**Project instructions**

**Step 1.1 Create S3 bucket and directory, upload csv files to S3**

**Step 1.2 Run Glue crawler to create database and tables according to S3 csv files**

**Step 1.3 Athena check tables and run SQL to create table products**

**Step 2 Athena run SQL to create feature tables**

**Step 3 Create Lambda Functions in Python to run above SQL queries separately.**

**Step 4 Step Function create state machine to orchestrate above Lambda functions**

**Step 5 Use Glue Job to join everything together then convert to spark dataframe**

**Step 6 Create a Lambda function to invoke Glue Job then add it to Step Function**

**(1)AWS Glue & Athena**

**1.1 Create S3 bucket and directory, upload csv files to S3**

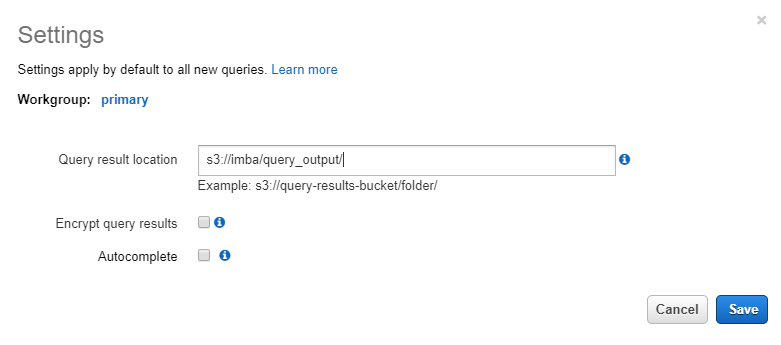
1. Download imba.zip from share drive, unzip all the files.
2. Create an s3 bucket with name imba-<put your name here> then create the following folders:   
   data/aisles   
   data/departments   
   data/orders   
   data/products   
   data/order\_products
3. Upload files to the corresponding directory (note: gzip both order\_products\_\_prior.csv and order\_products\_\_train.csv in gitbash, and upload all of them to data/order\_products).

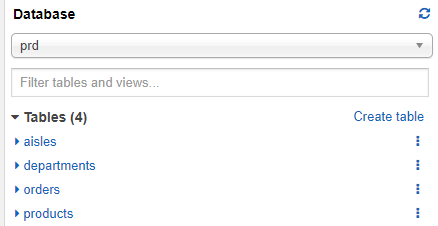
**1.2 Run Glue crawler to create database and tables base on S3 csv files**

1. Go to AWS Glue service, click on Crawlers and Add crawler.
2. Give the crawler a name, e.g. imba, click next.
3. Choose Data stores as Crawler source type, click next.
4. Choose s3 as data store and specify the Include path as: s3://ibma-<put your name here>/data, click next.
5. Do not add another data store and click next.
6. Select Create an IAM role and type a name (any name will do, or just use imba) in the text box, click next.
7. Specify Frequency as Run on demand, click next.
8. Create a new database by clicking Add database, name it as prod, click next.
9. Review the options and click Finish.
10. Select the crawler you just created and click Run crawler button.

**1.3 Athena check database and run SQL to create table products**

1. Go to AWS Athena service, click Settings button located on the top right of the page, type in the Query result location as s3://<your s3 bucket>/query\_output/:



1. Go to AWS Athena service, select prod data on the left pane, you should see four tables are created.
2. Check the fields for each table by expanding the table, as shown below: (note: below Database name should be prod not prd)



1. You might notice products table is malformed, this is because crawler has problems to recognize csv files with double quote in content. Instead, manually create the table by running below command in query pane(remember to update the s3 location to your file location ‘s3://imba-<put your name here>/data/products'):

* First drop the table by running query: DROP TABLE IF EXISTS products;
* Now re-create the table by running below query:

CREATE EXTERNAL TABLE `products` (

`product\_id` string COMMENT 'from deserializer',

`product\_name` string COMMENT 'from deserializer',

`aisle\_id` string COMMENT 'from deserializer’,

`department\_id` string COMMENT 'from deserializer')

ROW FORMAT SERDE

‘org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

'escapeChar'='\\',

‘quoteChar'='\"',

'separatorChar'=',')

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://imba-yibei/data/products'

TBLPROPERTIES (

'skip.header.line.count'='1')

1. Do some exploration on these tables and make sure you understand the context of them.
2. Design a query which join orders table and order\_products table together, and filter on eval\_set = ‘prior’ ( see below (2)1 create table order\_products\_prior )

**(2) Athena queries**

1. Create a table called **order\_products\_prior** by using the last SQL query you created from the previous assignment. It should be similar to below (note you need to replace the s3 bucket name “imba-yibei” to yours own bucket name):

CREATE TABLE order\_products\_prior WITH (  
 external\_location = ‘s3://imba-yibei/features/order\_products\_prior/'   
 format = ‘parquet'   
 )   
 as (  
 SELECT   
 a.\*, b.product\_id,   
 b.add\_to\_cart\_order,   
 b.reordered FROM orders a   
 JOIN order\_products b  
 ON a.order\_id = b.order\_id   
 WHERE a.eval\_set = 'prior'   
 )

2. Create a table called **user\_features\_1** as shown below, replace the <sql here> to the desired SQL query. Based on table **order\_products\_prior**, for each user, calculate the max order\_number, the sum of days\_since\_prior\_order and the average of days\_since\_prior\_order. (note you need to replace the s3 bucket name “imba” to yours own bucket name)

CREATE TABLE user\_features\_1 WITH (  
 external\_location = ‘s3://imba-yibei/features/user\_features\_1/',   
 format = 'parquet'  
 )   
 as (   
 SELECT   
 user\_id,  
 Max(order\_number) AS user\_orders,   
 Sum(days\_since\_prior\_order) AS user\_period,   
 Avg(days\_since\_prior\_order) AS user\_mean\_days\_since\_prior   
 FROM order\_products\_prior   
 GROUP BY user\_id  
 )

3. Create a table called **user\_features\_2**, similar to above, based on table order\_products\_prior, for each user calculate the total number of products, total number of distinct products, and user reorder ratio(number of reordered = 1 divided by number of order\_number > 1, hint: Cast(Sum(CASE WHEN order\_number > 1 THEN 1 ELSE 0 END) AS DOUBLE)   
(note you need to replace the s3 bucket name “imba” to yours own bucket name)

CREATE TABLE user\_features\_2 WITH (  
 external\_location = ‘s3://imba-yibei/features/user\_features\_2/',   
 format = 'parquet'  
 )   
 as (   
 SELECT   
 user\_id,   
 Count(\*) AS user\_total\_products,   
 Count(DISTINCT product\_id) AS user\_distinct\_products ,   
 Sum (   
 CASE WHEN reordered = 1 THEN 1 ELSE 0 END) /   
 Cast( Sum (CASE WHEN order\_number > 1 THEN 1 ELSE 0 END)   
 AS DOUBLE) AS user\_reorder\_ratio   
 FROM order\_products\_prior   
 GROUP BY user\_id   
 )

4. Create a table called **up\_features**, based on table order\_products\_prior, for each user and product(hint: group by user\_id and product\_id), calculate the total number of orders, minimum order\_number, maximum order\_number and average add\_to\_cart\_order.  
(note you need to replace the s3 bucket name “imba” to yours own bucket name)

CREATE TABLE up\_features WITH (  
 external\_location = ‘s3://imba-yibei/features/up\_features/',   
 format = 'parquet'  
 )   
 as (   
 SELECT

user\_id,   
 product\_id,   
 Count(\*) AS up\_orders,  
 Min(order\_number) AS up\_first\_order,   
 Max(order\_number) AS up\_last\_order,   
 Avg(add\_to\_cart\_order) AS up\_average\_cart\_position   
 FROM order\_products\_prior   
 GROUP BY   
 user\_id,   
 product\_id  
 )

5. Create a table called **prd\_features**, based on table order\_products\_prior, first write a sql query to calculate the sequence of product purchase for each user(hint: you should use window

function rank() over (partition by user\_id, product\_id order by user\_id, order\_number)) and name it product\_seq\_time. Then on top of this query, for each product, calculate the count, sum of reordered, sum of product\_seq\_time = 1 and sum of product\_seq\_time = 2.  
(note you need to replace the s3 bucket name “imba” to yours own bucket name)

CREATE TABLE prd\_features WITH (

external\_location = 's3://imba-yibei/features/prd\_features/',

format = 'parquet'

) AS (

SELECT

product\_id,

Count(\*) AS prod\_orders,

Sum(reordered) AS prod\_reorders,

Sum(CASE WHEN product\_seq\_time = 1 THEN 1 ELSE 0 END) AS prod\_first\_orders,

Sum(CASE WHEN product\_seq\_time = 2 THEN 1 ELSE 0 END) AS prod\_second\_orders

FROM

(

SELECT \*,

Rank() OVER (

partition BY user\_id,

product\_id

ORDER BY

user\_id,

order\_number

) AS product\_seq\_time

FROM

order\_products\_prior

)

GROUP BY

product\_id

)

**(3)AWS Lambda**

Note: Please replace **<your bucket name>** below to your own bucket name!

1. Create a lambda function with the code below (name it as **remove\_feature\_files** and choose **Python 3.8** as runtime environment), make sure you assigned **AmazonS3FullAccess** and **AmazonAthenaFullAccess** to the function role and increase the function timeout to **1 minute**.

import json

import boto3

def lambda\_handler(event, context):

# TODO implement

bucket = event['bucket']

prefix = event['prefix']

s3 = boto3.resource('s3')

bucket = s3.Bucket(bucket)

for key in bucket.objects.filter(Prefix=prefix):

key.delete()

return {

'statusCode': 200

}

Execute this function by clicking the test button on the top right of the page, for the test events please put below json object:

{   
 "bucket": "**<your bucket name>**",   
 "prefix": "features/"   
}  
Note: you need to replace <your bucket name> to the name of the bucket you created

2. Create another lambda function called **exe\_query\_order\_products\_prior** with the code below, please **re-use the role** you created before and increase the function timeout to **1 minute.** Please examine the query1 and query2 parameter, this function is intended to drop the relevant table if exists and re-create them.

import json

import boto3

import time

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

def lambda\_handler(event, context):

database = event['database']

query\_output = event['query\_output']

# TODO implement

query1 = """DROP TABLE IF EXISTS order\_products\_prior"""

query2 = """

CREATE TABLE order\_products\_prior WITH (

external\_location = 's3://imba-yibei/features/order\_products\_prior/',

format = 'parquet'

)

as (

SELECT

a.\*,

b.product\_id,

b.add\_to\_cart\_order,

b.reordered FROM orders a

JOIN order\_products b

ON a.order\_id = b.order\_id

WHERE a.eval\_set = 'prior'

)

"""

response1 = athena\_client.start\_query\_execution(

QueryString=query1,

QueryExecutionContext={'Database': database },

ResultConfiguration={'OutputLocation': query\_output}

)

# sleep 10 seconds to make sure the table is successfully dropped

time.sleep(10)

response2 = athena\_client.start\_query\_execution(

QueryString=query2,

QueryExecutionContext={'Database': database },

ResultConfiguration={'OutputLocation': query\_output}

)

# get the query execution id

execution\_id = response2['QueryExecutionId']

while True:

stats = athena\_client.get\_query\_execution(QueryExecutionId=execution\_id)

status = stats['QueryExecution']['Status']['State']

if status in ['SUCCEEDED', 'FAILED', 'CANCELLED']:

break

time.sleep(0.2) # 200ms

return {

'statusCode': status

}

Execute the function and put below json object as input:

{   
 "database": "prod",   
 "query\_output": “s3://imba-yibei/query\_output/“   
}

Note: you need to replace <your bucket name> to the name of the bucket you created. You may have noticed in the previous lambda function, query1 is merely to drop the table (order\_products\_prior) if exists, query2 is to re-create the table.

3. Please create 4 more lambda functions(exe\_query\_user\_features\_1, exe\_query\_user\_features\_2, exe\_query\_up\_features and exe\_query\_prd\_features) similar to above, but change the table name and query to the ones you did in project\_part2.  
**(Note: before testing these functions, make sure there are no files in this location 's3:// <your s3 bucket>/features/<your table name> by running the remove\_feature\_files function, otherwise you cannot re-create the table)**

**lambda functions exe\_query\_user\_features\_1**

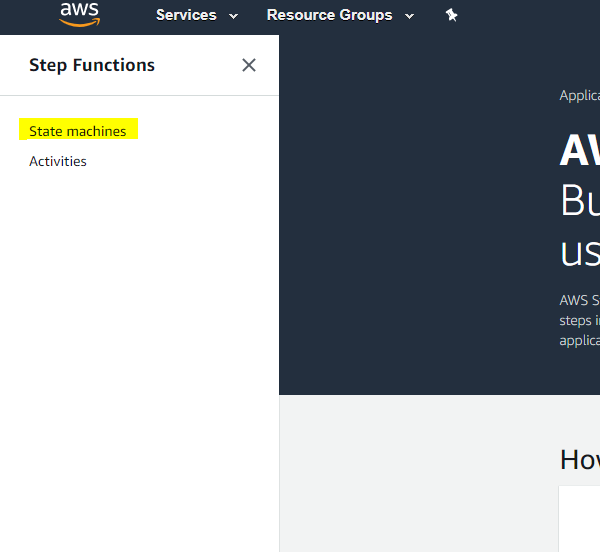
**lambda functions exe\_query\_user\_features\_2**

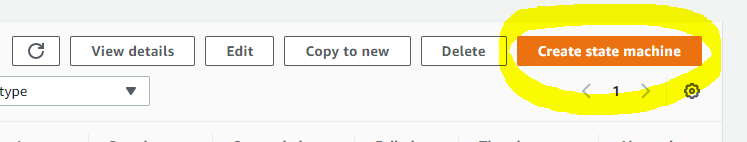
**lambda functions exe\_query\_up\_features**

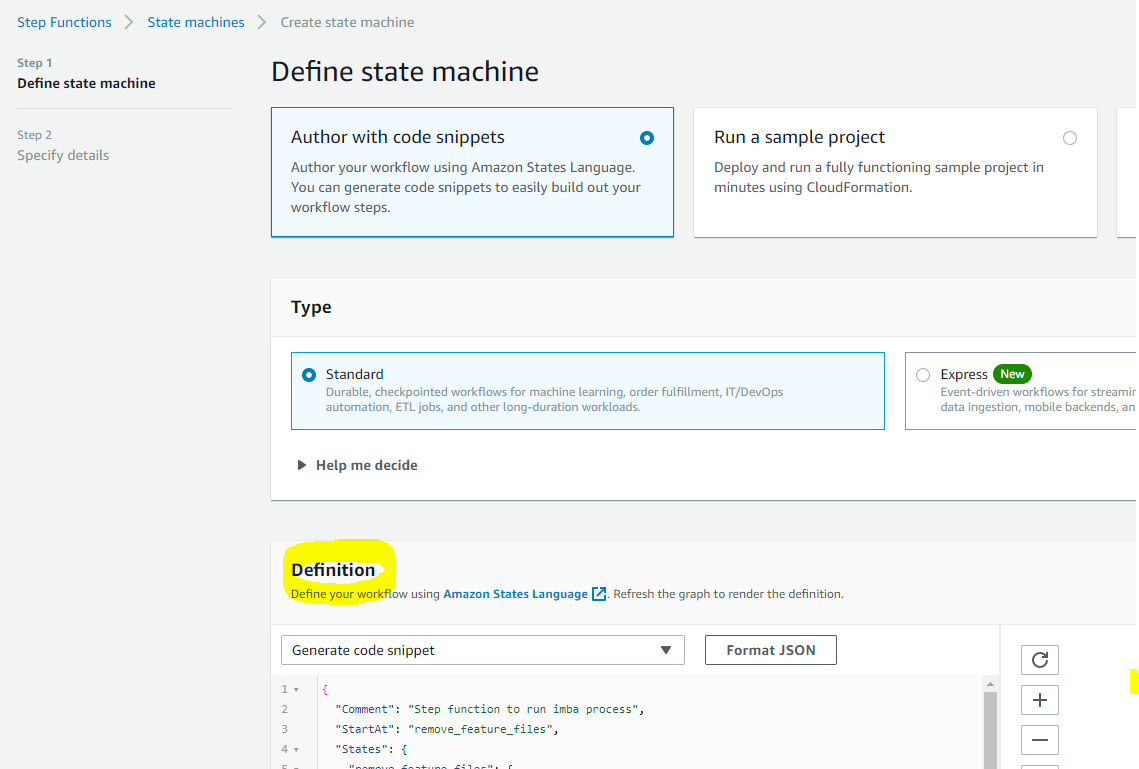
**lambda functions exe\_query\_prd\_features**

**Step 4 Step Function create state machine to orchestrate above Lambda functions**

1. Click Step function on AWS console, click State machines on the left side pane, see below:
2. Click Create state machine on the top right pane:
3. Leave the setting as default, in Definition section(marked as yellow below), copy and paste json below:







{

"Comment": "Step function to run imba process",

"StartAt": "remove\_feature\_files",

"States": {

"remove\_feature\_files": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:remove\_feature\_files:$LATEST",

"ResultPath": "$.remove\_feature\_files",

"Next": "exe\_query\_order\_products\_prior",

"TimeoutSeconds": 60

},

"exe\_query\_order\_products\_prior": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_order\_products\_prior:$LATEST",

"ResultPath": "$.exe\_query\_order\_products\_prior",

"Next": "exe\_query\_user\_features\_1",

"TimeoutSeconds": 60

},

"exe\_query\_user\_features\_1": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_user\_features\_1:$LATEST",

"ResultPath": "$.exe\_query\_user\_features\_1",

"Next": "exe\_query\_user\_features\_2",

"TimeoutSeconds": 60

},

"exe\_query\_user\_features\_2": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_user\_features\_2:$LATEST",

"ResultPath": "$.exe\_query\_user\_features\_2",

"Next": "exe\_query\_up\_features",

"TimeoutSeconds": 60

},

"exe\_query\_up\_features": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_up\_features:$LATEST",

"ResultPath": "$.exe\_query\_up\_features",

"Next": "exe\_query\_prd\_features",

"TimeoutSeconds": 60

},

"exe\_query\_prd\_features": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_prd\_features:$LATEST",

"ResultPath": "$.exe\_query\_up\_features",

"TimeoutSeconds": 60,

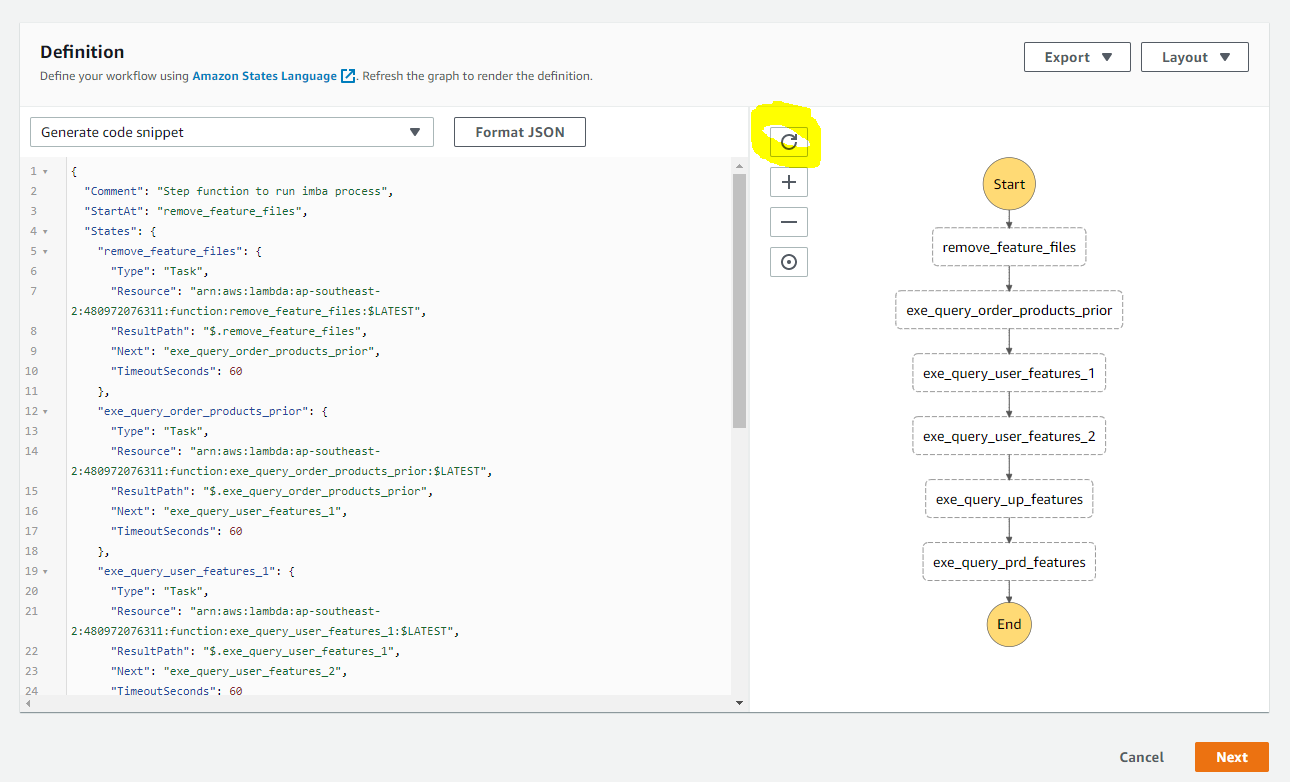
"End": true

}

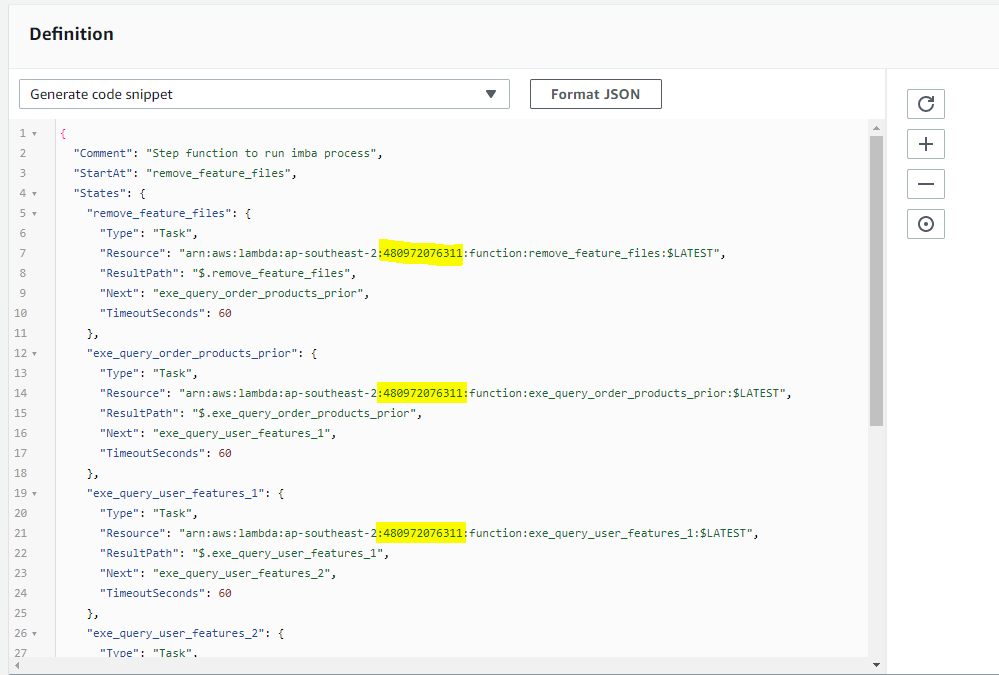
}

}

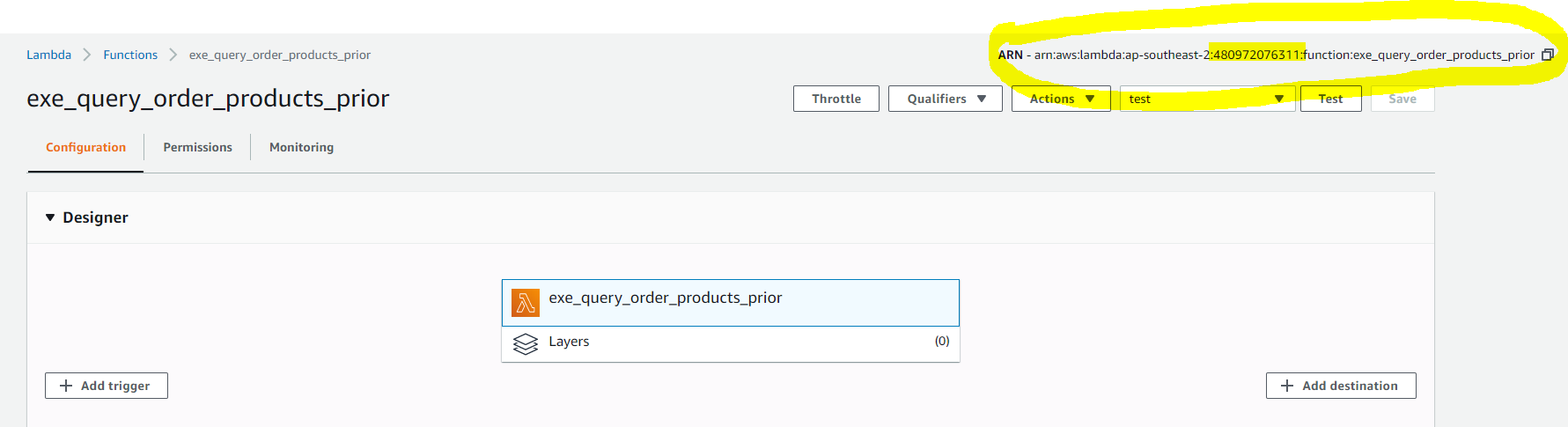
4. Click the refresh button(marked as yellow below), you should see the execution graph:



5. In the json object, you need to replace all the account\_id in the five lambda ARN to your account id, see below:

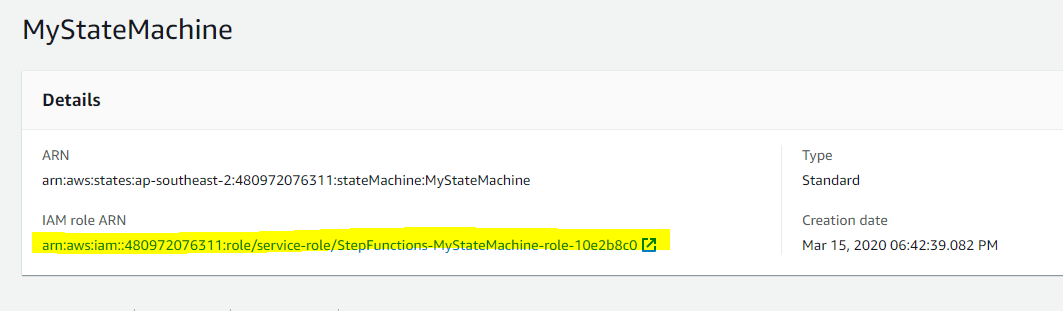


The account id or the lambda arn is located in your lambda function details, see below for one example:

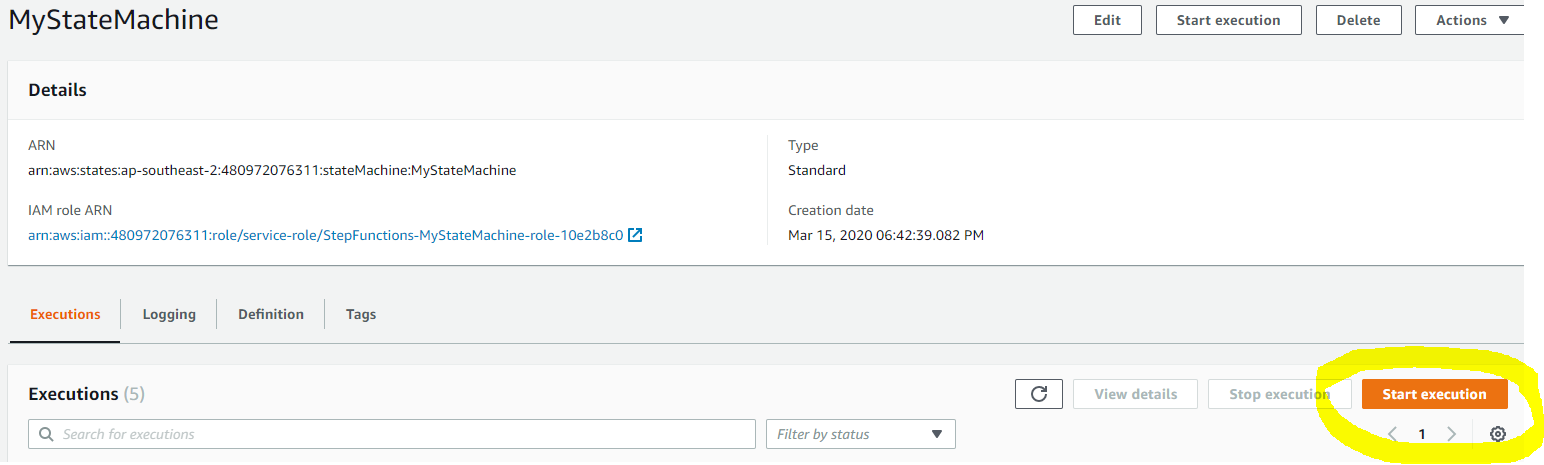


1. Click next, give your state machine a name and then click Create state machine in the bottom:
2. Give your step function the permission to invoke lambda function by clicking the IAM role ARN:





1. Attach AWSLambdaFullAcess permission to this role.
2. Go back to the state machine you just created, click Start execution:



Put below json as input and click Start execution:

{

"bucket": "**<your s3 bucket>**”,

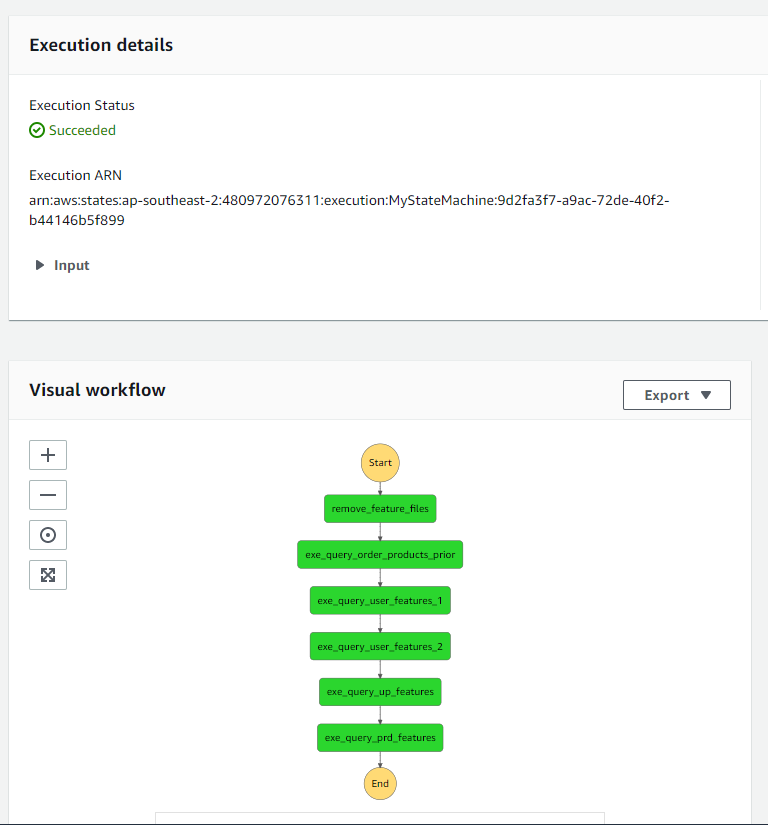
"prefix": "features/",

"database": "prd",

"query\_output": "s3://**<your s3 bucket>**/query\_results/"

}

10. You should see result similar to below if everything is configured correctly:



**Step 5 Use Glue Job to join everything together then convert to spark dataframe**

1. Open glue\_job.py, change <your s3 bucket> to your s3 bucket name, save this file and upload it to s3://<your s3 bucket>/scripts/.

import sys

from awsglue.transforms import \*

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

def main():

# create glue context first

glueContext = GlueContext(SparkContext.getOrCreate())

# creating dataframes from existing athena catelog

up\_features = glueContext.create\_dynamic\_frame.from\_catalog(database="prod", table\_name="up\_features")

prd\_features = glueContext.create\_dynamic\_frame.from\_catalog(database="prod", table\_name="prd\_features")

user\_features\_1 = glueContext.create\_dynamic\_frame.from\_catalog(database="prod", table\_name="user\_features\_1")

user\_features\_2 = glueContext.create\_dynamic\_frame.from\_catalog(database="prod", table\_name="user\_features\_2")

# join user features together

users = Join.apply(user\_features\_1.rename\_field('user\_id','user\_id1'), user\_features\_2, 'user\_id1', 'user\_id').drop\_fields(['user\_id1'])

# join everything together

df = Join.apply(Join.apply(up\_features,

users.rename\_field('user\_id','user\_id1'),

'user\_id','user\_id1').drop\_fields(['user\_id1']),

prd\_features.rename\_field('product\_id','product\_id1'),

'product\_id','product\_id1').drop\_fields(['product\_id1'])

# convert glue dynamic dataframe to spark dataframe

df\_spark = df.toDF()

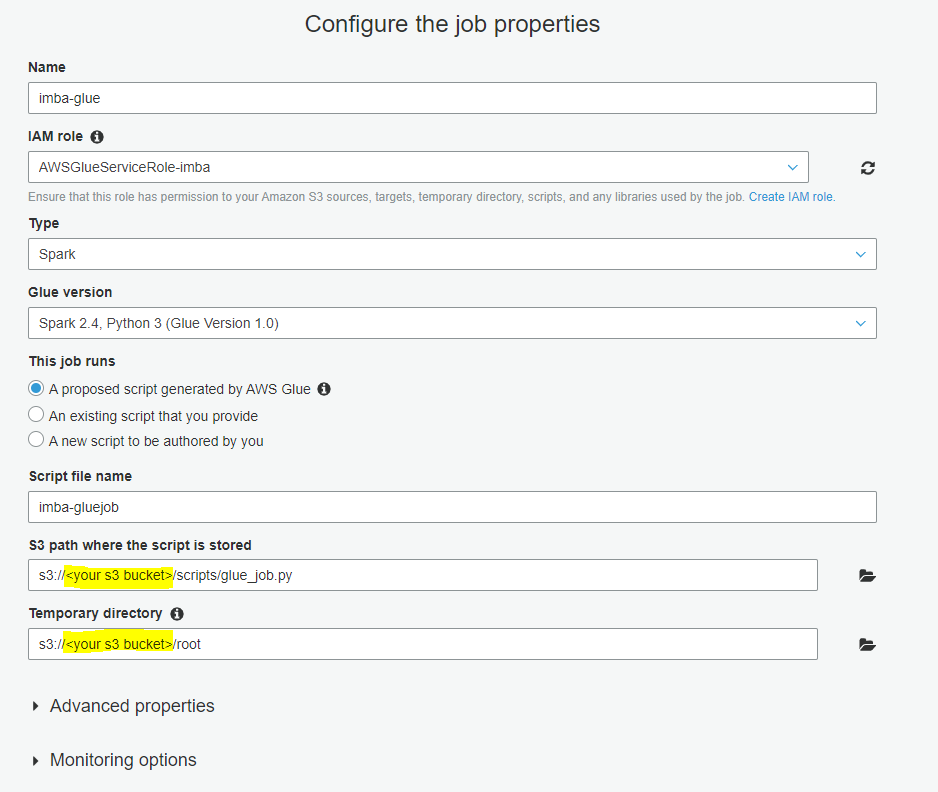
df\_spark.repartition(1).write.mode('overwrite').format('com.databricks.spark.csv').option("codec", "org.apache.hadoop.io.compress.GzipCodec").save("s3://imba-yibei/output", header = 'true')

if \_\_name\_\_ == '\_\_main\_\_':

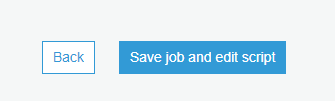
main()

1. Open Glue console in AWS, click Jobs on the left pane:
2. Click Add job and you should fill in the details similar to below, name the job to “imba- glue”, create a new IAM role or re-use an existing one (you just need to make sure AmazonS3FullAccess and AWSGlueServiceRole is attached). Make sure you select “An existing script that you provide” for “this job runs”. Specify the s3 path where your script is stored: s3://<your s3 bucket>/scripts/glue\_job.py and Temporary directory: s3://<your s3 bucket>/root. Leave everything else as default and click next.

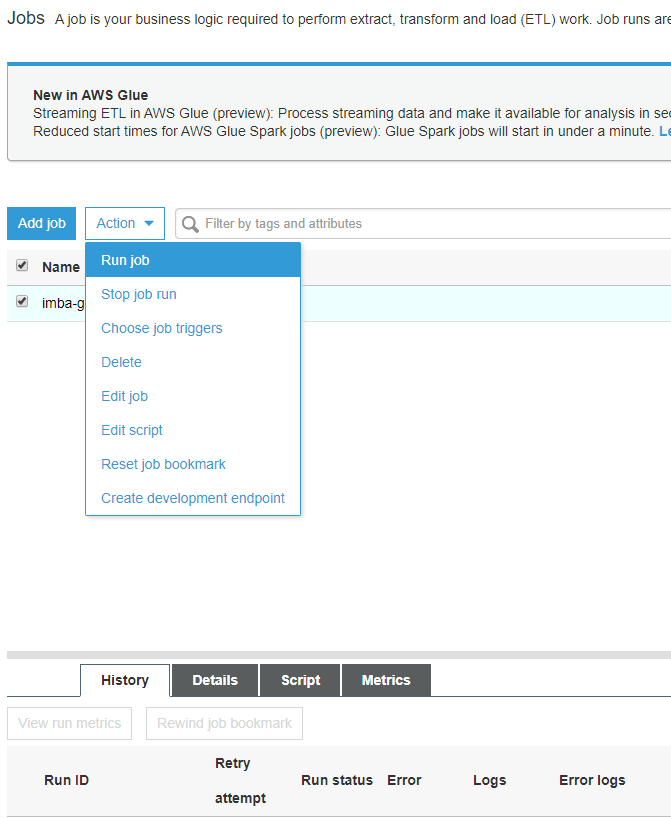




1. Click Save job and edit script:
2. Have a look at the script and close it by clicking the top right X button:
3. Select the job you created and click Run job from Action drop down menu:







1. Be patient, it should take around 10 minutes or so to finish, once done you should see an csv file is created in s3://<your s3 bucket>/output/part-xxxxxxxxxxx.csv
2. Download this file to your local desktop and rename it as “data.csv”

**Data pipeline enhancements:**

**Step 6 Create a Lambda function to invoke Glue Job then add it to Step Function**

1. Add one more step in step function to trigger the glue job automatically? (Look up boto3 documentation on glue client)

Create a Lambda function as below to invoke gluejob imba-glue. Noted to add permission AWSGlusServiceRole to the role.

# Set up logging

import json

import os

import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

# Import Boto 3 for AWS Glue

import boto3

client = boto3.client('glue')

# Variables for the job:

glueJobName = "imba-glue"

# Define Lambda function

def lambda\_handler(event, context):

logger.info('## TRIGGERED BY EVENT: ')

response = client.start\_job\_run(JobName = glueJobName)

logger.info('## STARTED GLUE JOB: ' + glueJobName)

logger.info('## GLUE JOB RUN ID: ' + response['JobRunId'])

return response

Add two states to the end of the Step function as below codes: wait\_ten\_seconds, invoke-imba-glue. Note to add permissions AmazonS3FullAccess, AWSGlusServiceRole.

Note that before invoke Glue Job, it is essencial to wait for 10 seconds letting the previous lambda function finish running. Otherwise there will be an error accur.

{

"Comment": "Step function to run imba process",

"StartAt": "remove\_feature\_files",

"States": {

"remove\_feature\_files": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:remove\_feature\_files:$LATEST",

"ResultPath": "$.remove\_feature\_files",

"Next": "exe\_query\_order\_products\_prior",

"TimeoutSeconds": 60

},

"exe\_query\_order\_products\_prior": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_order\_products\_prior:$LATEST",

"ResultPath": "$.exe\_query\_order\_products\_prior",

"Next": "exe\_query\_user\_features\_1",

"TimeoutSeconds": 60

},

"exe\_query\_user\_features\_1": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_user\_features\_1:$LATEST",

"ResultPath": "$.exe\_query\_user\_features\_1",

"Next": "exe\_query\_user\_features\_2",

"TimeoutSeconds": 60

},

"exe\_query\_user\_features\_2": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_user\_features\_2:$LATEST",

"ResultPath": "$.exe\_query\_user\_features\_2",

"Next": "exe\_query\_up\_features",

"TimeoutSeconds": 60

},

"exe\_query\_up\_features": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_up\_features:$LATEST",

"ResultPath": "$.exe\_query\_up\_features",

"Next": "exe\_query\_prd\_features",

"TimeoutSeconds": 60

},

"exe\_query\_prd\_features": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:exe\_query\_prd\_features:$LATEST",

"ResultPath": "$.exe\_query\_up\_features",

"Next": "wait\_ten\_seconds",

"TimeoutSeconds": 60

},

"wait\_ten\_seconds": {

"Type": "Wait",

"Seconds": 60,

"Next": "invoke-imba-glue"

},

"invoke-imba-glue": {

"Type": "Task",

"Resource": "arn:aws:lambda:ap-southeast-2:480972076311:function:invoke-imba-glue:$LATEST",

"ResultPath": "$.output",

"TimeoutSeconds": 900,

"End":true

}

}

}

1. Move more feature engineering processes into glue job rather than in R script.