Optimized Batch & Stream Data Processing for Enhanced Inventory Management

Submitted in partial fulfilment of the requirements for the degree of

Post Graduate Diploma in Data Engineering

by

Sudarshan P (G23AI1046)

Utkarsh Gupta (G23AI1048)

Under the guidance of

Dr. Pradip Samal

IIT Jodhpur

Indian Institute of Technology Jodhpur
Advance Data Engineering in Cloud
Trimester-3 (July 2024)

Assignment – 1

Problem Statement

An e-commerce company aims to enhance inventory management and supply chain efficiency through analysis of batched sales data from multiple online platforms. The objective is to optimize inventory levels, minimize stockouts, and streamline fulfilment processes.

Real-Life Use Case: Batch Data Processing for Inventory Management

Scenario:

- **Online Sales Platforms:** Transactional data including orders, product details, and customer information.
- **Inventory Systems:** Stock levels, warehouse locations, and logistics data.
- **Supplier Information:** Lead times, pricing, and availability.

Challenges:

1. **Data Integration:**
    - Integrating batch data from diverse online platforms and internal systems.
    - Ensuring data consistency and accuracy across different sources.

2. **Inventory Management:**
    - Providing insights into inventory levels and sales trends through batch processing.
    - Predicting demand fluctuations and adjusting inventory accordingly.

3. **Scalability:**
    - Handling large volumes of batched transactional data from global online platforms.

- Scaling to accommodate peak shopping periods and seasonal demand variations.

4. **Operational Efficiency:**
    - Optimizing procurement and fulfillment processes based on batched data insights.
    - Minimizing storage costs and reducing excess inventory.

Solution:

Using AWS Services to Build a Batch Data Inventory Management System:

1. **Data Ingestion:**
    - **Amazon S3:** Upload batch data files (CSV, JSON) containing sales, inventory, and supplier information.
    - **AWS Kinesis Data Streams or AWS Direct Connect:** Implement the data ingestion mechanism to stream data from a source to Amazon S3.

2. **Data Processing:**
    - **AWS Glue:** Develop and test the data processing pipeline using Apache Spark.
        - Apply data transformation and cleansing techniques to prepare the data for aggregation and analysis.
        - Implement data partitioning and indexing strategies to optimize query performance.
        - Update the GitHub repository with the code and configuration files for data ingestion and processing.

3. **Data Storage:**
    - **Amazon RDS (Relational Database Service):**
        - Set up an RDS instance (e.g., MySQL, PostgreSQL) to store structured transactional data:
            - Sales transactions.
            - Product details.

        - Inventory levels.
        - Supplier information.

4. **Analytics and Reporting:**
    - **Amazon Athena:**
        - Query data directly in Amazon S3 for ad-hoc analysis and reporting.
    - **Quicksight**
        - Visualize data using Amazon QuickSight for interactive dashboards.

5. **Security and Access Control:**
    - **AWS IAM:** Manage access to RDS and other AWS services:
        - Ensure data privacy and compliance with regulatory requirements.

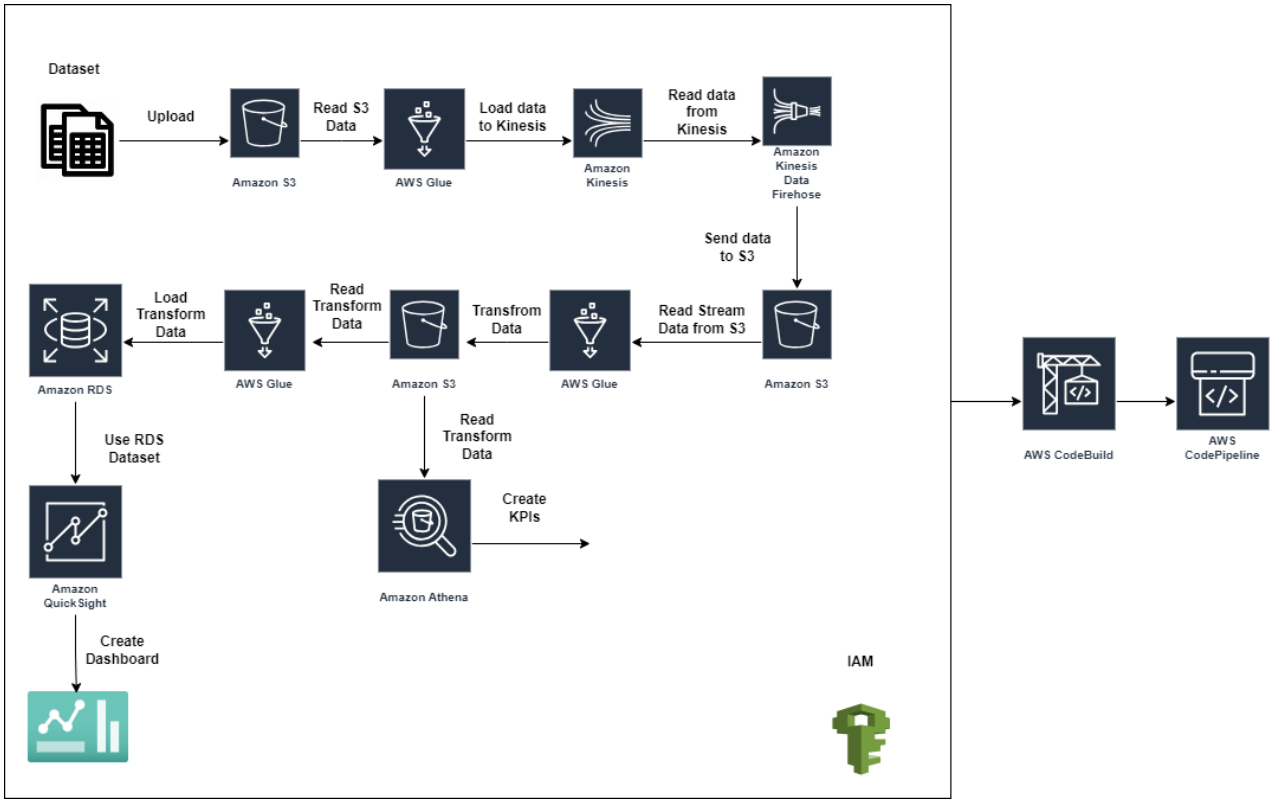
End-to-End Data Engineering Platform:

1. **Integration:**
    - Integrate the components developed to create a complete end-to-end data engineering platform.
2. **Data Management:**
    - Implement data retention policies and configure data lifecycle management in Amazon S3.
3. **Security Best Practices:**
    - Enable encryption for data at rest and in transit.
    - Set up proper access controls using AWS IAM for authentication and authorization.
4. **Cost Optimization:**
    - Utilize cost-effective storage options like Amazon S3 Glacier for long-term data retention.
    - Monitor and optimize resource utilization to reduce costs.
5. **Testing and Validation:**

- Conduct thorough testing of the platform, including data ingestion, processing, aggregation, and visualization, to ensure data integrity and performance.

6. **CI/CD Automation:**
   - Implement continuous integration and continuous deployment (CI/CD) processes using AWS CodePipeline and AWS CodeBuild to automate deployment and updates.

Architecture Diagram:



Key Performance Indicators (KPIs):

- **Inventory Turnover Ratio:** measures how efficiently inventory is managed by indicating how many times inventory is sold and replaced over a period.
- **Stockout Rate:** measures the percentage of time products are out of stock.

- **COGS to Revenue Ratio:** measures the efficiency of managing inventory costs relative to revenue.
- **Customer Acquisition Cost (CAC):** measures the average cost of acquiring a new customer.
- **Returning Customers**: The total number of unique customers who made purchases on July 11, 2011.
- **Average Order Value (AOV):** Measures the average amount spent per order over the course of the year 2011.
- **Gross Margin:** Measures the profitability of products by comparing the revenue (InvoiceTotal) to the cost of goods sold (Quantity * UnitPrice).
- **COGS to Revenue Ratio:** Efficiency in managing inventory costs relative to revenue.

Benefits:

- **Improved Inventory Management:** Enhanced control over inventory levels and reduced stockouts.
- **Efficient Supply Chain Operations:** Optimized procurement and fulfillment processes.
- **Scalability:** AWS services scale with business growth.
- **Cost Efficiency:** Managed services minimize infrastructure costs.
- **Security and Compliance:** Secure handling of data ensures compliance with regulations.