**Title: Automating Transaction Log Ingestion from S3 to Amazon Redshift Using AWS Lambda**

### **Step-by-Step Implementation Guide**

### **Step 1: Prerequisites**

* AWS Account with permissions to use S3, Lambda, IAM, and Redshift
* A Redshift cluster set up and accessible
* Redshift tables created:
  + upi\_transactions
  + neft\_transactions
  + imps\_transactions
* Sample JSON logs containing transaction data

### **Step 2: Create Required Redshift Tables**

CREATE TABLE upi\_transactions (  
 transaction\_id VARCHAR(50),  
 account\_id VARCHAR(50),  
 transaction\_type VARCHAR(10),  
 amount FLOAT,  
 timestamp TIMESTAMP,  
 merchant VARCHAR(50),  
 status VARCHAR(10)  
);  
  
CREATE TABLE neft\_transactions (  
 transaction\_id VARCHAR(50),  
 account\_id VARCHAR(50),  
 transaction\_type VARCHAR(10),  
 amount FLOAT,  
 timestamp TIMESTAMP,  
 merchant VARCHAR(50),  
 status VARCHAR(10)  
);  
  
CREATE TABLE imps\_transactions (  
 transaction\_id VARCHAR(50),  
 account\_id VARCHAR(50),  
 transaction\_type VARCHAR(10),  
 amount FLOAT,  
 timestamp TIMESTAMP,  
 merchant VARCHAR(50),  
 status VARCHAR(10)  
);

### **Step 3: Upload JSON Logs to S3**

* Create an S3 bucket (e.g., logsbank)
* Create a folder/prefix logs/
* Upload JSON log files inside the logs/ folder

### **Step 4: Create IAM Role for Lambda**

Attach the following AWS managed policies to the IAM role used by Lambda:

* AWSLambdaBasicExecutionRole
* AmazonS3FullAccess
* AmazonRedshiftFullAccess
* (Optional) CloudWatchEventsFullAccess if scheduled triggers are required

### **Step 5: Create the Lambda Function**

* Runtime: Python 3.12
* Role: Attach the IAM role created above

**Lambda Handler Code:**

import json  
import boto3  
  
s3\_client = boto3.client('s3')  
redshift\_client = boto3.client('redshift-data')  
  
# Configuration  
BUCKET\_NAME = 'logsbank'  
REDSHIFT\_CLUSTER\_ID = 'redshift-cluster-1'  
REDSHIFT\_DATABASE = 'dev'  
REDSHIFT\_DB\_USER = 'awsuser'  
REDSHIFT\_REGION = 'ap-south-1'  
  
def lambda\_handler(event, context):  
 print("Lambda triggered: Scanning all JSON files in bucket/logs/")  
  
 try:  
 response = s3\_client.list\_objects\_v2(Bucket=BUCKET\_NAME, Prefix='logs/')  
 if 'Contents' not in response:  
 print("No files found under logs/")  
 return {"statusCode": 200, "body": "No files to process"}  
  
 for obj in response['Contents']:  
 key = obj['Key']  
 if not key.startswith('logs/') or not key.endswith('.json'):  
 continue  
  
 print(f"Reading file: {key}")  
 try:  
 file\_obj = s3\_client.get\_object(Bucket=BUCKET\_NAME, Key=key)  
 content = file\_obj['Body'].read().decode('utf-8')  
 lines = content.strip().split('\n')  
  
 for line in lines:  
 print(f"Processing line: {line}")  
 log = json.loads(line)  
 transaction\_type = log.get("transaction\_type")  
  
 table\_map = {  
 "UPI": "upi\_transactions",  
 "NEFT": "neft\_transactions",  
 "IMPS": "imps\_transactions"  
 }  
  
 table\_name = table\_map.get(transaction\_type)  
 if not table\_name:  
 print(f"Skipping unknown transaction type: {transaction\_type}")  
 continue  
  
 txn\_id = log["transaction\_id"]  
 acct\_id = log["account\_id"]  
 txn\_type = log["transaction\_type"]  
 amount = log["amount"]  
 timestamp = log["timestamp"]  
 merchant = log["merchant"]  
 status = log["status"]  
  
 sql = f"""  
 INSERT INTO {table\_name} (  
 transaction\_id, account\_id, transaction\_type, amount, timestamp, merchant, status  
 )  
 VALUES (  
 '{txn\_id}', '{acct\_id}', '{txn\_type}', {amount}, '{timestamp}', '{merchant}', '{status}'  
 );  
 """  
 print(f"Executing SQL: {sql.strip()}")  
  
 result = redshift\_client.execute\_statement(  
 ClusterIdentifier=REDSHIFT\_CLUSTER\_ID,  
 Database=REDSHIFT\_DATABASE,  
 DbUser=REDSHIFT\_DB\_USER,  
 Sql=sql  
 )  
  
 print(f"Executed. Query ID: {result['Id']}")  
  
 # Archive processed file  
 archive\_key = key.replace('logs/', 'archive/logs/')  
 s3\_client.copy\_object(Bucket=BUCKET\_NAME, CopySource={'Bucket': BUCKET\_NAME, 'Key': key}, Key=archive\_key)  
 s3\_client.delete\_object(Bucket=BUCKET\_NAME, Key=key)  
 print(f"Archived file: {key} to {archive\_key}")  
  
 except Exception as e:  
 print(f"Error processing file {key}: {str(e)}")  
  
 return {  
 'statusCode': 200,  
 'body': 'Processed all JSON log files in logs/ folder (duplicates allowed)'  
 }  
  
 except Exception as e:  
 print(f"Fatal Error: {str(e)}")  
 return {  
 'statusCode': 500,  
 'body': 'Lambda failed to process files'  
 }

### **Step 6: Configure S3 Event Notification**

* Go to the logsbank bucket in the S3 console
* Add an **Event Notification**:
  + Prefix: logs/
  + Suffix: .json
  + Event type: **PUT**
  + Destination: your Lambda function

This makes the process fully **automated**.

### **Conclusion**

This setup automatically ingests new transaction logs uploaded to S3 into Amazon Redshift tables using Lambda, with support for UPI, NEFT, and IMPS logs. Processed files are archived to avoid duplication and ensure auditability.