**Purpose and Audience**

Visualizing and accessing organized e-commerce data in an insightful way is of the utmost importance to identifying strengths and weaknesses in supply chain, product line, and regional operations. E-commerce data is a constantly updating pool of information that is essential to capture, store, and interpret. Our team’s dashboard, along with our deployment of a MySQL Amazon Aurora database, is intended for use by internal stakeholders of an e-commerce organization. Specifically, it is intended to highlight where products are being shipped, which products are profitable, and where improvements to the supply chain can be made. Secure and robust storage and informative visuals are essential steps to understanding these core concepts.

**App Functionality**

Our e-commerce dashboard has simple functionality with an easy-to-use Amazon Quicksight interface:

* Users can filter visualizations by order date, region, product category, and minimum total profit using the control slicer located at the top of the dashboard.
* In each of the visualizations, users can select individual datapoints to ‘break out’ and highlight specific data. (e.g. a user can highlight ‘Binders’ in the Sankey diagram to see which regions binders are being shipped)
* Users can utilize genAI tool ‘Amazon Q’ to ask direct questions about the dataset and generate their own visualizations, tables, and insights based on their question as seen below:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Part 1. Amazon Aurora MySQL Database Service for OLTP**

The purpose of deploying our MySQL database on Amazon Aurora MySQL is to work with our database in an environment that is secure, highly available, operationally excellent, durable, resilient, and can connect to many other useful cloud services. It is made to optimize transactional workloads. The schema is normalized to 3NF for transactional workloads.

In the context of a larger data system, it would be used by apps for real-time reads and writes. For the purposes of this project it fits into the cloud architecture.

**Create Identity and Access Management (IAM) Project Group and Add Project Users**

Set up IAM group: Ecommerce-Project-Team

Add Users, gabe-tharp and marcos-fernandez, with PowerUserAccess IAM policy

Add User, owen-randolph, with AdministratorAccess IAM policy

**1. Create Aurora MySQL environment, starting with the cluster**

-Engine option: Aurora (MySQL Compatible)

Aurora MySQL provides high availability, fault tolerance, and MySQL compatibility, which makes it ideal for a production-level analytics project.

Credentials managed in AWS Secrets Manager

Aurora Standard cluster storage configuration

Instance engine: db.t3.medium (cost effective for this project), from Burstable classes

-Create Virtual Private Cloud (VPC) – VPC defines the network environment for the database cluster. It creates the foundation for a secure environment.

-Create a new subnet group for the database to live in inside the VPC. Two subnets in different availability zones (AZ) are selected for high availability.

-Allow public access for the database. This assigns a public IP address to the cluster, which will allow us to connect MySQL Workbench locally to Aurora MySQL.

-Create new security group. Security groups manage IP access at the instance level.

\* We will not create an Aurora replica due to scope and cost of this project. Multi-AZ deployment creates an Aurora replica for fast failover and high availability.

\* We will not create an RDS proxy for the scope of this project. This service allows applications to pool and share connections. This would be a great option to improve scalability.

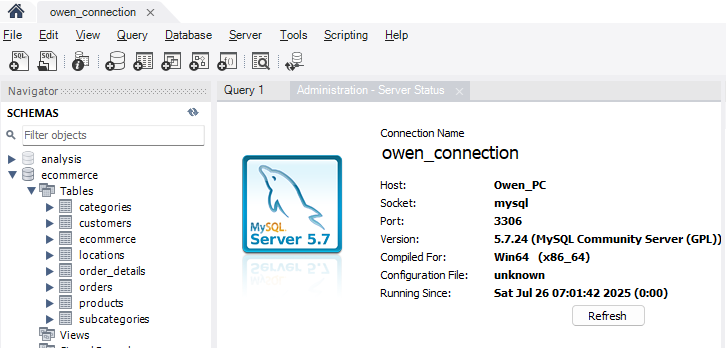
A screenshot of a computer

AI-generated content may be incorrect.

* A writer instance is created as a transactional database feature to add data to our database.

2. Deploy MySQL database on Aurora MySQL

Open MySQL Workbench, connection to local MySQL instance , port 3306



Export data using the ecommerce schema to Self-contained file

A screenshot of a computer

AI-generated content may be incorrect.

**Prepare Aurora connection**

Whitelist IP in VPC security group, use port 3306 for MySQL Workbench

Add inbound rules

A screenshot of a computer

AI-generated content may be incorrect.

Create a new connection in MySQL Workbench to connect to Amazon Aurora cluster

A screenshot of a computer

AI-generated content may be incorrect.

Import .sql Export file

A screenshot of a computer

AI-generated content may be incorrect.

Double Check that the schema worked by entering a query in MySQL Workbench. What’s happening here is that the query is being done on the Aurora instance, not the local instance. This is actually data coming from Aurora now:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Create S3 Bucket for data pipeline from Aurora to Quicksight**

Bucket Name: ecommerce-aurora-export

Uncheck “Block Public Access” to allow Quicksight access

Create an IAM and attach to read from the bucket

Role name: AuroraS3IntegrationRole

Add permission: AmazonS3ReadOnlyAccess

**S3 Data Lake**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Amazon Quicksight**

Upload data from S3 data lake files as a json manifest files for each of the tables which Quicksight requires as data ingestion format:

**A screenshot of a data source

AI-generated content may be incorrect.**

Add the csv files from the S3 buckets to the Quicksight builder and create joins to create star schema for OLAP

\* For a production-ready workflow we could add AWS Glue for automated ETL. For this project we will include that due to cost and scope

**Quicksight Star Schema Database model**

The star schema will make this database more efficient for OLAP querying. A table with the joins can be found in the appendix. The Quicksight GUI allows the user to configure the joins manually, rather than using SQL.

**A screenshot of a computer

AI-generated content may be incorrect.**

** QuickSight stores datasets in a compressed, columnar format (SPICE).**

** This reduces storage size drastically compared to raw CSV text files.**

**We started with a 2,253 KB csv file and after normalizing into seven tables, and compressing into SPICE data format, it’s 150.6 KB.**

Save & Publish

**Create new Analysis**

**Build Dashboard**

Amazon Quicksight is functionally like Microsoft Power BI and Tableau platforms. It has a simple drag and drop interface with a variety of visualization options.

**References:**

[Amazon Aurora MySQL reference - Amazon Aurora](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Reference.html)

[Connect to your Amazon Aurora MySQL DB cluster | AWS re:Post](https://repost.aws/articles/ARmIOpARjHTyKW0_dc8b3RVQ/connect-to-your-amazon-aurora-mysql-db-cluster)

[Introduction to AWS Simple Storage Service (AWS S3) - GeeksforGeeks](https://www.geeksforgeeks.org/devops/introduction-to-aws-simple-storage-service-aws-s3/)

[Overview of Performance Insights on Amazon RDS - Amazon Relational Database Service](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.Overview.html)

**Appendix:**

Star Schema Model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fact Table | Dimension Table | Join Key (Fact Table) | Join Key (Dimension Table) | Join Type |
| order\_details | orders | Order\_ID | Order\_ID | INNER JOIN |
| order\_details | products | Product\_ID | Product\_ID | INNER JOIN |
| orders | customers | Customer\_ID | Customer\_ID | INNER JOIN |
| products | subcategories | Subcategory\_ID | Subcategory\_ID | LEFT JOIN |
| subcategories | categories | Category\_ID | Category\_ID | LEFT JOIN |

**Queries:**

Top 10 Most Ordered Products

USE ecommerce

SELECT

COUNT(o.Order\_ID) AS NumberofOrders,

p.Product\_Name

FROM Order\_Details o

LEFT JOIN Products p

ON o.Product\_ID = p.Product\_ID

GROUP BY p.Product\_Name

ORDER BY NumberofOrders DESC

LIMIT 10;

Most Profitable Product Subcategories

SELECT

s.SubCategory\_Name,

SUM(od.Profit) AS TotalProfit

FROM Order\_Details od

JOIN Products p

ON od.Product\_ID = p.Product\_ID

JOIN Subcategories s

ON p.SubCategory\_ID = s.SubCategory\_ID

GROUP BY s.SubCategory\_Name

ORDER BY TotalProfit DESC;

Most Profitable Product Segments

SELECT

c.Segment,

SUM(od.Profit) AS TotalProfit

FROM

Order\_Details od

JOIN Orders o

ON od.Order\_ID = o.Order\_ID

JOIN Customers c

ON o.Customer\_ID = c.Customer\_ID

GROUP BY c.Segment

ORDER BY TotalProfit DESC;

**Reflections:**

Throughout this project, we discovered a technology stack that is particularly well-suited for our chosen dataset. Amazon Quicksight and Amazon Aurora integrate seamlessly with MySQL Workbench to create an intuitive storage solution with end user applications. The most challenging part of this project was the initial orchestration of all the different cloud services specifically, correctly configuring IAM permissions, VPC security groups, and subnet access to ensure all components of our data pipeline could communicate securely. This process highlighted the critical importance of secure access management in a cloud environment.

If we had more time, our team would explore more tools in the AWS ecosystem like AWS Glue for further automating data preparation, analyzation, and integration. Also, in a real-world scenario, our team would have liked to see this dashboard (or similar application) deployed to a dedicated website where intranet users within the e-commerce company could view and update the dashboard outside of Quicksight.