Real-Time Event-Driven Data Pipeline for an E-Commerce shop

Project Brief

You are tasked with designing and building a **real-time, event-driven data pipeline** to support operational analytics for an e-commerce platform.

The company deals with continuous inflow of transactional data (orders, products, etc.) in flat file format. These files arrive randomly in an **Amazon S3** bucket. The goal is to immediately validate, transform, and compute business KPIs, and store results for real-time querying via **Amazon DynamoDB**.

This is a **production-style project** requiring clear architecture, clean data flow, and automated orchestration. You must use AWS-native services in a containerized setup.

Core Services You Must Use

Your solution **must** use all four of the following services:

Service	**Purpose**
Amazon S3	Input file storage
Amazon ECS	Containerized data validation & transformation
AWS Step Functions	Workflow orchestration
Amazon DynamoDB	Real-time storage of KPIs and metrics

You are free to include other AWS services **as needed**

Your Mission

Build a data pipeline that:

1. **Detects** new data files arriving in S3.
2. **Validates** the contents using a containerized service.
3. **Transforms** valid data into business KPIs.
4. **Stores** KPIs in DynamoDB using an optimized schema.
5. **Archives** or logs processed files and outcomes.
6. **Automates** the entire workflow using Step Functions.

You are expected to make **architectural decisions** and simulate **realistic use cases**.

Required KPIs

You must compute and store the following two types of KPIs in DynamoDB as part of your transformation logic.

---

1. **Category-Level KPIs (Per Category, Per Day)**

Field	**Description**
category	Product category (e.g., Electronics)
order_date	Date of the summarized orders
daily_revenue	Total revenue from that category for the day
avg_order_value	Average value of individual orders in the category
avg_return_rate	Percentage of returned orders for the category

Store in a **DynamoDB table** optimized for querying by category and order_date.

---

2. Order-Level KPIs (Per Day)

Field	**Description**
order_date	Date of the summarized orders
total_orders	Count of unique orders
total_revenue	Total revenue from all orders
total_items_sold	Total number of items sold
return_rate	Percentage of orders that were returned
unique_customers	Number of distinct customers who placed orders

Store in a **separate DynamoDB table** optimized for querying by order_date.

Project Expectations

You are expected to:

- Demonstrate a deep understanding of **event-driven design**.
- Build a solution that can scale with **realistic data volumes**.
- Handle **validation, transformation, and storage** cleanly and modularly.
- Explain and justify all design choices.

System Requirements

ECS Tasks

- **Validation Task**:
  - Reject malformed or incomplete data.
  - Exit the pipeline gracefully on failure.
- **Transformation Task**:
  - Process clean data and compute KPIs.
  - Store results in DynamoDB.

Step Functions

- Orchestrate ECS tasks.
- Include failure paths, branching logic, and timeouts.

DynamoDB

- Design a table (or tables) optimized for querying KPIs.
- Use proper partition keys, sort keys, and secondary indexes if needed.

Logging & Monitoring

- Use **CloudWatch Logs** to track ECS task execution.
- Create error logs or alerts as appropriate.

Optional Bonus Challenges

These are recommended, and will push your learning even further:

- Add a **notification system** (e.g., email alert) for failures.
- Implement a **retry mechanism** or **dead-letter queue** for failed tasks.
- Add a simple **dashboard** to expose KPIs (can be mocked).

Need Help?

This project is designed to challenge you. If you're stuck:

- Start by sketching your **architecture on paper**.
- Break it down into services: **S3 → ECS → Step Functions → DynamoDB**.
- Simulate manually before fully automating.
- Reach out with specific blockers—i'll guide, not hand-hold.

I'm not sure where this should be; for the sake of consistency, please make sure your documentation includes **AT LEAST** the following points

- Your data format and sample schema
- Validation rules (e.g., missing fields, referential integrity)
- DynamoDB table structure and access patterns
- Step Function workflow explanation
- Any error-handling, retry, or logging logic
- Instructions to simulate or test the pipeline manually