E-Commerce Analytics with Amazon Redshift

Shubham Manisha Naik
San Jose State University

***Abstract*—This report provides a comprehensive overview of setting up and utilizing Amazon Redshift for e-commerce data analysis. The focus includes cluster setup, data ingestion, query optimization, and advanced analytics using Redshift Spectrum and materialized views. Each step is detailed to explain its purpose, implementation, and benefits, emphasizing the capabilities of Amazon Redshift in business intelligence and data-driven decision-making.**

I. Introduction

Amazon Redshift is a cloud-based data warehousing solution designed for high-performance analytical queries. Its ability to process petabyte-scale data makes it an essential tool for businesses handling large datasets. This report focuses on the practical application of Redshift for analyzing e-commerce data, detailing:

- The configuration and setup of a Redshift cluster.
- Integration with AWS services such as S3 for data storage.
- Query optimization techniques using distribution and sort keys.
- Advanced analytics with materialized views and Redshift Spectrum.

This approach demonstrates how Redshift supports scalable and efficient data analysis.

II. Cluster Setup

A. Launching a Redshift Cluster

To begin, a Redshift cluster was created through the AWS Management Console. The following configurations were applied:

- **Cluster Identifier:** ecomdatahw1, to uniquely identify the cluster.
- **Database Name:** ecomdatahw1, for database-level management.
- **Admin Username and Password:** Used to ensure secure administrative access.
- **Node Type:** dc2.large, chosen for its balance between cost and performance.
- **Number of Nodes:** 2, enabling distributed data processing for scalability.

Purpose: This setup ensures a multi-node cluster for handling complex queries efficiently. The publicly accessible configuration allowed external SQL clients to connect for query execution.

B. Configuring Security Groups

Security groups were configured to restrict access to the cluster. Specific inbound rules were added to allow connections only from the user's IP address on port 5439. **Purpose:** This step was critical for protecting the cluster from unauthorized access while enabling secure connections.

C. Connecting to the Cluster

Connection details, such as the endpoint and port, were retrieved from the Redshift console. SQL Workbench/J was used to establish the connection and execute SQL queries. **Benefits:** Verifying connectivity ensured readiness for data ingestion and query execution.

III. Data Preparation and Ingestion

A. Dataset Description

The dataset included customer, product, order, and order item details, sourced from Kaggle. The data was uploaded to an S3 bucket for seamless integration with Redshift. **Purpose:** Storing data in S3 allows Redshift to efficiently access and process it, leveraging AWS's robust storage infrastructure.

B. Granting Redshift Access to S3

An IAM role with **AmazonS3ReadOnlyAccess** was created and attached to the Redshift cluster. **Purpose:** This role enabled secure access to S3, ensuring Redshift could directly ingest data using the COPY command without manual transfers.

C. Loading Data into Redshift

Tables were created with the following schema:

- **customers:** Customer ID and country.
- **products:** Product ID, name, and price.
- **orders:** Order ID, customer ID, and order date.
- **order_items:** Line item details including quantity and unit price.

Benefits: Organizing data into a star schema optimized it for analytical queries.

IV. Data Analysis and Queries

A. Key Queries

- **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;
```

This query revealed that the *Regency Cake Stand 3 Tier* was the highest revenue-generating product with $261K in sales.

- **Customer Purchase History:**

```
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;
```

This highlighted customer ID 14911 as the most active customer with 512 orders.

Purpose: These queries provided insights into product performance and customer behavior.

B. Derived Insights

- **Popular Products:** *World War 2 Gliders* was the most purchased item by quantity (103,505 units).
- **Revenue Distribution:** The United Kingdom contributed the highest sales revenue ($13M).
- **Seasonal Trends:** Sales spikes in December and September indicated holiday-driven demand.

Benefits: These insights supported targeted marketing and inventory planning.

V. Optimization and Advanced Features

A. Query Optimization

- **DISTKEY(customer_id):** Optimized joins between orders and customers.
- **SORTKEY(order_date):** Accelerated date-range queries for sales trends.
- **ANALYZE and VACUUM:** Maintained table performance by updating statistics and reclaiming storage.

Purpose: These optimizations reduced query execution time and resource usage.

B. Materialized Views

A materialized view precomputed total sales and order counts for faster analytics:

```
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;
```

Benefits: This significantly reduced query latency for frequently accessed data.

C. Redshift Spectrum Integration

External tables were created to query data stored in S3 directly:

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

Purpose: Spectrum enabled real-time analytics without loading data into Redshift, enhancing flexibility.

VI. Conclusion

This project demonstrated Amazon Redshift's effectiveness in managing and analyzing e-commerce data at scale. By combining advanced features such as distribution keys, sort keys, materialized views, and Spectrum integration, Redshift optimized query performance and provided actionable insights. These methodologies highlight its value as a critical tool for data-driven decision-making in business intelligence.

References

1) Amazon Redshift Documentation: https://docs.aws.amazon.com/redshift
2) Kaggle Dataset: Online Retail Dataset.