Project 1 :

Project Title	Data Migration and Transformation Tool for Amazon RDS Data Warehouses
Technologies	Python/PySpark,Requests,Zipfile,boto3,pandas,sqlalchemy, Amazon S3,Amazon RDS

Problem Statement:

You have a URL that points to a zip file. The zip file contains multiple JSON files. The JSON files contain multiple documents with various data structures. Your goal is to download the zip file from the URL, extract the data from the JSON files, store it in Amazon S3, and load it into Amazon RDS. You want to use Python or PySpark to perform these tasks. You may use any libraries or tools that are necessary to complete the task.

Approach:

To extract the data from a zip file that is available at a URL and load it into Amazon S3 and Amazon RDS (NoSQL), you can follow these steps:

1. Use the requests library to download the zip file from the URL.
2. Use the zipfile module to extract the data from the zip file.
3. Use the boto3 library or PySpark to store the data in Amazon S3.
4. Use the pandas library and sqlalchemy or PySpark to load the data from S3 into Amazon RDS (NoSQL).

Results:

The result of following these steps should be that the data from the zip file is extracted and stored in a list of dictionaries (if you are using Python) or a DataFrame (if you are using PySpark). Each dictionary or DataFrame row will represent a document from one of the JSON files in the zip file.

The data in the list or DataFrame will then be stored in Amazon S3 as JSON files. You will be able to access these JSON files using the boto3 library or the Amazon S3 web interface.

The data from the JSON files will also be loaded into Amazon RDS (NoSQL). You will be able to access the data in RDS using SQL queries. The data will be stored in a table in RDS, and the schema of the table will be determined by the structure of the JSON documents.

I hope this helps. Let me know if you have any further questions or need more assistance.

Project 2 :

Project Title	**Designing an Automatic Data Collection and Storage System with AWS Lambda and Slack Integration for Server Availability Monitoring and Slack Notification**
Technologies	**AWS Lambda, Amazon RDS, CloudWatch, Slack API**

Problem Statement:

You are tasked with creating an AWS Lambda function that will periodically fetch data from an API and store it in an Amazon RDS instance. The function should be triggered by an Amazon CloudWatch Event that occurs every 15 seconds.

To fetch the data from the API, the function should use the requests library (or a similar library) to make a GET request to the API. The function should then use a library such as psycopg2 to connect to the Amazon RDS instance and store the data in the database.

In addition to fetching and storing the data, the function should also use Amazon CloudWatch to monitor the server and send an alert to a Slack community if the server goes down. This can be done using the Slack API.

Overall, the function should be able to run indefinitely and continue to fetch and store the data on a regular basis.

Approach:

1. Create an AWS Lambda function and configure it to be triggered by an Amazon CloudWatch Event that occurs every 15 seconds.
2. In the function's code, use the requests library to make a GET request to the API to fetch the data.

3. Use a library such as psycopg2 to connect to the Amazon RDS instance and store the data in the database.
4. Use Amazon CloudWatch to set up a monitoring alarm that will trigger when the server is unavailable.
5. Use the Slack API to send a message to your Slack community when the alarm is triggered.
6. Test the function to ensure that it is able to fetch and store the data correctly, and that the monitoring and alerting functionality is working as expected.
7. Deploy the function to run indefinitely, continuing to fetch and store the data on a regular basis.

The result of the above approach would be an AWS Lambda function that is continuously running and performing the following tasks:

· Fetching data from an API on a regular basis (every 15 seconds).

· Storing the fetched data in an Amazon RDS database.

· Monitoring the server's availability using a CloudWatch Alarm.

· Sending a notification to a Slack channel if the server becomes unavailable.

The function will continue to run and perform these tasks until it is stopped or modified. The Amazon RDS database will contain the data fetched from the API, and the CloudWatch Alarm will be triggered if the server becomes unavailable. The Slack notification will alert users that the server is unavailable, and provide details on the status of the server. The function and the database can be monitored to ensure that they are running and storing data correctly.

Results: