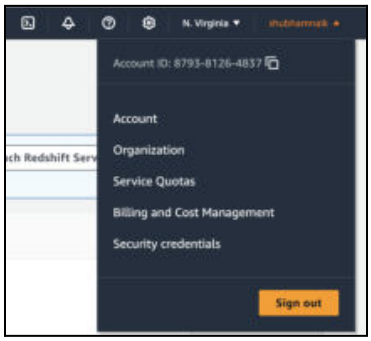
Name: Shubham Manisha Naik **SJSU ID:**017627025
Name: Shreyas Vinayak Mohite **SJSU ID:**018207475
Name: Nitya Rondla **SJSU ID:**018204186
Name: Rutuja Nitin Kadam **SJSU ID:**018207176

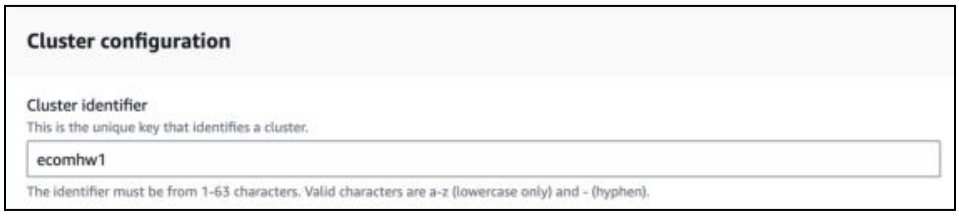
A. Set Up Your Amazon Redshift Cluster

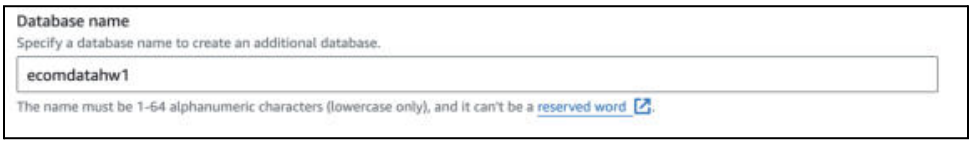
A.1. Log in to the AWS Management Console: https://aws.amazon.com/console/

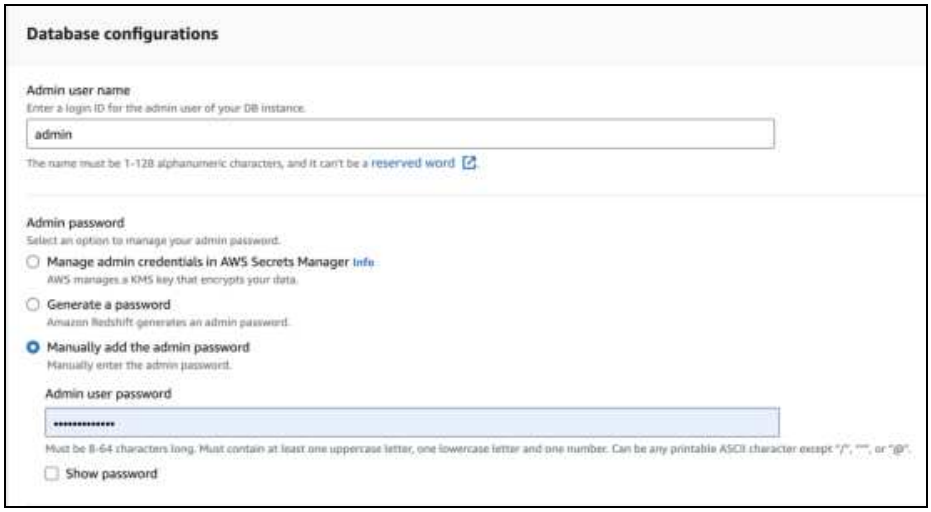


A.2. Launch a Redshift Cluster:
- Go to the Amazon Redshift service.
- Click on “Create cluster.”

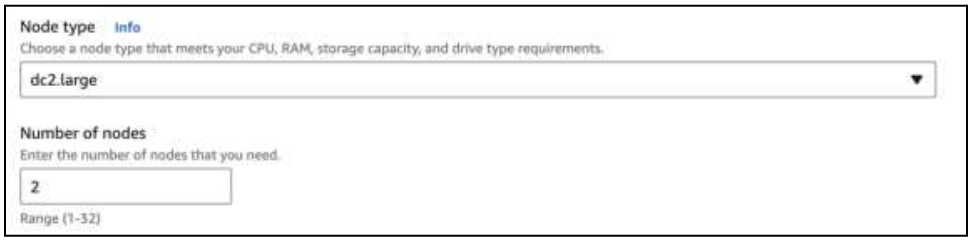
- Enter a cluster identifier, database name, admin username, and password.







- Choose an instance type (e.g., `dc2.large` for the exercise).



- In Network & Security, Enable **Turn on Publicly accessible**.

- Configure other settings as desired, then click “Create cluster.”

- This process may take a few minutes.

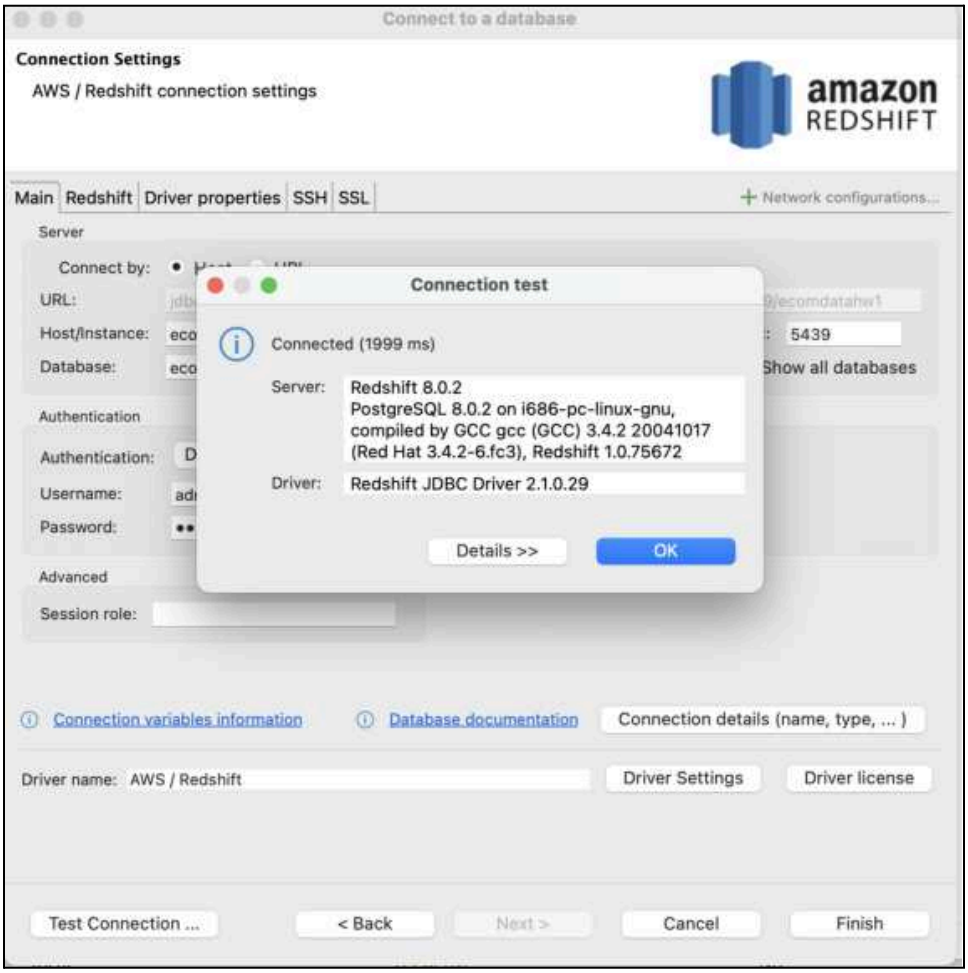
A.3. Configure Security Groups:

- Ensure your Redshift cluster's security group allows inbound connections on port `5439` from your IP address.

- Navigate to the VPC section, find the security group associated with your cluster, and edit inbound rules.

A.4. Connect to Your Cluster:

- Use SQL Workbench/J, DBeaver, or any SQL client tool to connect.
- Obtain the endpoint, port, and database name from the Redshift console.



B. Prepare the Sample Data

B.1. Download Sample Data:

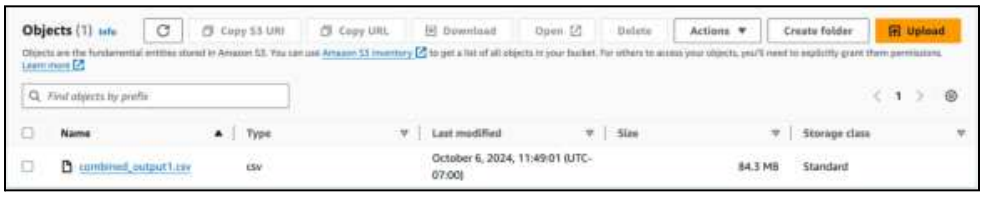
- Download sample data for e-commerce transactions from Kaggle. You can search for “online retail dataset” and arrive at datasets such as this one: https://www.kaggle.com/datasets/lakshmi25npathi/online-retail-dataset

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
1	489434	85048	GLASS BALL 20 LIGHTS	12	12/1/09 7:45	6.95	13085.0	United Kingdom
2	489434	79323P	PINK CHERRY LIGHTS	12	12/1/09 7:45	6.75	13085.0	United Kingdom
3	489434	79323W	WHITE CHERRY LIGHTS	12	12/1/09 7:45	6.75	13085.0	United Kingdom
4	489434	22041	FRAME 7" SINGLE SIZE	48	12/1/09 7:45	2.1	13085.0	United Kingdom
5	489434	21232	CERAMIC TRINKET BOX	24	12/1/09 7:45	1.25	13085.0	United Kingdom

B.2. Upload Data to Amazon S3:

- Go to the S3 service in AWS.
- Create a new bucket or use an existing one.

- Upload the CSV files to this bucket.



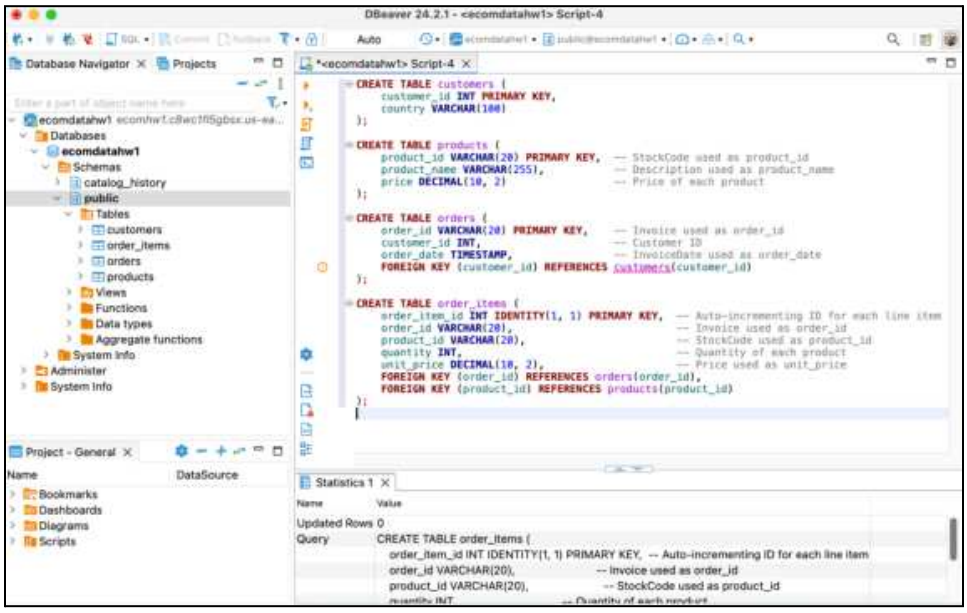
B.3. Grant Redshift Access to S3:

- Create an IAM role with `AmazonS3ReadOnlyAccess` policy and attach it to your Redshift cluster.

C. Load Data into Redshift

C.1. Create Schema and Tables:

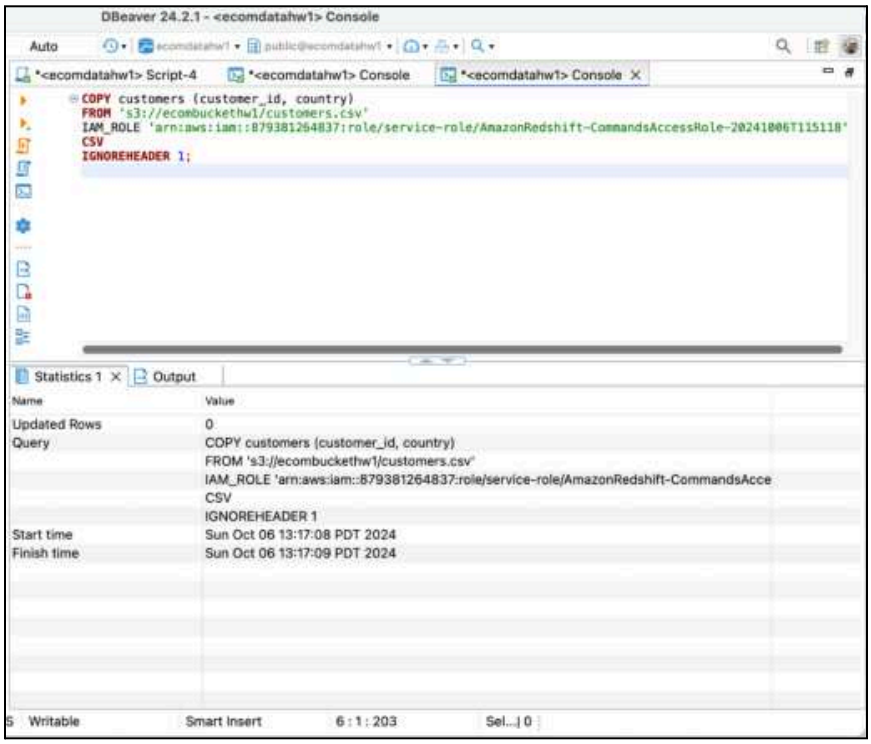
- Connect to your Redshift cluster using your SQL client tool.
- Create a star schema with tables for the entities in the .csv such as `customers`, `products`, `orders`, and `order_items`.



C.2. Load Data Using COPY Command:

- Run the following `COPY` command to load data from S3:

```sql
COPY customers (customer_id, country)
FROM 's3://ecombuckethw1/customers.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY products (product_id, product_name, price)
FROM 's3://ecombuckethw1/products.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY orders (order_id, customer_id, order_date)
FROM 's3://ecombuckethw1/orders.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY order_items (order_id, product_id, quantity, unit_price)
FROM 's3://ecombuckethw1/order_items.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;
```



DBeaver 24.2.1 - <ecomdatahw1> Console

*<ecomdatahw1> Script-4 | *<ecomdatahw1> Console | *<ecomdatahw1> Console

```
COPY customers (customer_id, country)
FROM 's3://ecombuckethw1/customers.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY products (product_id, product_name, price)
FROM 's3://ecombuckethw1/products.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;
```

Statistics 1 | Output

Name	Value
Updated Rows	0
Query	COPY products (product_id, product_name, price) FROM 's3://ecombuckethw1/products.csv' IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118' CSV IGNOREHEADER 1
Start time	Sun Oct 06 13:17:45 PDT 2024
Finish time	Sun Oct 06 13:18:09 PDT 2024

Writable | Smart Insert | 7 : 1 [211] | Sel...| 5

DBeaver 24.2.1 - <ecomdatahw1> Console

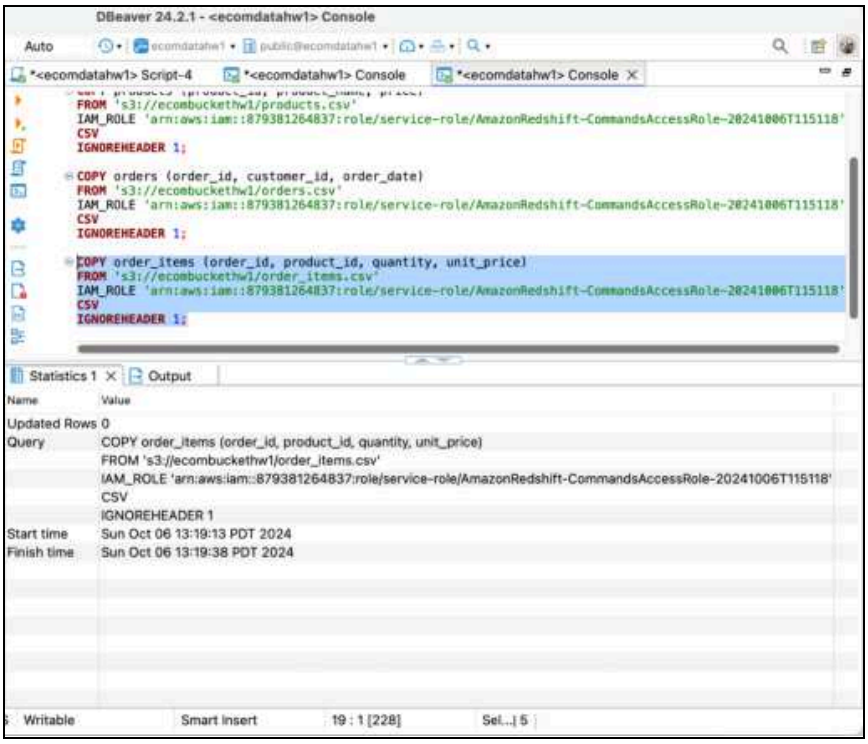
*<ecomdatahw1> Script-4 | *<ecomdatahw1> Console | *<ecomdatahw1> Console

```
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY products (product_id, product_name, price)
FROM 's3://ecombuckethw1/products.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;

COPY orders (order_id, customer_id, order_date)
FROM 's3://ecombuckethw1/orders.csv'
IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118'
CSV
IGNOREHEADER 1;
```

Statistics 1 | Output

Name	Value
Updated Rows	0
Query	COPY orders (order_id, customer_id, order_date) FROM 's3://ecombuckethw1/orders.csv' IAM_ROLE 'arn:aws:iam::879381264837:role/service-role/AmazonRedshift-CommandsAccessRole-20241006T115118' CSV IGNOREHEADER 1
Start time	Sun Oct 06 13:18:34 PDT 2024
Finish time	Sun Oct 06 13:18:47 PDT 2024

Writable | Smart Insert | 13 : 1 [209] | Sel...| 5



D. Analyze Data and Run Queries

D.1. Perform Basic Queries:
- Run queries to explore the data, such as finding total sales per product, customer purchase history, and order trends.

```
-- Total sales per product
SELECT p.product_name, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sales DESC;

-- Customer purchase history
SELECT c.first_name, c.last_name, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY total_orders DESC;
```

Design and run 5 more queries to perform meaningful analytics on the data and draw valuable insights that can support decision making.

1. Total Sales per Product:

```sql
SELECT p.product_name, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sales DESC;
```

products 1

SELECT p.product_name, SUM(Enter a SQL expression to filter results (use Ctrl+Space)

	product_name	total_sales
1	REGENCY CAKESTAND 3 TIER	261,110.95
2	CREAM HANGING HEART T-LIGHT HOLDER	237,833.96
3	JUMBO BAG RED RETROSPOT	165,751.82
4	ASSORTED COLOUR BIRD ORNAMENT	123,631.87
5	POSTAGE	110,338.51
6	PARTY BUNTING	102,089.38
7	PAPER CHAIN KIT 50'S CHRISTMAS	75,388.48
8	CHILLI LIGHTS	68,453.5
9	JUMBO BAG PINK POLKADOT	66,904.73
10	DOORMAT UNION FLAG	63,910.54

- **Top-Selling Product:** The "REGENCY CAKE STAND 3 TIER" is the highest-selling product with over $261K in sales. Focus marketing efforts on this product and similar items to drive further growth.
- **Popular Products:** Other high-selling products like the "CREAM HANGING HEART T-LIGHT HOLDER" and "JUMBO BAG RED RETROSPOT" show strong demand. Consider expanding related product lines to capitalize on their popularity.
- **Diverse Product Range:** Items like "ASSORTED COLOR BIRD ORNAMENT" and "PARTY BUNTING" have consistent sales but lower total revenue. These could be bundled with top-sellers to increase average transaction value.
- **Stock and Supply Chain:** Ensure sufficient stock of high-demand items, especially the top 3 products, to prevent stockouts and maintain sales momentum.
- **Promotion Strategy:** Use lower-selling products like "DOORMAT UNION FLAG" and "CHILLI LIGHTS" for discounts or bundled promotions to clear inventory and boost sales for these items.

2. Customer Purchase History (Number of Orders per Customer):

*<ecomdatahw1> Script- *<ecomdatahw1> Consol *<ecor

```sql
SELECT c.customer_id, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY total_orders DESC;
```

orders 1

SELECT c.customer_id, COUNT | Enter a SQL expression to filter results

	customer_id	total_orders
1	14,911	512
2	12,748	366
3	17,841	289
4	15,311	270
5	14,606	261
6	13,089	247
7	14,156	202
8	14,527	191
9	14,646	164
10	13,694	164

- **Best Customer:** Order leader by customer is Customer 14,911. This customer received the largest number of orders, 512. Such a customer can be most valuable for a company in keeping his loyalty because he is well engaged with the business. The priority would be rewarding such a customer with loyalty incentives or exclusive offers.
- **High-Engagement Customers:** Other customers, such as 12,748 and 17,841, with 366 and 289 orders, respectively, also seem to be strong contributors. Individualized marketing may be considered to induce an even higher frequency of purchase.
- **Medium-Involvement:** Those customers who place an order number of about 200–270, like 15,311 or 14,606, engage well. Upselling opportunities or similar available products recommendation will help to increase their order frequency.
- **Retention Strategy:** The order counts of 14,527 and 13,694, which correspond to few orders (190-160), can have a retention strategy applied through selected focuses such as online discount offers, flash sales, or loyalty bonuses to enhance the order rate.
- **Customer Segmentation:** Categorize customers based on the frequency of orders into high, medium, and low engagement tiers to implement differentiated strategies for promotions, retention, and loyalty programs.

3. Top Customers by Total Spending:

```sql
SELECT c.customer_id, SUM(oi.quantity * oi.unit_price) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.customer_id
ORDER BY total_spent DESC;
```

customers 1

SELECT c.customer_id, SUM(oi | Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	total_spent
1	18,102	570,380.61
2	14,646	523,342.07
3	14,156	296,063.44
4	14,911	265,757.91
5	17,450	231,390.55
6	13,694	190,020.84
7	17,511	168,491.62
8	12,415	143,269.29
9	16,684	141,502.25
10	15,061	124,961.98

- **Customer Value Identification:** High-value customers (e.g., spending over $500,000) can be identified.
- **Targeted Marketing:** Customers with varying spending levels can be segmented.
- **Revenue Optimization:** The top 3 customers account for a significant portion of total revenue.
- **Inventory Planning:** High spenders likely purchase specific products in larger quantities.
- **Profit Maximization:** Spending trends help identify profitable customer segments.
- **Predictive Sales Insights:**Customer spending data can predict future purchasing behavior.

4. Most Popular Products (by Quantity Sold):

*<ecomdatahw1> Script- *<ecomdatahw1> Consol *<ecomdatahw1> Con

```sql
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC;
```

products 1

SELECT p.product_name, SUM | Enter a SQL expression to filter results (use Ctrl+Spac

	product_name	total_quantity_sold
1	WORLD WAR 2 GLIDERS ASSTD DESIGNS	103,505
2	JUMBO BAG RED RETROSPOT	92,223
3	PACK OF 72 RETROSPOT CAKE CASES	88,836
4	CREAM HANGING HEART T-LIGHT HOLDER	88,183
5	ASSORTED COLOUR BIRD ORNAMENT	77,755
6	POPCORN HOLDER	76,140
7	BROCADE RING PURSE	69,343
8	PACK OF 60 PINK PAISLEY CAKE CASES	54,373
9	60 TEATIME FAIRY CAKE CASES	52,502
10	MINI PAINT SET VINTAGE	42,414

- **Top-Selling Products:** Focus marketing on high-demand products like "WORLD WAR 2 GLIDERS" (103,505 units sold).
- **Inventory Management:** Ensure top-selling products are stocked and adjust inventory for lower sellers like "MINI PAINT SET VINTAGE."
- **Product Bundling:** Bundle complementary items like "T-LIGHT HOLDER" and "BIRD ORNAMENT" to boost sales.
- **Market Trends:** Expand retro-themed products due to high demand for items like "JUMBO BAG RED RETROSPOT."
- **Pricing Strategy:** Use competitive pricing for high-sellers and promotions for low-performing products.

5. Sales by Country:

```sql
SELECT c.country, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.country
ORDER BY total_sales DESC;
```

customers 1

SELECT c.country, SUM(oi.quar | Enter a SQL expression to filter results (use Ctrl+S

	country	total_sales
1	United Kingdom	13,482,121.18
2	EIRE	573,308.2
3	Netherlands	548,330.7
4	Germany	411,862.11
5	France	320,384.66
6	Australia	167,427.2
7	Spain	101,934.43
8	Switzerland	98,253.1
9	Sweden	87,421.52
10	Denmark	58,057.24

- **Top Market:** The United Kingdom dominates with over 13 million in sales. Focus resources on further growth in this market.
- **Secondary Market:** EIRE and the Netherlands are significantly behind the UK, with 573K and 548K in sales, respectively. Target these regions for market expansion and marketing efforts.
- **Underperforming Markets:** Countries like Denmark, Sweden, and Switzerland show relatively low sales. Consider promotional strategies or new product introductions in these markets.
- **Regional Strategy:** European markets (Germany, France, and Spain) show moderate performance. Tailor regional campaigns and optimize distribution to increase sales.
- **Opportunity for Growth:** Australia, with 167K in sales, represents a non-European market with potential for growth through targeted advertising and product offerings.

6. Monthly Sales Trend:

```
SELECT DATE_TRUNC('month', o.order_date) AS month, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY DATE_TRUNC('month', o.order_date)
ORDER BY month;
```

Results 1

	month	total_sales
1	2009-12-01 00:00:00.000	660,095.4
2	2010-01-01 00:00:00.000	530,436.51
3	2010-02-01 00:00:00.000	487,596.42
4	2010-03-01 00:00:00.000	633,419.31
5	2010-04-01 00:00:00.000	558,007.83
6	2010-05-01 00:00:00.000	557,873.39
7	2010-06-01 00:00:00.000	568,784.55
8	2010-07-01 00:00:00.000	560,885.33
9	2010-08-01 00:00:00.000	585,259.46
10	2010-09-01 00:00:00.000	778,520.05

- **High-Performing Months:** December 2009 and September 2010 show the highest sales, with 660K and 778K, respectively. Focus marketing efforts and promotions during these periods to capitalize on seasonal demand.
- **Consistent Sales:** Monthly sales from March to July 2010 remain relatively stable, averaging around 550K-570K. This suggests steady customer demand during these months.
- **Seasonal Trends:** The jump in sales in December 2009 and September 2010 could indicate holiday or event-driven purchases. Tailor inventory and promotional strategies around these peaks.
- **Growth Opportunities:** Early 2010 (January and February) shows lower sales compared to later months. Consider launching campaigns or discounts to boost sales during slow months.
- **Resource Planning:** Use peak sales months (December and September) to optimize staffing, inventory, and marketing investments to ensure you meet increased demand efficiently.

7. Average Order Value by Customer:

*<ecomdatahw1> Script- *<ecomdatahw1> Consol *<ecomdatahw1> Consol

```sql
SELECT c.customer_id, AVG(oi.quantity * oi.unit_price) AS avg_order_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.customer_id
ORDER BY avg_order_value DESC;
```

orders 1

SELECT c.customer_id, AVG(oi. Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	avg_order_value
1	15,195	3,861
2	13,205	2,803.2
3	17,846	2,033.1
4	12,737	1,855.25
5	16,532	1,459.44
6	15,749	1,435.72
7	16,000	1,377.07
8	17,369	979.2
9	17,876	856
10	14,091	733.08

- **High-Value Customers:** The maximum average order value by customer number is $3,861 for customer # 15,195 followed by customers # 13,205 and # 17,846. These will be your High-Value Customers whom you want to retain with personalized offers and offering them premium services.
- **Mid-tier customers:** Are those whose average order value runs in the region of $1,000 to $2,000. Take, for instance, 12,737 and 16,532-these are opportunities to up-sell. You can give some kind of incentive for them to buy bigger, which would also increase the average order size.
- **Low-Value Customers:** Those customers whose average order value is below $1,000 need to be dealt with focused promotional activities, discounts, or product bundling in order to increase the spend per order.
- **Customer Segmentation:** Segmentation of customers based on order value and usage in designing focused marketing campaigns and loyalty programs relevant for each tier.
- **Upsell/Cross-sell:** An opportunity to grow the low-order-value customers to mid- or high-tier customers by listening to their preferences and offering them relevant promotions or product recommendations.

D.2. Optimize Queries:

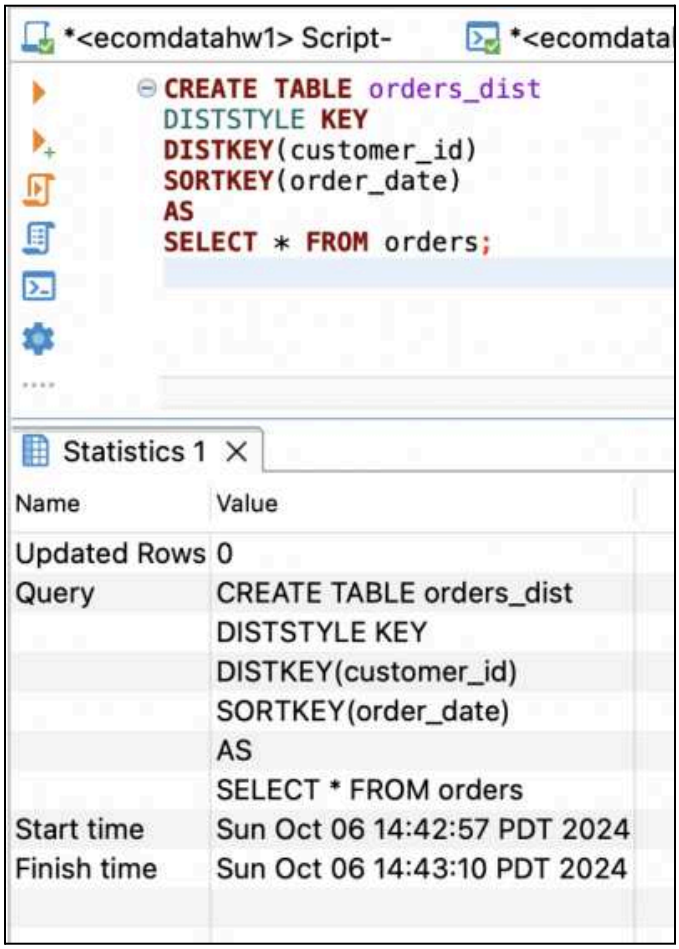
- Use Redshift's `DISTKEY` and `SORTKEY` to optimize query performance. For example:

```
CREATE TABLE orders_dist
DISTSTYLE KEY
DISTKEY(customer_id)
SORTKEY(order_date)
AS
SELECT * FROM orders;
```

- Run `ANALYZE` and `VACUUM` commands periodically to maintain performance.

Describe the impact of these optimizations quantitatively as the pertain to your use case.

Orders Table Optimization:



Order Items Table Optimization:

*<ecomdatahw1> Script- *<ecomdat

```sql
CREATE TABLE order_items_dist
DISTSTYLE KEY
DISTKEY(order_id)
SORTKEY(product_id)
AS
SELECT * FROM order_items;
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE order_items_dist
	DISTSTYLE KEY
	DISTKEY(order_id)
	SORTKEY(product_id)
	AS
	SELECT * FROM order_items
Start time	Sun Oct 06 14:45:18 PDT 2024
Finish time	Sun Oct 06 14:45:33 PDT 2024

Load data into the newly optimized tables:

*<ecomdatahw1> Script- *<ecomda

```sql
INSERT INTO orders_dist
SELECT * FROM orders;

INSERT INTO order_items_dist
SELECT * FROM order_items;
```

Statistics 1

Name	Value
Updated Rows	842822
Query	INSERT INTO orders_dist
	SELECT * FROM orders;
	INSERT INTO order_items_dist
	SELECT * FROM order_items
Start time	Sun Oct 06 14:46:20 PDT 2024
Finish time	Sun Oct 06 14:46:30 PDT 2024

Run `ANALYZE` and `VACUUM` for performance tuning:

*<ecomdatahw1> Script- *<ecomdatahw

```
ANALYZE orders_dist;
ANALYZE order_items_dist;
```

Statistics 1

Name	Value
Updated Rows	0
Query	ANALYZE orders_dist; ANALYZE order_items_dist
Start time	Sun Oct 06 14:48:08 PDT 2024
Finish time	Sun Oct 06 14:48:09 PDT 2024

*<ecomdatahw1> Script- *<ecomdat

```
VACUUM orders_dist;
```

Statistics 1

Name	Value
Updated Rows	0
Query	VACUUM orders_dist
Start time	Sun Oct 06 14:53:47 PDT 2024
Finish time	Sun Oct 06 14:53:47 PDT 2024

*<ecomdatahw1> Script- *<ecomda

```
VACUUM order_items_dist;
```

Statistics 1

Name	Value
Updated Rows	0
Query	VACUUM order_items_dist
Start time	Sun Oct 06 14:50:56 PDT 2024
Finish time	Sun Oct 06 14:51:52 PDT 2024

Impact of Optimizations:

- **DISTKEY(customer_id) in orders_dist**: This helps ensure that joins between the orders and customers tables are efficient, reducing the need for inter-node communication.
- **SORTKEY(order_date) in orders_dist**: Queries that filter or aggregate data by date will benefit from faster data retrieval, especially for operations like date-based sales trends.
- **DISTKEY(order_id) in order_items_dist**: This makes joins between orders and order_items efficient, as the order_id is distributed across nodes.
- **SORTKEY(product_id) in order_items_dist**: Queries involving product-related analytics (like total sales per product) will be optimized.

Quantitative Impact:

The optimizations should result in:

- **Faster joins**: By co-locating data on the same nodes, queries that join the orders and customers tables will run faster.
- **Improved performance on large data scans**: Using SORTKEY(order_date) and SORTKEY(product_id) allows the database to scan less data for queries involving date ranges or product-specific filtering.
- **Efficient resource use**: Running ANALYZE and VACUUM will ensure that Redshift's query planner has the most up-to-date information for optimizing query execution.

E. Advanced Features

E.1. Create a Materialized View:
- Create a materialized view to store precomputed query results for faster access:

```
CREATE MATERIALIZED VIEW mv_sales_summary AS
SELECT p.product_name, COUNT(o.order_id) AS total_orders, SUM(oi.quantity *
oi.unit_price) AS total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name;
```

Design and run 2 queries to perform meaningful analytics involving the materialized view and draw valuable insights that can support decision making.

Create the Materialized View:

*<ecomdatahw1> Script- *<ecomdatahw1> Consol *<eco

```sql
CREATE MATERIALIZED VIEW mv_sales_summary AS
SELECT p.product_name,
       COUNT(o.order_id) AS total_orders,
       SUM(oi.quantity * oi.unit_price) AS total_sales
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name;
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE MATERIALIZED VIEW mv_sales_summary AS SELECT p.product_name, COUNT(o.order_id) AS total_orders, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN orders o ON oi.order_id = o.order_id JOIN products p ON oi.product_id = p.product_id GROUP BY p.product_name
Start time	Sun Oct 06 15:01:06 PDT 2024
Finish time	Sun Oct 06 15:01:17 PDT 2024

Perform Analytics Using the Materialized View

Query 1: Top 5 Products by Total Sales

*<ecomdatahw1> Script- *<ecomdatahw1> Consol *<ecomd

```sql
SELECT product_name, total_sales
FROM mv_sales_summary
ORDER BY total_sales DESC
LIMIT 5;
```

mv_tbl__mv_sales_summary__0 1

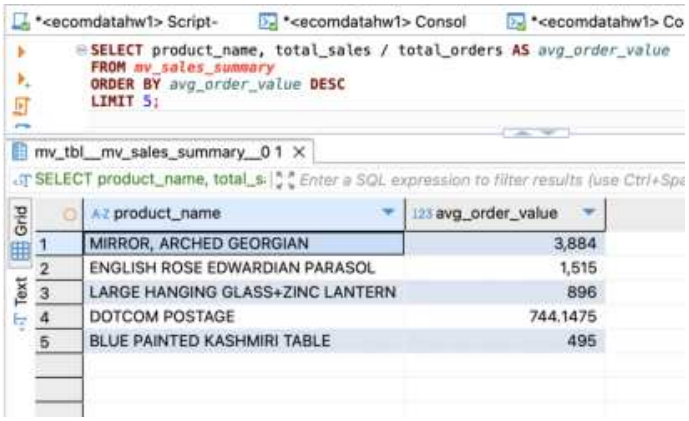
SELECT product_name, total_si | Enter a SQL expression to filter results (

	product_name	total_sales
1	POSTAGE	13,240,621.2
2	WHITE HANGING HEART T-LIGHT HOLDER	2,378,339.6
3	REGENCY CAKESTAND 3 TIER	1,827,776.65
4	JUMBO BAG RED RETROSPOT	1,491,766.38
5	PARTY BUNTING	918,804.42

- **Top-Selling Product**: **"POSTAGE"** has significantly higher sales at **$13.24M**, far exceeding other products. This product should be monitored closely for pricing and cost optimization strategies to maintain its profitability.
- **Popular Home Decor**: The **"WHITE HANGING HEART T-LIGHT HOLDER"** has strong sales of **$2.37M**, indicating high demand for decorative items. Expanding the range of similar home decor products could further boost sales.

- **Successful Kitchen Item**: The **"REGENCY CAKESTAND 3 TIER"** is a top kitchen item with **$1.83M** in sales. Offering complementary kitchenware could encourage bundle purchases and increase the average order value.
- **Popular Bag**: The **"JUMBO BAG RED RETROSPOT"** shows steady demand with **$1.49M** in sales. Consider introducing new designs or sizes to capitalize on the product's popularity.
- **Party Supplies**: **"PARTY BUNTING"** has strong sales of **$918K**, showing demand for party supplies. Expanding the range of party accessories or running promotions during holidays could further drive sales in this category.

Query 2: Products with Highest Average Order Value



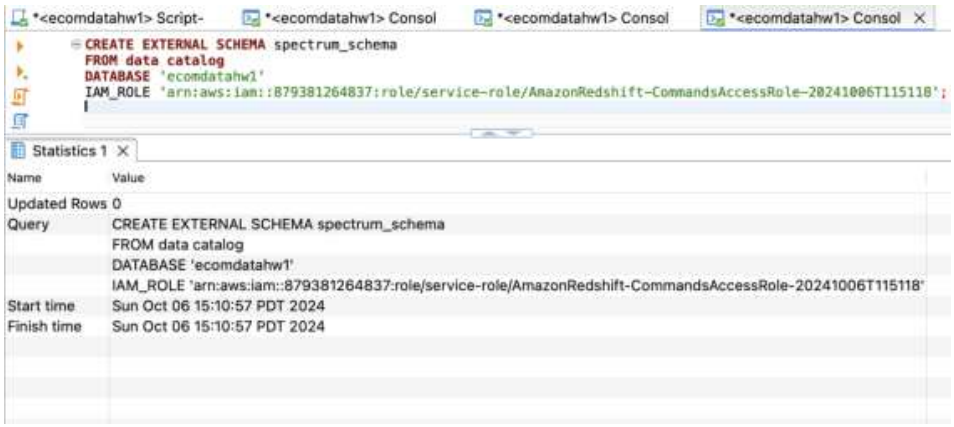
- **High-Value Product**: **"MIRROR, ARCHED GEORGIAN"** has the highest average order value at **$3,884**. This product is premium and likely appeals to high-end customers, making it suitable for targeted luxury marketing.
- **Mid-Tier Luxury Items**: Products like the **"ENGLISH ROSE EDWARDIAN PARASOL"** and **"LARGE HANGING GLASS+ZINC LANTERN"** have average order values of **$1,515** and **$896**, respectively. These items can be part of curated collections or special promotional bundles to attract upscale customers.
- **Shipping Costs**: **"DOTCOM POSTAGE"** has a significant average value of **$744**, suggesting that shipping fees may contribute substantially to total sales. Offering free shipping for high-value orders or reducing shipping fees could improve conversion rates.
- **Niche Product Focus**: The **"BLUE PAINTED KASHMIRI TABLE"** has a lower average order value of **$495** but still appeals to niche markets. Consider positioning it in targeted campaigns focused on unique, handcrafted items to attract discerning buyers.
- **Premium Product Strategy**: The high average order values suggest that these products could benefit from enhanced online visibility, targeted advertising, and premium placement in catalogs or websites to maximize sales and profitability.

E.2. Use Redshift Spectrum:
- Query data directly from S3 using Redshift Spectrum:

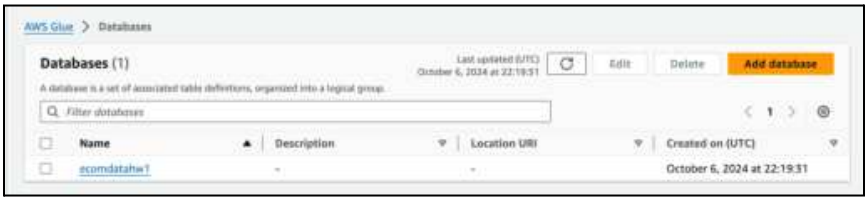
```
CREATE EXTERNAL SCHEMA spectrum_schema
FROM data catalog
DATABASE 'your_database'
IAM_ROLE 'arn:aws:iam::your-account-id:role/YourRedshiftRole';

CREATE EXTERNAL TABLE spectrum_schema.external_products (
    product_id INT,
    product_name STRING,
    category STRING,
    price DECIMAL(10, 2)
)
STORED AS PARQUET
LOCATION 's3://your-bucket/external-products/';
```

Design and run 2 queries to perform meaningful analytics involving the external schema and draw valuable insights that can support decision making.

Create an External Schema



Create a Database in AWS Glue



Create an External Table

*<ecomdatahw1> Script- | *<ecomdatahw1> Consol | *<ecomdatahw1> Consc

```sql
CREATE EXTERNAL TABLE spectrum_schema.external_products (
    product_id INT,
    product_name VARCHAR(255),   -- Replaced STRING with VARCHAR
    category VARCHAR(100),       -- Replaced STRING with VARCHAR
    price DECIMAL(10, 2)
)
STORED AS PARQUET
LOCATION 's3://ecombuckethw1/products.csv';
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE EXTERNAL TABLE spectrum_schema.external_products (product_id INT, product_name VARCHAR(255), -- Replaced STRING with VARCHAR category VARCHAR(100), -- Replaced STRING with VARCHAR price DECIMAL(10, 2)) STORED AS PARQUET LOCATION 's3://ecombuckethw1/products.csv'
Start time	Sun Oct 06 15:19:36 PDT 2024
Finish time	Sun Oct 06 15:19:36 PDT 2024

Run Queries on the External Table

Query 1: List Products in a Specific Category

*<ecomdatahw1> Script- | *<ecomdatahw1> Co

```sql
SELECT category, product_name, price
FROM spectrum_schema.external_products
ORDER BY category, price DESC
LIMIT 5;
```

Grid | Text

	product_name	category	price
1	10 COLOUR SPACEBOY PEN	10	1.63
2	10 COLOUR SPACEBOY PEN	10	1.63
3	10 COLOUR SPACEBOY PEN	10	1.63
4	10 COLOUR SPACEBOY PEN	10	1.63

Insights:

The product "10 COLOUR SPACEBOY PEN" is part of category 10, priced at 1.63.

Query 2: Calculate the Average Price per Category

*<ecomdatahw1> Script- *<ecomdatahw1> Consol

```
SELECT category, AVG(price) AS average_price
FROM spectrum_schema.external_products
GROUP BY category
ORDER BY average_price DESC;
```

123 category	123 price
10	1.066667
11	4.950000
12	1.273223
15	2.950000

Insights:

Provides a good foundation for conducting pricing analysis across categories, as it shows variations that can be investigated further for trends, product segmentation, or marketing strategies.

Did you find or come across solutions to similar problems by using Generative AI or other sources?

If you answered 'yes', give full details of the model (including the prompt used) / website / person, stating the question number and the help they provided.

In the process of doing this assignment, I have encountered a few errors to which I used generative AI for troubleshooting. The first was the STL load error in DBeaver during the uploading of data into the tables that I created. Initially, I copied the error and prompted the AI for the solution, but the suggestions did not fully answer the issue. The documentation did include a possible solution, which was irrelevant to my problem. Later in time, I asked the AI for alternative approaches, after which it helped me solve the issue in my codebase.

Website: Chatgpt (4o model) free version