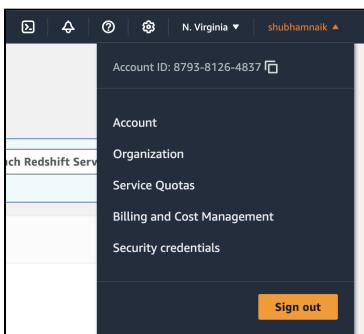


**Name:** Shubham Manisha Naik

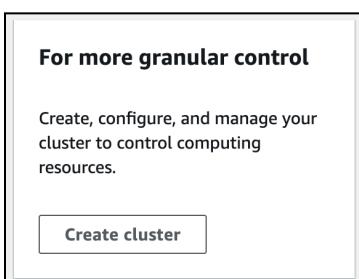
## A. Set Up Your Amazon Redshift Cluster

**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.

**Cluster configuration**

**Cluster identifier**  
This is the unique key that identifies a cluster.  
  
The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

**Database name**  
Specify a database name to create an additional database.  
  
The name must be 1-64 alphanumeric characters (lowercase only), and it can't be a reserved word [?](#) .

**Database configurations**

**Admin user name**  
Enter a login ID for the admin user of your DB instance.  
  
The name must be 1-128 alphanumeric characters, and it can't be a reserved word [?.](#)

**Admin password**  
Select an option to manage your admin password.  
 Manage admin credentials in AWS Secrets Manager [Info](#)  
AWS manages a KMS key that encrypts your data.  
 Generate a password  
Amazon Redshift generates an admin password.  
 Manually add the admin password  
Manually enter the admin password.

**Admin user password**  
  
Must be 8-64 characters long. Must contain at least one uppercase letter, one lowercase letter and one number. Can be any printable ASCII character except "/", "", or "@".  
 Show password

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

**Node type** [Info](#)  
Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.

**Number of nodes**  
Enter the number of nodes that you need.  
  
Range (1-32)

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

**Publicly accessible**  
For more information, see [Learn more about Redshift clusters security groups](#) [?.](#)

Turn on Publicly accessible  
Allow public connections to Amazon Redshift.

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

[Cancel](#) **Create cluster**

- This process may take a few minutes.

Clusters (1) <a href="#">Info</a>						
<a href="#">C</a> <a href="#">Query data</a> <a href="#">Actions</a> <b>Create cluster</b>						
<input type="text"/> <a href="#">Find clusters</a>						
Cluster	Status	Cluster namespace	Availability Zone	Multi-AZ	Storage capacity us...	CPU utiliz
ecomhw1 dc2.large   2 nodes   320 GB	<span>Available</span>	d46d4c3f-a85a-4fe3-b...	us-east-1c	No	0 %	0 %

### 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.

Inbound rules	Info	Type	Info	Protocol	Info	Port range	Source	Info	Description - optional	Info
sgr-098a21512a3a4f9e7		All traffic		All		All	Custom		Q sg-0c83dd708ec1150d2	X
-		Redshift		TCP		5439	My IP			Delete

### 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.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

**Create bucket**

General purpose buckets (1) [Info](#) [AWS Regions](#)  
Buckets are containers for data stored in S3.  
Name: ecombuckethw1 | AWS Region: US East (N. Virginia) us-east-1 | IAM Access Analyzer: View analyzer for us-east-1 | Creation date: October 6, 2024, 11:46:35 (UTC-07:00)

- Upload the CSV files to this bucket.

Objects (1) [Info](#) [C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.  
Learn more [\[?\]](#)  
Find objects by prefix:   
Name | Type | Last modified | Size | Storage class  
 combined\_output1.csv | csv | October 6, 2024, 11:49:01 (UTC-07:00) | 84.3 MB | Standard

### B.3. Grant Redshift Access to S3:

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

Cluster permissions  
Create an IAM role as the default for this cluster that has the [AmazonRedshiftAllCommandsFullAccess](#) policy attached. This policy includes permissions to run SQL commands to COPY, UNLOAD, and query data with Amazon Redshift. The policy also grants permissions to run SELECT statements for related services, such as Amazon S3, Amazon CloudWatch logs, Amazon SageMaker, and AWS Glue.

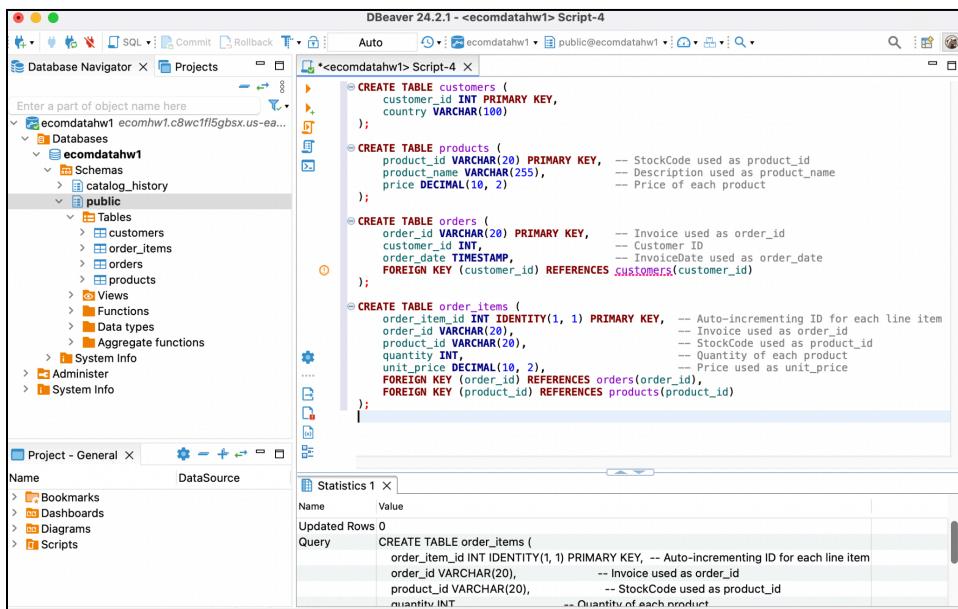
Associated IAM roles (1) [Info](#)  
Create, associate, or remove an IAM role. You can associate up to 50 IAM roles. You can also choose an IAM role and set it as the default for this cluster.  
Actions [Associate IAM roles](#) [Create IAM role](#)  
Find associated iam roles  
IAM roles | Status | Role type  
 AmazonRedshift-CommandsAccessRole-20241006T115118 | In-sync | Default

Policy attached to entity AmazonRedshift-CommandsAccessRole-20241006T115118.

## 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`.



The screenshot shows the DBeaver interface. On the left, the Database Navigator displays a database structure for 'ecomdatahw1'. Under the 'public' schema, there are tables for 'customers', 'order\_items', 'orders', and 'products'. A script editor window titled 'ecomdatahw1 > Script-4' contains the following SQL code:

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    country VARCHAR(100)
);

CREATE TABLE products (
    product_id VARCHAR(20) PRIMARY KEY, -- StockCode used as product_id
    product_name VARCHAR(255), -- Description used as product_name
    price DECIMAL(10, 2) -- Price of each product
);

CREATE TABLE orders (
    order_id VARCHAR(20) PRIMARY KEY, -- Invoice used as order_id
    customer_id INT, -- Customer ID
    order_date TIMESTAMP, -- InvoiceDate used as order_date
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE order_items (
    order_item_id INT IDENTITY(1, 1) PRIMARY KEY, -- Auto-incrementing ID for each line item
    order_id VARCHAR(20), -- Invoice used as order_id
    product_id VARCHAR(20), -- StockCode used as product_id
    quantity INT, -- Quantity of each product
    unit_price DECIMAL(10, 2), -- Price used as unit_price
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Below the script editor, a 'Statistics 1' window shows a query for creating the 'order\_items' table with 0 updated rows.

### C.2. Load Data Using COPY Command:

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

## 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

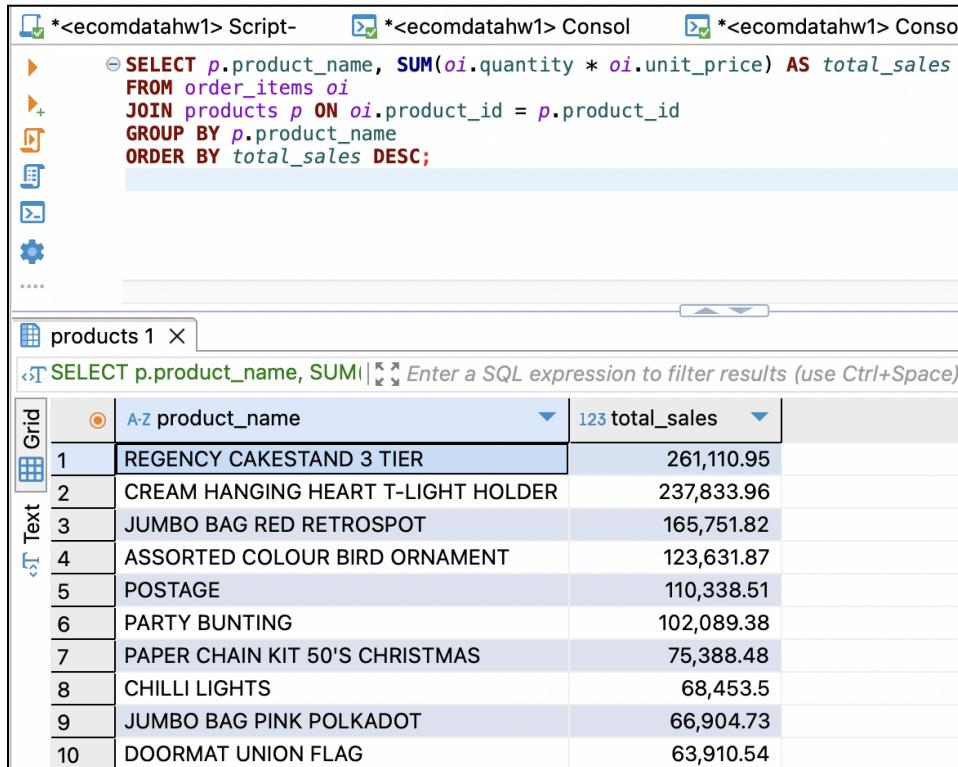
```
SELECT p.product_name, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sales DESC;
```

-- 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:



The screenshot shows a database interface with three tabs at the top: 'Script', 'Console', and 'Console'. The 'Script' tab contains a SQL query:

```
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;
```

The 'Console' tab shows the results of the query in a grid format. The columns are 'product\_name' and 'total\_sales'. The data is sorted by total sales in descending order. The results are as follows:

	A-Z product_name	123 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):

The screenshot shows a database interface with three tabs at the top: 'Script', 'Console', and 'Results'. The 'Script' tab contains a SQL query:

```
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;
```

The 'Results' tab displays the output of the query in a grid format. The columns are 'customer\_id' and 'total\_orders'. The data is as follows:

	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:

The screenshot shows a database interface with a query editor and a results grid. The query is:

```
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;
```

The results grid displays 10 rows of data:

customer_id	total_spent
18,102	570,380.61
14,646	523,342.07
14,156	296,063.44
14,911	265,757.91
17,450	231,390.55
13,694	190,020.84
17,511	168,491.62
12,415	143,269.29
16,684	141,502.25
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):

The screenshot shows a database management system interface with three tabs at the top: 'Script', 'Console', and 'Console'. The 'Script' tab contains the following SQL query:

```
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;
```

The 'Console' tab shows the results of the query in a grid format. The columns are 'product\_name' and 'total\_quantity\_sold'. The results are:

	A-Z product_name	123 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:

The screenshot shows a database interface with three tabs at the top: 'Script', 'Console', and 'Results'. The 'Script' tab contains a SQL query:

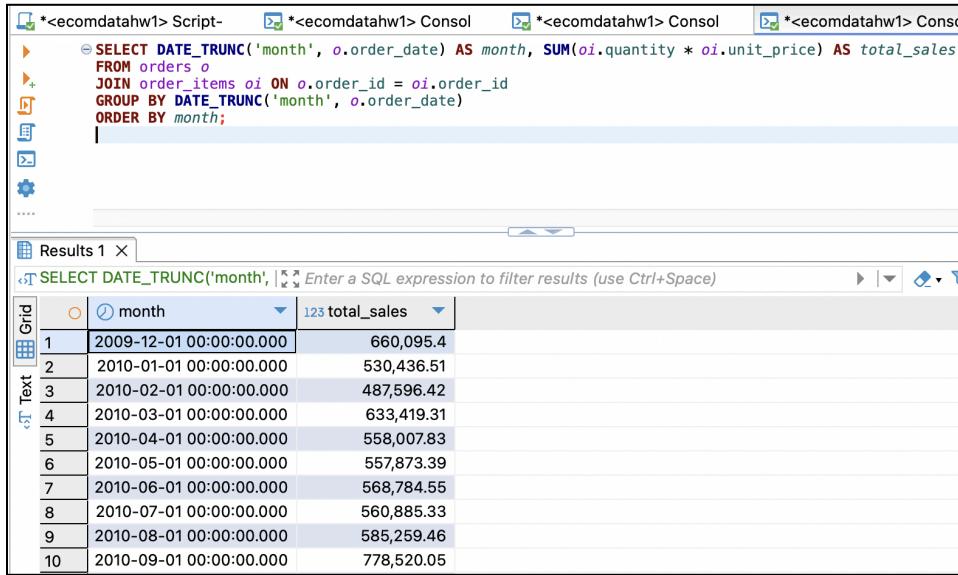
```
SELECT c.country, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.country
ORDER BY total_sales DESC;
```

The 'Console' tab is empty. The 'Results' tab shows a table titled 'customers 1' with 10 rows of data. The columns are 'Grid' (checkbox), 'country' (sorted by A-Z), and 'total\_sales'. The data is as follows:

Grid	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:



The screenshot shows a database interface with four tabs at the top: Script, Console, Consol, and Consol. The Script tab contains a SQL query:

```
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;
```

The Results tab displays the output of the query:

month	total_sales
2009-12-01 00:00:00.000	660,095.4
2010-01-01 00:00:00.000	530,436.51
2010-02-01 00:00:00.000	487,596.42
2010-03-01 00:00:00.000	633,419.31
2010-04-01 00:00:00.000	558,007.83
2010-05-01 00:00:00.000	557,873.39
2010-06-01 00:00:00.000	568,784.55
2010-07-01 00:00:00.000	560,885.33
2010-08-01 00:00:00.000	585,259.46
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:

The screenshot shows a database interface with three panes. The top-left pane is titled 'Script' and contains the following SQL query:

```

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;

```

The top-right panes are titled 'Console' and show the results of the query. The results are displayed in a grid table titled 'orders 1 X'.

	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:

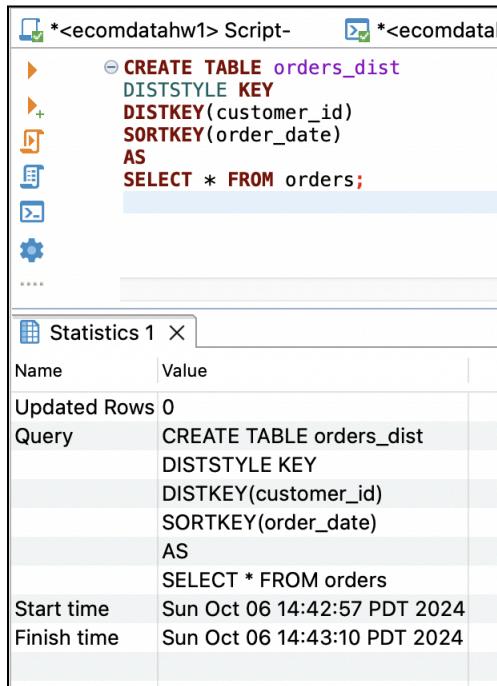
- Use Redshift's `DISTKEY` and `SORTKEY` to optimize query performance. For example:

```
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 they pertain to your use case.

## Orders Table Optimization:



The screenshot shows a database management interface with two main panes. The left pane is a script editor titled "Script" containing the SQL code for creating a distributed table "orders\_dist". The right pane is a "Statistics 1" panel showing execution details:

Name	Value
Updated Rows	0
Query	CREATE TABLE orders_dist DISTSTYLE KEY DISTKEY(customer_id) SORTKEY(order_date) AS SELECT * FROM orders
Start time	Sun Oct 06 14:42:57 PDT 2024
Finish time	Sun Oct 06 14:43:10 PDT 2024

## Order Items Table Optimization:

The screenshot shows a database interface with two tabs: 'Script-' and 'ecomdatahw1'. The 'Script-' tab contains the following SQL code:

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

The 'Statistics 1' section below the code provides the following information:

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:

The screenshot shows a database interface with two tabs: 'Script-' and 'ecomdatahw1'. The 'Script-' tab contains the following SQL code:

```
INSERT INTO orders_dist
SELECT * FROM orders;
INSERT INTO order_items_dist
SELECT * FROM order_items;
```

The 'Statistics 1' section below the code provides the following information:

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- *<ecomdatahw1>
  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- *<ecomdatahw1>
  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- *<ecomdatahw1>
  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:

```

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;
    
```

Name	Value
Updated Rows	0
Query	<pre> 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;     </pre>
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

```

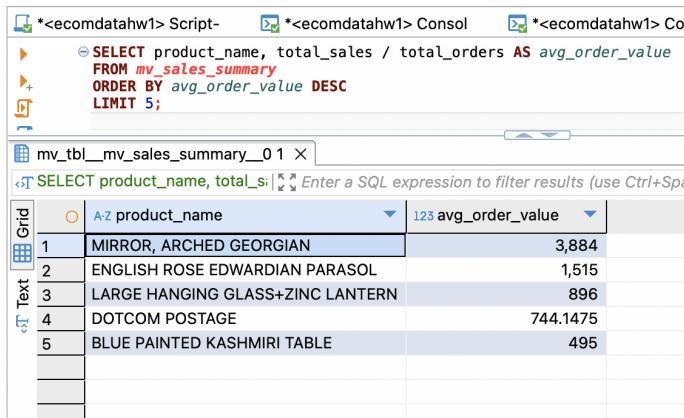
SELECT product_name, total_sales
FROM mv_sales_summary
ORDER BY total_sales DESC
LIMIT 5;
    
```

Grid	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



The screenshot shows a database interface with three tabs: Script, Console, and Co. The Script tab contains a SQL query:

```

SELECT product_name, total_sales / total_orders AS avg_order_value
FROM mv_sales_summary
ORDER BY avg_order_value DESC
LIMIT 5;

```

The Console tab shows the results of the query:

product_name	avg_order_value
MIRROR, ARCHED GEORGIAN	3,884
ENGLISH ROSE EDWARDIAN PARASOL	1,515
LARGE HANGING GLASS+ZINC LANTERN	896
DOTCOM POSTAGE	744.1475
BLUE PAINTED KASHMIRI TABLE	495

- **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:

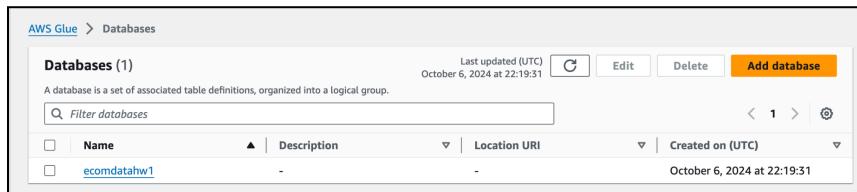
```
CREATE EXTERNAL SCHEMA spectrum_schema
FROM data catalog
DATABASE 'your_database'
IAM_ROLE 'arn:aws:iam::your-account-id:role/YourRedshiftRole';

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.

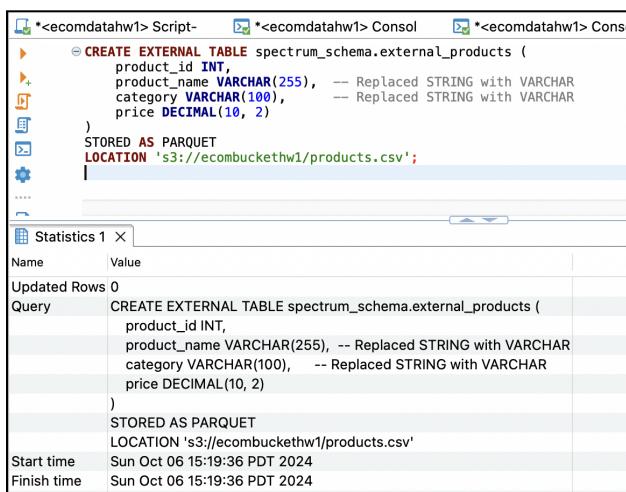
## Create an External Schema

## Create a Database in AWS Glue



The screenshot shows the AWS Glue Databases console. At the top, there's a header with 'AWS Glue > Databases'. Below it, a table lists a single database entry: 'ecomdatahw1'. The table has columns for Name, Description, Location URI, and Created on (UTC). The 'Created on (UTC)' column shows 'October 6, 2024 at 22:19:31'. There are buttons for 'Edit' and 'Delete' at the top right, and a 'Add database' button. A 'Filter databases' input field is also present.

## Create an External Table



The screenshot shows the AWS Glue Script Editor. The main pane displays a script with the following SQL code:

```
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';
```

Below the script, a 'Statistics 1' section provides details about the query execution:

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

```
Script- *<ecomdatahw1> Co
SELECT category, product_name, price
FROM spectrum_schema.external_products
ORDER BY category, price DESC
LIMIT 5;
```

Grid	A-Z product_name	123 category	123 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

```
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.