Amazon SageMaker PyTorch & Analytics Workshop

Workshop Overview

In this workshop, we will first use Amazon SageMaker to train and deploy a PyTorch model on images of products from the Zappos 50k dataset. This model will be able to learn and predict the similarity between products. Given two products, the model is able to generate a similarity score between 0-1, which determines how similar the products were. Once we have the inferences from the model we will merge this data with information we have on our products to perform analytics and answer questions such as “What stylistically similar products are unpopular? Can we retire them?”

Workshop Scratchpad

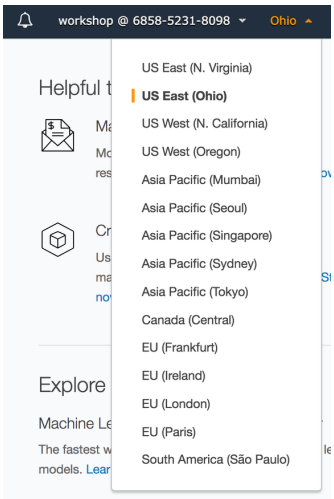
You will be running some commands from the terminal during this workshop, which may not copy over easily from this workshop guide. We've provided a text file with these commands that you can use for copy and pasting here.

Workshop Part 1: Workshop Setup

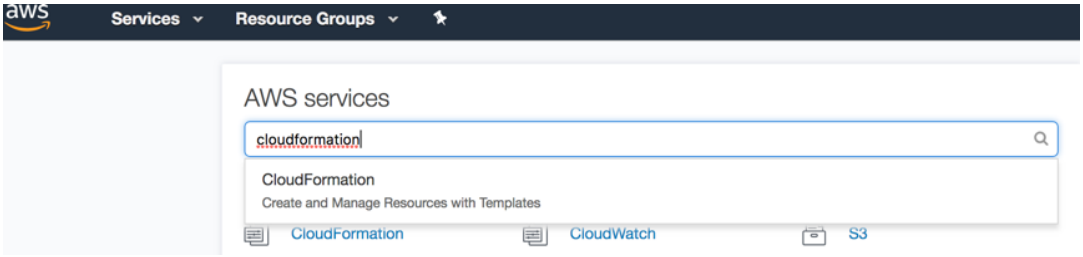
To set up the AWS resources that we will use for this lab, we have already ran a CloudFormation template. This template launches resources such as a SageMaker notebook instance, S3 bucket, AWS Glue Job, and a Redshift cluster that we will use for this lab.

Step 1: Login to the AWS console using the link and credentials provided in the hand out.

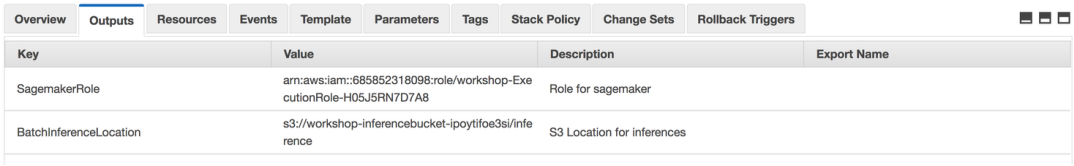
Step 2: Change your AWS region to the region indicated on your hand out by clicking the region menu on the top right side of the screen and selecting the appropriate region.



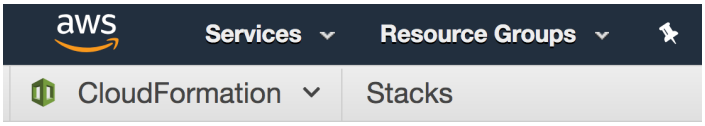
Step 3: Go to AWS **CloudFormation** by searching for the service in the console.



Step 4: In the CloudFormation console you will see a stack created for the workshop. **Click the check box** next to the stack name. Then **click on the "Outputs" tab** at the bottom of the screen. There are two output values from the CloudFormation stack, one for "SagemakerRole" the other one for "BatchInferenceLocation". **Copy and paste** both of these values to a clipboard to save them. You will use these values later in the lab.

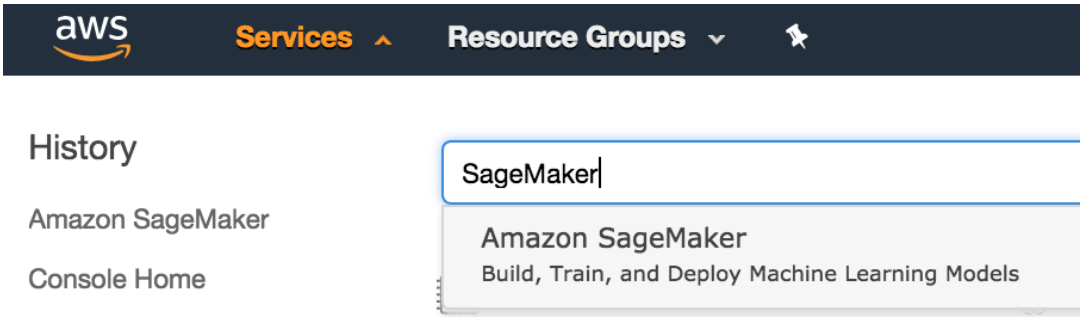


Step 5: Go back to the AWS console by clicking the AWS logo on the top left side of the screen.



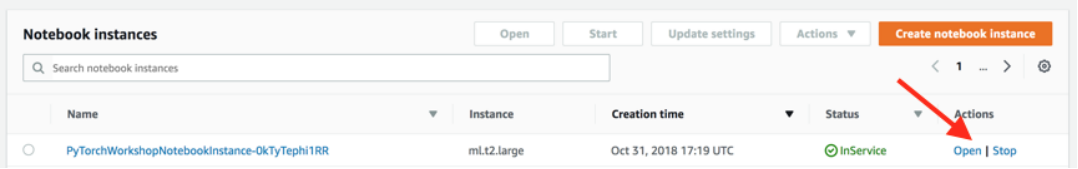
Workshop Part 2: Launch Jupyter Notebook for model training and development

Step 1: Navigate to the SageMaker service in the AWS console.



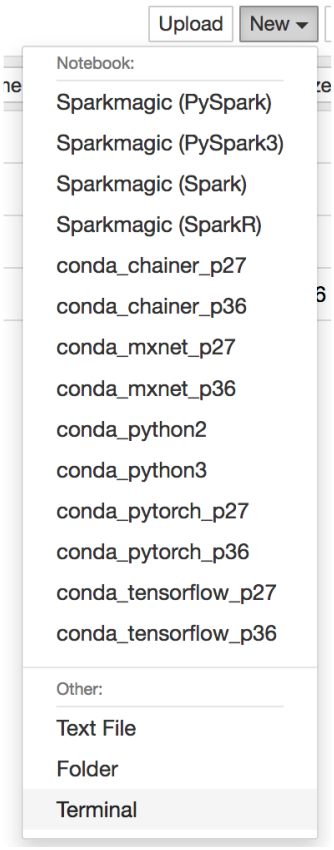
Step 2: The CloudFormation template launched a SageMaker notebook instance for us to use for our development environment. Open up this notebook.

- Select **Notebook instances** from the left navigation under the Notebook section.
- Find the notebook instance starting with the name "PyTorchWorkshopNotebookInstance-XXX" and click **Open** when it is in InService status



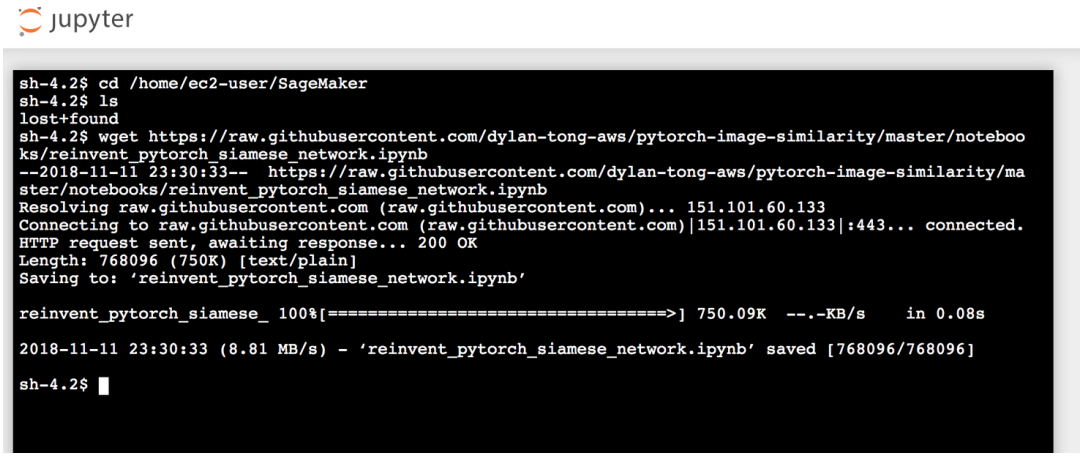
Step 4: Download the workshop Jupyter notebook from Github.

- Click on the drop down located the far right labeled **New**, and select **Terminal**.



- From the **terminal**, switch to the SageMaker directory:
  - cd /home/ec2-user/SageMaker

- Download the notebook by running the following **wget command from the terminal** (Recall the text file that has been provided here to avoid copy and paste issues):
    - *wget https://raw.githubusercontent.com/dylan-tong-aws/pytorch-image-similarity/master/notebooks/reinvent_pytorch_siamese_network.ipynb*



Step 5: Start-up the **reinvent_pytorch_siamese_network.ipynb** Jupyter notebook:

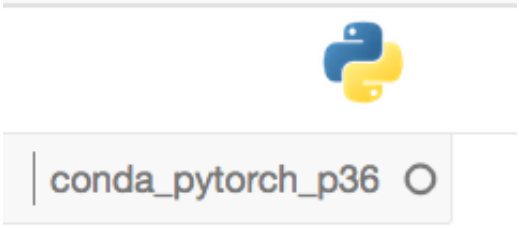
Return to the Jupyter dashboard, and click on **reinvent_pytorch_siamese_network.ipynb** to launch the notebook.

Workshop Part 3: Prepare the data and artifacts

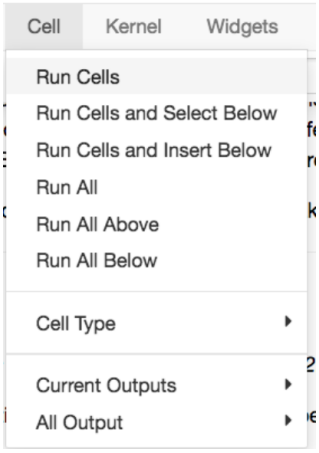
In the next section, we're going to prepare your SageMaker environment with the data, scripts, and artifacts, which will be used to build, train and deploy your image similarity model. We will perform the steps by running the steps laid out in the cells of the **Setup** and **Data** sections of your Jupyter notebook.

Step 1: Cells contain code snippets in your notebook, and can be executed in isolation. One way to run a cell is to select **Run Cells** from the **Cell** dropdown located on the top navigation bar.

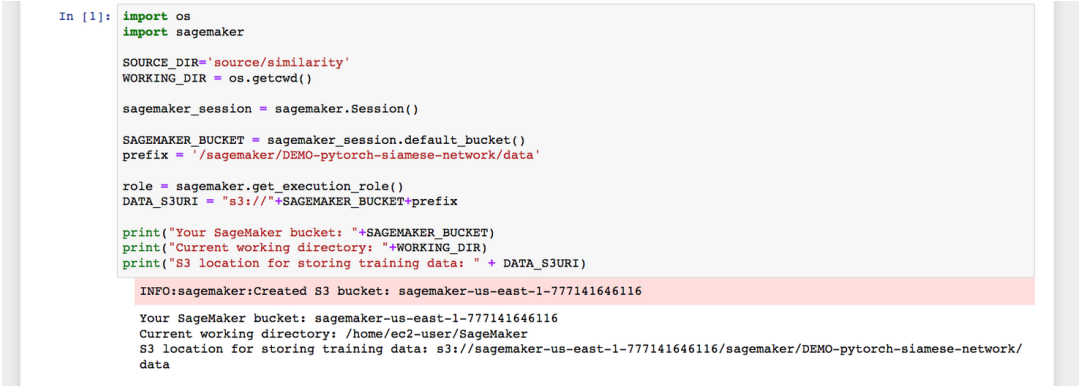
Confirm that the notebook's kernel is set to **conda_pytorch_p36**. The current kernel is displayed at the top right hand corner of your notebook. This should be the default for this notebook. This kernel provides you with access to python 3.6, common python data science libraries like Scipy and Pandas as well as PyTorch.



If it isn't running this kernel for unexpected reasons, switch to this kernel from the top navigation by selecting **Kernel > Change Kernel > conda_pytorch_p36**. Next, select the first cell, and **run the cell.**



This will create a SageMaker session, a reference to the execution role to grant permissions, and create a default S3 bucket in your current region to hold your data and artifacts. Take note of the name of your SageMaker bucket. You will need to reference it later.



In [1]:
```
import os
import sagemaker

SOURCE_DIR='source/similarity'
WORKING_DIR = os.getcwd()

sagemaker_session = sagemaker.Session()

SAGEMAKER_BUCKET = sagemaker_session.default_bucket()
prefix = '/sagemaker/DEMO-pytorch-siamese-network/data'

role = sagemaker.get_execution_role()
DATA_S3URI = "s3://"+SAGEMAKER_BUCKET+prefix

print("Your SageMaker bucket: "+SAGEMAKER_BUCKET)
print("Current working directory: "+WORKING_DIR)
print("S3 location for storing training data: " + DATA_S3URI)
```

```
INFO:sagemaker:Created S3 bucket: sagemaker-us-east-1-777141646116

Your SageMaker bucket: sagemaker-us-east-1-777141646116
Current working directory: /home/ec2-user/SageMaker
S3 location for storing training data: s3://sagemaker-us-east-1-777141646116/sagemaker/DEMO-pytorch-siamese-network/
data
```

Step 2: Run the second cell and take note of the region that you're running the workshop in. You'll need to reference this region in future commands.

```
In [5]: import boto3
        WORKSHOP_REGION = boto3.session.Session().region_name
        print("Your current region is "+WORKSHOP_REGION)

          Your current region is ap-southeast-2
```

Step 3: Follow the instructions in the notebook, and run the cells up until the **Train** section. As described in the notebook, these steps will copy the data, scripts and artifacts that will later be used to build, train and deploy your models.

The start of this step appears in the notebook as displayed in the following screenshot:

Step 3

For the next part of the workshop, work through the "Data" section of this notebook up until the next section "Train."

Data

The original dataset can be downloaded from UT Zappos50k. The dataset has been made available at the following S3 bucket to offload the original site.

```
DOWNLOAD_S3URI = "s3://reinvent2018-sagemaker-pytorch"
```

The end of this section appears in this notebook as shown in the screenshot below:

This concludes part 3 of the workshop. The next section of the notebook corresponds to part 4.

Workshop Part 4: Train the model

Train

We need to provide a training script that can run on the SageMaker platform. The training script is very similar to a training script you might run outside of SageMaker, but you can access useful properties about the training environment through various environment variables.

A typical training script loads data from the input channels, configures training with hyperparameters, trains a model, and saves a model to model_dir so that it can be hosted later. Hyperparameters are passed to your script as arguments and can be retrieved with an argparse.ArgumentParser instance.

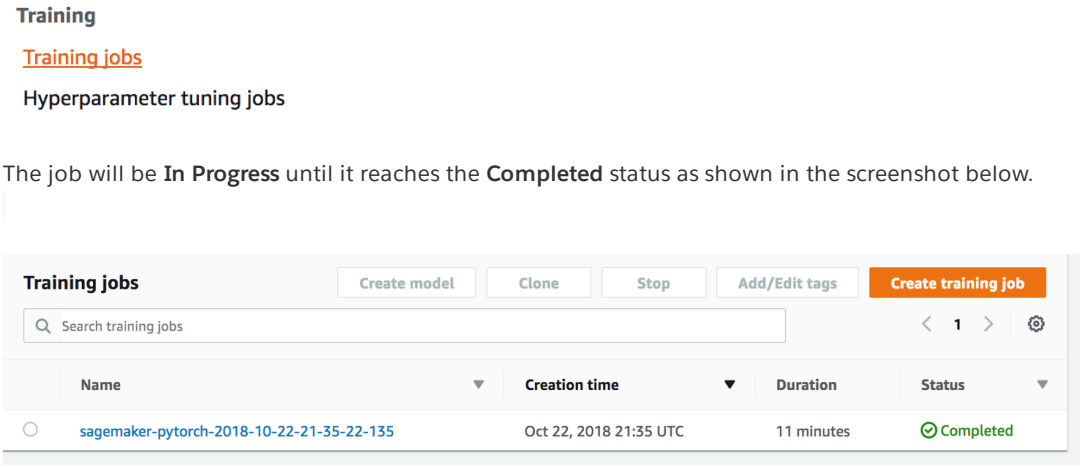
In the following steps, scripts are provided to build and train the following model:

Workshop Part 4: Train the model

In the next section of your notebook, we will go through the process of remote SageMaker training by launching a training job from your Notebook instance via the programatic interface. Note that there are other training options including local training on an GPU enabled SageMaker Notebook instance for prototyping, or launching remote training jobs through the console or AWS CLI.

Step 1: Follow the instructions in the notebook and **run the cells up until the end of the section**. The last step will launch a remote training job via the command: `estimator.fit({'train':DATA_S3URI}).`

The training job is configure to only one epoch, but will take at least 11 minutes on GPU to complete. Going through these steps should be enough for your to understand the model training process.
You can monitor the progress of the training job from your notebook as well as the console. To view your job in the console, select **Training jobs** under the **Training** section of the left navigation.

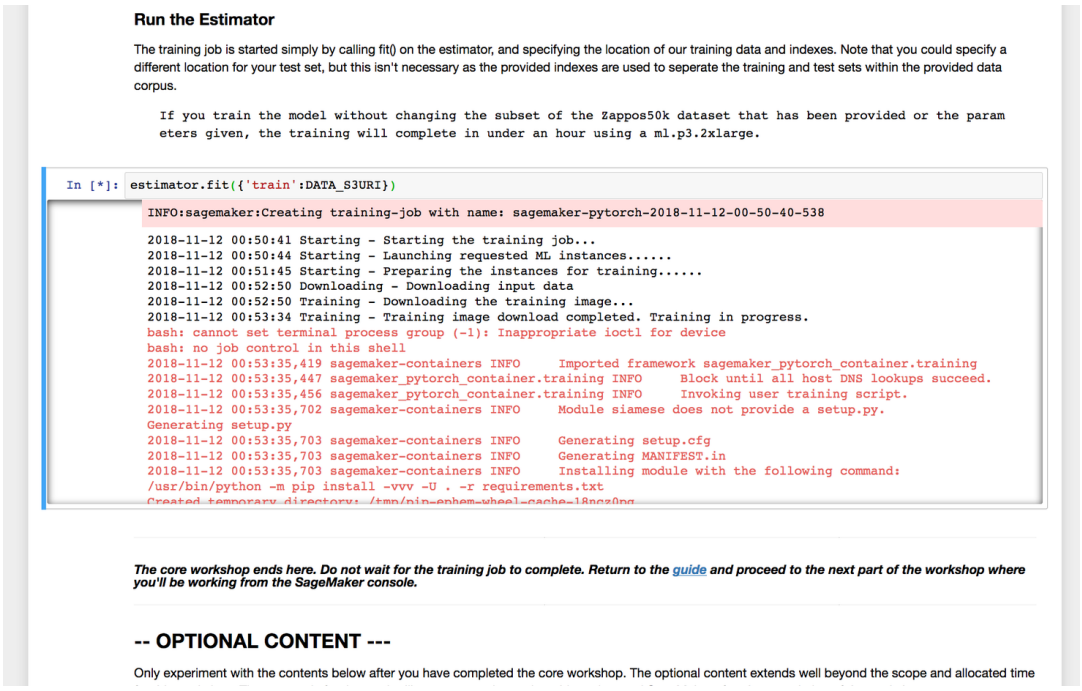


The job will be **In Progress** until it reaches the **Completed** status as shown in the screenshot below.

Instead of waiting for 11+ minutes of training, immediately move on to the next section of this lab where we leverage a pre-train model for inference.

Note that the remainder of the notebook only contains optional content. You are free to experiment with the optional content after you complete the lab. We will spend the remainder of the SageMaker portion of this workflow experiencing the console interface.

The end of this part of the workshop appears in the notebook as follows:



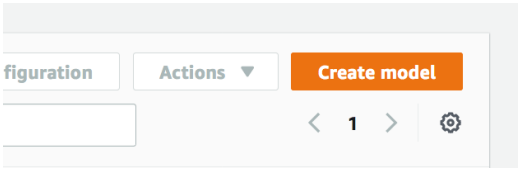
Workshop Part 5: Register a Pre-trained Model with SageMaker

Since fully training a model is a time consuming process, we're going to leverage a pre-trained model, so that we can continue on our journey without waiting on the training process. In this section, we'll learn how to register a trained model with SageMaker through the console.

Step 1: Return to the **SageMaker console**

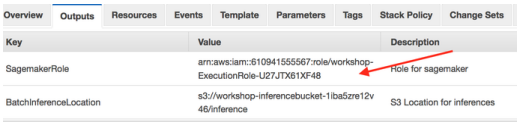
Step 2: Select Models under the Inference section of the left hand navigation.

Step 3: Click on the **Create model** button.



Step 4: Provide a name for your model.

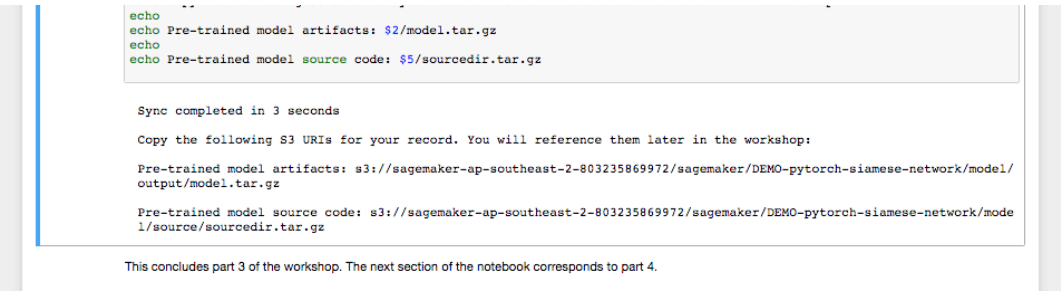
Step 5: Select the **Enter a custom IAM Role ARN** for the IAM Role in the drop down. Use the role provided under “SagemakerRole” in the outputs section of the CloudFormation template that was referenced at the start of this workshop.



Step 6: Provide the following inference code image: **520713654638.dkr.ecr.<<replace with your-region>>.amazonaws.com/sagemaker-pytorch:0.4.0-cpu-py3.**

The SageMaker PyTorch container is region specific so you'll need to substitute **<<replace with your-region>>** above with the AWS region you're running SageMaker in. For example**,** if you're running the workshop the workshop from the us-west-2 region, the image will be 520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch:0.4.0-cpu-py3.

Step 7: Previously, you copied pre-trained model artifacts over to your account's default SageMaker S3 bucket.



Enter the S3 location of the model artifacts. It should have this form: **s3://<<Your SageMaker Bucket>>/sagemaker/DEMO-pytorch-siamese-network/model/output/model.tar.gz.**

Note that once the training job that you created in the previous section completes, it will create artifacts for your model packaged in the same manner.

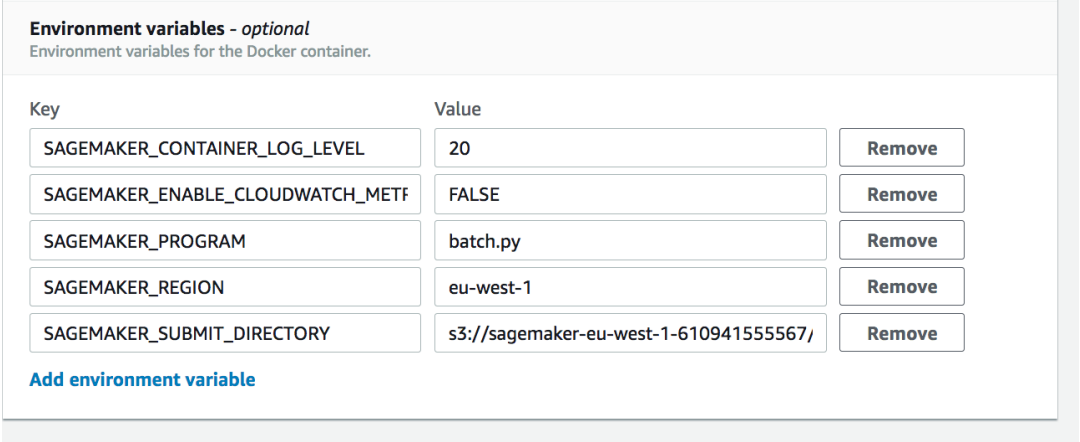
Step 8: Set the following **Environment Variables**:

KEY	VALUE
SAGEMAKER_CONTAINER_LOG_LEVEL	20
SAGEMAKER_ENABLE_CLOUDWATCH_METRICS	FALSE
SAGEMAKER_PROGRAM	batch.py
SAGEMAKER_REGION	**<<your region>> eg. us-west-2**
SAGEMAKER_SUBMIT_DIRECTORY	**s3://<<Your SageMaker Bucket>>/sagemaker/DEMO-pytorch-siamese-network/model/source/sourcedir.tar.gz**

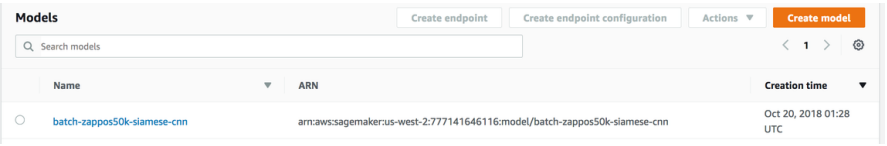
The **SAGEMAKER_SUBMIT_DIRECTORY** key should be set to the directory that contains the model's scripts. These were dowloaded to your S3 bucket along with the model artifacts.

The location should look something like:
s3://<<Your SageMaker Bucket>>/sagemaker/DEMO-pytorch-siamese-network/model/source/sourcedir.tar.gz

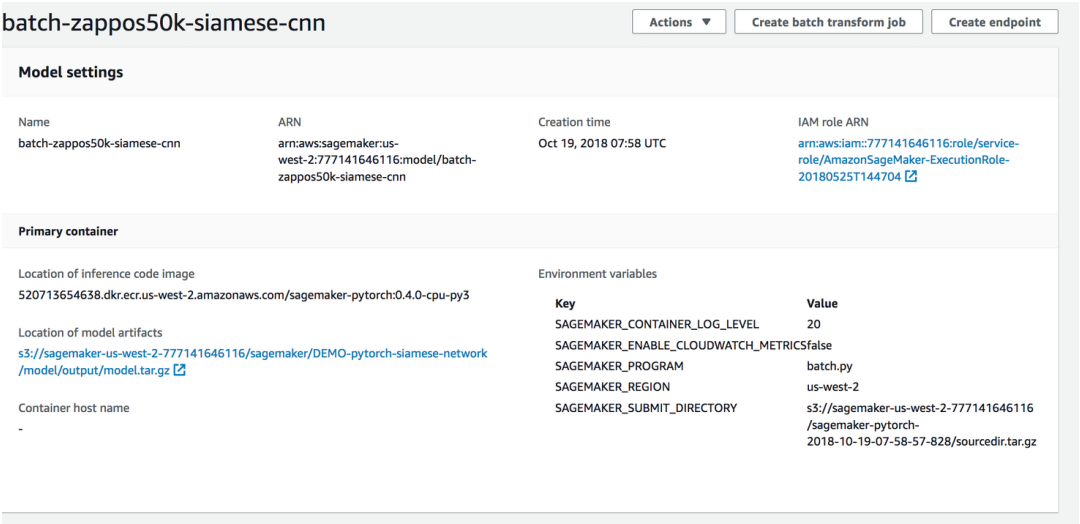
The screenshot below illustrates how the completed section appears in the console for the eu-west-1 region:



Step 9: Select **Create model** at the bottom of the page. You should see your model listed in the SageMaker console:



If you click on the model name, you will be linked to the details page of your model. It should look similar to the following:



Workshop Part 6: Performing Batch Inference

Now that we have a model, we can infer the visual similarity between each product and the rest of the Zappos50K product catalog. SageMaker supports both real-time inference and batch inference. A real-time inference point can be useful for on-demand similarity queries between a pair of products. However, in this case, we want to perform some large scale similarity analysis across our customer base and entire product catalog, so we're going to infer the similarity values between our products in batch via **SageMaker Batch Inference jobs.**

The notebook has an optional section that demonstrates how a SageMaker Batch Inference job can be initiated programmatically. Once your training job completes, you can optionally start a batch inference job from the notebook to experience the process.

In the meantime, lets learn how to **create a batch inference job from the console**.

The batch inference job that we're going to create expects inputs in NPY format. The images have been pre-processed by packing pairs of image tensors in NPY format into gzip files, so that images can be sent to a SageMaker Batch Transform job cluster in batches. The pre-processed images will be provided for you.

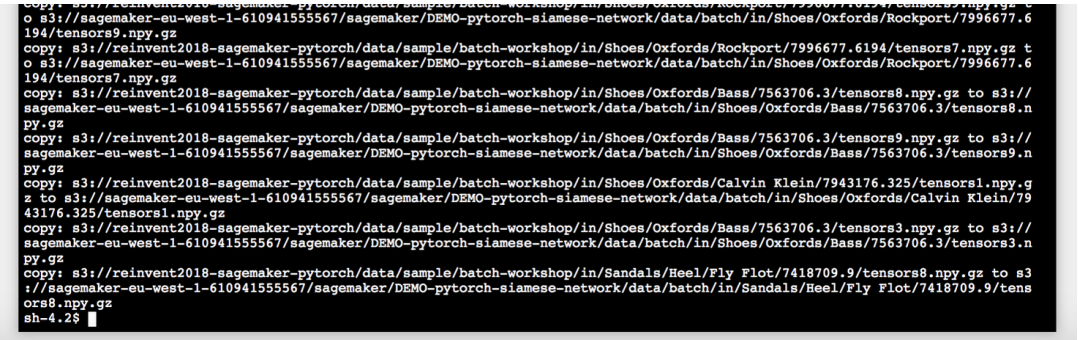
Step 1: Copy the pre-processed inputs that have been created for a sample subset of the Zappos50k images. We'll accomplish this through the AWS CLI from the terminal on your SageMaker Notebook Instance.

Open your Sagemaker notebook instance, and select **Terminal** from the far right drop down labeled **New**.

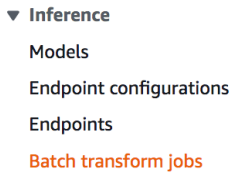
Step 2: Run the following **S3 sync command** from the terminal to copy all the pre-processed data over to your SageMaker default bucket. Take note of this S3 URI.

aws s3 sync s3://reinvent2018-sagemaker-pytorch/data/sample/batch-workshop/in s3://<<Your SageMaker Bucket>>/sagemaker/DEMO-pytorch-siamese-network/data/batch/in

The sync will look similar to the following in your terminal:

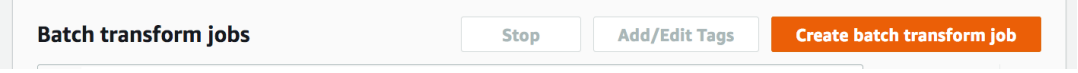


Step 3: Begin creating a Batch Transform Job from the console by clicking on the **"Batch transform jobs"** option from the left navigation under the **Inference** section.

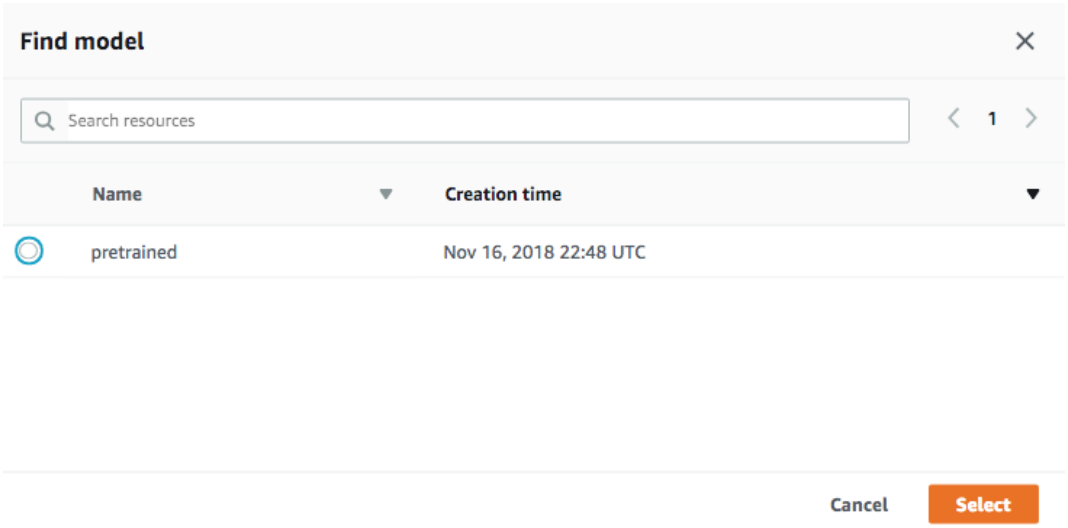


Step 4: Press the **Create batch transform job button**.

Step 5: Enter a **Job Name** in the **Batch transform job configuration** form.



Step 6: Press the **Find Model** button. Select the model that you created in the previous section, and press the **Select** button.



Step 7: Select *ml.c5.xlarge* as the **instance type**, and set the **instance count** to **1**.

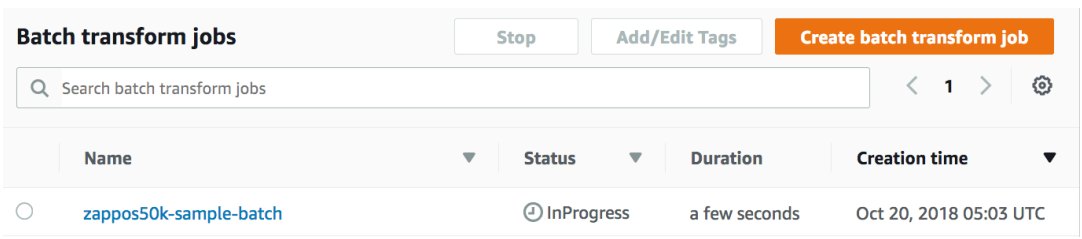
Step 8: Under **Input data configuration:**

- Select **S3Prefix** as the **S3 data type.**
- Leave **Split type** and **Compression** as **None**.
- Set **Content-type** to **application/x-npy**.
- Set the **S3 URI** to the location of the sample batch input files that you downloaded earlier (**step 2**). The URI should resemble:
    - *s3://<<Your SageMaker Bucket>>/sagemaker/DEMO-pytorch-siamese-network/data/batch/in*

Step 9: Under **Output data configuration:**

- Reference the Output tab of your CloudFormation template, and copy the value of "BatchInferenceLocation" in the CloudFormation script outputs to the **S3 output path.** The S3 location will look something like:
    - *s3://<<bucket-name>>/inference*
- Set **Accept** to **text/csv**.
- Leave **Assemble with** as **None**.

Step 10: Press **Create job**. You should see your batch transform job listed in the **InProgress** status.



The Batch transform job will take some time to complete. Instead of waiting for it to complete, move on to the next step and you can come back and check on the job at a later time.

Workshop Part 7: ETL the Inference Data

Now that we have inferences, we can perform some transformations on this data in order to run analytics on our results. We will be using AWS Glue to run an ETL (Extract, transform, load) job that will merge our inference data with some product catalog data in order to put this data into Redshift, which we can use for running analytical queries against. The final output will be a table that looks like this:

1	productId1	productId2	Similarity
2	95883	100134	0.95883
3	99344	90559	0.78995
	A	B	C

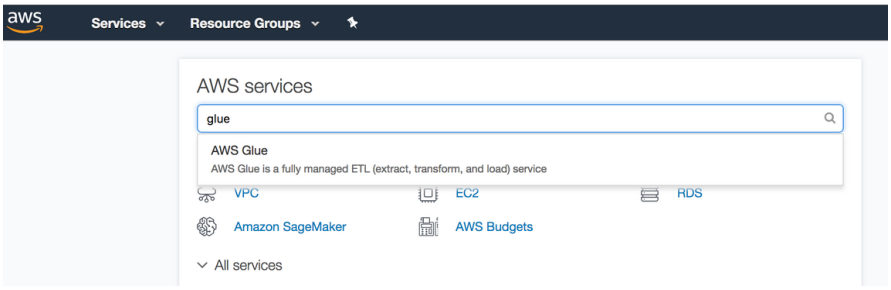
This table contains all the pairs of products that we ran inferences against and their similarity scores. We will build this table by joining our inference data with data from the product catalog of all the products in the Zappos 50k dataset. This data will ultimately reside in Redshift along with other data on these products, such as the number of orders for each product. With this data in Redshift, we can then create dashboards for analysis.

If you want to take a look at the exact tables we are joining, the raw inference data can be found under this directory: https://s3-us-west-2.amazonaws.com/reinvent2018-sagemaker-pytorch/data/sample/batch-32/out/
We will be using this as the source of the inferences, this way if your batch transform job has not yet completed, you can still run the ETL script as this contains the the data that will be produced by the batch inference job.

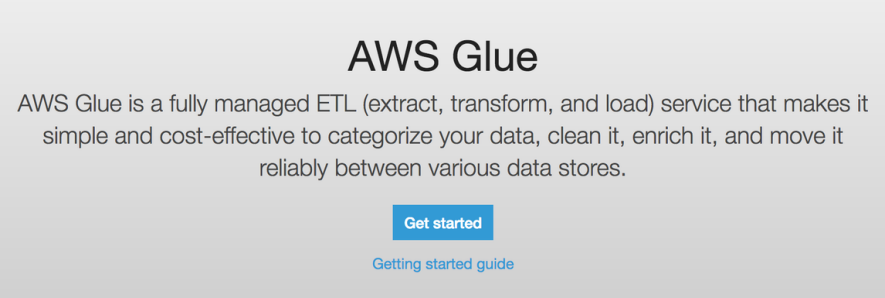
The product catalog can be found here: https://s3-us-west-2.amazonaws.com/reinvent2018-sagemaker-pytorch/data/workshop/tables/product_catalog.txt

To save time in the workshop, our ETL script will write out the data out to S3 instead of loading it into a Redshift cluster. Instead, the CloudFormation template we launched creates a Redshift cluster with this data already populated so that we can start creating analytical dashboards against it. This script, however, goes though all the steps required to ETL the raw inference data so that it can be loaded into Redshift.

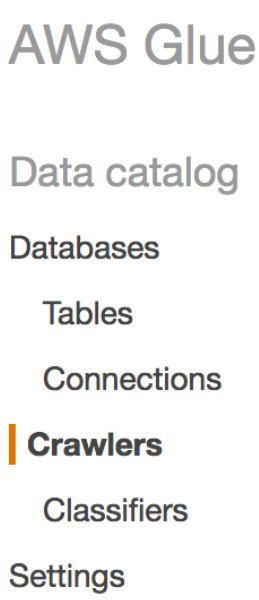
Step 1: Go to the AWS Glue console.



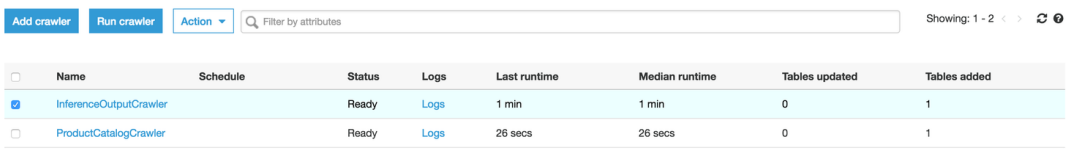
If you come across the screen below, click **Get started** to go to the Glue console



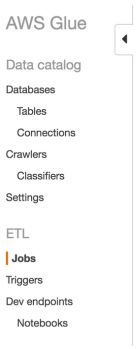
Step 2: On the menu on the left of the Glue console, select **Crawlers**



Step 3: You should see two crawlers already created by the CloudFormation template. These crawlers read over the output of our ML inference, as well as a product catalog of all our products and classify their schema. The schema is then placed into the Glue data catalog, and is used for easy access to the data by our Glue ETL job. Our Glue ETL script can easily access this data once the data catalog is populated with it's metadata. **Click the check box next to each crawler and then click "Run crawler" to run the crawler.** When the crawlers complete running, you should see that a table has been added by each crawler to the data catalog.

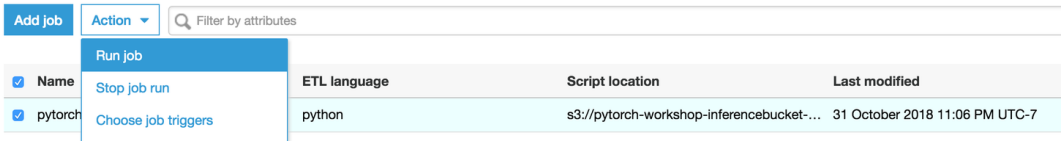


Step 4: On the menu on the left of the Glue console, select **ETL Jobs**

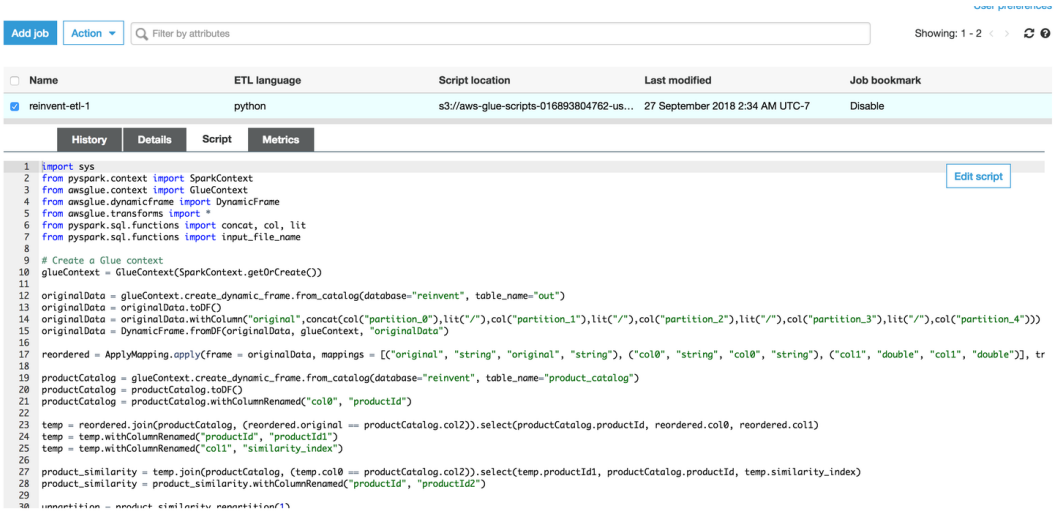


Step 5: The CloudFormation template you launched earlier created a Glue Job named **pytorch-workshop-etl**. Select this Glue Job by clicking the checkbox next to the name of the job, and then click **Action > Run Job**. This will go ahead and start the Glue ETL Job.

This job will take the raw inferences from our model and transform and join them with data in a product catalog table that is stored on S3 so we have a list of productId pairs and a similarity score between the pairs of products that we ran the ML model on. This output table will be stored in S3, and would ultimately be copied to Redshift.



Step 6 (optional): Select the **scripts tab** with the Glue Job select it in order to inspect the ETL code and see the exact Spark ETL code that is being run in order to transform the raw inference results into an analytics table for loading into Redshift. You can also view the output of the ETL script here: https://s3-us-west-2.amazonaws.com/reinvent2018-sagemaker-pytorch/data/workshop/etl/part-00000-701121d8-f653-4466-8e97-7e354b8c9aae-c000.csv

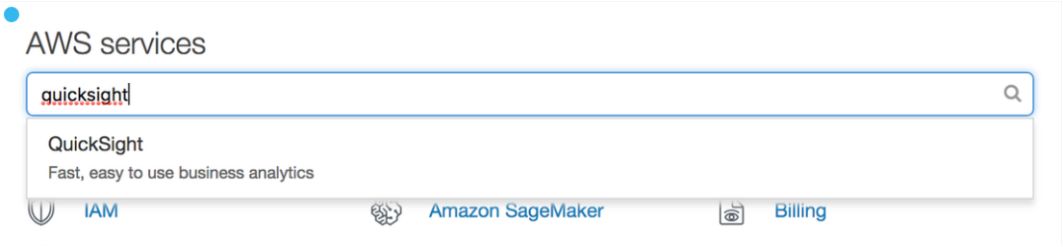


Workshop Part 8: Create Dashboards for Analysis

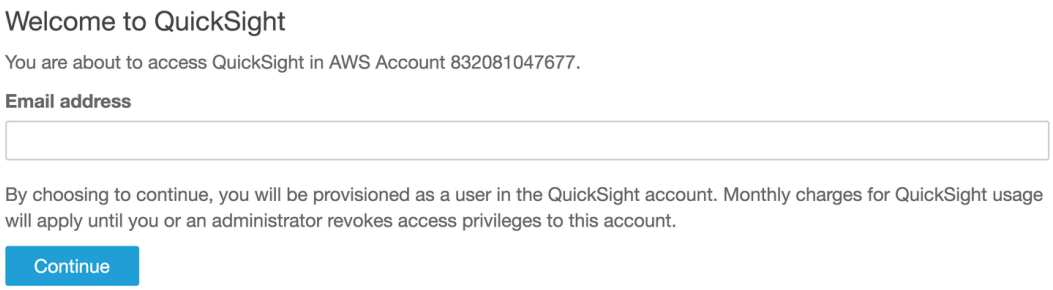
We will now try to visualize the similarity data in such a way that we can take some informed decisions such as:

a) What products from my catalog can I retire based on the sales/revenue?
b) Can I categorize my products based on the similarity scores?

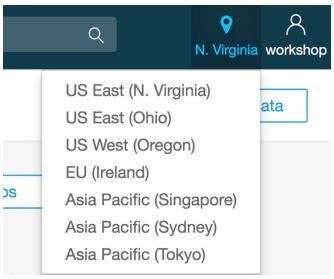
Step 1: Setup access to Amazon Quicksight. Go to the AWS console and search for QuickSight. Click on QuickSight to go to the Amazon QuickSight console.



Step 2: Setup a Quiksight user by entering your email address in the box below. Then click continue.

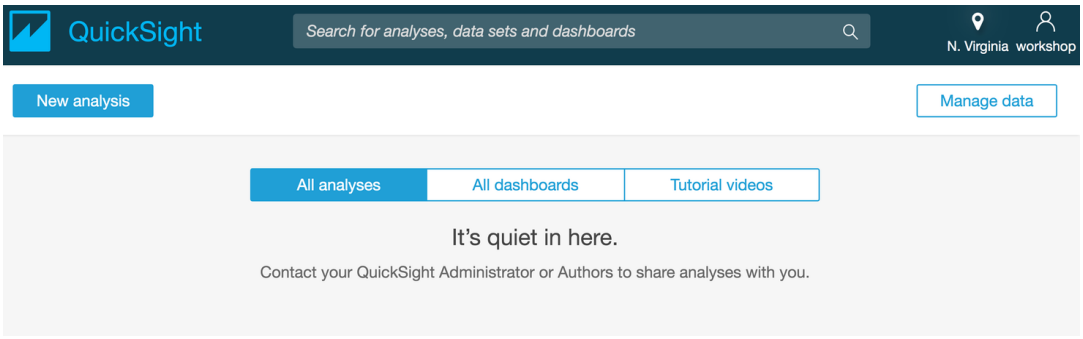


Step 3: Change the region you are running quick sight in by clicking on the region in the top right corner of the screen and click the region you are running the workshop in.

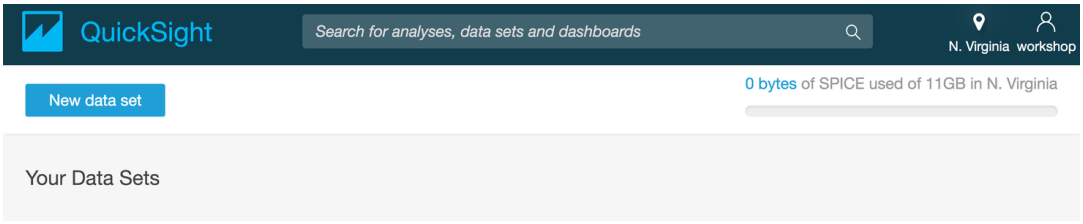


Step 4: Ensure quicksight has access to your redshift cluster

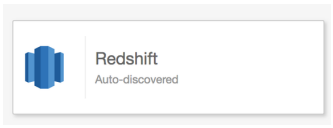
Click on **"Manage Data"** on the top right of the screen:



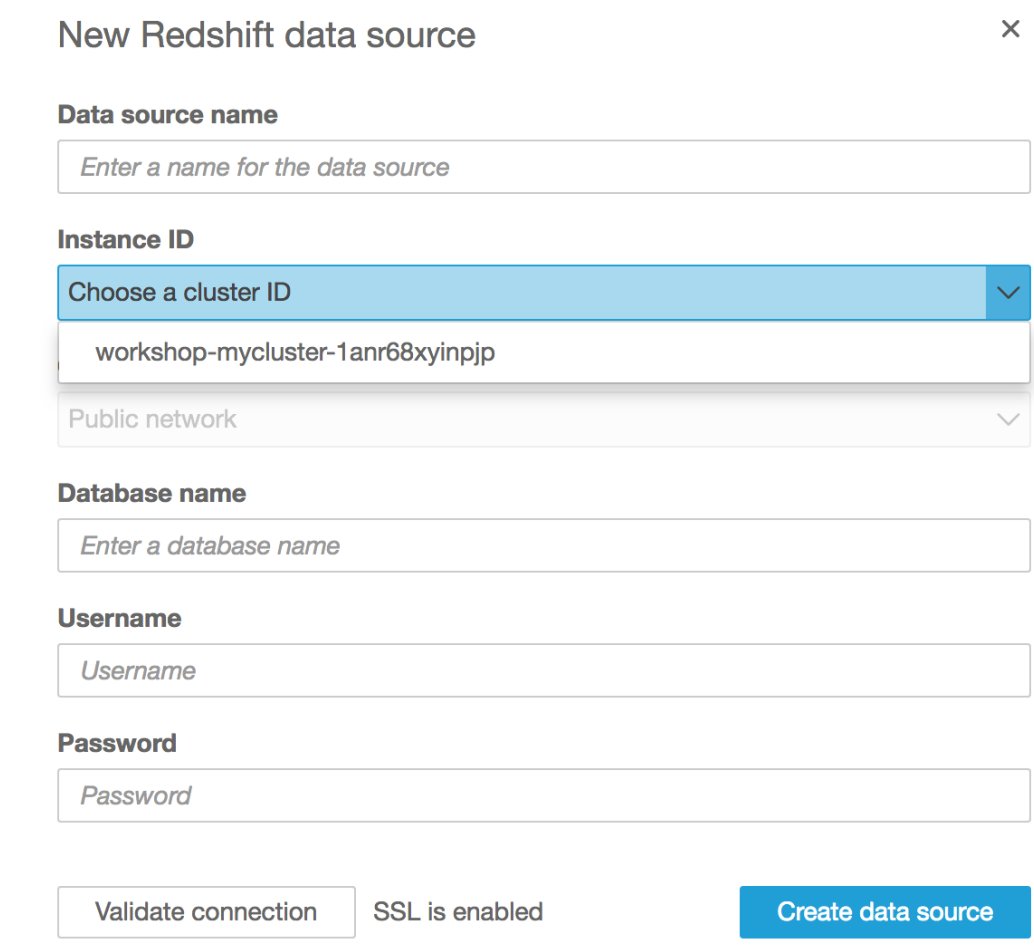
Click on **"New data set"** on the top left of the screen



In the menu, select **"Redshift - Auto-discovered".**

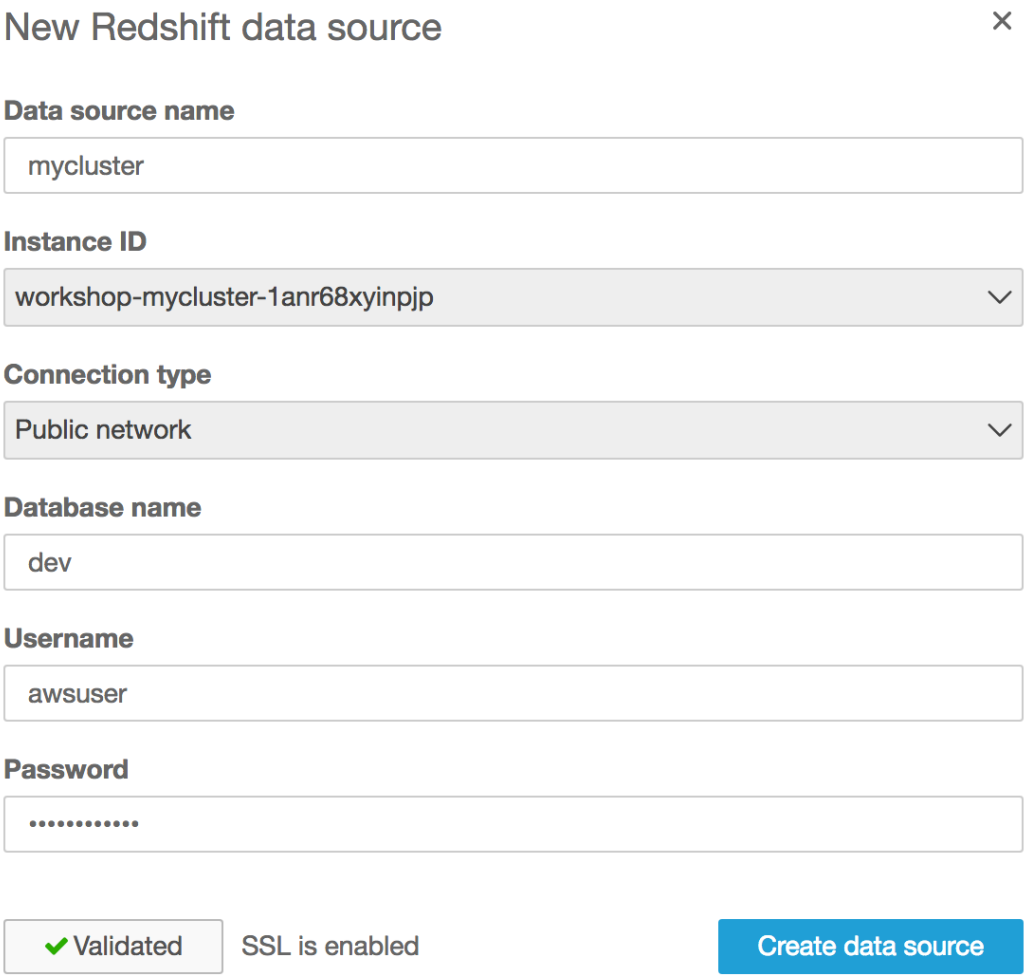


You will be taken to the connection window

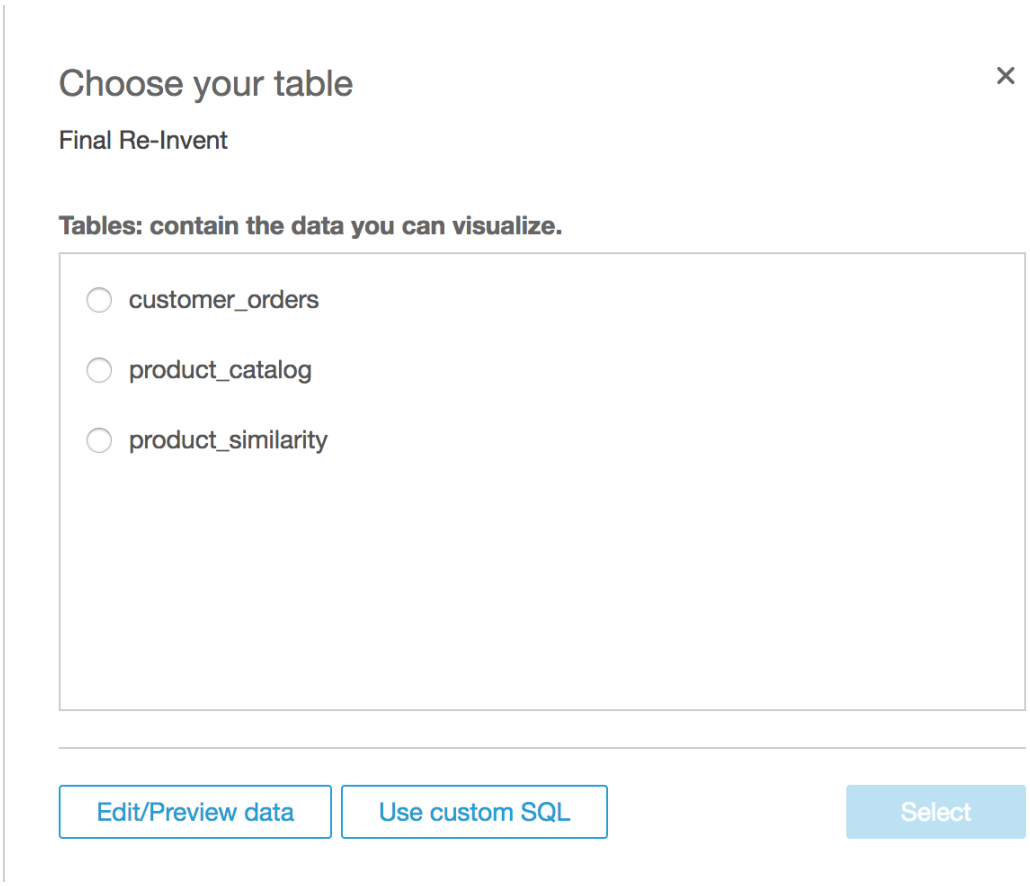


Select the Redshift cluster that is populated in the "Instance ID" dropdown menu
Enter a data source name, such as **my cluste**r
For database name, enter **dev**
For username enter **awsuser**
For password enter: **Redshift1234**

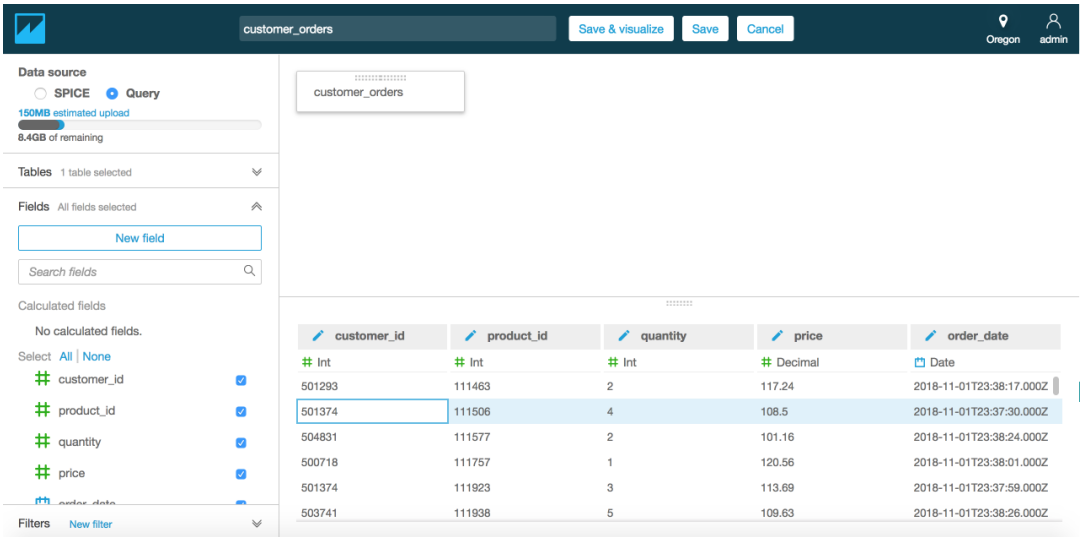
Click **validate** to ensure that you have a connection to the Redshift cluster. Once you see a green check mark, click **Create Data source**



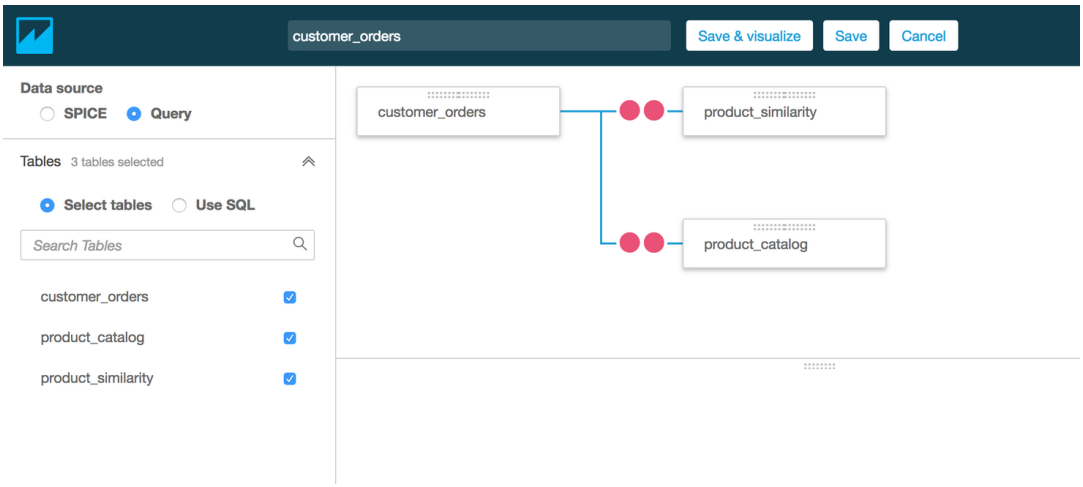
Once you enter all the details and click “Ok”, you should be taken to the following window:



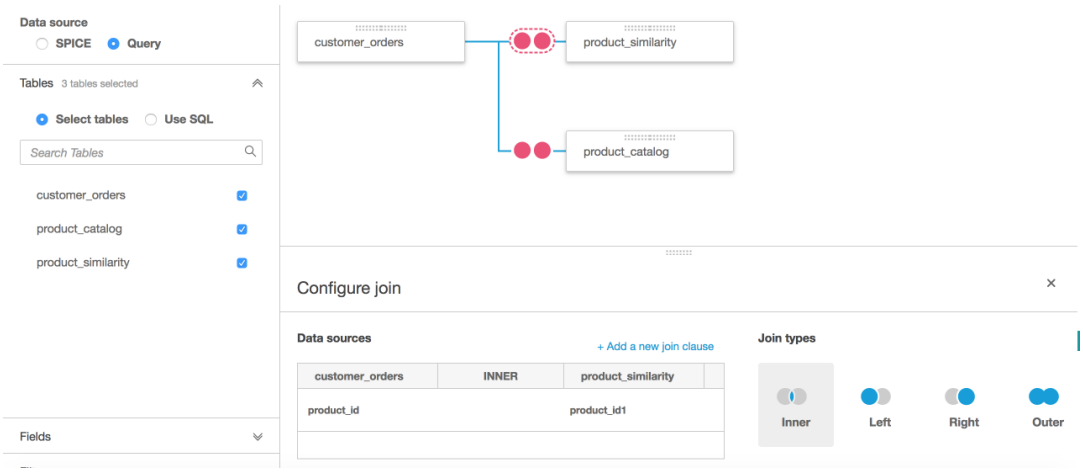
Click on **Edit/Preview data**. You should be taken to the following window:



Click on **"Tables"** on the left menu and ensure all the tables are selected:

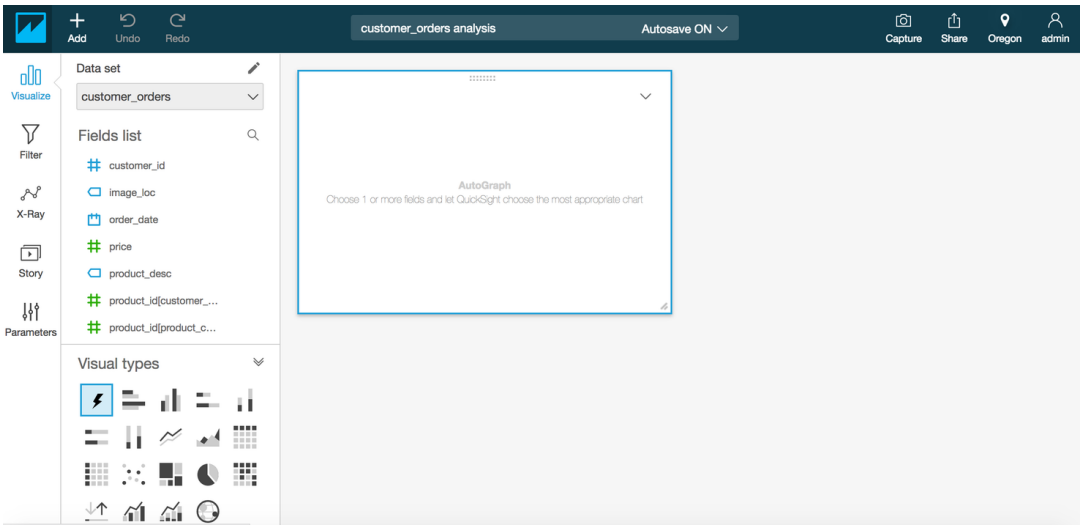


Click on the first of the two red circles in the graph to define the join between the tables. Observe that I have used **product_id** and **product_id1** for an “inner” join. Select these in the “Data sources” section of the screen

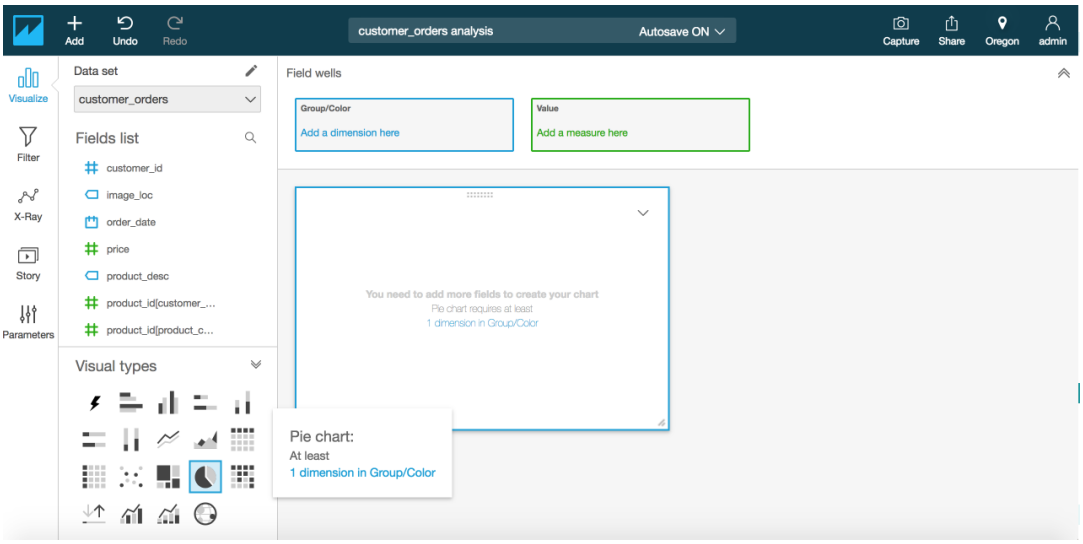


Repeat the same process for the second join (between customer_orders and product_catalog) this time joining between **product_id** of customer_orders table and **product_id** from product_catalog table.

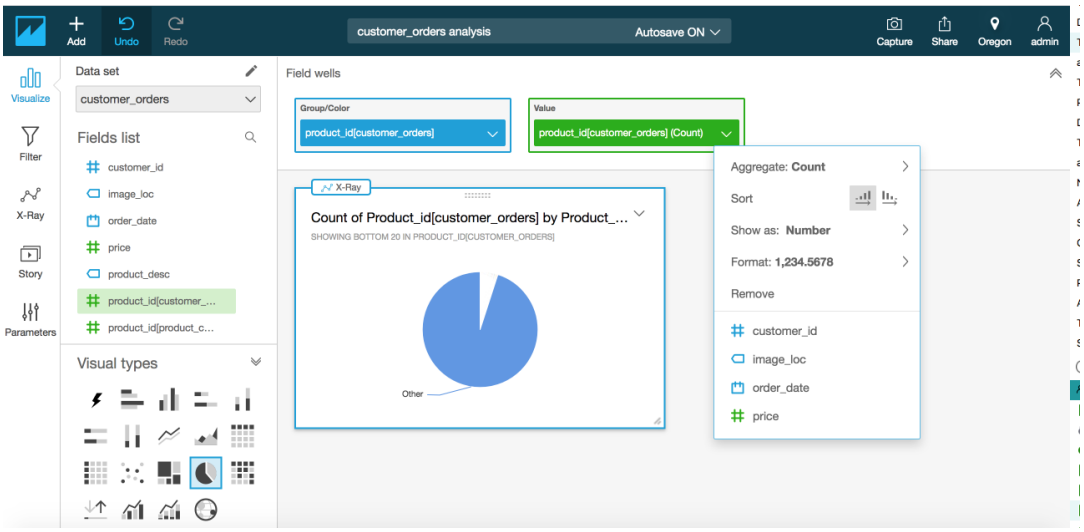
Once that is done, **click on “Save & visualize”** on the top. You should be taken to the following window:



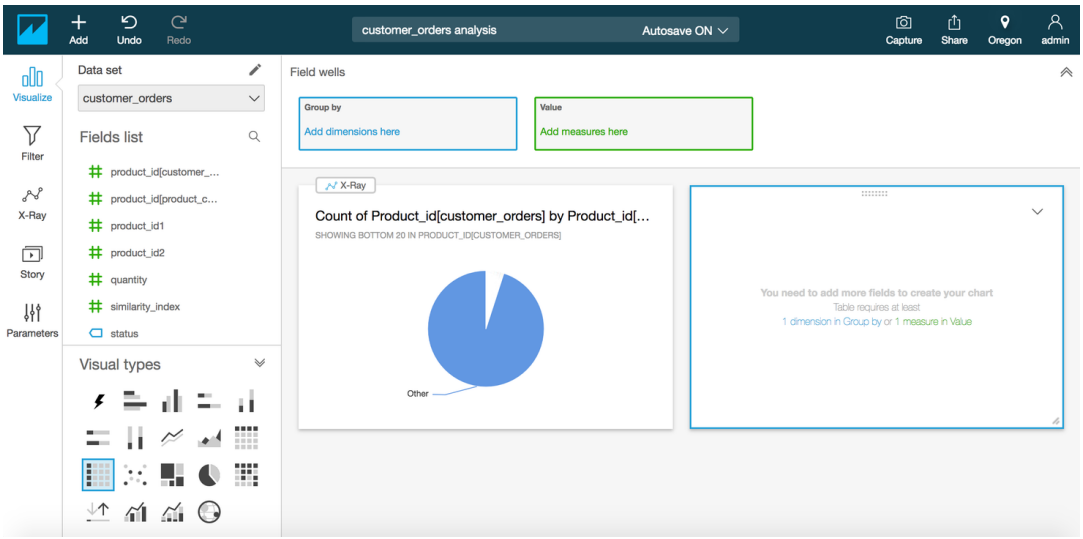
At the bottom left corner, you should see a list of “Visual types”. **Select the “Pie Chart”**. You should be taken to the following window:



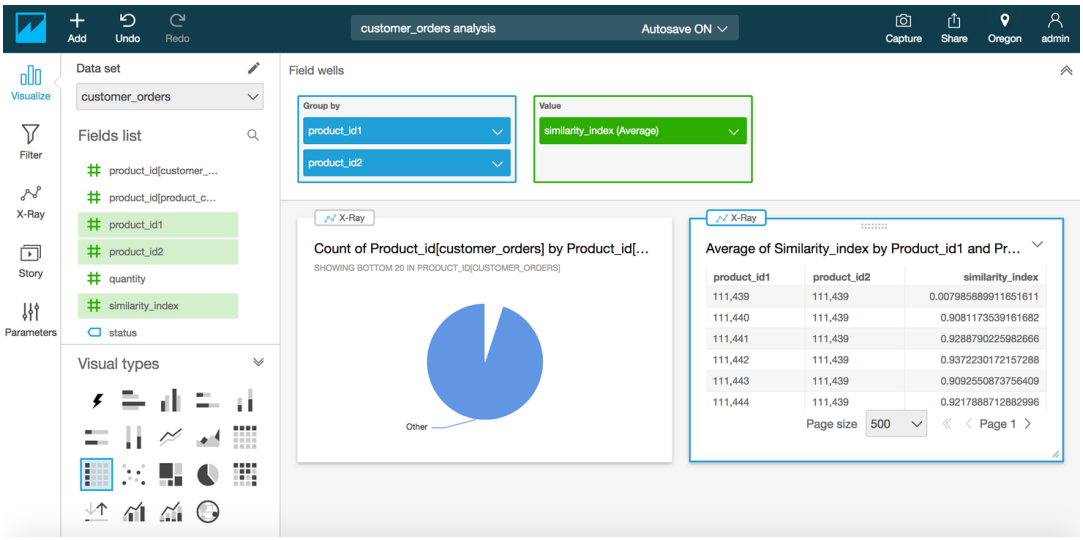
Add “product_id” from customer_orders as the group/color and count(product_id) as the value. In the value, ensure you are sorting from low to high values so that you can know the bottom 20 selling products:



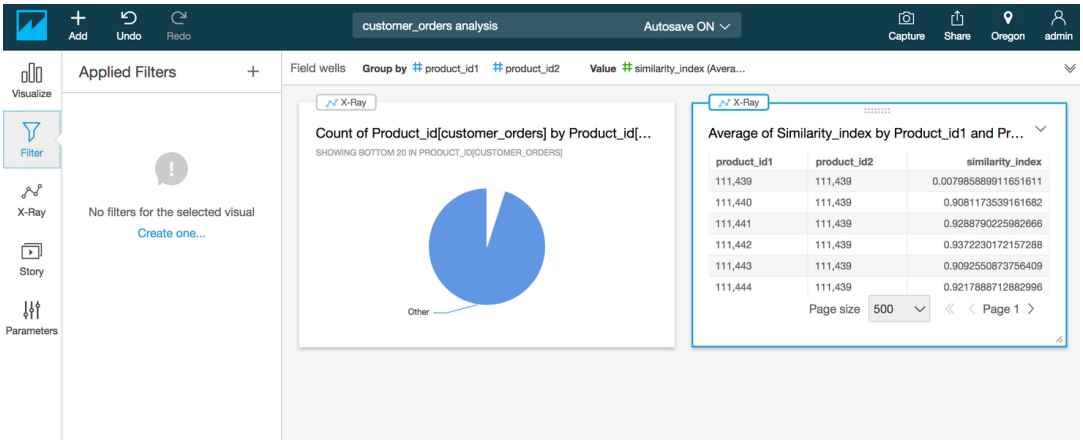
Click on “+/Add” button on top left corner to add another visual to this dashboard. This time select “Table” for the visual type:



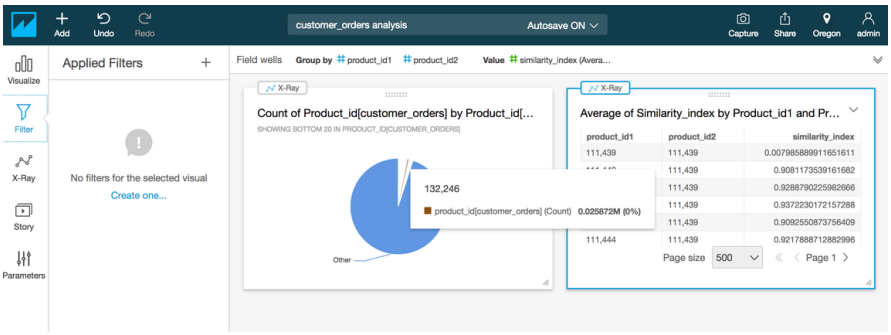
For “Group by” dimensions, drag and drop product_id1 and product_id2. For “Value”, drag and drop similarity_index. You can use the “Average” value. You should see the following changes to the dashboard:



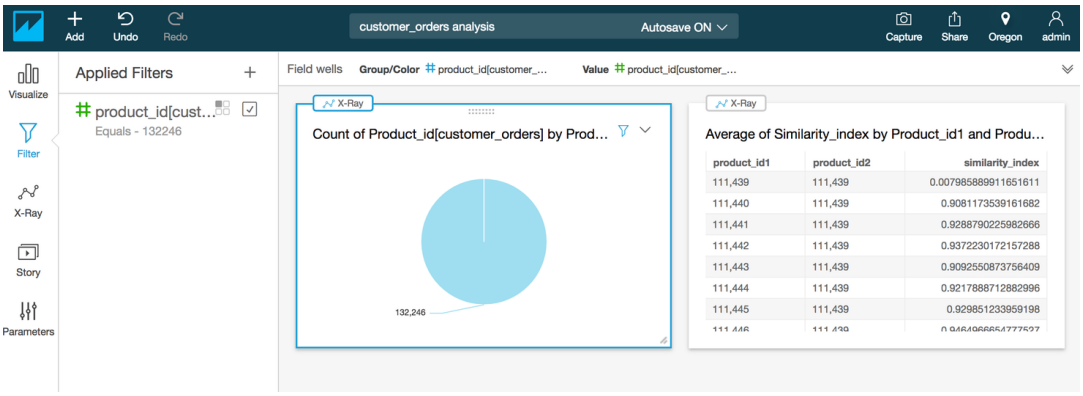
Now, we can start our analysis by filtering data by product_ids that are least popular. Click on “Filter” on the left hand pane of the window:



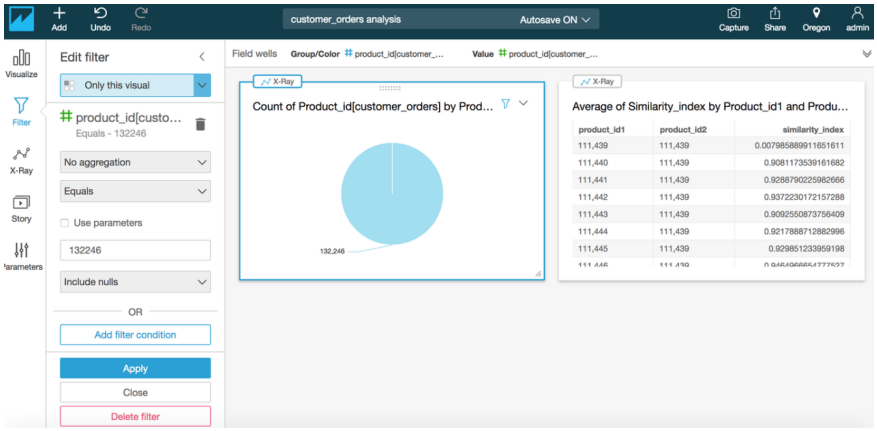
Hover over the small slice of data in the first visual to determine one of the product_ids in the bottom 20 list.



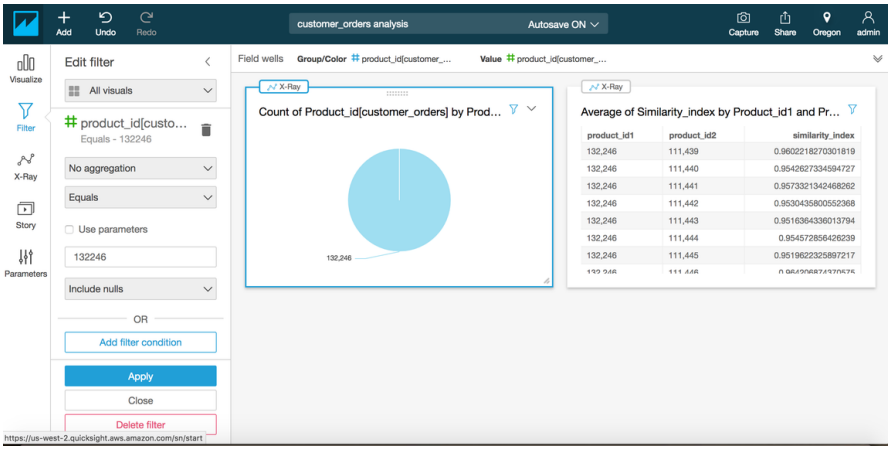
Click on on of the product_ids and click “Focus only on...” after that. You should be taken to the following window:



Click on the filter “#Product_id” in the left hand side pane to make modifications:

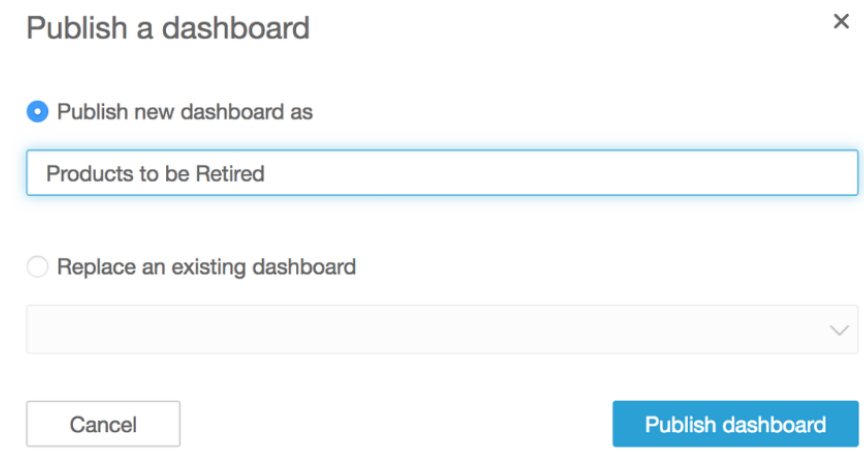


Change the “Only this visual” scroll down to “All visuals”. You should now see both the visuals updated. Then, click “Apply”:



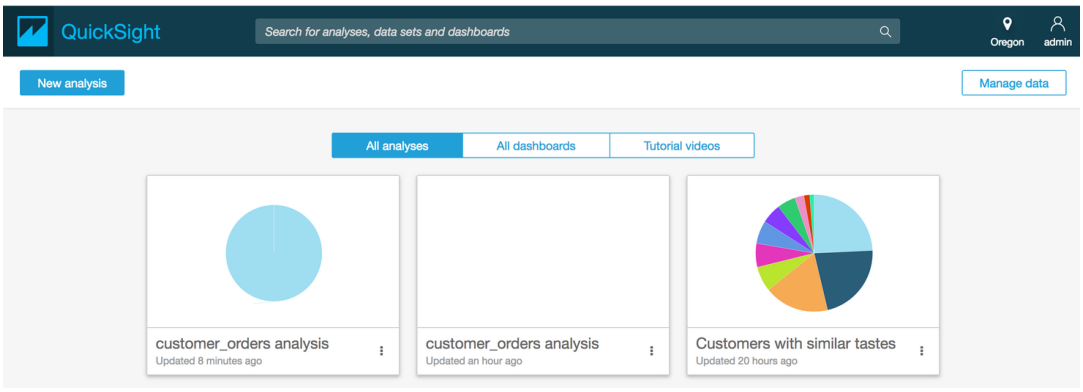
Now you can determine other products that have a similarity scores that are high with this product from the bottom 20. You can then take suitable action on whether to retire the item or not.

You can then click on “Share” on top right hand corner to create a dashboard and share it with users in your account:

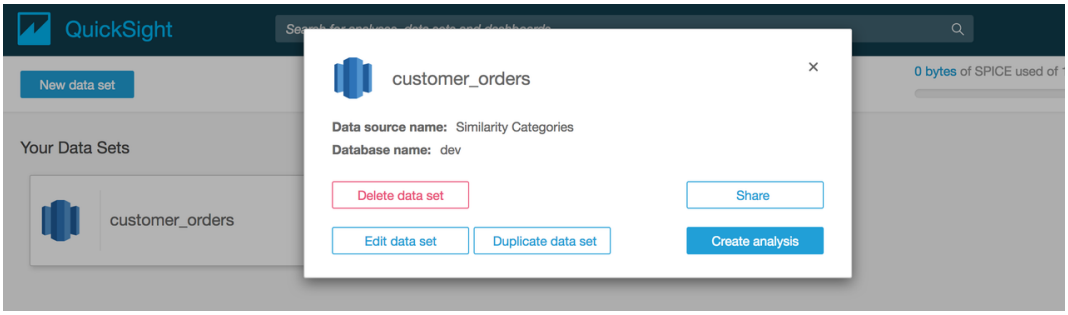


Now, lets try to categorize the data based on the similarity scores. For that, we will use a custom SQL query that will be used to create a dashboard.

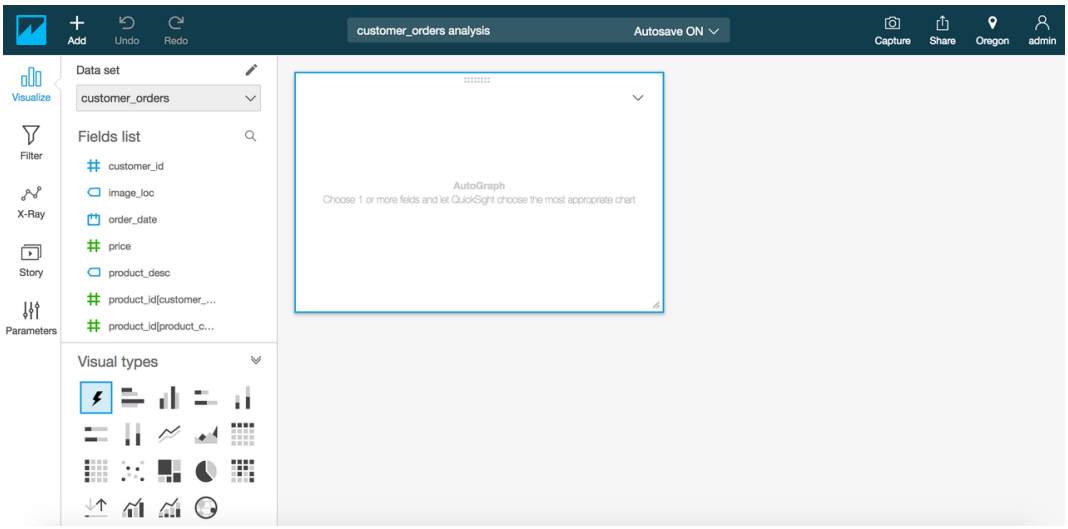
From the QuickSight homepage, click on “New analysis”



Click on the same dataset that was used in the first visualization (in this case, customer_orders):

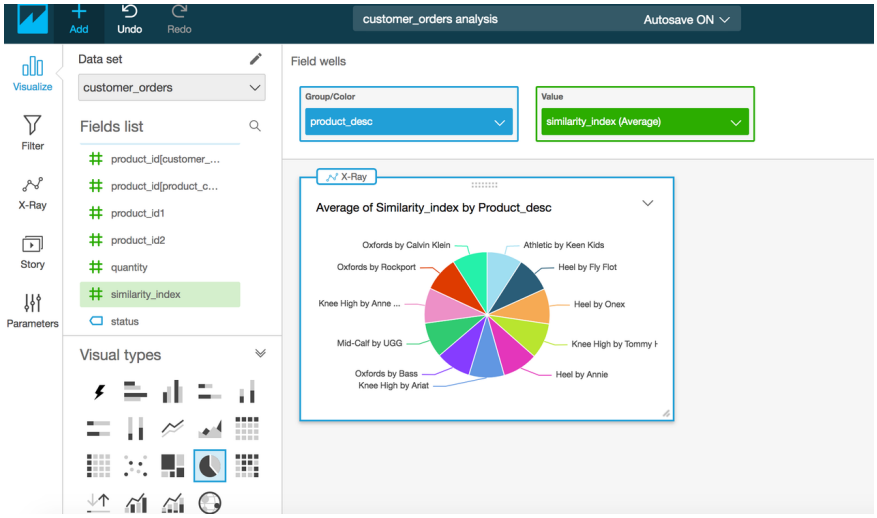


Click on “Create analysis” and you should be taken to the following screen:

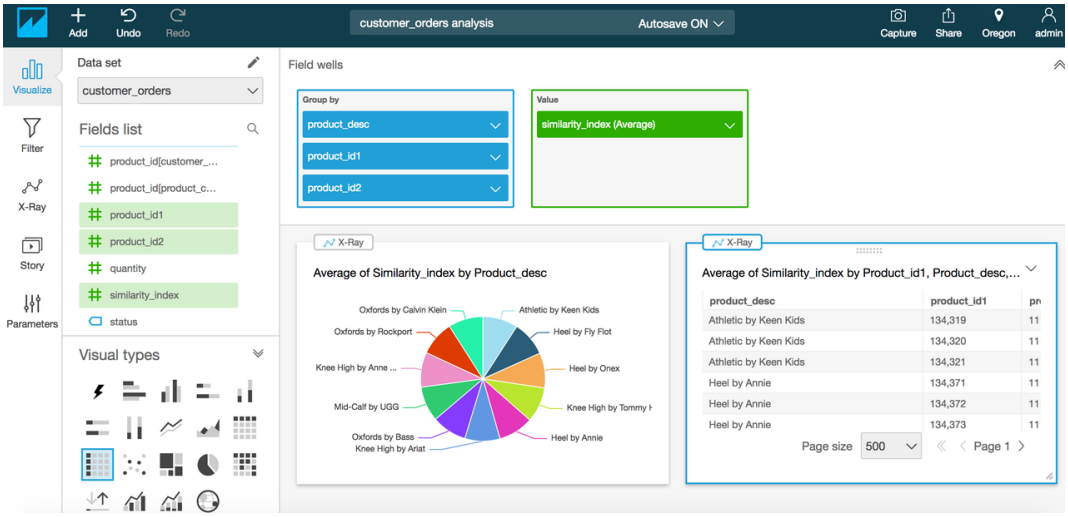


Click on the “Pie chart” from the “Visual type”.

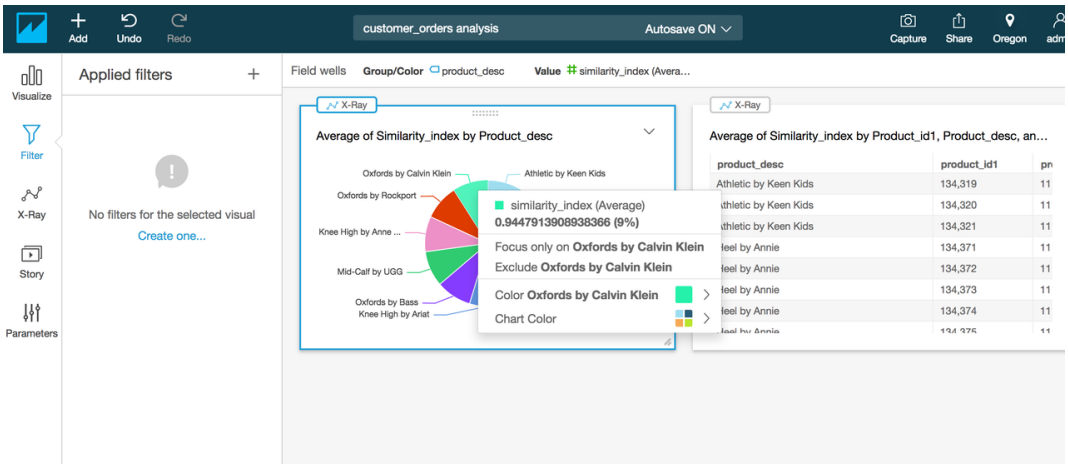
Drag and drop “product_desc” into the “Group” and similarity_index(Average) into the “Value” field. You should see the categorization visual as shown below:



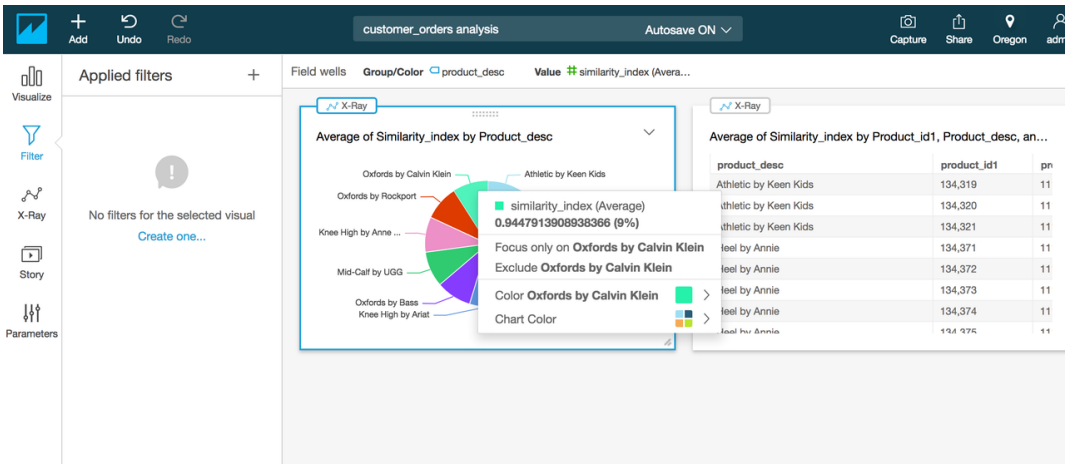
Now click on the "+ Add" and then "add Visual" in the top left hand corner of the screen. Click on the "Table" from the "Visual type".
Drag and drop "product_desc", "product_id1" and "product_id2" into the "Group by" field and similarity_index(Average) in the "value" field. You should see the following second visual alongside the first pie chart:



Now, you can start your analysis by finding the product categories with the least similarity index value. For example, hover over "Oxfords by Calvin Klein" in the pie chart and you will see that it is one of the few categories with a score of < 0.95. Click on that pie and focus only on that category:



You will see that a new filter is created for that value of “product_desc” field. Now click on the filter tab on the left hand panel and select the filter. Then, choose “All visuals” from the scroll down at the top:



You should now see the similarity index values for all the products in the category of “Oxfords by Calvin Klein”. Now you will be able to assess the similarities and take informed decision based on the orders data that you have for the same category.