Getting Started with Pyspark on AWS EMR and Athena

CookBook

Table of Content:

Introduction

FinTech companies are businesses that use technology to provide financial services to customers. These companies leverage innovative technologies to offer financial services in a more efficient and cost-effective way than traditional financial institutions. FinTech companies are disrupting the traditional financial services industry, and their services include payments, lending, insurance, wealth management, and many others.

One of the most significant advantages of FinTech companies is their ability to offer financial services to underserved markets, such as small businesses and individuals with poor credit histories. They use innovative technologies to analyze customer data and assess risk, enabling them to offer loans and other financial products to customers who may not have been eligible for these services in the past.

FinTech lending is a growing sector of the FinTech industry, and it has been enabled by the rise of big data and machine learning technologies. These technologies allow FinTech companies to analyze large volumes of data to assess the creditworthiness of borrowers accurately. FinTech lenders use a range of data sources, including bank statements, payment history, tax returns, and credit reports, to determine the creditworthiness of borrowers.

The use of data analytics and machine learning enables FinTech lenders to make lending decisions quickly and efficiently. This allows them to offer loans with lower interest rates and better terms than traditional lenders, making them attractive to borrowers. Furthermore, the use of digital platforms enables borrowers to apply for loans and receive funding quickly and easily.

However, FinTech lenders also face challenges, particularly around regulatory compliance. As they deal with sensitive financial data, they need to ensure that they comply with all relevant laws and regulations around data protection and privacy. In addition, the use of machine learning algorithms for credit assessments can raise issues around transparency and fairness. FinTech lenders must be transparent about the data they use and how they use it to ensure that their lending decisions are fair and non-discriminatory.

In conclusion, FinTech companies are transforming the financial services industry by leveraging technology to provide innovative and cost-effective financial services. FinTech lending, in particular, is growing rapidly, enabled by big data and machine learning technologies. While FinTech lenders face regulatory and ethical challenges, they are well-positioned to offer financial services to underserved markets and provide borrowers with faster and more convenient access to credit.

Financial technology (FinTech) companies are rapidly growing and offering various services to customers. One of the most significant services is lending money to individuals and businesses. To determine the creditworthiness of borrowers, these companies use a rigorous analysis of their past transactions and remittances. However, the analysis cannot proceed further without appropriate data preparation and staging from raw transactional logs. In this article, we will explore the importance of data preparation in FinTech lending.

The Significance of Data Preparation in FinTech Lending:

Data preparation involves collecting, cleaning, and organizing raw data to ensure it is ready for analysis. It is a crucial step in any data-driven project, including FinTech lending. Inaccurate or incomplete data can lead to wrong conclusions, which can have severe consequences for the lending company and the borrower. Therefore, data preparation is vital to ensure the accuracy of the analysis and, ultimately, the lending decision.

Data preparation starts with collecting the necessary data from various sources, including bank statements, payment receipts, tax returns, and credit reports. Once collected, the data is cleaned to remove duplicates, errors, and inconsistencies. This process can be automated using machine learning algorithms to identify and fix data issues. The data is then transformed into a format that is suitable for analysis. This transformation may involve filtering, aggregating, or merging data from different sources.

Once the data is transformed, it is staged, which involves organizing and storing it in a way that makes it easily accessible for analysis. This process includes selecting the appropriate database management system, creating data tables, and indexing data for efficient querying. The staged data is then ready for analysis.

Data Description

OpenExchangeRates.org API provides access to real-time and historical foreign exchange rates for over 200 currencies. The API provides endpoints that allow users to retrieve exchange rate data in various formats, including JSON, XML, and CSV.

The API supports various types of requests, including single currency conversion, multiple currency conversions, and time-series data for historical exchange rates.

To access the OpenExchangeRates.org API, users need to sign up for an API key, which can be obtained for free or by subscribing to a paid plan with additional features and usage limits.

The API provides various parameters and options to customize the response, such as the base currency, date range, and currency symbols. Additionally, the API supports various programming languages and libraries, making it easy to integrate with various applications and systems.

Services used in the project

AWS S3

Amazon S3 (Simple Storage Service) is a cloud-based object storage service provided by Amazon Web Services (AWS). It allows businesses and individuals to store and retrieve data from anywhere on the internet. Amazon S3 is designed to be highly scalable, reliable, and cost-effective, making it a popular choice for storing a wide variety of data, including images, videos, documents, and backups.

Some of the key features of Amazon S3 include:

1. **Scalability:** Amazon S3 is highly scalable and can handle storage needs ranging from a few gigabytes to multiple petabytes.
2. **Security:** Amazon S3 provides several security features to protect data, including encryption in transit and at rest, access controls, and multi-factor authentication.
3. **Durability and availability:** Amazon S3 store data across multiple data centers, providing high durability and availability. It also provides a service-level agreement (SLA) for availability of 99.999999999% (11 nines).
4. **Integration:** Amazon S3 integrates with a wide range of AWS services, including compute, database, analytics, and machine learning services.
5. **Cost-effective:** Amazon S3 offers a pay-as-you-go pricing model, allowing customers to pay only for the storage and data transfer they use.

Here are some disadvantages of using Amazon S3:

1. **Complexity:** The various features and configuration options of Amazon S3 can make it difficult to set up and manage for users without a technical background.
2. **Cost:** Although Amazon S3 can be cost-effective for smaller storage needs, costs can quickly add up for large volumes of data and high usage.
3. **Data transfer fees:** There are additional fees for transferring data in and out of Amazon S3, which can increase costs for businesses with high data transfer needs.
4. **Performance:** While Amazon S3 offers high durability and availability, its performance can be slower compared to other storage options for certain use cases.
5. **Vendor lock-in:** Moving data out of Amazon S3 to another storage solution can be challenging, potentially creating a vendor lock-in situation.

Overall, Amazon S3 is a powerful and flexible storage solution that can be used for a wide range of use cases, from backup and disaster recovery to big data analytics and web hosting.

Amazon EMR

Amazon EMR (Elastic MapReduce) is a fully-managed big data processing service provided by Amazon Web Services (AWS). It simplifies the processing of large amounts of data using popular big data frameworks such as Apache Hadoop, Apache Spark, and Presto. Amazon EMR can automatically provision, scale, and manage clusters of virtual servers running Hadoop, Spark, and other big data frameworks. It provides an easy-to-use web interface and command-line tools to create and manage EMR clusters, as well as APIs to automate cluster creation and management.

Here are some key features of Amazon EMR:

1. **Fully managed:** Amazon EMR is a fully-managed service that takes care of the underlying infrastructure, software installation, configuration, and maintenance of the big data frameworks and tools.
2. **Scalable:** Amazon EMR can automatically scale the processing capacity up or down based on the size and complexity of the data processing workloads.
3. **Flexible:** Amazon EMR supports a wide range of big data frameworks and tools, such as Apache Hadoop, Apache Spark, Apache Hive, Apache Pig, and Presto.
4. **Cost-effective:** Amazon EMR uses a pay-as-you-go pricing model, allowing you to pay only for the processing resources you use. You can also use spot instances to reduce costs.
5. **Integration:** Amazon EMR integrates with other AWS services such as Amazon S3, Amazon DynamoDB, and Amazon Redshift, making it easy to move data in and out of EMR.

Here are some disadvantages of using Amazon EMR:

1. **Complexity:** Setting up and configuring EMR clusters can be complex, especially for users who are new to big data processing. It requires knowledge of big data frameworks and the Amazon Web Services environment.
2. **Cost:** While EMR uses a pay-as-you-go pricing model, costs can add up quickly, especially if you are processing large volumes of data or using high-performance instances.
3. **Data transfer costs:** If you are moving data between AWS services or from outside AWS into EMR, there may be additional data transfer costs that you need to consider.
4. **Limited control:** As a managed service, EMR may not give you full control over the underlying infrastructure or software. This may limit your ability to customize or optimize your clusters for specific workloads.

5. **Limited support for certain tools:** While EMR supports many big data frameworks and tools, there may be certain tools or features that are not fully supported or integrated.
6. **Reliance on AWS:** If you are using EMR, you are relying on AWS for your big data processing needs. This may limit your ability to use other cloud providers or run big data workloads on-premises.

Some of the use cases for Amazon EMR include:

1. **Data processing and analysis:** Amazon EMR can be used for processing and analyzing large volumes of data, such as log files, social media data, and sensor data.
2. **ETL (Extract, Transform, Load):** Amazon EMR can be used for ETL workloads, such as transforming and loading data from various sources into a data warehouse.
3. **Machine learning:** Amazon EMR can be used for building and training machine learning models using frameworks like Apache Spark MLlib and TensorFlow.
4. **Real-time processing:** Amazon EMR can be used for real-time processing of streaming data using tools like Apache Kafka and Apache Flink.

In summary, Amazon EMR is a powerful, flexible, and cost-effective big data processing service that can help businesses process and analyze large amounts of data efficiently. Its ease of use and integration with other AWS services make it a popular choice for data engineers, data analysts, and data scientists.

Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL queries. It is a serverless, fully managed service that doesn't require any infrastructure setup or maintenance, making it easy to get started with querying data immediately.

Here are some key features of Amazon Athena:

1. **Serverless:** Amazon Athena is a serverless service, which means that it automatically scales up or down based on your query workload. You don't need to provision any servers or manage any infrastructure.
2. **Fully managed:** Amazon Athena is a fully managed service, which means that Amazon takes care of all the underlying infrastructure, including backups, security, and patching.
3. **Pay-as-you-go pricing:** Amazon Athena uses a pay-as-you-go pricing model, which means that you only pay for the queries that you run.
4. **Integration:** Amazon Athena integrates seamlessly with other AWS services, including Amazon S3, AWS Glue, and Amazon Redshift.
5. **SQL-based queries:** Amazon Athena supports standard SQL queries, making it easy for data analysts and business users to get started with querying data without having to learn new query languages or tools.
6. **Schema-on-read:** Amazon Athena uses a schema-on-read approach, which means that you can store data in Amazon S3 without having to define a schema beforehand. This allows for greater flexibility and agility when working with data.

Here are some disadvantages of using Amazon Athena:

1. **Performance:** Amazon Athena performance can be affected by the size and complexity of the data you are querying, as well as the complexity of the SQL queries you are running.
2. **Limited functionality:** Amazon Athena is designed for ad hoc querying and exploration of data in Amazon S3 using standard SQL. While it can be used for more complex analytics and data processing tasks, it has limited functionality compared to other big data tools and platforms.
3. **Cost:** While Amazon Athena uses a pay-as-you-go pricing model, costs can add up quickly, especially if you are querying large volumes of data or running complex queries.
4. **Integration:** Amazon Athena is designed to work with data stored in Amazon S3, which may limit its usability if you need to work with data stored in other locations or formats.
5. **Data formats:** Amazon Athena works best with data stored in columnar formats like Apache Parquet or Apache ORC. If your data is stored in other formats, you may need to convert it to a supported format before querying it with Athena.

6. **Data availability:** Amazon Athena requires that your data be stored in Amazon S3 and be available at the time of querying. If your data is not immediately available, you may need to wait for it to be processed by other AWS services like AWS Glue.

Some of the use cases for Amazon Athena include:

1. **Ad hoc querying:** Amazon Athena is ideal for ad hoc querying and exploration of large datasets in Amazon S3.
2. **Business intelligence:** Amazon Athena can be used for business intelligence and reporting, allowing business users to easily access and analyze data without requiring IT support.
3. **Log analysis:** Amazon Athena can be used for analyzing log data stored in Amazon S3, allowing you to gain insights into application and system performance.
4. **Machine learning:** Amazon Athena can be used for preparing and transforming data for machine learning models, allowing you to quickly iterate and refine your models.

In summary, Amazon Athena is a powerful and easy-to-use query service that makes it easy to analyze data stored in Amazon S3 using standard SQL queries. Its serverless and fully managed architecture, pay-as-you-go pricing model, and integration with other AWS services make it a popular choice for data analysts, business users, and data scientists.

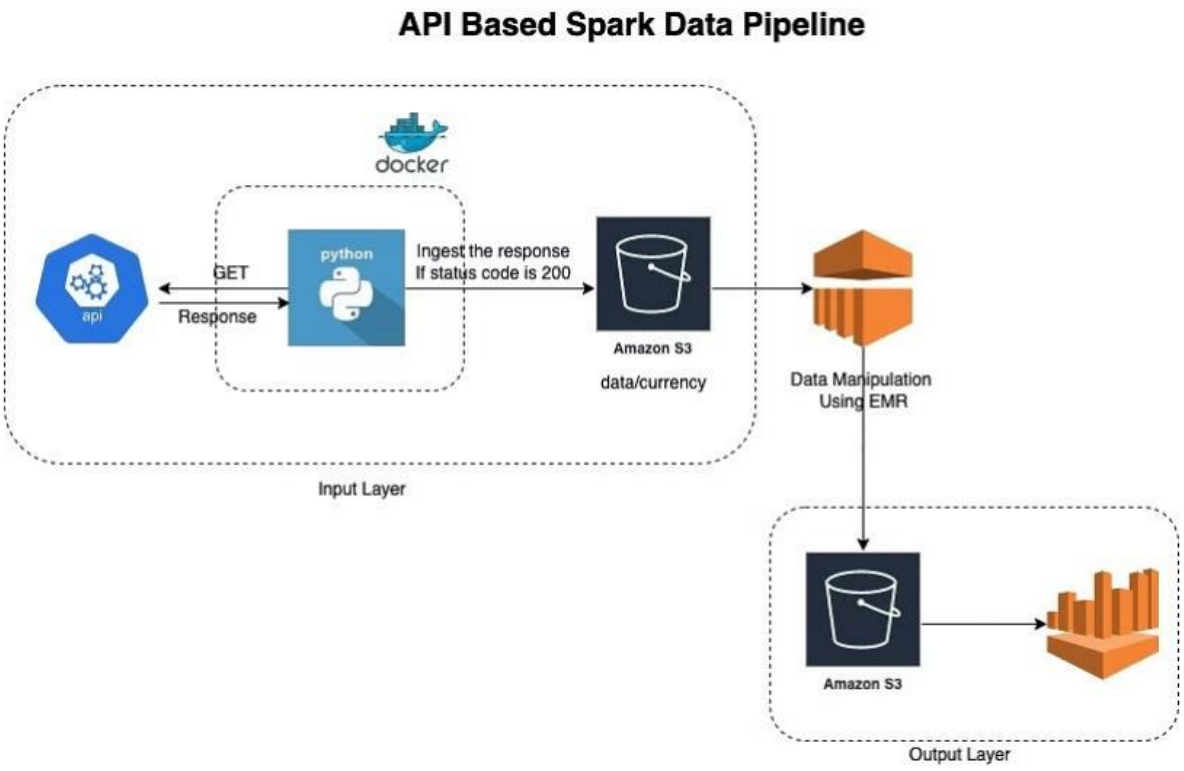
Use-case depicted in this project

The objective of this project is to conduct Spark Transformations on banking transactions data. The data will be obtained from a real-time currency ticker API, which will enable the application to retrieve accurate and up-to-date exchange rates for various currencies. By incorporating the currency data into the processing pipeline, the application will be able to perform the required transformations, such as currency conversion and aggregation, on the transaction data.

Once the processing is complete, the transformed data will be loaded into Athena, a serverless data warehousing solution offered by AWS. To automate the loading process, Glue Crawler will be utilized to crawl the processed data and create a table schema for Athena. By using Athena, the application will be able to efficiently store and query the transformed data, allowing for real-time analysis and reporting.

Overall, this project aims to showcase the power and flexibility of Spark, real-time APIs, and cloud-based data warehousing solutions like Athena, to efficiently process, store, and analyze large amounts of financial data.

The Architecture of the pipeline is as follows:



Breakdown of the Use-Case

- Setup required:
  - Create an AWS account
  - Download the dataset
  - Download the Code
  - This is an OS-independent project, as the whole pipeline is set up on AWS Cloud

- Ways to interact with the AWS services:
  - Console- Using the Web-based UI
  - CLI- Using the AWS CLI tool in Terminal/CMD
  - CDK- Using infrastructure-as-a-code, e.g., CloudFormation
  - SDK- Programmatically, e.g., 'boto3' in python
  - For this project, we will not be using the CDK(CloudFormation); instead, the rest of the three methods will be used for execution.

- Best Practices for AWS Account
  - Enable Multi-Factor Authentication(MFA)
  - Protect the account using IAM policies
  - Avoid using the root account
  - Choose the AWS region that is closest to you
  - Create a Budget and set up a notification
  - Rotate all keys and passwords periodically
  - Follow least privilege principle

AWS S3

We will upload the data to AWS S3 bucket for preprocessing.

Create AWS S3 bucket

To create an S3 bucket in AWS, follow these steps:

1. Log in to your AWS Management Console.
2. Navigate to the S3 service by typing "S3" in the search box at the top of the page and selecting "S3" from the list of services.
3. Click on the "Create bucket" button in the top-left corner of the page.
4. Enter a unique name for your bucket. Note that bucket names must be globally unique across all of AWS, so you may need to try a few different names before finding one that is available.
5. Select the region where you want your bucket to be located. Choose a region that is geographically close to your users to minimize latency.
6. Choose the appropriate settings for your bucket. For example, you can choose whether to enable versioning or object-level logging.
7. Click on the "Create bucket" button at the bottom of the page to create your bucket.

Your S3 bucket will now be created, and you can start uploading objects to it.

Upload the raw data in the AWS S3 bucket folder

To upload data to an S3 bucket in AWS, you can follow these steps:

1. Log in to your AWS Management Console and navigate to the S3 service.
2. Click on the name of the bucket you want to upload data to.
3. Click on the "Upload" button in the top-left corner of the page.
4. In the "Upload" dialog box, click on the "Add files" button to select the file(s) you want to upload.
5. Choose the file(s) you want to upload and click on the "Open" button. You can also drag and drop the files into the dialog box.
6. In the next step, you can configure some settings for your upload, such as setting permissions, adding tags or metadata, and enabling server-side encryption.
7. Once you have configured the settings, click on the "Upload" button to start the upload process.
8. You will see a progress bar that shows the status of the upload. Once the upload is complete, you will see a message confirming that the file(s) have been successfully uploaded.

That's it! Your data has been uploaded to your S3 bucket in AWS. You can now access it through the S3 console or programmatically using the AWS SDK or API.

Amazon EMR

We will process the data stored in AWS S3 bucket using PySpark code running on Amazon EMR cluster. Before we create the Amazon EMR cluster, we need to create EC2 key pair.

To create an EC2 key pair, you can follow these steps:

1. Log in to the AWS Management Console and navigate to the EC2 dashboard.
2. Click on the "Key Pairs" link in the left-hand menu.
3. Click on the "Create Key Pair" button.
4. Enter a name for your key pair in the "Key pair name" field.
5. Choose the file format for your key pair. The recommended format is "PEM."
6. Click on the "Create Key Pair" button.
7. The key pair will be downloaded to your local computer. Save it in a secure location as you will need it to access your EC2 instances.

By following these steps, you can easily create an EC2 key pair that can be used to securely access your EC2 instances. The key pair is used to authenticate your SSH (Secure Shell) connections to your EC2 instances, and it's important to keep it secure and protected.

Create Amazon EMR cluster

To create an EMR cluster using the AWS Management Console, you can follow these steps:

1. Log in to the AWS Management Console and navigate to the EMR dashboard.
2. Click on the "Create cluster" button.
3. On the "Create Cluster" page, select the desired options for your cluster, including:
    - Cluster name: Enter a name for your EMR cluster.
    - Software configuration: Choose the software and version you want to use for your cluster, such as Hadoop or Spark.
    - Instance type: Select the type of EC2 instances to use for your cluster.
    - Number of instances: Choose the number of instances you want to use for your cluster.
    - Security and access: Configure security groups and SSH key pair to access the cluster.
    - Additional options: Choose any additional options you want to configure, such as debugging or logging.
4. Click on the "Create cluster" button at the bottom of the page.
5. Wait for the EMR cluster to be created. This may take a few minutes.
6. Once the EMR cluster is created, you can navigate to the "Cluster List" page to see a list of your clusters.
7. From the "Cluster List" page, you can select your cluster and view details about it, such as the status, instances, and applications running on it.
8. To access your EMR cluster, you can use SSH to connect to the master node using the SSH key pair you specified during cluster creation.

By following these steps, you can easily create an EMR cluster using the AWS Management Console. The console provides an easy-to-use interface for configuring and launching EMR clusters, and allows you to manage and monitor your clusters from a central location.

We have successfully created an EMR cluster, let's add a SSH port to the security group of the EMR cluster. To add an SSH port (port 22) to the security group of your EMR cluster, you can follow these steps:

1. Go to the Amazon EMR console.
2. Select your EMR cluster.
3. Click on the "Security and access" tab.
4. Under "Security groups", click on the link to the security group assigned to your EMR cluster.
5. In the security group details page, click on the "Edit inbound rules" button.
6. Click on the "Add rule" button and select "SSH" from the list of predefined rules.
7. Set the source to "My IP" to restrict access to your own IP address. Alternatively, you can set the source to "Anywhere" to allow access from any IP address.
8. Click on the "Save rules" button to apply the changes.

By following these steps, you can easily add the SSH port to the security group of your EMR cluster, allowing you to connect to the master node of the cluster using SSH. It's important to restrict access to the SSH port to only the IP addresses that require access, in order to improve the security of your EMR cluster.

To connect to an EMR cluster using SSH via the command prompt or terminal, you can follow these steps:

1. Open the command prompt or terminal application on your local machine.
2. Retrieve the public DNS name of the master node for your EMR cluster from the EMR console. To do this, select your cluster from the cluster list and navigate to the "Summary" tab.
3. Navigate to the directory where your SSH key pair is stored using the cd command.

   ```
   cd /path/to/your/ssh/keypair
   ```

   Replace /path/to/your/ssh/keypair with the local file path to your SSH key pair.

4. Use the following command to connect to the master node of your EMR cluster via SSH:

   ```
   ssh -i keypair.pem hadoop@<master-node-public-dns>
   ```

   Replace keypair.pem with the name of your SSH key pair file, and <master-node-public-dns> with the public DNS name of the master node for your EMR cluster.

5. If prompted, type "yes" to confirm the connection to the server.
6. You are now connected to the master node of your EMR cluster via SSH. From here, you can run commands and manage your EMR cluster using the command line.

By following these steps, you can easily connect to your EMR cluster using SSH via the command prompt or terminal application. It's important to keep your SSH key pair secure and protected, as it provides access to your EMR cluster.

To create a Python file using the vi editor in an EMR cluster, you can follow these steps:

1. Connect to the master node of your EMR cluster using SSH. You can use the command prompt or terminal application as described in the previous steps.
2. Navigate to the directory where you want to create the Python file. You can use the cd command to navigate to the desired directory.
3. Use the vi command to create the Python file. For example, to create a file named main.py, you can use the following command:

```
vi main.py
```

4. Press the **`i`** key to enter the insert mode.
5. Type your Python code into the file.
6. Press the **`Esc`** key to exit the insert mode.
7. Type **`:wq`** and press Enter to save and exit the file.
    - The **`:`** puts **`vi`** into command mode.
    - The **`w`** command writes the file to disk.
    - The **`q`** command quits vi.

Now you have successfully created a Python file using vi in your EMR cluster. You can run the Python file using the spark-submit command to execute it on your EMR cluster.

Use the following command to run the Python file in EMR cluster:

```
spark-submit main.py
```

Please watch these videos to understand the data manipulation part in detail:

1. Data Manipulation – Part 1
2. Data Manipulation – Part 2

Amazon Athena

We will create tables on top of the processed data stored in the AWS S3 bucket by providing the relevant schema in Athena.

To create tables in Amazon Athena on top of data stored in AWS S3, follow these steps:

1. Open the Athena console in the AWS Management Console.
2. Click on the "Query Editor" tab in the left navigation pane.
3. In the query editor, click on the "Database" dropdown in the top-left corner and select the database where you want to create the table.
4. Click on the "Create Table" button in the top-right corner of the query editor.
5. In the "Create Table" wizard, select the "From S3 bucket data" option.
6. In the "From S3 bucket data" form, enter the following information:
    - S3 location: The location of the S3 bucket and folder where your data is stored.
    - File format: The file format of your data (e.g. CSV, JSON, Parquet).
    - Table name: The name of the table you want to create.
    - Database name: The database where you want to create the table.
    - IAM role: The IAM role that has access to your S3 bucket and folder.
    - Additional options: Additional options to specify, such as custom delimiter, compression, or schema.
7. Click on the "Preview table" button to preview the data in your S3 file.
8. Once you have confirmed that the preview looks correct, click on the "Create table" button to create the table.
9. After the table is created, you can query the data using SQL queries in the query editor.

When creating a table in Athena on top of data stored in S3, make sure to specify the correct data format, delimiter, and file type used in the S3 file. Additionally, you must have the appropriate IAM permissions to access the S3 bucket and folder where your data is stored. Now you can perform various queries to perform analysis on data. Please watch this video to understand this in more detail.

Summary

- In this training, we performed spark transformations on banking transactions data.
- Fetched real-time data from an API
- Batch processing is a technique used in data processing where a large volume of data is processed at once, rather than processing it individually or in real-time. Wikipedia, being a massive online encyclopedia, generates a tremendous amount of data every day.
- The reusability of the proposed pipeline in the various industries was discussed.
- We understood each service used in this project in detail, their advantages, disadvantages and use cases.
- We created an EMR cluster and processed raw data using PySpark.
- Performed queries on Amazon Athena to analyse the data.
- Various AWS services were leveraged to implement the concepts with the most cost-optimized configurations that can even serve 100x the amount of demo data used.