# Car Rental Marketplace Data Pipeline Documentation

## Overview

This documentation describes a big data processing pipeline for a car rental marketplace using AWS EMR (Elastic MapReduce), Spark, Glue, Athena, and Step Functions. The pipeline processes raw data from S3, transforms it to derive key business metrics, and makes it available for analysis.

**The pipeline follows this workflow**:

1. Raw data is extracted from S3
2. EMR processes the data with Spark jobs
3. Transformed data is loaded back to S3
4. Glue Crawler catalogs the data
5. Athena queries the processed data
6. Step Functions orchestrates the entire workflow

## Prerequisites

- AWS account with appropriate permissions
- S3 buckets for raw data, processed data, and logs
- IAM roles for EMR (EMR_DefaultRole and EMR_EC2_DefaultRole)
- AWS Glue Crawler configured
- Athena workgroup created

## Pipeline Components

### 1. AWS Step Functions State Machine

The state machine orchestrates the entire workflow with these steps:

States:

- **CreateEMRCluster**: Creates an EMR cluster with Spark and Hive
- **ProcessEMRJobs**: Runs parallel Spark jobs:
    - Location and Vehicle Performance Metrics
    - User and Transaction Analysis
- **RunGlueCrawler**: Triggers the Glue Crawler

- **ExecuteAthenaQueries**: Runs analytical queries:
    - Highest revenue locations
    - Most rented vehicle types
    - Top spending users
- **TerminateCluster**: Shuts down the EMR cluster

Error Handling:

- Automatic cluster termination on failure
- Retry logic for crawler operations
- Detailed error logging

**2. Spark Jobs**

Location and Vehicle Performance Metrics (location_vehicle_kpi.py)

**Inputs:**

- s3://<raw-data-bucket>/raw_data/vehicles.csv
- s3://<raw-data-bucket>/raw_data/locations.csv
- s3://<raw-data-bucket>/raw_data/rental_transactions.csv

**Outputs:**

- s3://<processed-data-bucket>/processed_data/location_performance_metrics/
- s3://<processed-data-bucket>/processed_data/vehicle_type_performance_metrics/

**Key Metrics Calculated:**

- Revenue per location
- Transactions per location
- Average transaction amounts
- Unique vehicles per location
- Rental duration by vehicle type

User and Transaction Analysis (user_transaction_kpi.py)

**Inputs:**

- s3://<raw-data-bucket>/raw_data/users.csv

- s3://<raw-data-bucket>/raw_data/rental_transactions.csv

**Outputs:**

- s3://<processed-data-bucket>/processed_data/daily_transaction_metrics/
- s3://<processed-data-bucket>/processed_data/user_engagement_metrics/

**Key Metrics Calculated:**

- Daily transactions and revenue
- User spending patterns
- Rental duration metrics
- Transaction value statistics

**3. AWS Services Configuration**

**EMR Cluster Configuration**

- **Release Label**: emr-6.9.0
- **Applications**: Spark, Hive
- **Instance Types**: m5.xlarge
    - 1 Master node
    - 2 Core nodes
- **Spark Configuration**:
    - Driver memory: 4g
    - Executor memory: 4g
    - Executor cores: 2
- **Logging**: Enabled with aggregation

**Glue Crawler**

- Name: crawl_rentals_processed_data
- Targets processed data in S3
- Creates/updates Glue Data Catalog

**Athena Queries**

- Workgroup: primary
- Output location: s3://<query-results-bucket>/query-results/

- Sample queries:
    - Top revenue locations
    - Most popular vehicle types
    - Highest spending users

## Setup Instructions

1. **Upload Data to S3**:
    - Create buckets for raw data, processed data, and logs
    - Upload CSV files to raw data bucket:
        - vehicles.csv
        - locations.csv
        - users.csv
        - rental_transactions.csv
2. **Upload Spark Scripts**:
    - Place location_vehicle_kpi.py and user_transaction_kpi.py in s3://<scripts-bucket>/scripts/
3. **Configure Glue Crawler**:
    - Set up crawler to target processed data locations
    - Configure database (e.g., car_rental_db)
4. **Deploy Step Functions**:
    - Create state machine using provided JSON definition
    - Update bucket names in the configuration
    - Set appropriate IAM permissions

## Execution

1. **Start Pipeline**:
    - Execute the Step Functions state machine
    - Monitor progress in AWS Console
2. **Expected Outputs**:
    - Processed Parquet files in S3

    - Glue Data Catalog updates
    - Athena query results
    - Cluster termination upon completion

## Monitoring and Troubleshooting

### Logging Locations

- EMR cluster logs: s3://<logs-bucket>/emr-logs/
- Spark application logs: Included in EMR logs
- Step Functions execution history: Available in AWS Console
- Athena query results: s3://<query-results-bucket>/query-results/

### Common Issues and Solutions

1. **Cluster Creation Failure**:
    - Verify IAM roles exist
    - Check EC2 instance limits
    - Validate subnet configurations
2. **Spark Job Failures**:
    - Check EMR step logs
    - Verify input data exists in S3
    - Validate file formats and schemas
3. **Glue Crawler Issues**:
    - Verify crawler has permissions to access S3
    - Check processed data exists before running crawler
    - Review crawler logs for schema inference errors
4. **Athena Query Failures**:
    - Verify tables exist in Glue Data Catalog
    - Check table names in queries match catalog
    - Validate output bucket permissions

## Maintenance

1. **Cost Optimization**:

- Monitor and adjust cluster size
- Set appropriate auto-termination policies
- Clean up unnecessary S3 data

2. **Updates**:
    - Review and update Spark scripts as needed
    - Adjust metrics calculations based on business needs
    - Update Athena queries for new analytical requirements

## Sample Queries for Analysis

```
-- Top 5 revenue generating locations
SELECT location_name, city, state, total_revenue
FROM car_rental_db.location_performance_metrics
ORDER BY total_revenue DESC
LIMIT 5;

-- Most active vehicle types
SELECT vehicle_type, rental_count
FROM car_rental_db.vehicle_type_performance_metrics
ORDER BY rental_count DESC
LIMIT 5;

-- User engagement metrics
SELECT first_name, last_name, total_user_transactions, total_user_spending
FROM car_rental_db.user_engagement_metrics
ORDER BY total_user_spending DESC
LIMIT 10;
```

## Conclusion

This pipeline provides a complete solution for processing car rental marketplace data, from raw data ingestion to analytical insights. The automated workflow ensures

consistent processing while the modular design allows for easy maintenance and extension.