Perfect, Ashok! Here's the **real-time TCS-style project** titled **"Insurance Claims Intelligence Platform"**, presented as a full **question paper style challenge** with clearly **labeled sections**, **AWS setup instructions**, and **task-by-task questions**, exactly like **Wings1 T15 format**.

**📝 Full Model Questions: Insurance Claims Intelligence Platform**

**🌐 Challenge Overview:**

This challenge simulates a real-world insurance analytics and ML problem. It consists of two independent parts:

1. **Analytics Part (AL)** – Fraud analysis using PySpark on EMR
2. **Machine Learning Part (ML)** – Claim approval prediction using SageMaker

You will work with CSV datasets stored in a dedicated S3 bucket:  
s3://insurance-data-<your-random-id>/

**🧭 Pipeline Overview:**

* **Analytics Pipeline (45 mins)**  
  S3 → EMR (Spark) → Redshift & S3
* **ML Pipeline (45 mins)**  
  S3 → SageMaker → S3

**🌎 Region:**

Use only **US East (N. Virginia)** (us-east-1)

**🧱 Configured for Challenge**

**✅ S3 Bucket:**

* Name: insurance-data-<unique\_id>
* Folders:
  + claims/claims\_data.csv
  + claims\_ml\_ready/claims\_ml\_ready.csv

**✅ IAM Role:**

* Role Name: TCS-ins-analytics-role
* Permissions: Full access to S3, EMR, Redshift, SageMaker

**✅ Redshift Cluster:**

* Cluster Name: ins-redshift-cluster
* DB Name: claimsdb
* Username: retailuser
* Password: RetailUser1

**✅ EMR Cluster:**

* Name: insurance-analytics-emr
* EMR Version: 7.1.0
* Node Type: m4.large
* Single node setup
* Applications: Spark, Hadoop, Hive
* Key Pair: retail\_emr.pem (download and store locally)

**🔷 Analytics Part – Fraud Detection (EMR + Redshift + S3)**

**📁 Dataset:**

claims/claims\_data.csv  
Contains columns like:

claim\_id, customer\_id, claim\_amount, incident\_type, claim\_status,

payout, region, incident\_severity, police\_report\_available, fraud\_reported

**✅ Task 1: read\_claim\_data()**

📌 **Objective:**  
Load the claims CSV file into a PySpark DataFrame.

📋 **Instructions:**

* Use a custom schema
* Read from: s3://insurance-data-<id>/claims/claims\_data.csv
* Use header=True, schema=customSchema
* Return the DataFrame

**✅ Task 2: clean\_claims()**

📌 **Objective:**  
Clean the loaded data for further processing.

📋 **Instructions:**

* Drop rows with nulls in claim\_amount, incident\_type, customer\_id
* Remove duplicate rows
* Filter out rows with invalid regions (e.g., region is null or unknown)

**✅ Task 3: compute\_avg\_payout\_by\_incident()**

📌 **Objective:**  
Analyze average payouts across incident types.

📋 **Instructions:**

* Group by incident\_type
* Compute average payout
* Return DataFrame sorted by descending avg\_payout

**✅ Task 4: flag\_suspicious\_claims()**

📌 **Objective:**  
Tag potentially fraudulent claims.

📋 **Instructions:**

* Create new column fraud\_score
* Conditions for fraud score = 1:
  + claim\_amount > 50000
  + payout > claim\_amount
  + police\_report\_available == "NO"
* Else, fraud\_score = 0

**✅ Task 5: save\_suspicious\_to\_s3()**

📌 **Objective:**  
Store flagged suspicious claims in S3.

📋 **Instructions:**

* Use coalesce(1) to write a single CSV
* Add header=True
* Output path: s3://insurance-data-<id>/output/suspicious\_claims.csv

**✅ Task 6: load\_summary\_to\_redshift()**

📌 **Objective:**  
Load suspicious claim summary into Redshift.

📋 **Instructions:**

* Use JDBC with:
  + DB: claimsdb
  + Username: retailuser
  + Password: RetailUser1
* Table: fraud\_summary
* Columns: claim\_id, fraud\_score, incident\_type, claim\_amount
* Mode: overwrite

**🔶 Machine Learning Part – Claim Approval Prediction (SageMaker)**

**📁 Dataset:**

s3://insurance-data-<id>/claims\_ml\_ready/claims\_ml\_ready.csv

**✅ Task 1: Launch Notebook Instance**

📌 **Objective:**  
Launch SageMaker notebook for ML tasks.

📋 **Instructions:**

* Notebook name: insurance-ml-notebook
* Instance type: ml.t3.medium
* Role: TCS-ins-analytics-role
* Wait for InService, then launch JupyterLab

**✅ Task 2: Upload Dataset**

📌 **Objective:**  
Upload the ML dataset into SageMaker.

📋 **Instructions:**

* Use "Upload Files" in JupyterLab
* Load claims\_ml\_ready.csv into working directory

**✅ Task 3: Load ML Dataset**

📌 **Objective:**  
Read CSV into a pandas DataFrame.

📋 **Instructions:**

* Use boto3, pandas
* Load from S3 path
* Use .head() and .info() to inspect

df=pd.read\_csv(S3 path)

print(df)

**✅ Task 4: Preprocess Features**

📌 **Objective:**  
Encode categorical variables.

📋 **Instructions:**

* Columns to encode: incident\_type, region, severity
* Use pd.get\_dummies()
* Ensure final data is all numeric

df=df.get\_dummies(df,columns[‘incident\_type’,’ region’,’ severity’],dtype=int)

Print(df)

**✅ Task 5: Handle Class Imbalance**

📌 **Objective:**  
Balance the claim\_approved target column.

📋 **Instructions:**

* Use resample() from sklearn.utils
* Upsample minority class
* Combine with majority class
* Validate balanced dataset using .value\_counts()

Df\_majority=df.drop[df[‘not\_fully\_null]==1]

Df\_minority=df[df[‘not\_fully\_null]==0]

Df\_sample\_minority=resample(Df\_minority,

Replace=True,

N\_sample=len(Df\_majority),random\_state=42)

Df=df.concat([Df\_sample\_minority, Df\_minority])

Print(df)

**✅ Task 6: Train Classifier**

📌 **Objective:**  
Train Random Forest model.

📋 **Instructions:**

* Features: All columns except claim\_id, claim\_approved
* Target: claim\_approved
* Split: 70% train, 30% test
* Use RandomForestClassifier(random\_state=42)

X=df.drop([‘claim\_id’,’ claim\_approved’],axis=1)

Y=df[df[‘claim\_approved’]]

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_case=0.3,random\_state=42)

* Df= RandomForestClassifier(random\_state=42)

Rf=Df.fit(X\_train,y\_train)

Pred=rf.predict(X\_test)

**✅ Task 7: Evaluate Model**

📌 **Objective:**  
Assess prediction performance.

📋 **Instructions:**

* Use classification\_report, confusion\_matrix, accuracy\_score
* Report model precision and recall

**✅ Task 8: Save Model to S3**

📌 **Objective:**  
Save the trained model in S3 for future inference.

📋 **Instructions:**

* Serialize with joblib.dump()
* Save as a temp file using tempfile.TemporaryFile()
* Use boto3.client().put\_object() to upload as:  
  s3://insurance-data-<id>/models/claim\_model.pkl

**✅ Final Summary Table**

| **Section** | **Task** | **Task Name** | **Output** |
| --- | --- | --- | --- |
| AL | 1 | Read Claim Data | Spark DF |
| AL | 2 | Clean Claim Data | Cleaned DF |
| AL | 3 | Avg Payout By Incident Type | Summary DF |
| AL | 4 | Flag Suspicious Claims | Fraud Scored DF |
| AL | 5 | Save Suspicious Claims to S3 | CSV in S3 |
| AL | 6 | Load Fraud Summary to Redshift | Redshift Table |
| ML | 1 | Launch SageMaker Instance | Jupyter Notebook |
| ML | 2 | Upload Dataset | Local File |
| ML | 3 | Load & Inspect Data | Pandas DF |
| ML | 4 | Encode Features | Encoded DF |
| ML | 5 | Balance Target Column | Balanced DF |
| ML | 6 | Train Random Forest Classifier | Model |
| ML | 7 | Evaluate Predictions | Metrics Report |
| ML | 8 | Save Model to S3 | model.pkl in S3 |

✅ Let me know if you now want:

* ✍ Python code templates for these tasks
* 📊 Diagram of full architecture
* 📂 Sample data (CSV format) for practice

Shall I proceed with the code templates next?

# ------------------------------------------------------------

# 📦 IMPORTS

# ------------------------------------------------------------

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

from sklearn.utils import resample

import joblib

# ------------------------------------------------------------

# 🟦 PART 1: ANALYTICS – FRAUD DETECTION

# ------------------------------------------------------------

# 🧾 Simulated Claim Data (like from S3)

claims\_data = pd.DataFrame({

'claim\_id': ['CL001', 'CL002', 'CL003', 'CL004', 'CL005', 'CL006'],

'customer\_id': ['C01', 'C02', 'C03', 'C01', 'C05', None],

'claim\_amount': [20000, 60000, 30000, 52000, np.nan, 35000],

'incident\_type': ['Collision', 'Theft', 'Fire', 'Collision', 'Fire', 'Collision'],

'claim\_status': ['Approved', 'Denied', 'Approved', 'Approved', 'Denied', 'Approved'],

'payout': [18000, 65000, 29000, 50000, 0, 36000],

'region': ['North', 'South', 'Unknown', None, 'East', 'South'],

'incident\_severity': ['Minor', 'Major', 'Severe', 'Major', 'Minor', 'Major'],

'police\_report\_available': ['YES', 'NO', 'YES', 'NO', 'YES', 'NO'],

'fraud\_reported': ['N', 'Y', 'N', 'Y', 'N', 'Y']

})

# ✅ Step 1: Clean claim data

claims\_cleaned = claims\_data.dropna(subset=['claim\_amount', 'incident\_type', 'customer\_id'])

claims\_cleaned = claims\_cleaned.drop\_duplicates()

claims\_cleaned = claims\_cleaned[~claims\_cleaned['region'].isin([None, 'Unknown'])]

# ✅ Step 2: Compute average payout per incident type

avg\_payout\_df = claims\_cleaned.groupby('incident\_type')['payout'].mean().reset\_index()

avg\_payout\_df.columns = ['incident\_type', 'avg\_payout']

avg\_payout\_df = avg\_payout\_df.sort\_values(by='avg\_payout', ascending=False)

# ✅ Step 3: Flag suspicious claims

def fraud\_score\_fn(row):

return int(row['claim\_amount'] > 50000 and row['payout'] > row['claim\_amount'] and row['police\_report\_available'] == 'NO')

claims\_cleaned['fraud\_score'] = claims\_cleaned.apply(fraud\_score\_fn, axis=1)

# ✅ Step 4: Save suspicious claims (fraud\_score = 1)

suspicious\_claims = claims\_cleaned[claims\_cleaned['fraud\_score'] == 1]

suspicious\_claims[['claim\_id', 'fraud\_score', 'incident\_type', 'claim\_amount']].to\_csv("suspicious\_claims.csv", index=False)

# ✅ Step 5: Redshift simulation print

print("\n📌 Suspicious Claims Summary (for Redshift):")

print(suspicious\_claims[['claim\_id', 'fraud\_score', 'incident\_type', 'claim\_amount']])

print("\n📊 Average Payout by Incident Type:")

print(avg\_payout\_df)

# ------------------------------------------------------------

# 🔶 PART 2: MACHINE LEARNING – CLAIM APPROVAL PREDICTION

# ------------------------------------------------------------

# 🧾 Simulated ML Dataset

ml\_data = pd.DataFrame({

'claim\_id': ['CL001', 'CL002', 'CL003', 'CL004', 'CL005', 'CL006', 'CL007'],

'incident\_type': ['Collision', 'Theft', 'Fire', 'Collision', 'Fire', 'Collision', 'Theft'],

'region': ['North', 'South', 'East', 'West', 'North', 'East', 'West'],

'severity': ['Minor', 'Major', 'Severe', 'Minor', 'Major', 'Severe', 'Major'],

'claim\_amount': [10000, 60000, 45000, 30000, 25000, 50000, 70000],

'claim\_approved': [1, 0, 1, 1, 1, 0, 0]

})

# ✅ Step 1: One-hot encoding

ml\_encoded = pd.get\_dummies(ml\_data, columns=['incident\_type', 'region', 'severity'])

X = ml\_encoded.drop(columns=['claim\_id', 'claim\_approved'])

y = ml\_encoded['claim\_approved']

# ✅ Step 2: Handle class imbalance

df\_full = pd.concat([X, y], axis=1)

majority = df\_full[df\_full.claim\_approved == 1]

minority = df\_full[df\_full.claim\_approved == 0]

minority\_upsampled = resample(minority, replace=True, n\_samples=len(majority), random\_state=42)

df\_balanced = pd.concat([majority, minority\_upsampled])

X\_bal = df\_balanced.drop('claim\_approved', axis=1)

y\_bal = df\_balanced['claim\_approved']

# ✅ Step 3: Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_bal, y\_bal, test\_size=0.3, random\_state=42)

# ✅ Step 4: Train RandomForestClassifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# ✅ Step 5: Evaluate model

y\_pred = model.predict(X\_test)

print("\n📈 Classification Report:")

print(classification\_report(y\_test, y\_pred))

print("✅ Accuracy:", accuracy\_score(y\_test, y\_pred))

print("📌 Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# ✅ Step 6: Save model to file (simulate S3 upload)

joblib.dump(model, "claim\_model.pkl")

print("\n💾 Model saved as claim\_model.pkl (simulated S3 upload)")

practice AL code

print("Task1")

df=spark.read.csv(s3://insurance-data-<id>/claims/claims\_data.csv,header=True, schema=customSchema)

print(df)

print("Task2")

df=df.drop\_duplicates()

print(df)

df=df.dropna(subset=['claim\_amount','incident\_type','customer\_id'])

print(df)

df=df.filter[df['region'].notnull()&df['region'].str.lower()!='unknown']

print(df)

print("Task3")

df=df.groupBY("incident\_type",agg

(sum("payout").alias("Total\_payout")

,orderBy("Total\_payout",ascending=False)))

print(df)

print("Task4")

df=df.withcolumns("fraud\_score",

when(

    col("claim\_amount") > 50000 |

o   col("payout")>col("claim\_amount") |

(upper(col("police\_report\_available")) == "NO"),

            1

).otherwise(0)

))

print(df)

print("Task5")

df==data.colasce(1).write.mode("overwrite").csv(s3://insurance-data-<id>/output/suspicious\_claims.csv,header=True)

print(df)

print("TAsk6")

df=data.write.mode("overwrite").csv(url,dbname,password).save()