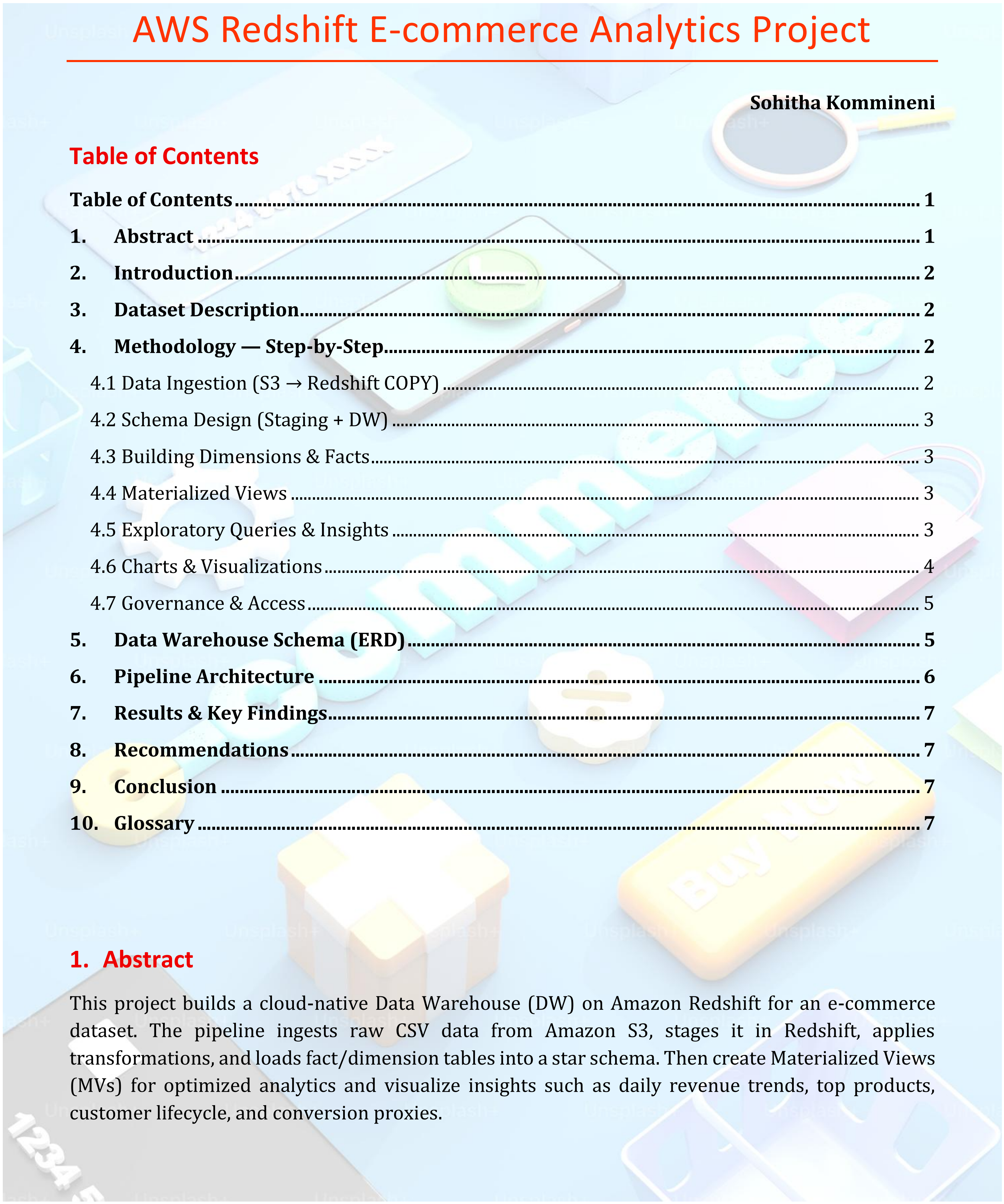


AWS Redshift E-commerce Analytics Project

Sohitha Kommineni

Table of Contents

1. Abstract

This project builds a cloud-native Data Warehouse (DW) on Amazon Redshift for an e-commerce dataset. The pipeline ingests raw CSV data from Amazon S3, stages it in Redshift, applies transformations, and loads fact/dimension tables into a star schema. Then create Materialized Views (MVs) for optimized analytics and visualize insights such as daily revenue trends, top products, customer lifecycle, and conversion proxies.

2. Introduction

E-commerce platforms generate vast amounts of customer, product, order, and web traffic data. The challenge is turning this raw data into fast, queryable insights for decision-making. This project demonstrates how Redshift can serve as a scalable analytics platform using a star schema + materialized views for performance.

3. Dataset Description

- customers.csv → Customer demographics and metadata
- products.csv → Product catalog (category, subcategory, brand, price, active flag)
- orders.csv → Orders with customer, channel, coupon, and revenue
- order_items.csv → Items per order with price and quantity
- web_events.csv → User web sessions and events

Row counts after load:

Table	Rows
web_events	360,000
products	900
customers	3,000
orders	60,000
order_items	149,796

4. Methodology — Step-by-Step

4.1 Data Ingestion (S3 → Redshift COPY)

What: The raw .csv files (customers, products, orders, order_items, web_events) are stored in Amazon S3.
Why: Redshift cannot directly read local files. Using S3 as a staging layer ensures scalable, cost-efficient storage.
How:

- Created an IAM Role (RedshiftS3AccessRole) with AmazonS3ReadOnlyAccess.
- Associated this IAM role with the Redshift Serverless workgroup.
- Used the COPY command to bulk load data from S3 into Redshift staging tables. COPY automatically handles parallel loading, compression, and error logging, making it the industry-standard method.

4.2 Schema Design (Staging + DW)

What: Designed a two-layer schema:

- stg schema → staging/landing area (raw ingestion).
- dw schema → curated data warehouse for analytics.
  Why: Separating staging from DW ensures data quality checks, transformations, and normalization before analytics. It prevents raw data issues from polluting the DW.
  How:
- **Staging tables:** mirror raw CSV structure, loaded directly with COPY.
- **DW schema (star schema):** designed around fact/dimension modeling.
  - **Dimensions** (who/what/when): dim_customer, dim_product, dim_date.
  - **Facts** (measures/transactions): fact_order, fact_order_item, fact_web_event.
    This approach improves query performance and supports BI-friendly reporting.

4.3 Building Dimensions & Facts

What: Populated the DW schema from staging with clean, structured data.
Why: Dimensions give descriptive context; fact tables provide numeric measures. Together they form a star schema for analytics.

How:

- Inserted distinct, cleaned data from staging into dimension tables (dim_customer, dim_product).
- Built dim_date using SQL date series (covering last 2 years).
- Inserted transactional data into fact tables:
  - fact_order: aggregated revenue/order-level metrics.
  - fact_order_item: line-item level granularity.
  - fact_web_event: user-level digital interactions.

4.4 Materialized Views

What: Created Materialized Views (MVs) to store pre-aggregated metrics.
Why: Redshift MVs boost query performance by persisting computed results. Unlike normal views, they don't recompute everything each query.

How:

- mv_daily_revenue: daily revenue + order counts for trend analysis.
- mv_product_sales: total units and sales per product, enriched with brand/category.
- mv_customer_lifecycle: customer first/last order, order count, lifetime revenue.
- MVs refreshed after data loads using REFRESH MATERIALIZED VIEW.

4.5 Exploratory Queries & Insights

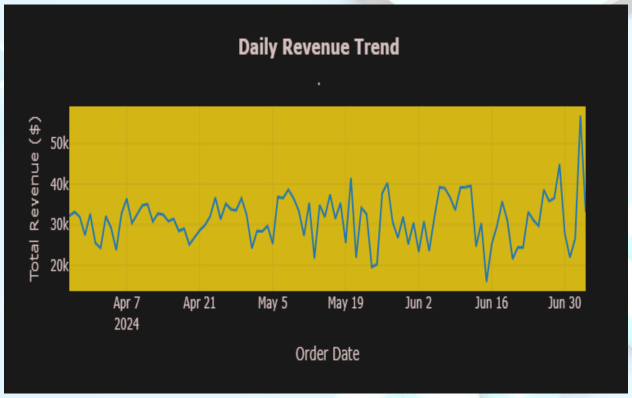
What: Analytical queries to derive insights from the DW.
Why: Helps validate the schema, test aggregations, and generate business KPIs.

How:

- Monthly Revenue & AOV (average order value).
- Revenue by Category (identify high-performing categories).
- Conversion Proxy (orders per web event).
  These queries feed Redshift charts for visualization.

4.6 Charts & Visualizations

What: Used Redshift Query Editor v2's built-in charting to visualize insights.
Why: Charts provide quick validation & stakeholder-friendly summaries.



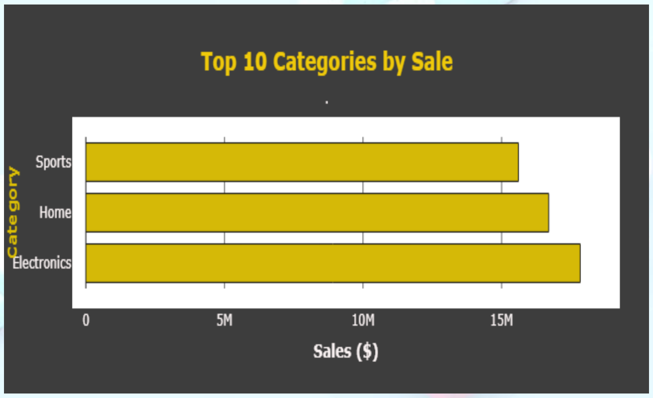
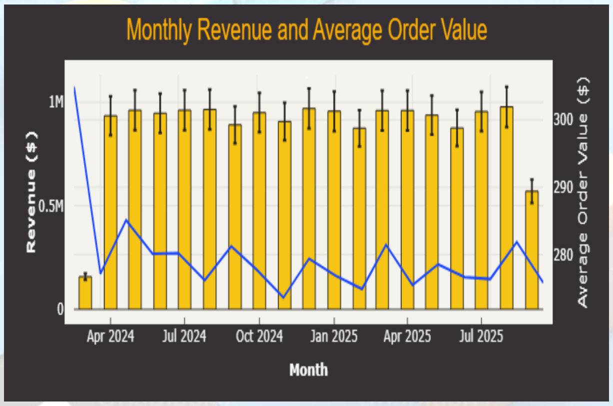


Fig.1. Daily Revenue Trend

Fig. 7. Revenue by Category



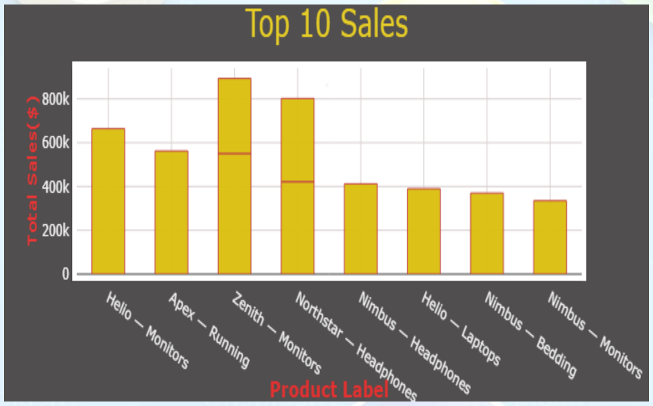
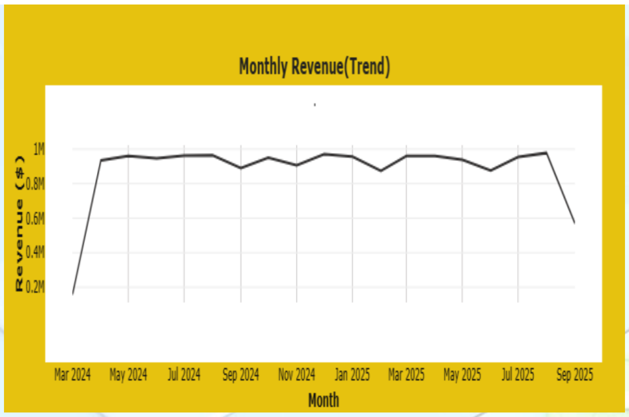


Fig. 2. Monthly Revenue and AOV

Fig.6. Top 10 Product Sales



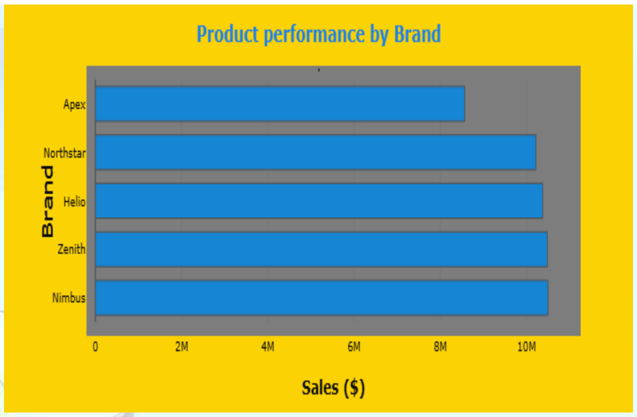
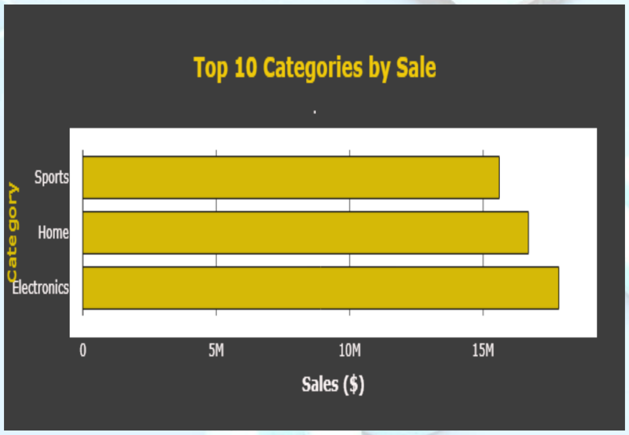


Fig.3. Monthly Revenue trend

Fig.7. Product Performance by brand



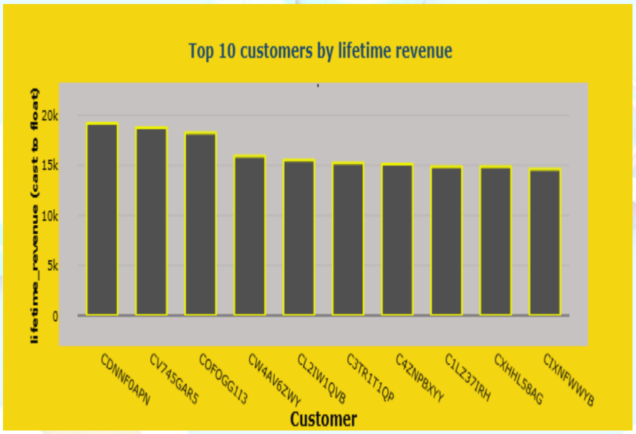


Fig .4. Top 10 Categories by sales

Fig.8. Top 10 customers

4.7 Governance & Access

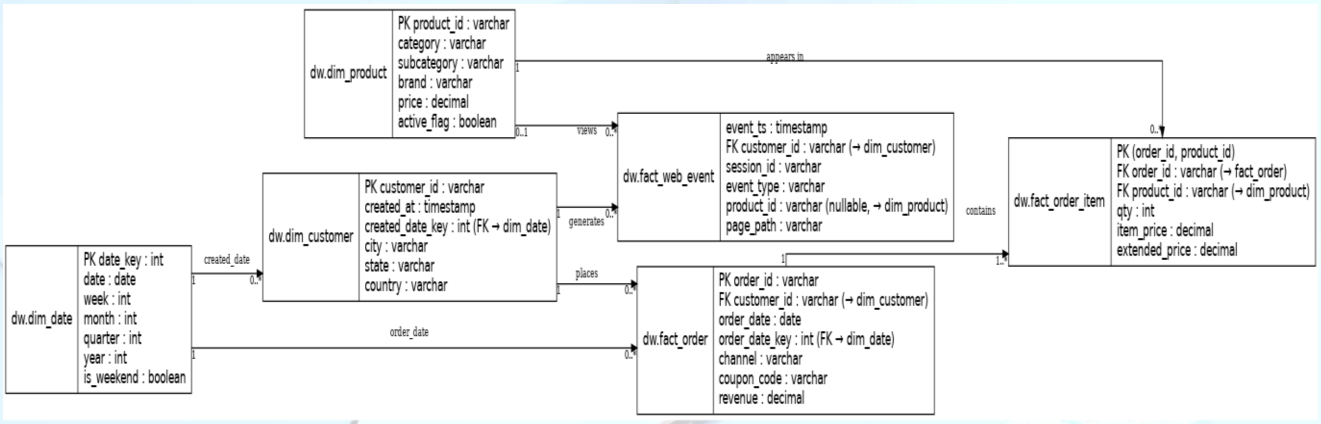
What: Security and access design for analytics users.
Why: In real projects, analysts shouldn’t modify schema/data, only query.

How:

- Granted USAGE on schema dw to BI users.
- Granted SELECT on all tables/views in dw.
- Applied default privileges so new objects inherit read-only access.
  This ensures controlled, read-only analytics access without exposing ingestion/transformation logic.

5. Data Warehouse Schema (ERD)

The ERD below shows the dimensional model with facts and dimensions:

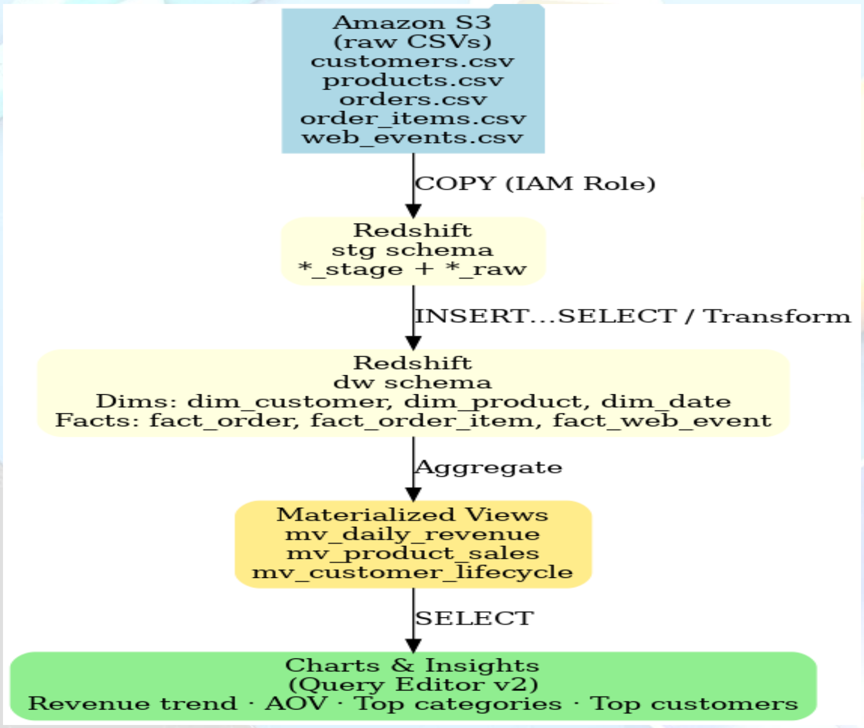


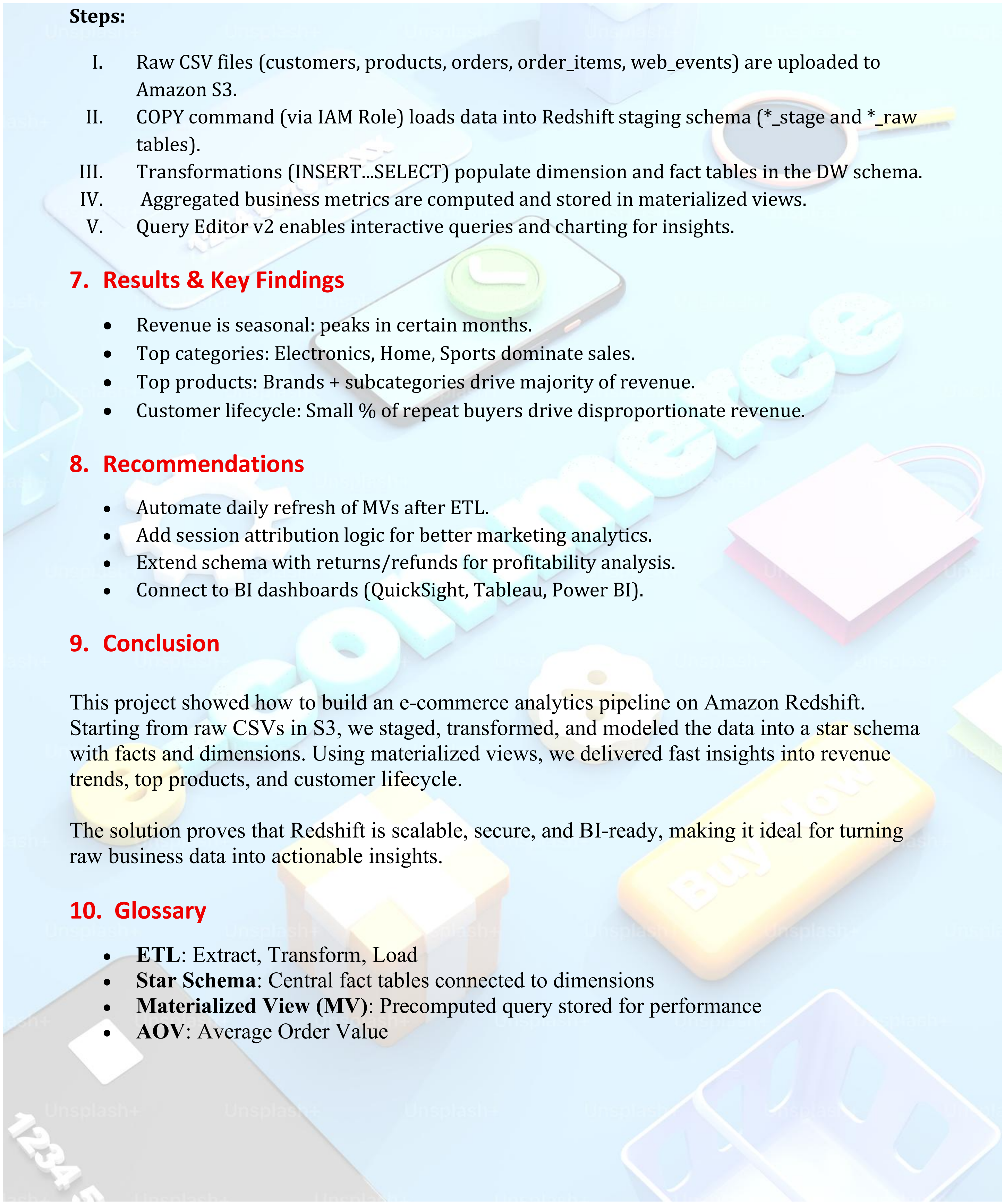
Key Tables:

- **dim_customer:** Customer attributes such as ID, city, state, and country.
- **dim_product:** Product attributes such as category, subcategory, brand, and price.
- **dim_date:** Date dimension supporting calendar-based analysis.
- **fact_order:** Order facts, including revenue and coupons.
- **fact_order_item:** Line-item level details of orders.
- **fact_web_event:** Web events tied to customer sessions.

6. Pipeline Architecture

The following diagram illustrates the flow of data from ingestion to analytics:





Steps:

I. Raw CSV files (customers, products, orders, order_items, web_events) are uploaded to Amazon S3.
II. COPY command (via IAM Role) loads data into Redshift staging schema (*_stage and *_raw tables).
III. Transformations (INSERT...SELECT) populate dimension and fact tables in the DW schema.
IV. Aggregated business metrics are computed and stored in materialized views.
V. Query Editor v2 enables interactive queries and charting for insights.

7. Results & Key Findings

- Revenue is seasonal: peaks in certain months.
- Top categories: Electronics, Home, Sports dominate sales.
- Top products: Brands + subcategories drive majority of revenue.
- Customer lifecycle: Small % of repeat buyers drive disproportionate revenue.

8. Recommendations

- Automate daily refresh of MVs after ETL.
- Add session attribution logic for better marketing analytics.
- Extend schema with returns/refunds for profitability analysis.
- Connect to BI dashboards (QuickSight, Tableau, Power BI).

9. Conclusion

This project showed how to build an e-commerce analytics pipeline on Amazon Redshift. Starting from raw CSVs in S3, we staged, transformed, and modeled the data into a star schema with facts and dimensions. Using materialized views, we delivered fast insights into revenue trends, top products, and customer lifecycle.

The solution proves that Redshift is scalable, secure, and BI-ready, making it ideal for turning raw business data into actionable insights.

10. Glossary

- **ETL**: Extract, Transform, Load
- **Star Schema**: Central fact tables connected to dimensions
- **Materialized View (MV)**: Precomputed query stored for performance
- **AOV**: Average Order Value