with various AWS services, I am confident this expertise will significantly enhance my ability to contribute to the ongoing success of any technical team. This journey is just beginning, and I look forward to refining these skills and embracing new opportunities within the world of AWS and technical support.