

Lambda Code

import json

import boto3

import pandas as pd

import pymysql

import io

import os

from datetime import datetime

def fmt\_inr(x):

return "₹{:,.2f}".format(x)

s3 = boto3.client('s3')

sns\_client = boto3.client('sns')

def lambda\_handler(event, context):

# 1) GET S3 object that triggered

record = event['Records'][0]

bucket = record['s3']['bucket']['name']

key = record['s3']['object']['key']

# get file from s3

obj = s3.get\_object(Bucket=bucket, Key=key)

df\_raw = pd.read\_csv(io.BytesIO(obj['Body'].read()))

# 2) CLEAN DATA

df\_raw['total\_value'] = df\_raw['quantity'] \* df\_raw['price']

df\_clean = df\_raw.copy()

# \*\*\* FIX NaN \*\*\*

df\_clean = df\_clean.fillna(0)

df\_clean = df\_clean.where(pd.notnull(df\_clean), None)

# write cleaned to processed folder

cleaned\_key = "processed/cleaned\_orders\_detailed.csv"

csv\_buffer = io.StringIO()

df\_clean.to\_csv(csv\_buffer, index=False)

s3.put\_object(

Bucket=bucket,

Key=cleaned\_key,

Body=csv\_buffer.getvalue()

)

# 3) INSERT INTO RDS

conn = pymysql.connect(

host=os.environ['DB\_HOST'],

user=os.environ['DB\_USER'],

passwd=os.environ['DB\_PASSWORD'],

db=os.environ['DB\_NAME'],

connect\_timeout=10

)

cur = conn.cursor()

# create table if not exists

create\_sql = """

CREATE TABLE IF NOT EXISTS orders\_cleaned(

order\_id int,

customer\_name varchar(100),

city varchar(50),

product varchar(100),

quantity int,

price float,

channel varchar(50),

payment\_mode varchar(50),

discount\_code varchar(20),

order\_date date,

total\_value float

);

"""

cur.execute(create\_sql)

conn.commit()

insert\_sql = """INSERT INTO orders\_cleaned VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""

rows = [tuple(x) for x in df\_clean.to\_numpy()]

cur.executemany(insert\_sql, rows)

conn.commit()

# 4) ADVANCED ANALYTICS FOR EXECUTIVE EMAIL

total\_revenue = df\_clean['total\_value'].sum()

city\_sales = df\_clean.groupby('city')['total\_value'].sum().sort\_values(ascending=False)

top3 = city\_sales.head(3)

category\_sales = df\_clean.groupby('product')['total\_value'].sum().sort\_values(ascending=False)

top\_category = category\_sales.head(1).index[0]

avg\_order\_value = df\_clean['total\_value'].mean()

fmt\_total = fmt\_inr(total\_revenue)

fmt\_avg = fmt\_inr(avg\_order\_value)

today = datetime.now().strftime("%Y%m%d")

sns\_message = f"""

DAILY BUSINESS PERFORMANCE SUMMARY

Region: India

Date: {today}

Total Revenue Today: {fmt\_total}

Top 3 Cities by Revenue:

{top3.to\_string()}

Top Performing Product Category:

{top\_category}

Average Order Value (AOV):

{fmt\_avg}

Action Recommendation:

Prioritize stock allocation towards the top cities and the {top\_category} category.

This is a system generated executive summary.

"""

sns\_client.publish(

TopicArn=os.environ['SNS\_TOPIC\_ARN'],

Message=sns\_message.strip()

)

# 5) WRITE SUMMARY JSON TO S3

summary\_obj = {

"date": today,

"total\_revenue": total\_revenue,

"top\_3\_cities": top3.to\_dict(),

"top\_category": top\_category,

"avg\_order\_value": avg\_order\_value

}

summary\_key = f"analytics/summary\_{today}.json"

s3.put\_object(

Bucket=bucket,

Key=summary\_key,

Body=json.dumps(summary\_obj, indent=2)

)

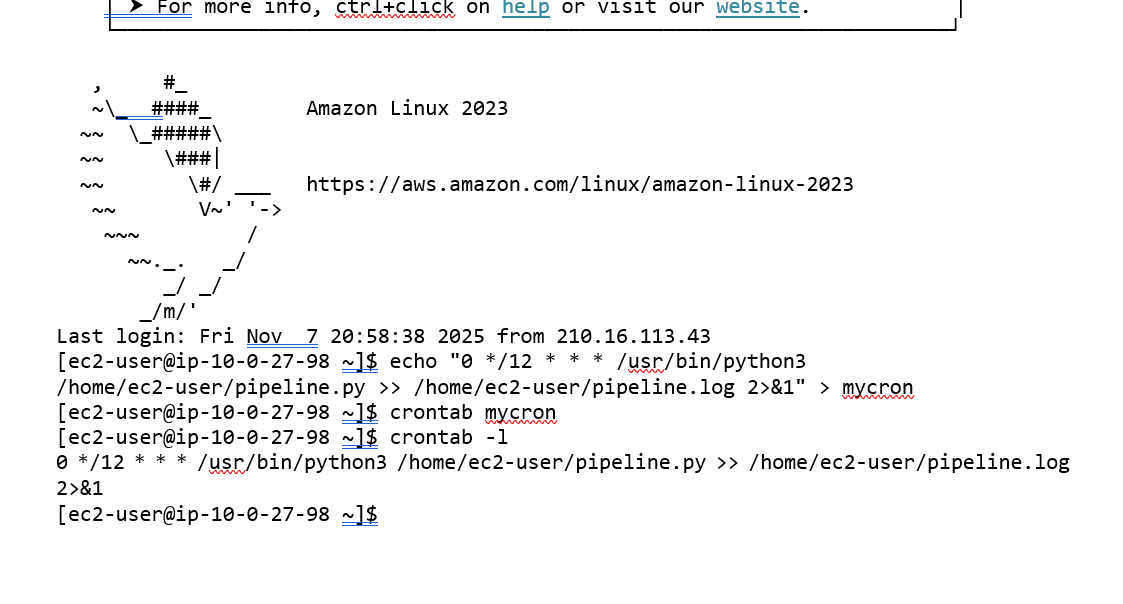
return {

"statusCode": 200,

"body": "Pipeline success"

}

Cron code :



Json file :

