New York Institute of Technology

College of Engineering and Computing Sciences

Big Data Project Summary Report

Summer 2023 | Professor Liangwen Wu

Kamran Adil | Mysura Reddy Kuchuru

Project Goal: The Goal of the Project is to visualize the Sales Patterns Understanding, Channel Effectiveness Assessment, and Product Performance Evaluation from the given dataset

Data Lake vs. Data Warehouse.

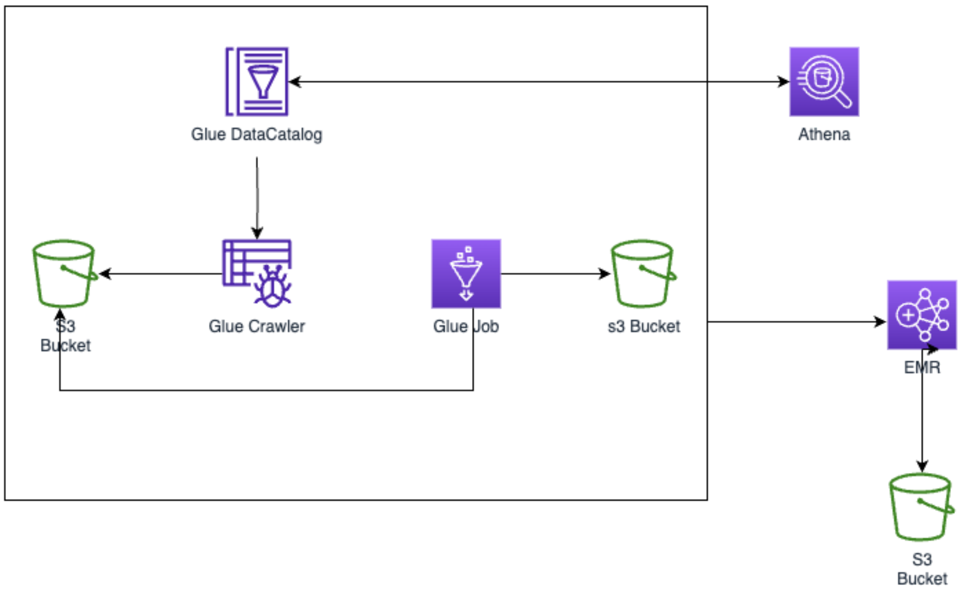
For this project, let's choose a Data Lake for storing the data. Data lakes are suited for storing large amounts of raw data in their native format, which can then be processed and transformed as needed. This flexibility is advantageous when dealing with diverse data sources and uncertain analysis requirements. Additionally, since we'll be using Spark for processing, Data Lakes can seamlessly integrate with Spark's processing capabilities.

SNo	Criteria	Data Lake	Data Warehouse
1	Storage Approach	Stores raw data in its native format	Stores structured, processed data
2	Flexibility	Suited for diverse data sources and formats	Best for structured, well-defined data
3	Data Processing	Processing and transformation as needed	Often involves predefined schema
4	Analysis Requirements	Flexible for uncertain analysis needs	Better for predefined analytical queries
5	Integration with Processing	Seamlessly integrates with Spark processing	Integration depends on the DW technology
6	Use Case	Suited for big data and unstructured data	Well-suited for business intelligence

Data Set Derived From the Below Link :

https://www.kaggle.com/datasets/talhabu/us-regional-sales-data

Our Datalake would be AWS Services like S3, Glue, Athena, and EMR.



Connecting Data Lake to AWS Services.

We will deploy a cloud formation template to provision the architecture shown above. It can be found on the GitHub link below.

https://github.com/kamranadil436/bigdata-finalproject
https://github.com/mysurakuchuru/BigData_FinalProject.git

Running cloudformation.yml

Resource Definition and Parameters: The code defines parameters that allow customization during deployment, such as Subnet ID, Security Groups, and EC2 Key Pair. It establishes the foundational components needed for the data lake setup.

S3 Buckets Setup: The template creates S3 buckets for various purposes. The "RawDataBucket" holds the initial data. The "ScriptBucket" stores Glue scripts. The "AnalyticsBucket" and "ResultBucket" are meant for storing analytics output and final results, respectively. The "EmrLogBucket" captures logs from the EMR cluster.

AWS Glue Configuration: AWS Glue components are configured. "GlueDatabase" defines a database named "my_database" in the Glue Data Catalog. "GlueCrawler" scans the raw data bucket, extracts schema information, and populates the Glue Catalog for future processing.

AWS Glue Job Configuration: "GlueJob" defines a Glue Job named "my-glue-job" that utilizes the Glue ETL (Extract, Transform, Load) functionality. It specifies a Python script located in the "my-glue-data-bucket-bdpro" bucket.

AWS EMR Cluster Setup: The "EMRCluster" resource provisions an EMR cluster with "m5.xlarge" instances. It configures a master instance group and a core instance group, both using the ON_DEMAND market. The cluster runs Spark applications and leverages roles like "EmrServiceRole" and "EmrJobFlowInstanceProfile."

The CloudFormation code sets up a versatile data lake infrastructure on AWS. It defines parameters for customization and creates S3 buckets for data storage. It configures AWS Glue components for data cataloging and transformation tasks. Additionally, it provisions an EMR cluster for large-scale data processing using Spark applications. This technical blueprint serves as a foundation for implementing data pipelines and advanced analytics solutions.

Running the spark code over our distributed services.

Initialization and Configuration: The script begins by importing the required modules (sys, getResolvedOptions, SparkContext, GlueContext, Job) for AWS Glue and PySpark. It obtains and initializes various components, including SparkContext (sc), GlueContext (glueContext), SparkSession (spark), and a Glue job (job). Command-line arguments are resolved using getResolvedOptions().

Data Reading: The script employs the SparkSession's read.csv() method to load CSV data from "s3://my-raw-data-bucket-bdpro/dataset.csv" into a DataFrame (data_frame). It automatically detects headers and infers the schema from the data.

Data Transformation: The script uses the dropna() function to create a new DataFrame (data_frame_no_nulls) by removing rows with null values from data_frame. This step ensures data quality by eliminating incomplete records.

Data Writing: The cleaned DataFrame is written back to Amazon S3 in Parquet format using the write.parquet() function. The data is stored at "s3://my-analytics-bucket-bdpro/dataset.parquet". The "overwrite" mode replaces any existing data, and "snappy" compression is applied for efficient storage.

Job Execution: The Glue job is committed using job.commit(). This action finalizes the processing, ensuring the tasks are successfully executed and data is saved.

The provided Python script demonstrates the power of AWS Glue and PySpark for data processing. It ingests CSV data, performs data cleansing by removing null values, and writes the processed data to Amazon S3 in Parquet format. This technical implementation showcases the integration of data transformation and storage in a scalable and efficient manner, setting the groundwork for more complex data workflows.

Exploring Sales Data Through Spark and Visualizations

Spark Initialization and Data Loading The code begins by initializing a Spark session and loading CSV data into a DataFrame from an S3 location. This data frame will serve as the foundation for subsequent analysis and visualization tasks.

Time Series Analysis
The code performs time series analysis on sales data, aggregating order quantities over different dates. It converts the data to a Pandas DataFrame for plotting and uses Matplotlib to visualize the trend of total quantities over time. The graph helps identify sales patterns and trends, aiding in understanding demand fluctuations.

Sales Channel Comparison
The code compares sales performance across different sales channels. It groups data by sales channels, calculates total quantities, and creates a bar plot using Pandas. This visualization enables a quick comparison of sales distribution among various channels, providing insights into channel effectiveness.

Product Analysis
The code analyzes the distribution of sales across different products. It groups data by product IDs and calculates total quantities, creating a pie chart with percentages. This visualization helps in understanding the popularity of products and their contribution to overall sales.

Discount Analysis
The code explores the impact of discounts on sales by plotting a scatter plot. It groups data by discount levels, calculates total quantities, and then visualizes how discounts influence sales. This analysis provides insights into the effectiveness of discount strategies.

Conclusion:
The code showcases the process of loading and analyzing sales data using PySpark and visualizing insights using Matplotlib and Pandas. Through time series analysis, sales channel comparison, product distribution visualization, and discount impact analysis, the code demonstrates how businesses can derive actionable insights from their sales data to enhance decision-making and sales strategies.

End Summary/Context :

In conclusion, our Big Data project revolves around achieving key objectives through effective visualization and data modeling. We intend to gain deeper insights into sales patterns, assess the effectiveness of various sales channels, and evaluate the performance of different products using the provided dataset.

Choosing a Data Lake as our storage solution, leveraging AWS services such as S3, Glue, Athena, and EMR, aligns well with our project's goals. The inherent flexibility of Data Lakes enables us to efficiently manage diverse data sources and adapt to evolving analysis requirements.

Our project architecture, deployed through a CloudFormation template, establishes the necessary infrastructure to seamlessly integrate data processing across our distributed services. The Spark script plays a pivotal role in ingesting, transforming, and storing data, leveraging the capabilities of AWS Glue and PySpark.

Through these implementations, our project aims to empower businesses with actionable insights, foster data-driven decision-making, and optimize sales strategies. By visualizing sales patterns, assessing channel effectiveness, and evaluating product performance, we are poised to drive informed choices and enhance sales performance in a dynamic market landscape.