CREATE EXTERNAL TABLE maciedetail\_all\_jobs( accountid string,

category string,

classificationdetails originType:string>, createdat string,

description string, id string,

partition string, region string, resourcesaffected

struct<s3Bucket:struct<arn:string,name:string,createdAt:string,owner:struct<displayName:string,id:st ring>,tags:array<string>,defaultServerSideEncryption:struct<encryptionType:string,kmsMasterKeyI d:string>,publicAccess:struct<permissionConfiguration:struct<bucketLevelPermissions:struct<

schemaversion string,

severity struct<description:string,score:int>, title string,

type string, updatedat string)

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe' WITH SERDEPROPERTIES (

'paths'='accountId,category,classificationDetails,createdAt,description,id,partition,region,resourcesAf fected,schemaVersion,severity,title,type,updatedAt')

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT

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

's3://<RESULTS-BUCKET-NAME>/AWSLogs/'

select sensitive\_data.category, detections\_data.type,

sum(cast(detections\_data.count as INT)) total\_detections from maciedetail\_all\_jobs,

unnest(classificationdetails.result.sensitiveData) as t(sensitive\_data), unnest(sensitive\_data.detections) as t(detections\_data)

where classificationdetails.result.sensitiveData is not null and resourcesaffected.s3object.embeddedfiledetails is null group by sensitive\_data.category, detections\_data.type order by total\_detections desc

LIMIT 5

{ "Sid": "Allow Quicksight Service Role to use the key", "Effect": "Allow",

"Principal": {

"AWS": <QUICKSIGHT\_SERVICE\_ROLE\_ARN>

},

"Action": "kms:Decrypt", "Resource": <KMS\_KEY\_ARN>

}

Guard Duty tool:

PUT \_template/gdt

{

"template": "gdt\*",

"settings": {}, "mappings": {

"\_default\_": { "properties": {

"detail.service.action.portProbeAction.portProbeDetails.remoteIpDetails.geoLocation": { "type": "geo\_point"

},

"detail.service.action.networkConnectionAction.remoteIpDetails.geoLocation": { "type": "geo\_point"

}

}

}

}

}

def path = doc['detail.type.keyword'].value; if (path != null) {

int firstColon = path.indexOf(":"); if (firstColon > 0) {

return path.substring(0,firstColon);

}

}

return "";

{

"Schema": {

"Name": "CloudWatchLogRule", "Version": 1

},

"AggregateOn": "Count", "Contribution": {

"Filters": [

{

"Match": "$.eventName", "NotIn": [

"AssumeRole"

]

}

],

"Keys": [

"$.eventName"

]

},

"LogFormat": "JSON", "LogGroupNames": [

"CloudTrail/DefaultLogGroup"

]

}

filter @type="REPORT"

| stats avg(@billedDuration) as mean\_billed\_duration, min(@billedDuration) as min\_billed\_duration, max(@billedDuration) as max\_billed\_duration, percentile(@billedDuration, 95) as Percentile95

filter @type="REPORT"

| stats avg(@maxMemoryUsed/1024/1024) as mean\_MemoryUsed, min(@maxMemoryUsed/1024/1024) as min\_MemoryUsed, max(@maxMemoryUsed/1024/1024) as max\_MemoryUsed, percentile(@maxMemoryUsed/1024/1024, 95) as Percentile95

ilter @type="REPORT"

| stats min(@billedDuration) as min\_billed\_duration,

(min\_billed\_duration + (max\_billed\_duration - min\_billed\_duration)/5) as bucket1, (min\_billed\_duration + 2 \* (max\_billed\_duration - min\_billed\_duration)/5) as bucket2, (min\_billed\_duration + 3 \* (max\_billed\_duration - min\_billed\_duration)/5) as bucket3, (min\_billed\_duration + 4 \* (max\_billed\_duration - min\_billed\_duration)/5) as bucket4, max(@billedDuration) as max\_billed\_duration

filter @type="REPORT"

| fields (@billedDuration>=200 and @billedDuration<360) as R200\_360, (@billedDuration>=360 and @billedDuration<520) as R360\_520, (@billedDuration>=520 and @billedDuration<680) as R520\_680, (@billedDuration>=680 and @billedDuration<840) as R680\_840, (@billedDuration>=840 and @billedDuration<=1000) as R840\_1000

| stats sum(R200\_360) as billedDuration200\_360, sum(R360\_520) as billedDuration360\_520, sum(R520\_680) as billedDuration520\_600, sum(R680\_840) as billedDuration680\_840,

Return REPORT}