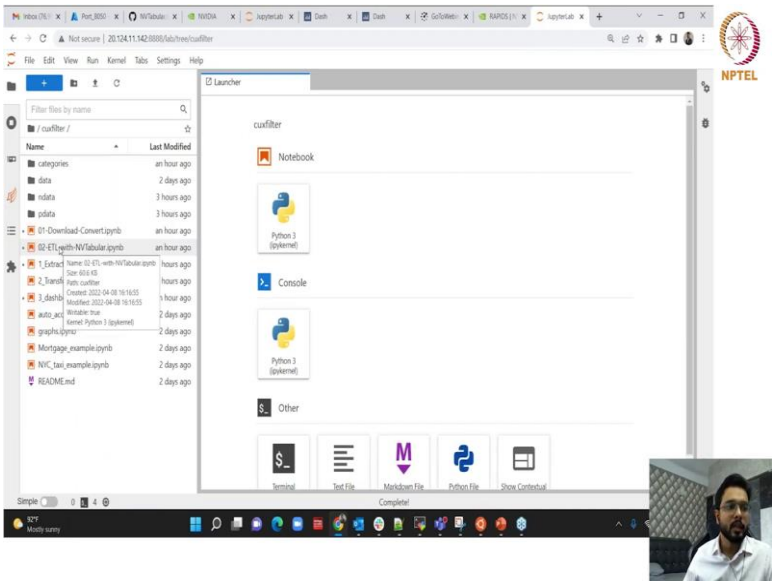
Applied Accelerated Artificial Intelligence
Prof. Saurav Agarwal
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

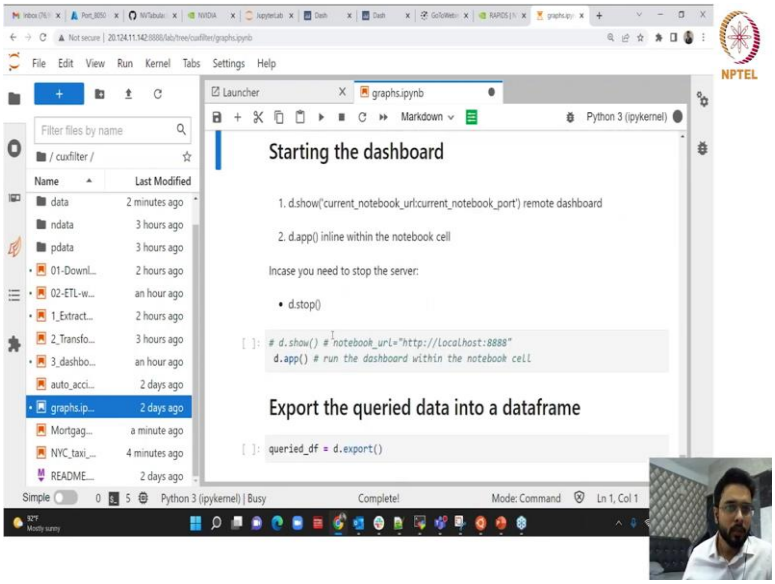
Lecture - 51
Web visualizations to GPU accelerated crossfiltering part 2

(Refer Slide Time: 00:14)



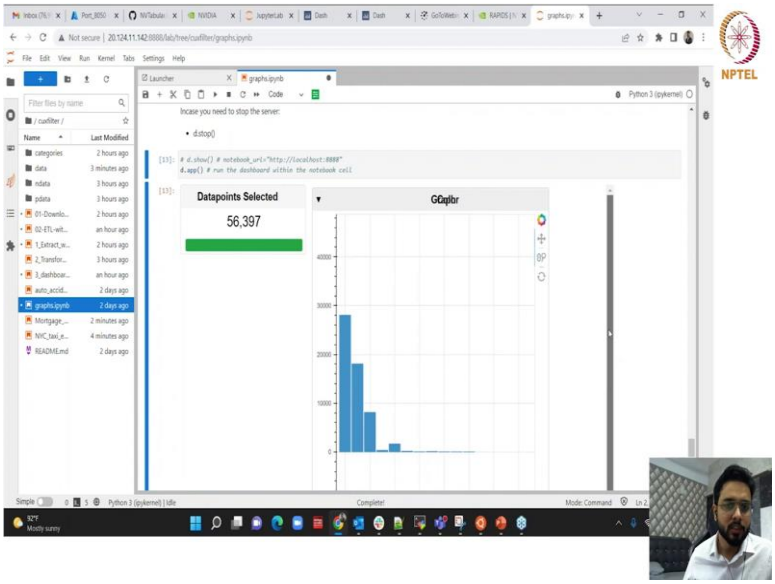
So, this is the Jupyter Lab Environment, where you can do the hands on and here mainly we will be focusing on visualization today and then NVtabular. So, let us go inside this. So, in the visualization you see that already some notebooks are there. So, for example, I will go one of the notebooks.

(Refer Slide Time: 01:24)



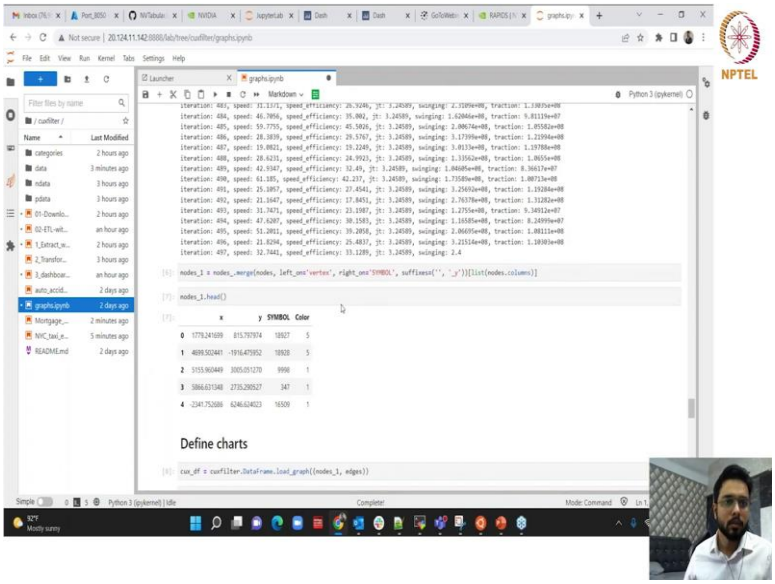
So, for this particular chart we do not need any API key. So, this should work ok.

(Refer Slide Time: 01:56)

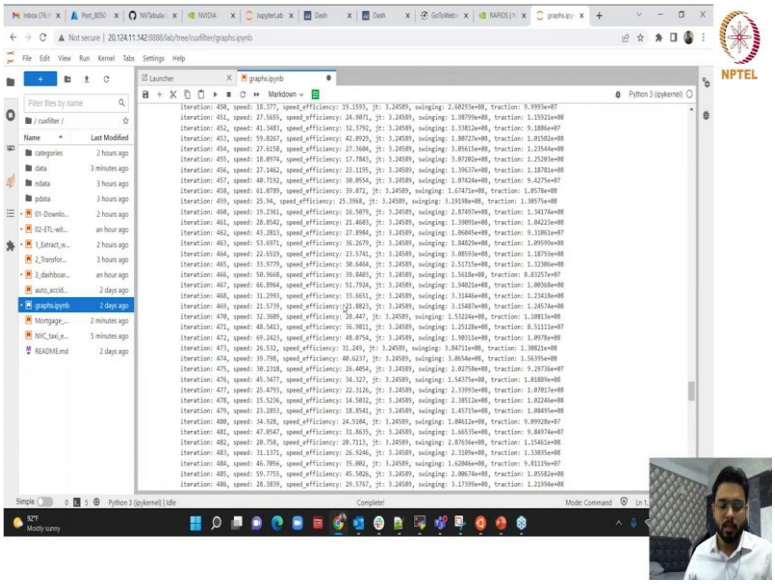


We can see some chart here. So, let me try to show you in full screen, yeah. So, this is how it looks the data point selected, then the graph this is a simple bar chart which is being plotted. So, just ignore this one we do not need to see the preview.

(Refer Slide Time: 02:24)

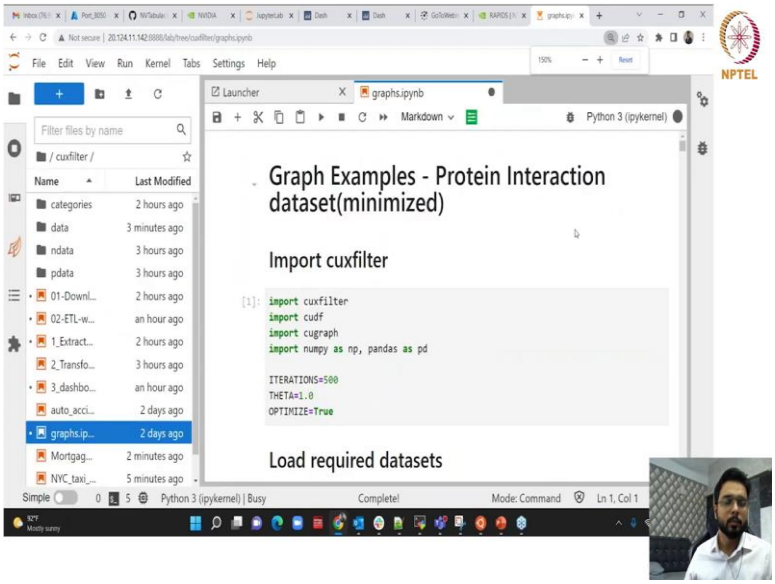


(Refer Slide Time: 02:27)

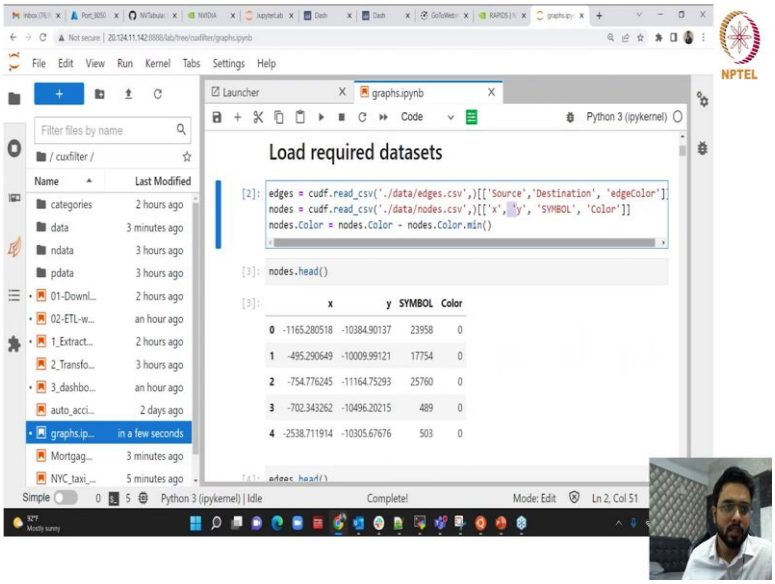


So, just try to show what we did here.

(Refer Slide Time: 02:31)

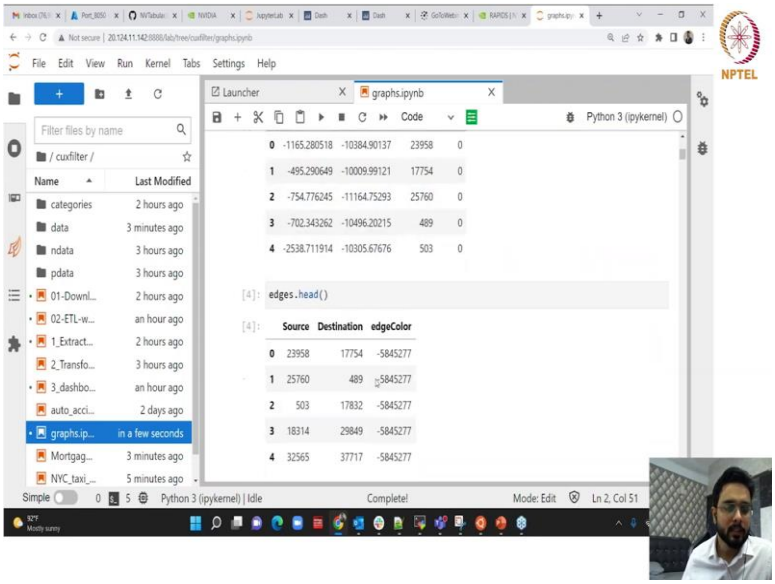


(Refer Slide Time: 02:35)



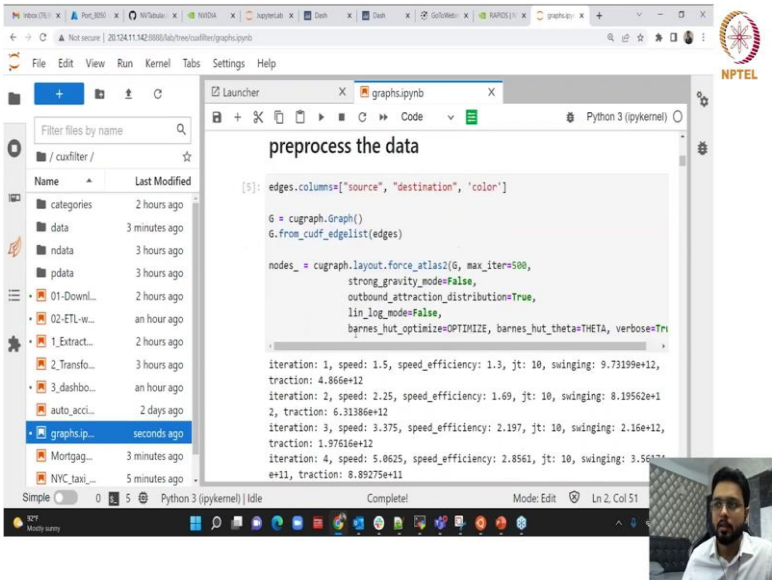
So, it is a protein interaction dataset which is the basically biological data, we have graphical format of data where we have edges, source, destination, edge color and then x, y symbol color as nodes.

(Refer Slide Time: 02:47)



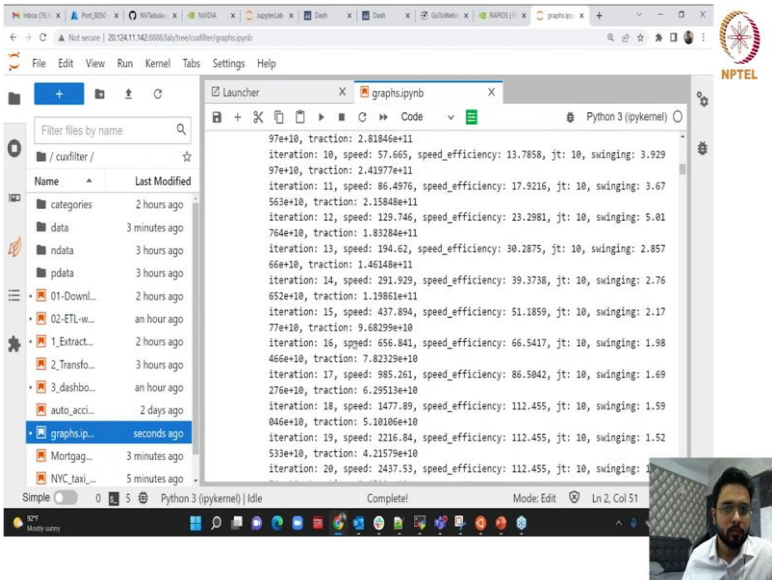
So, this is the dataset we have in hand.

(Refer Slide Time: 02:48)

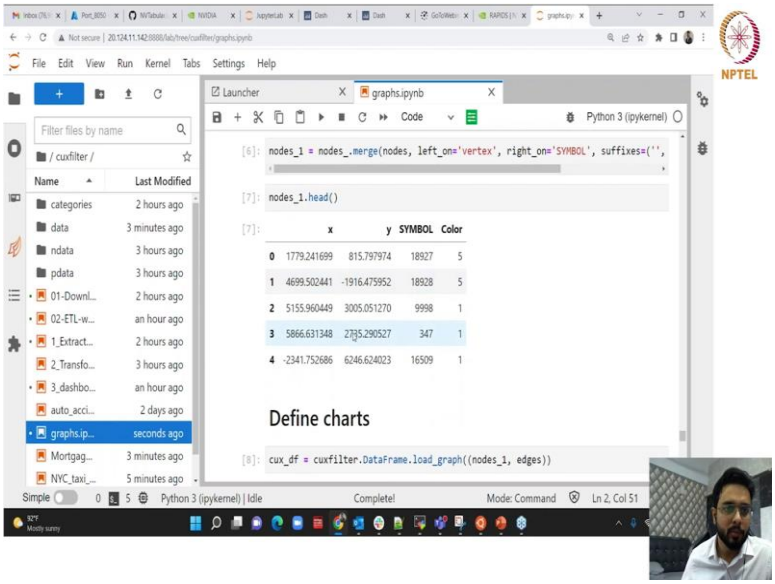


Then we pre process the data using cugraph, we do some basic processing of the data.

(Refer Slide Time: 02:57)

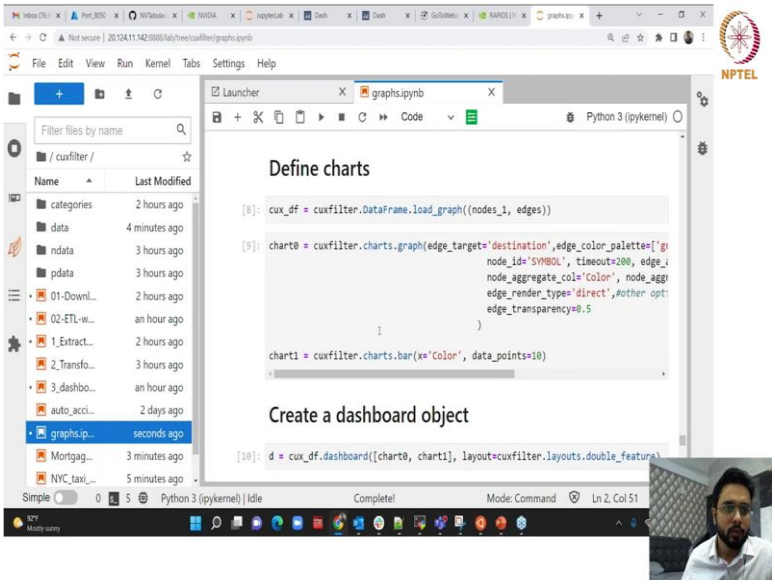


(Refer Slide Time: 03:05)

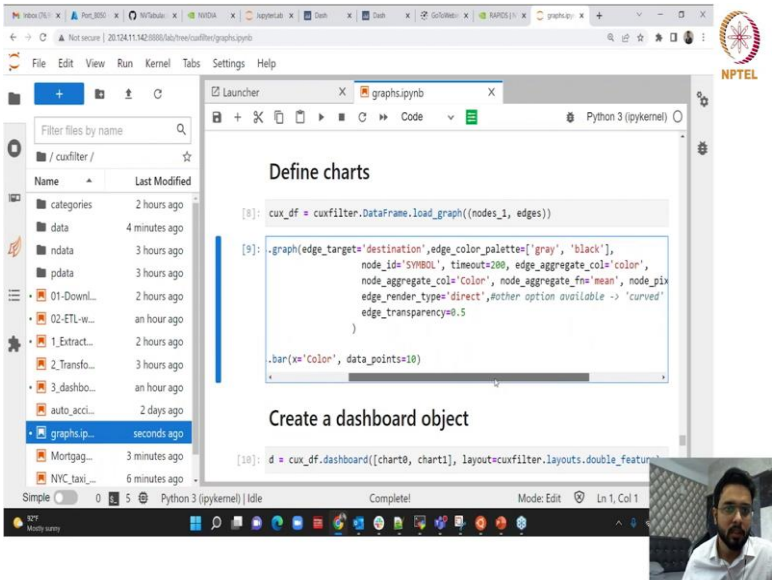


And then after doing all the processing. So, it is not necessary that you use cugraph. So, it is just one of the examples, where cugraph is used. You can use normal cuDF or DASK to do that, then after processing we have this format x, y, symbol and color.

(Refer Slide Time: 03:13)

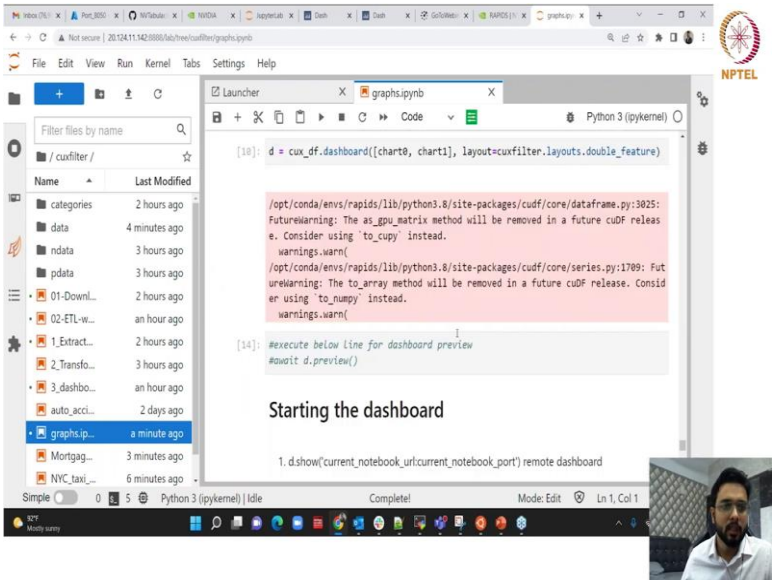


(Refer Slide Time: 03:20)



So, using that we plot a data using graph chart, destination is the edge target, edge color is grey, black node id SYMBOL.

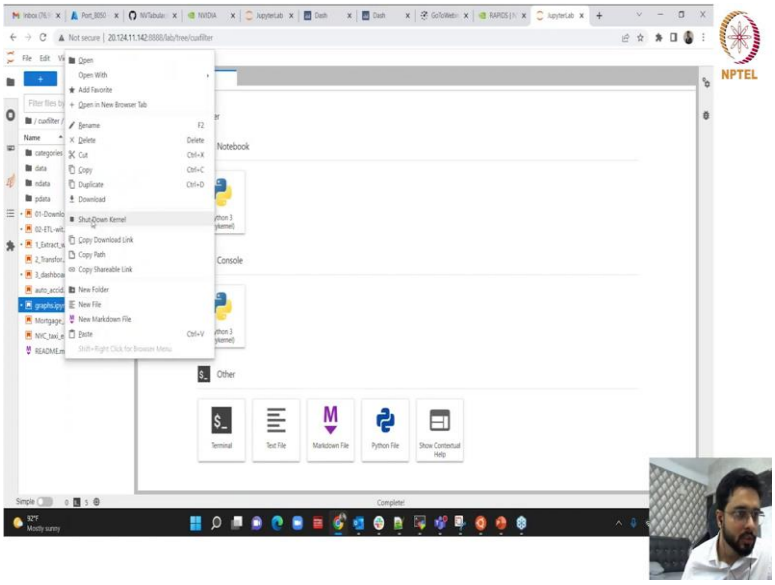
(Refer Slide Time: 03:32)



So, all these details you give to put the chart, again all this parameter values or parameter description you can refer the cuXfilter documentation to understand it in detail and then you can d.app() to run the dashboard within the notebook cell. So, there are 56397 data points and we have this graph created for that.

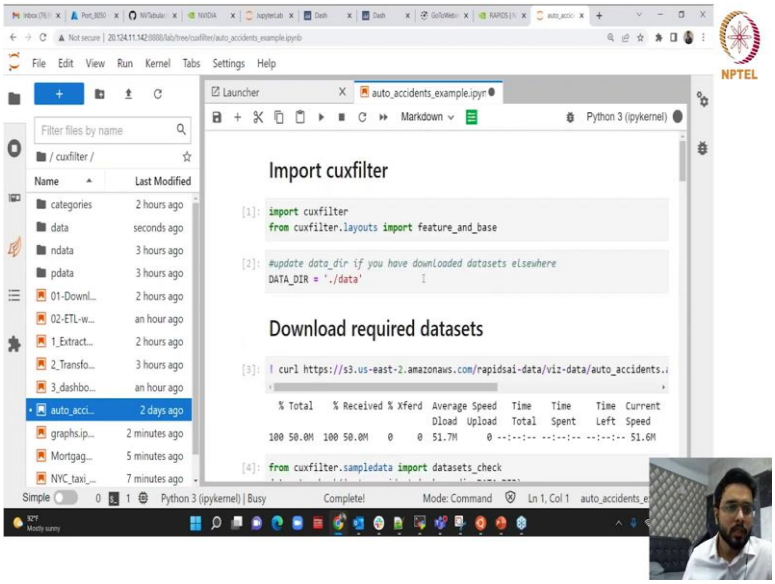
So, here you see this is the x axis, y axis respectively and I will just show one more if possible.

(Refer Slide Time: 04:06)

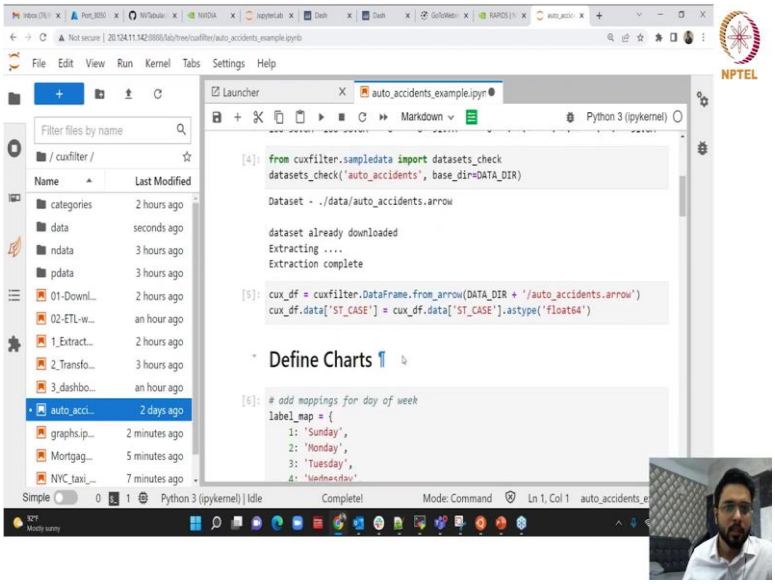


So, I will just close this one to all other.

(Refer Slide Time: 05:06)

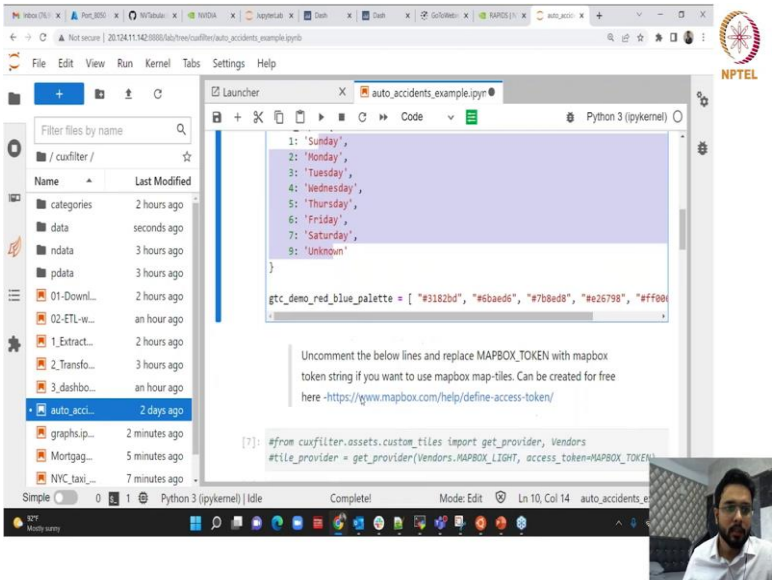


(Refer Slide Time: 05:11)



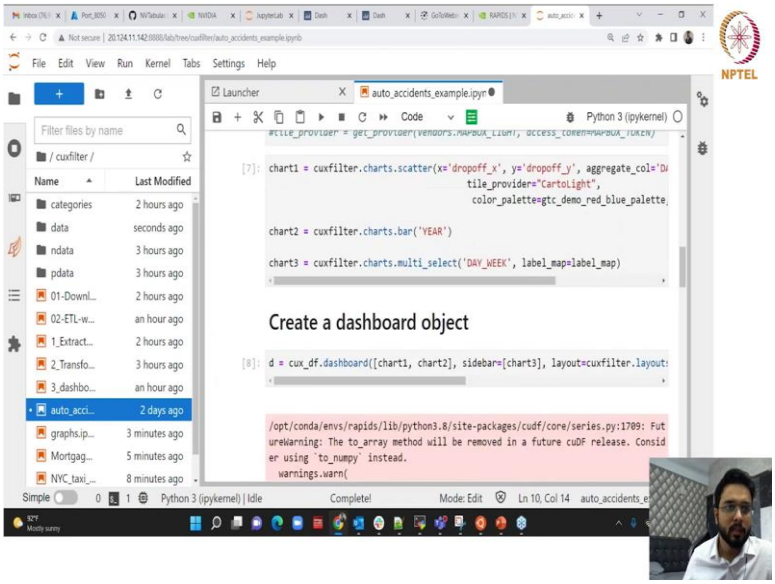
So, this again is auto accident dataset where we are getting all the data reprocessing it.

(Refer Slide Time: 05:14)



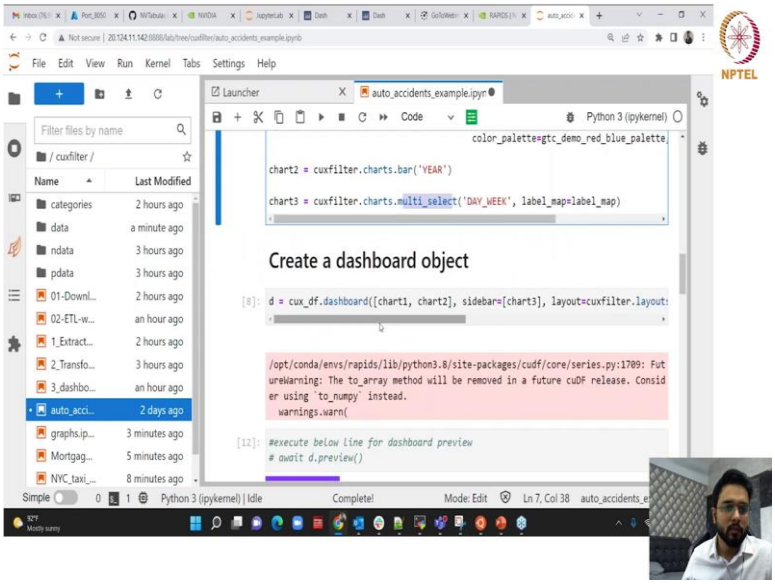
Where we have label Sunday to Saturday.

(Refer Slide Time: 05:21)

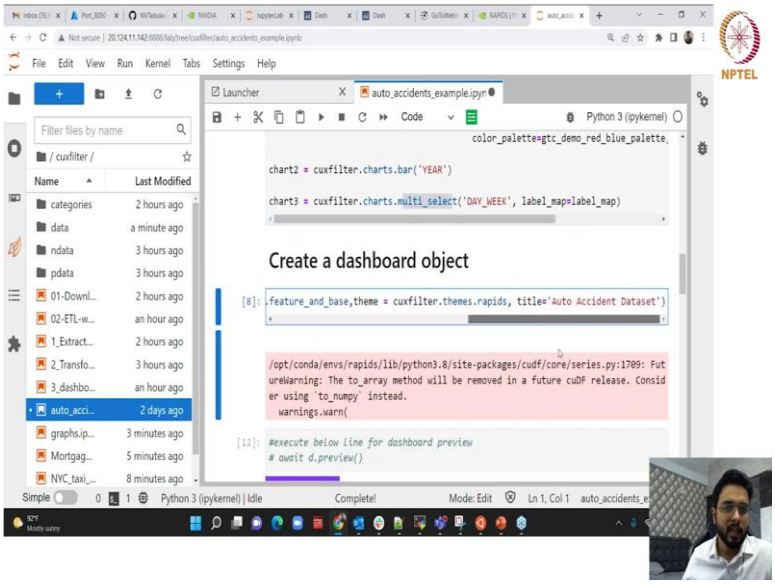


And then pilot, then there are three charts we have scatter chart, bar chart and then we have a multi select. So, we can we are now this time we are not creating only one chart, we are creating two charts and one multi select filter.

(Refer Slide Time: 05:36)

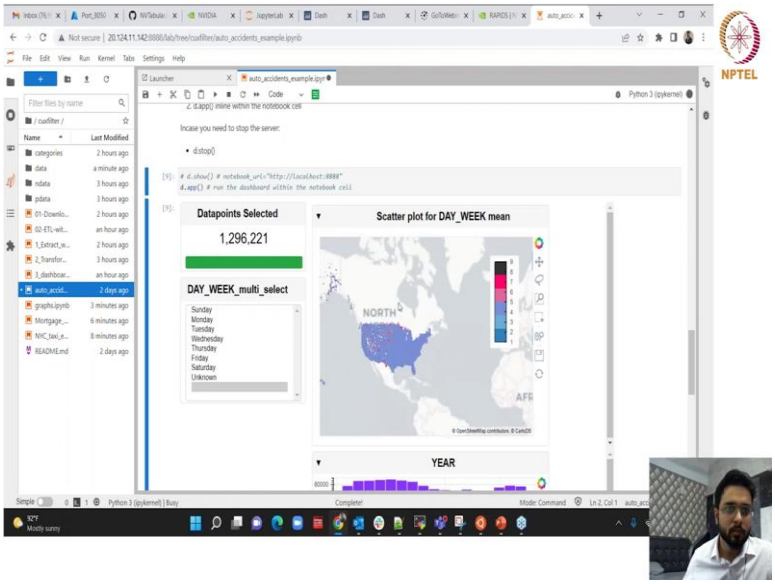


(Refer Slide Time: 05:38)



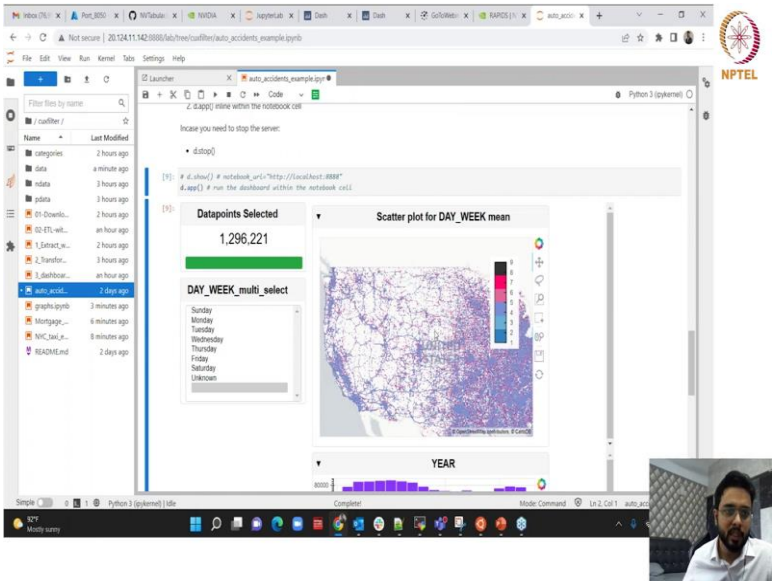
So, and then we are initializing the dashboard, dashboard is a combination of multi select filter and also various charts together.

(Refer Slide Time: 05:48)



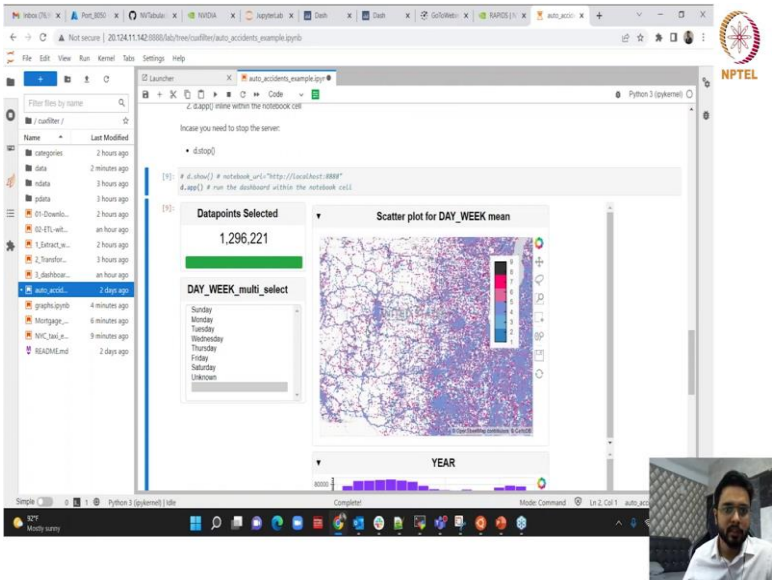
So, here is the chart, if you I will just put it in full screen. So, here is the chart which has been created Sunday, Monday up to Saturday here it is, on the right hand side if you see this is the scatter plot for the auto accident and how many auto accidents happened in which region.

(Refer Slide Time: 06:09)



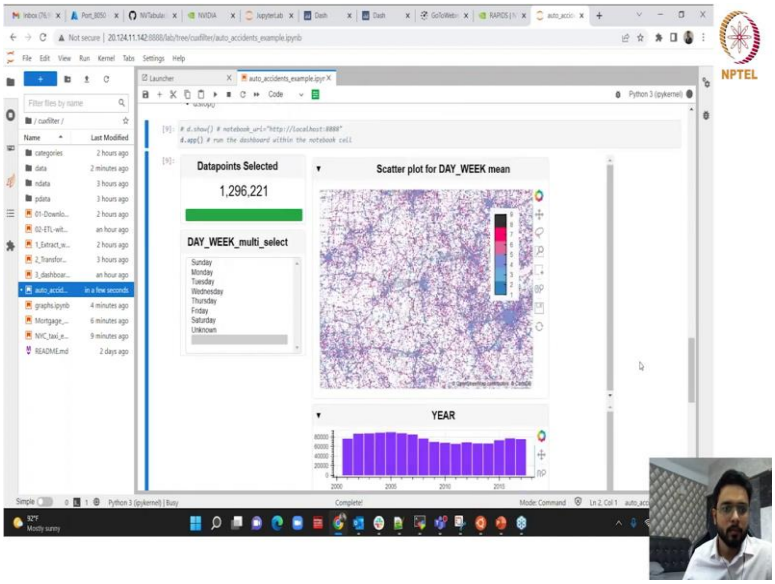
So, if you see the blue ones are the least, why there is no color; that means, no data around that and there where we have the dark color; that means, there were a lot of accidents there.

(Refer Slide Time: 06:23)



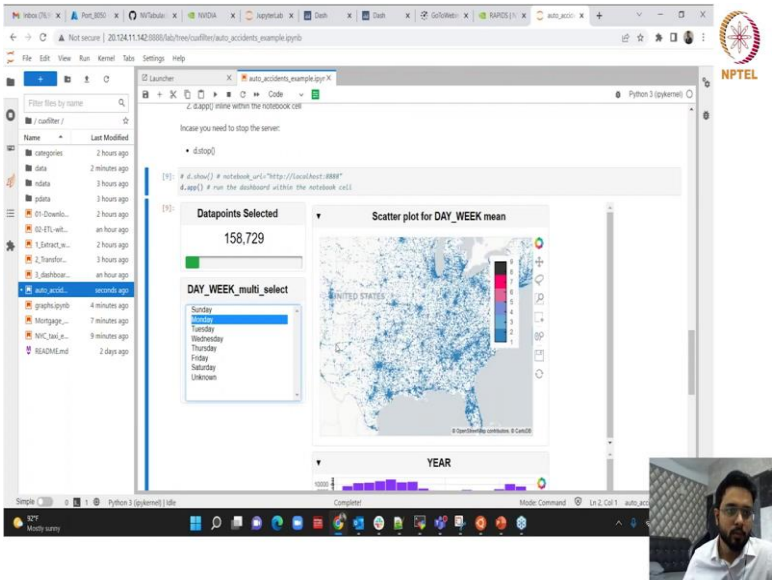
So, this is this is how fast it is seen, how fast it is filtering.

(Refer Slide Time: 06:29)



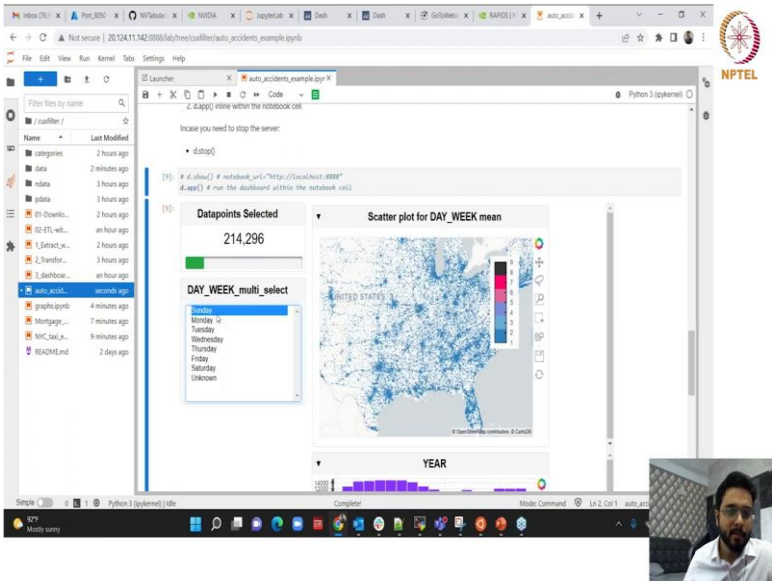
How fast I am able to do the magnification and all the processing, which is which may not be possible for large datasets on CPUs easily. And the other chart is about a simple bar graph which is showing year based how many accidents had happened per year.

(Refer Slide Time: 06:51)



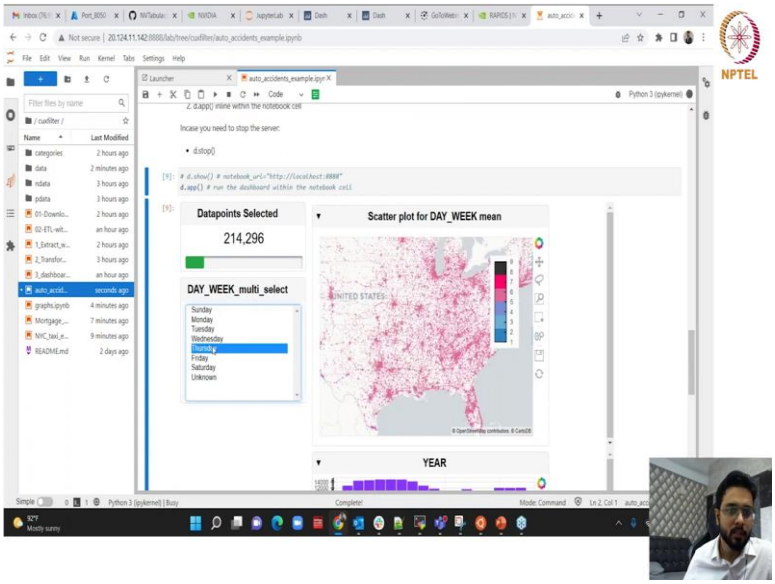
And then, if you want to see the day of the week so we just filter on Monday, the graph will change, see the graph has changed. So, Monday only few accidents happened so it is just 1.

(Refer Slide Time: 07:04)



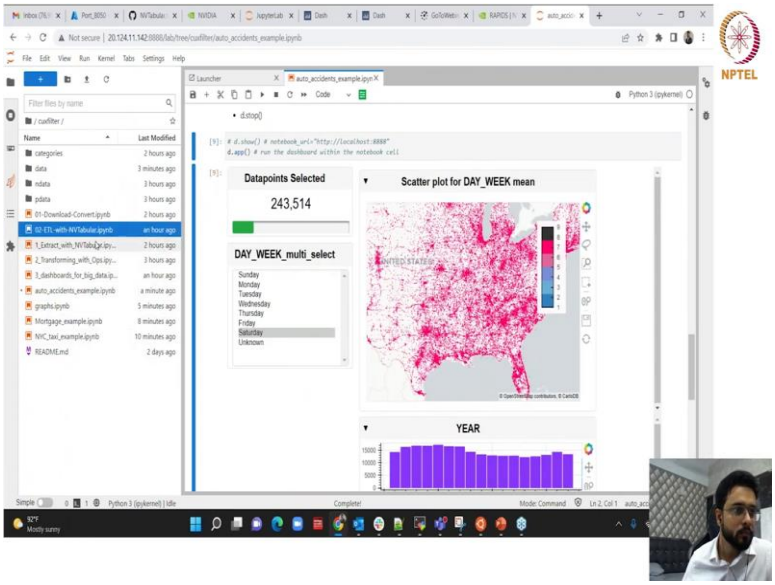
Sunday a little more.

(Refer Slide Time: 07:09)



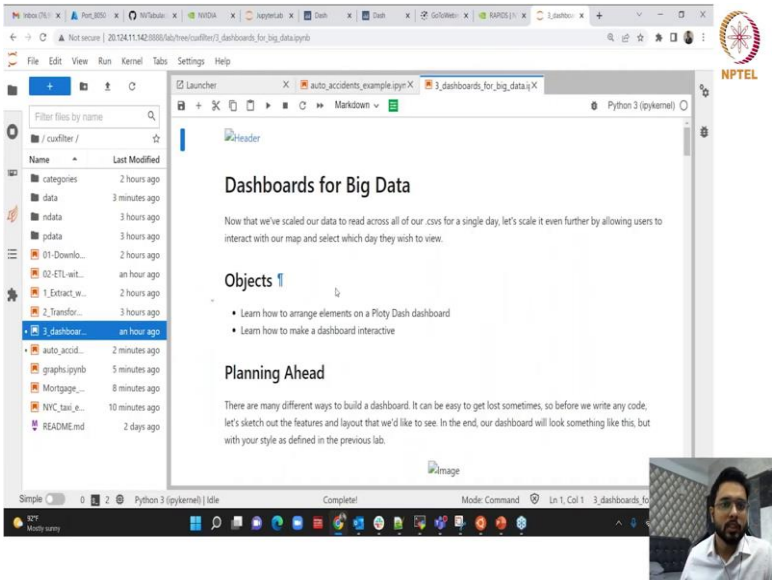
Let us go on Saturday, Thursday, Thursday a lot of accidents. So, you can see based on the day of weeks, the number of accidents are changing it is red; that means, 6 or 7 or something around that, right. This is how good visualizations are to understand the patterns inside the data, alright.

(Refer Slide Time: 07:37)



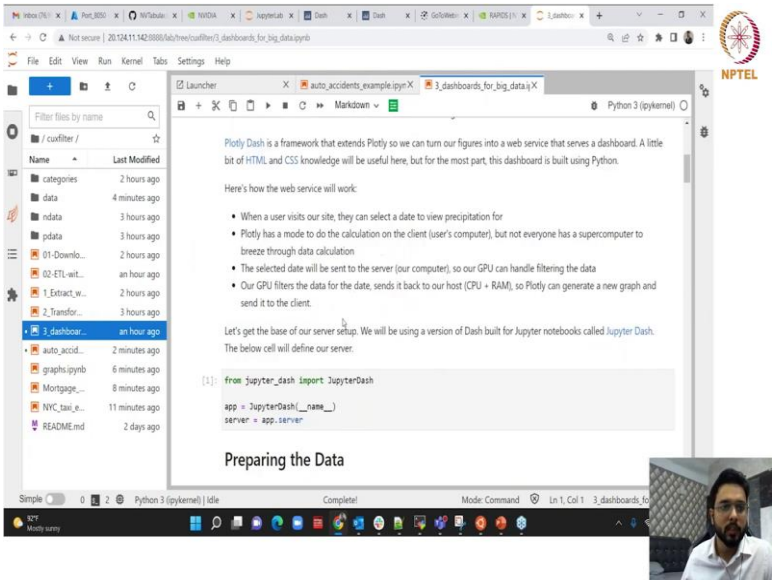
Saturday also there are a lot of accidents ok. So, these are the simple visualizations using cuXfilter. So, if you want to do using plotly also, I will just show you how it works at a notebook level.

(Refer Slide Time: 08:10)

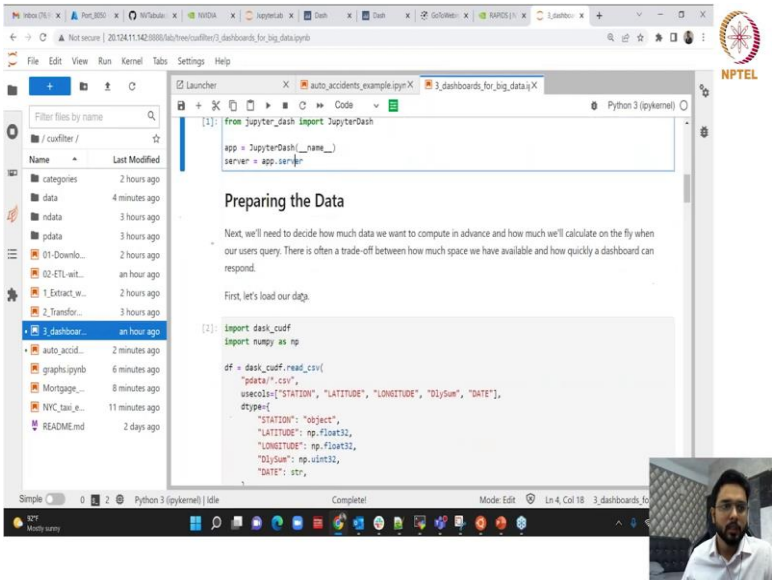


So, this is for using plotly, it is a bit different, you do not have to import cuXfilter.

(Refer Slide Time: 08:19)

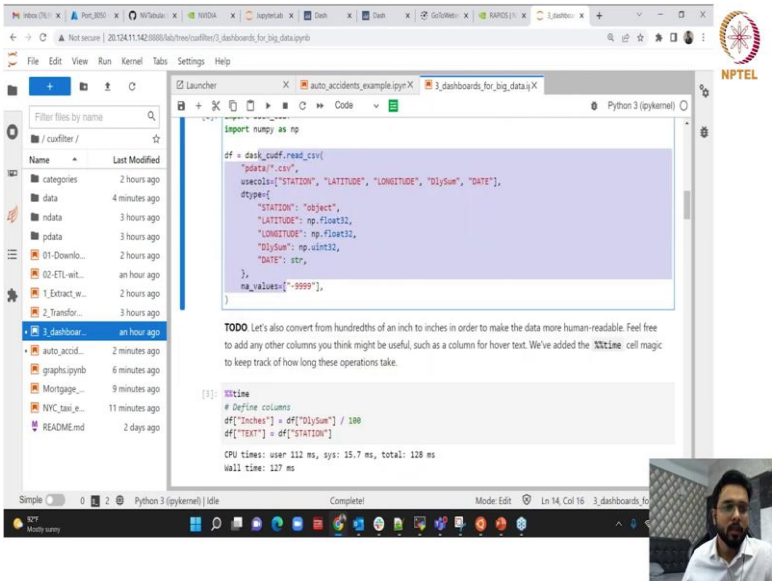


(Refer Slide Time: 08:21)



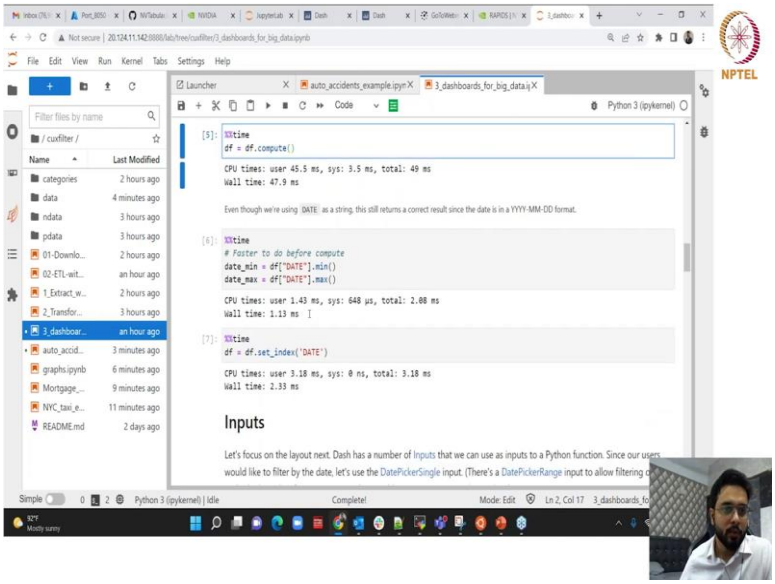
So, first you have to like import Jupyter dash. So, you have to install Jupyter dash, if you have if you do not have it in by default in NGC container docker. So, keep install Jupyter_dash and then you initialize the server.

(Refer Slide Time: 08:36)



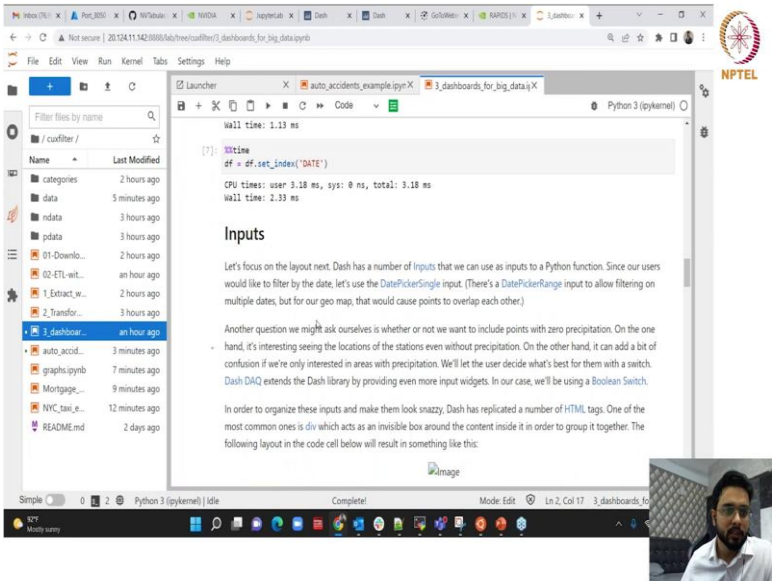
And then we have the dashboard here, where you import the data so where your station latitude longitude. So, basically create a DASK data frame.

(Refer Slide Time: 08:49)



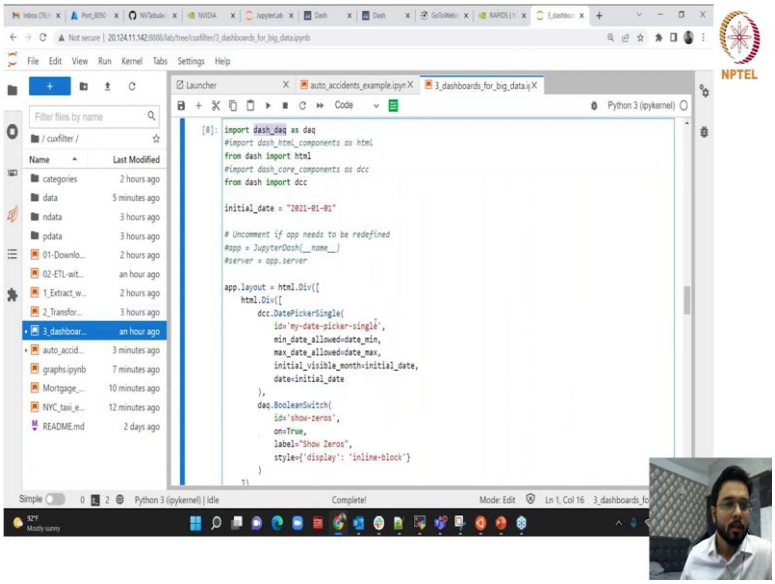
And then basic do some basic transformations and then do compute to get the data frame out of it. So, this is important. So, plotly will not take in us direct DASK data frame as an input, you have to compute it that we get a normal cuDF data frame before you pass that on to plotly, that is what I was seeing it works on the CPU.

(Refer Slide Time: 09:16)



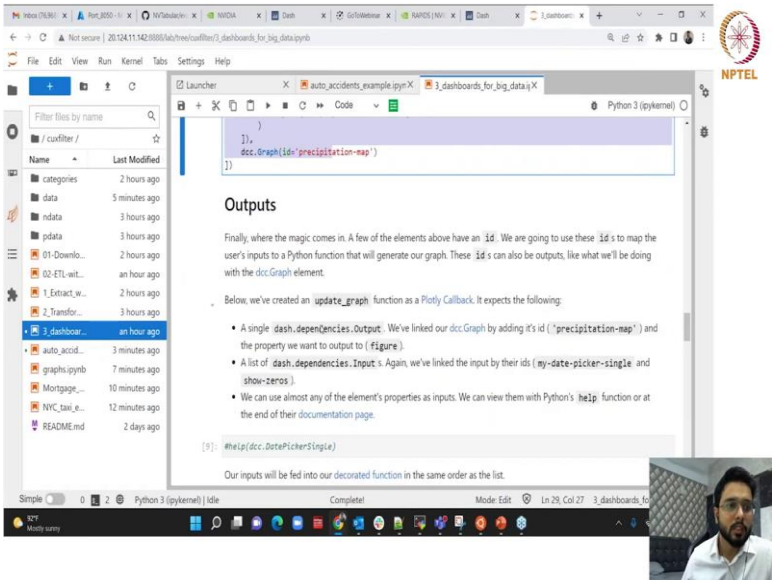
Though the pre-processing and all the back end happens on the GPU, but the plotly takes the CPU data flip.

(Refer Slide Time: 09:26)

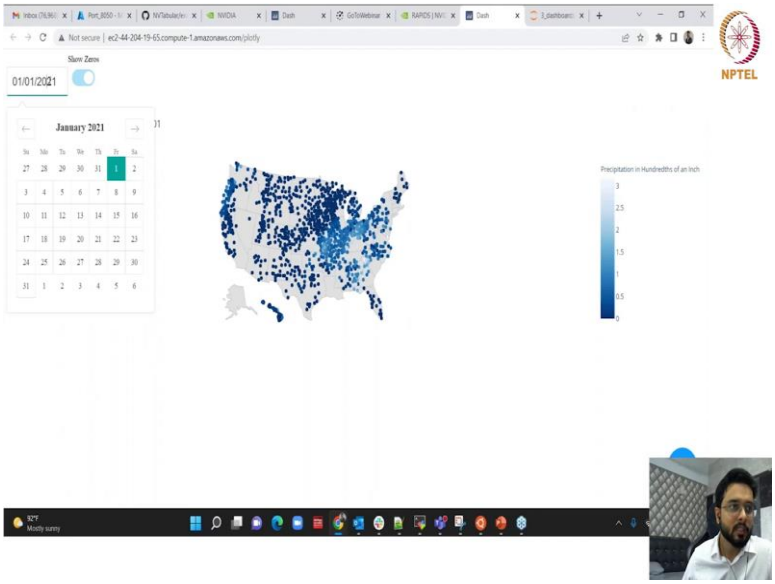


So, here if set index and all is there. So, here the I mean, the plotly dash part start again for dash you have to use dash underscore DAQ. So, Dash DAQ is the extend the Dash library for providing even more input feature. So, it is just more feature full plotly Dash library version. So, after importing you have to create that app.layout which I was explaining you. So, this is html Div, date picker, BooleanSwitch and all that stuff.

(Refer Slide Time: 09:48)

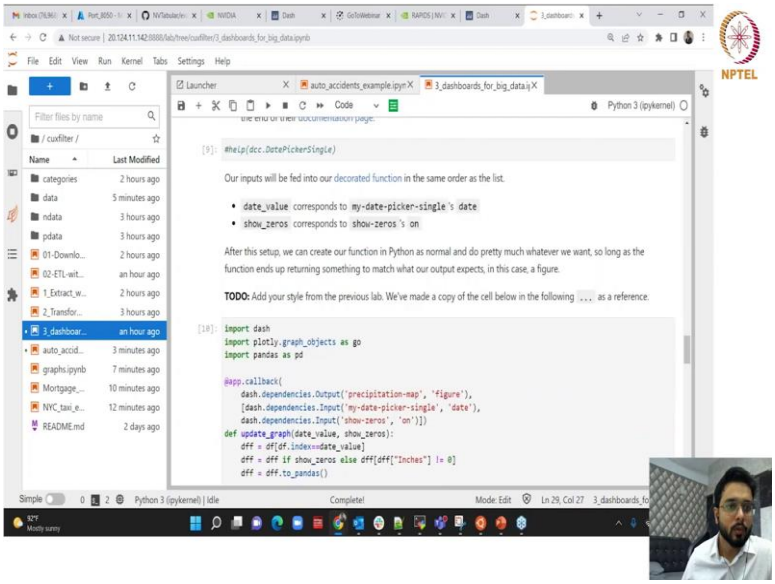


(Refer Slide Time: 10:03)

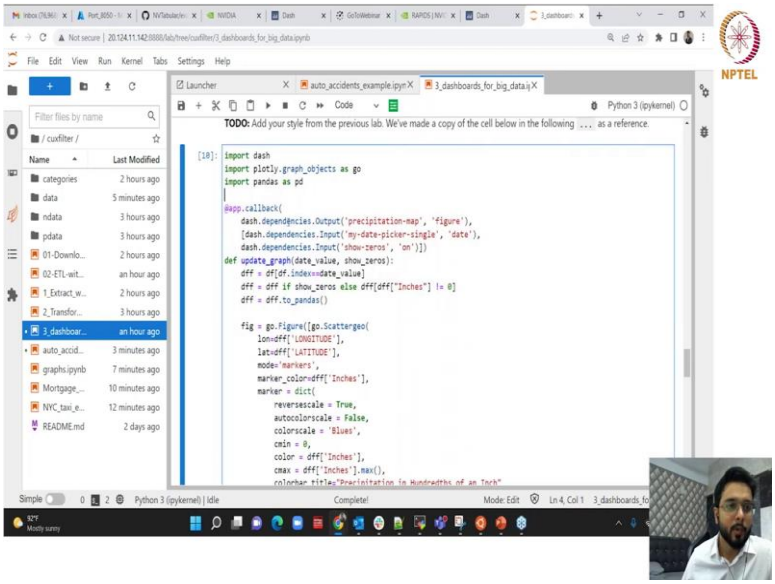


So, I will just show you how it works yeah.

(Refer Slide Time: 09:18)

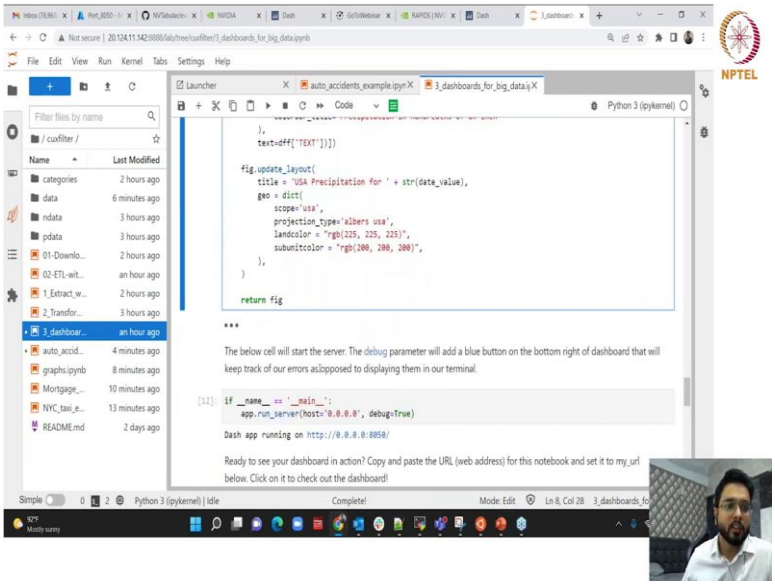


(Refer Slide Time: 10:20)



So, after you run all this then you create app.callback, which I was showing in the slides, where you pass the precipitation map, date picker and show-zeros elements of the dashboard and then you host that ok.

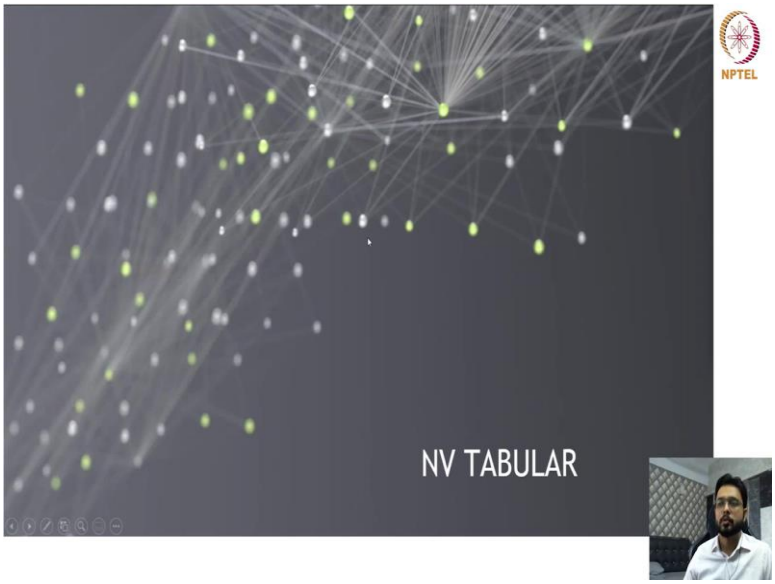
(Refer Slide Time: 10:29)



And then if you see, you want to see that how it looks, it will look like this. This is the date picker, this is the show-zeros and this is the precipitation map that ok, what is the amount of precipitation, 0 is the lowest, 3 is the highest.

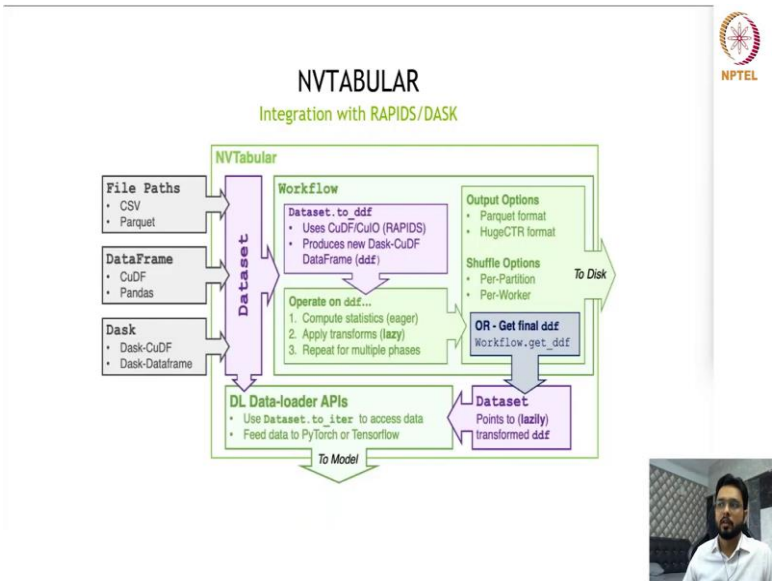
So, light we here the precipitation was a lot and the dark was, dark means its it was a relatively less. So, if you see this, this is like a new window altogether; that means, contrary to the other though cuXfilter can also be created in a new window. So, it is very similar to cuXfilter as well.

(Refer Slide Time: 11:19)



Now, going to another framework called NV TABULAR.

(Refer Slide Time: 11:35)



So, NV TABULAR is again very popular framework to do data pre-processing. So, you know that ok you have been doing data pre-processing using RAPIDS, cuDF or DASK

ah, but again using cuDF and DASK you have to write all the functions one by one like you do in pandas programming or you say sklearn programming or DASK programming. However, if you want to do pre processing in a shortcut manner by saving a lot of time.

Because we know that in a particular data science project, there are few numbers of operations, a few numbers of models which are more required ok. So, for example, if you see this NV Tabular architecture. So, what it experts is like file paths can be their CSV based path or parquet file path, super CSV and parquet are the data formats data file formats. It can have cuDF as input it can take pandas data frame as input or DASK, cuDF as input or a DASK data frame as an input.

And then we have all these inputs can be created to create something called data set NVTabular data set. And using this NVTabular data set we create something called workflow which is used to define the desired data transformation pipeline. So, workflow is something which defines the, what kind of transformations you will be doing, what kind of processing you will be doing and all that stuff.

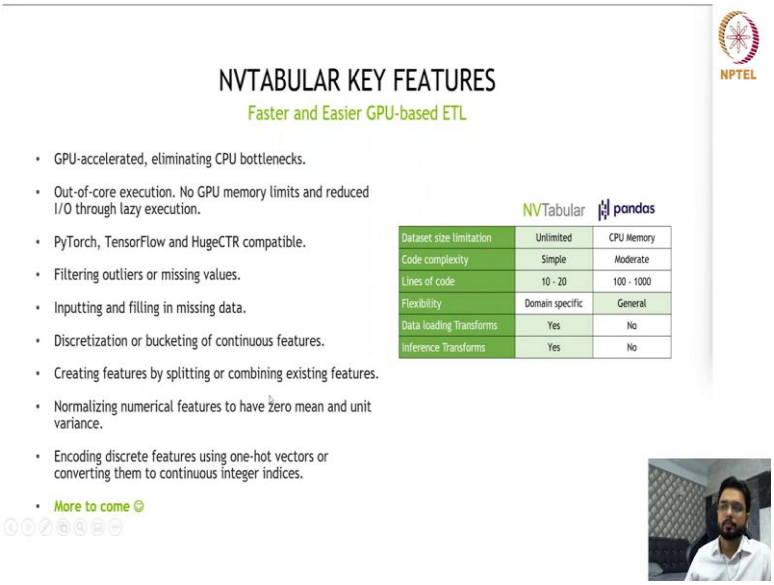
And then we have the data loader part, where we used to feed into a tabular data source to a deep learning based model. So, no matter it can be a Keras based you can create some neural network model or LSTM or something like that. So, you can create using the load the data or convert the DASK data frame or a normal data frame into a TensorFlow data or a PyTorch data or something like that using the deep learning data loader API of NVTabular.

And then we have the other things like we have output options, if you just want to load the data into a deep learning model just you want to make it as output, you can use our output as a parquet format or a HugeCTR format. So, HugeCTR format is a format which accepts which you can do recommendation system.

So, if you want to create a recommender system using HugeCTR model, you want to train it then the data can be exported directly to a HugeCTR compatible format. Then we have like we have various operations on the data frame possible. Like for example, if you want to do compute stats like what are the maximum minimum value, aggregate values and all that stuff you want to apply transformations.

Like maybe bucketing, maybe normalization and all that stuff you can do that or using very shortcut, like shortcut methods of NVTabular. So, basically NVTabular accelerates further your development process of data science pipeline.

(Refer Slide Time: 14:48)



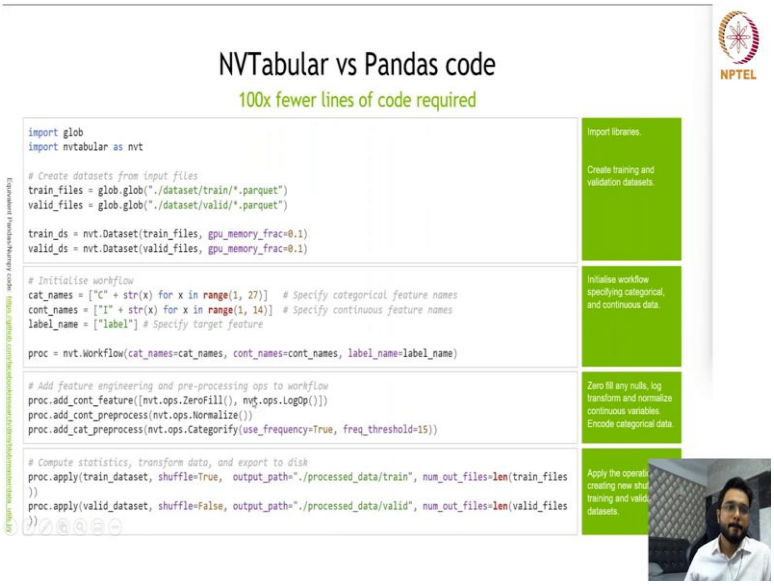
Key features are its completely GPU accelerated, it supports out of core execution; that means, the data volume is more then also it will not fail then the GPU memory, then it supports PyTorch, TensorFlow, HugeCTR. It filters the outliers or missing values, it helps to do the transformations.

So, all these transformations are possible, filtering out layers missing value removal, then we have input the and, filling the missing data discretization of bucketing, creating features by splitting and exist combining the existing features. So, merging and all that stuff. Normalizing numerical features to have zero mean and unit variance, then encoding discrete features using one hot encoding or converting them to continuous integer indices.

And there are more and more which are coming in every release and there is a whole list of operations which are possible, which you can see in the documentation again open source. Comparison from NVTabular to pandas, even cuDF data size limitation it is based on CPU memory, but here it is unlimited, code complexity is very simple. So, this is one of the biggest thing that only 10 to 20 lines of code will be needed as per 100 to 1000 lines of code in pandas.

Flexibility is domain specific; that means, if you want to do in the retail based recommender system or if you want to do some other forecasting model, it will be different set of operations it is domain specific transformations are available. Data loading is possible to deep learning model which is not possible in pandas. Then inferencing; that means, if you want to input the data while predicting the data and do some real time pre-processing and then do the prediction then also it is possible using NVTabular.

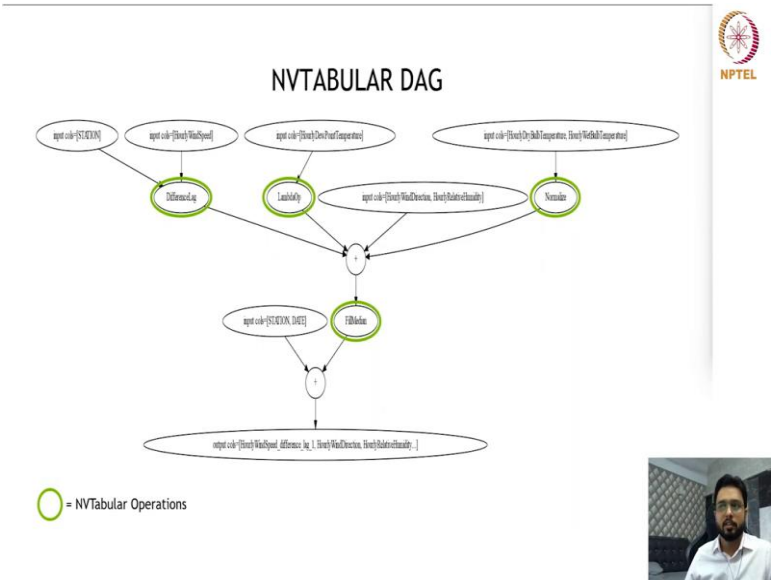
(Refer Slide Time: 16:42)



So, if you see let us understand the code example that how easy it is to do it. So, on the left-hand side if you see this is the NVTabular code, where we import nvtabular import the files then create the nvtabular data set, which I was explaining. Then we create the category names so category features, then create the workflow and in the workflow we do some feature engineering in pre-processing like continuous features we do the zero filling or the log operations.

We do the for the pre-processing we do the normalization. For categorification; that means, if you want to change numerical values into categories you can do that. So, using categorify and then apply all this on the training and validation set and then pass it to the model. So, this is how easy it is. So, entire code is fitted into this PPT, hence you can understand how easy it is to do the data science workflow using NVTabular.

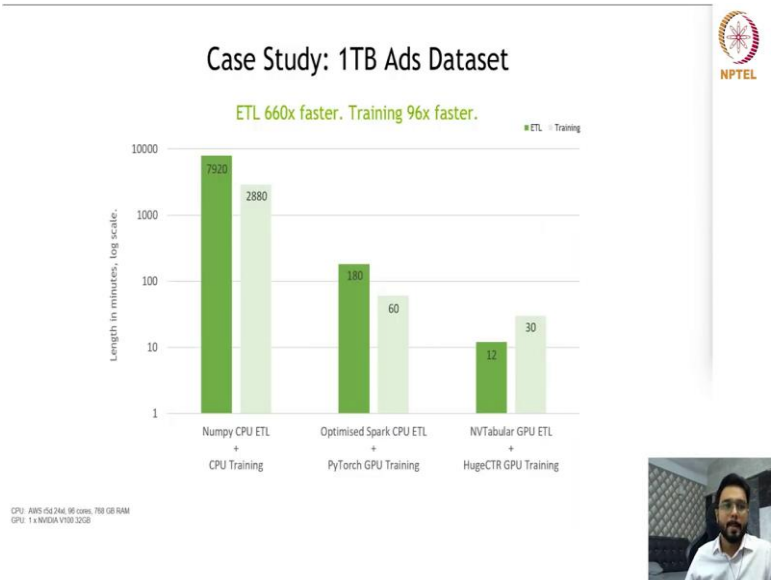
(Refer Slide Time: 17:52)



So, it creates a DAG like this. So, which is the cyclic graph; that means, you input the column, station, you input the column hourly wind speed. For these two columns if you want to do one kind of difference logging operations, for another column you have to do the lambda operation lambda operation means that any custom function you can write, like the lambda function of Python. And then if you want to include some other columns and you want to normalize for that, so all that can be done.

And then finally, you can get the output columns. So, these are the NVTabular operations, difference logging, lambda operations normalization fill missing values with median values. So, all this the, NVTabular has a graphing tool as well to you can see the DAG that is called Graphviz, that I will show you in the hands on ok.

(Refer Slide Time: 18:44)

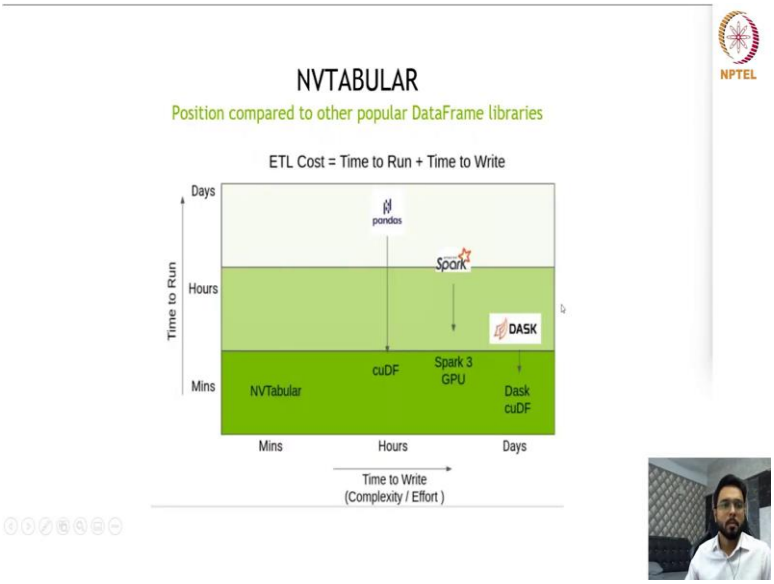


So, if you see like this particular slide, where we have done some benchmarking. So, apart from the ease of writing the code, how fast it is so the another biggest advantage is it is 660x faster based on the 1 TB Ads dataset case study we did.

So, we took the open source Criteo data set and we did ETL and then model training using HugeCTR. So, we compared with Numpy CPU ETL versus CPU based training and then we also use frameworks like Spark, which is very scalable and multi node framework. And then PyTorch, PyTorch GPU training and then we compared with NVTabular GPU. And we use similar like from the cost perspective, similar costing machines.

It is not that we use a very costly GPU machine but we use a very cheap CPU machine not like that it was very comparable still we got this kind of better performance out of it. So, just the pre processing run for 12 minutes versus 180 minutes in a Spark, 13 minutes versus 16 minutes in a Spark the model training.

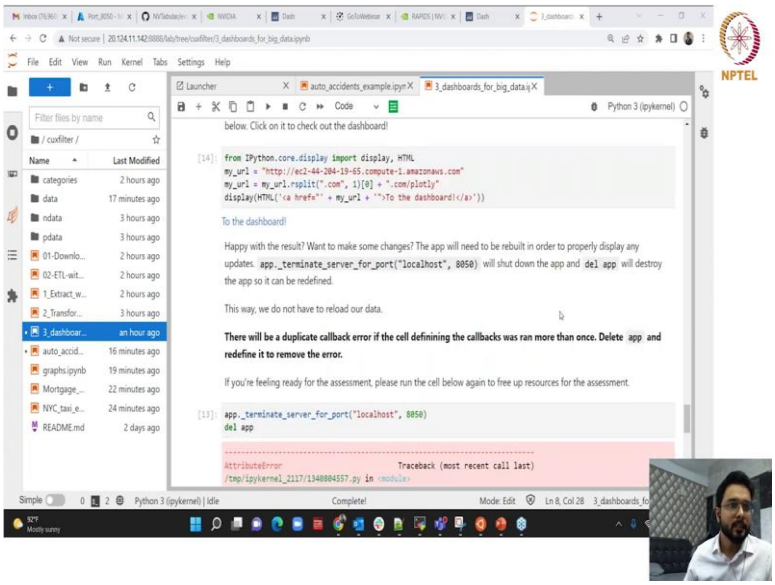
(Refer Slide Time: 20:04)



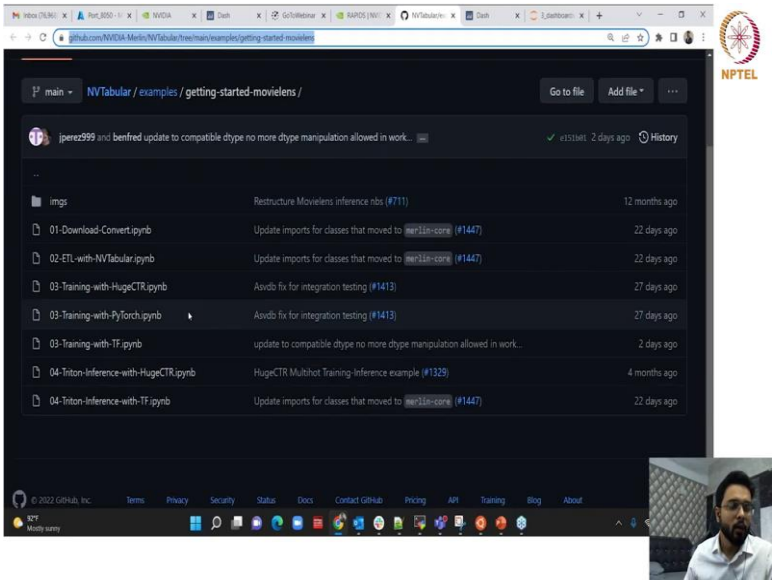
So, this is the how the performance compares. So, NVTabular is one of the fastest has comparable to DASK, cuDF. Then second fastest is that Spark GPU and then we have the cuDF and finally, the slowest is pandas.

So, this is the performance aspect of it. Let me show you some hands on, before I move to the next part.

(Refer Slide Time: 20:57)

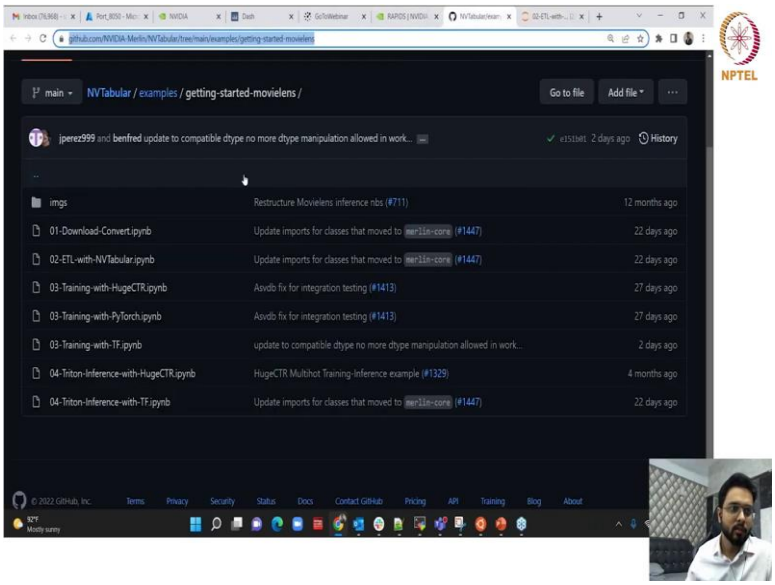


(Refer Slide Time: 20:58)



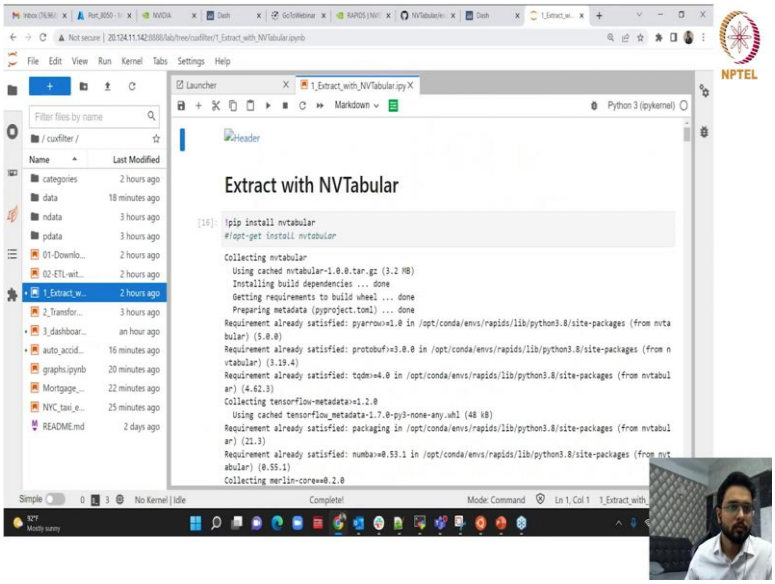
So, for this hand on, I just went to the open source NVTabular repository getting started movie lens, particular folder.

(Refer Slide Time: 21:07)



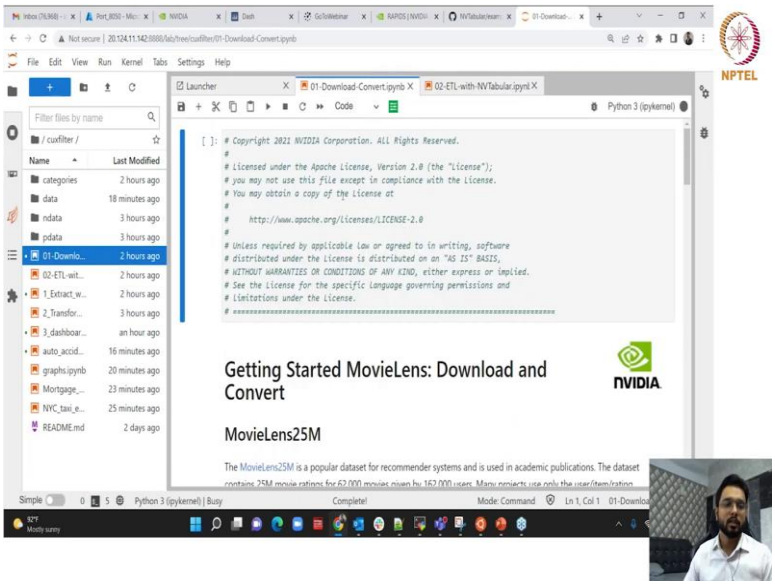
And then I downloaded these two notebooks, download convert ipynb and ETL with and NVTabular. So, again this is open source.

(Refer Slide Time: 21:18)



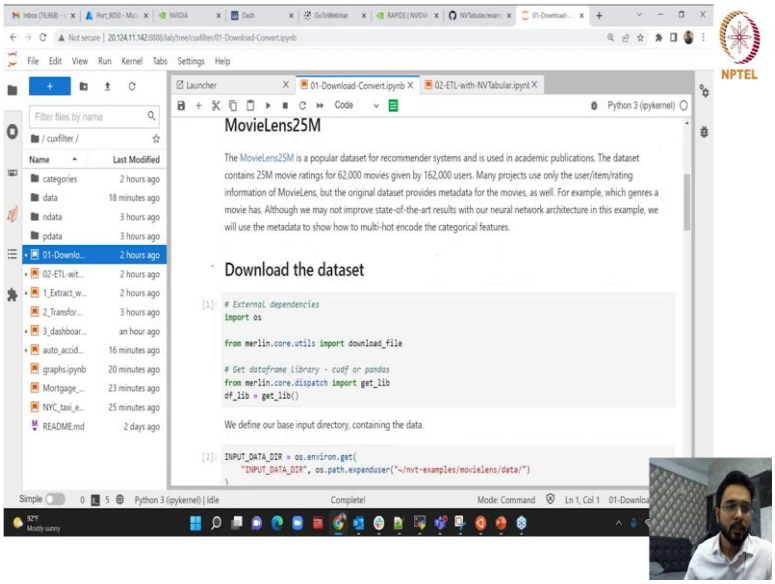
So, before that so, I will just open that folder. So, there are two 01 download convert and 02. So, I will open that 01 and 02.

(Refer Slide Time: 21:37)

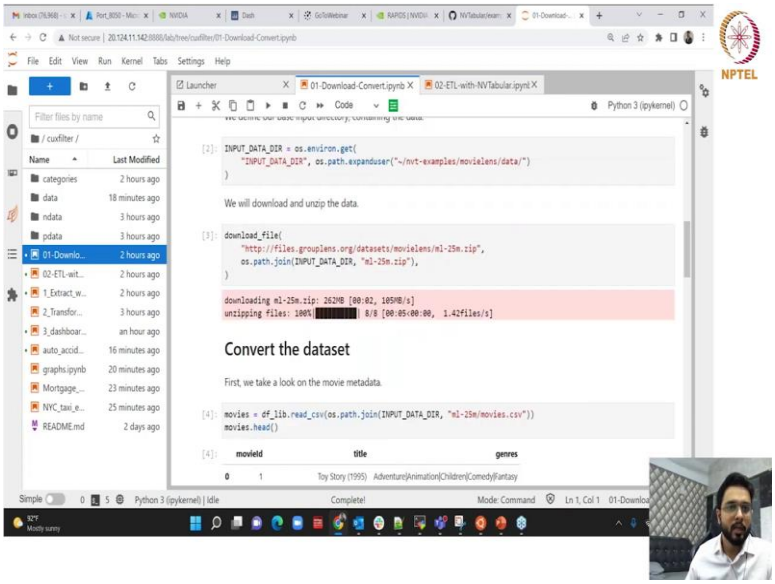


So, in the 01 we will download the movie lens data set.

(Refer Slide Time: 21:40)

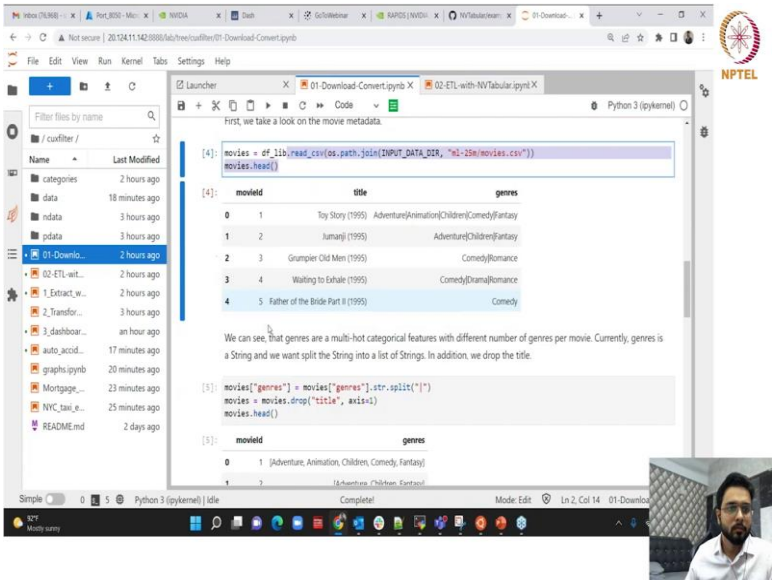


(Refer Slide Time: 21:41)



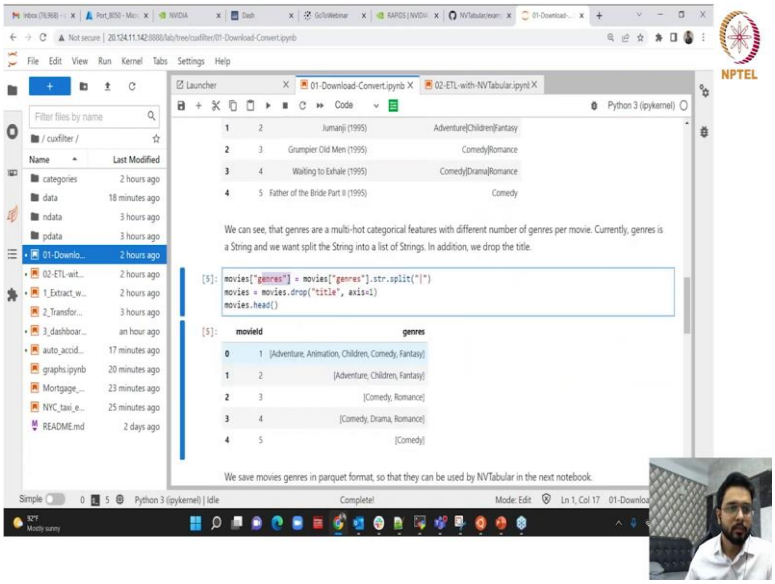
So, here it is. So, if you see here, we download the data set merlin core dispatch get lib.

(Refer Slide Time: 21:54)



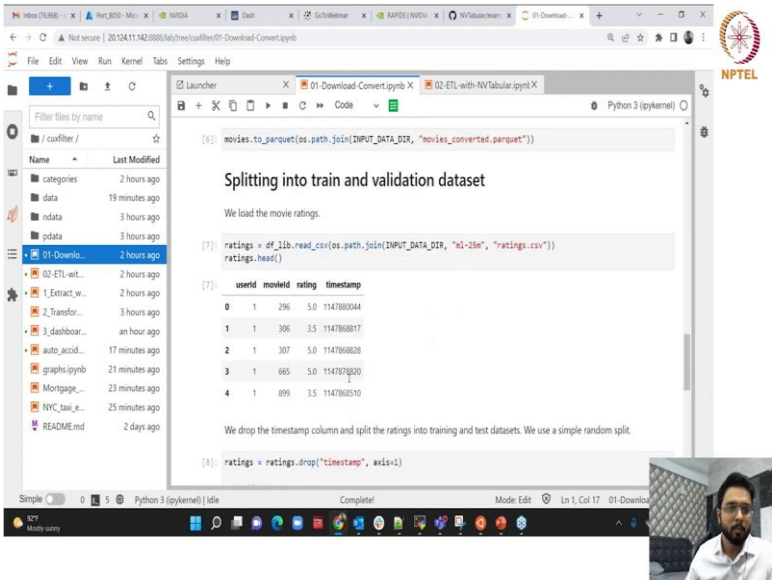
And then input the data directory, download the file, convert the data set.

(Refer Slide Time: 22:00)



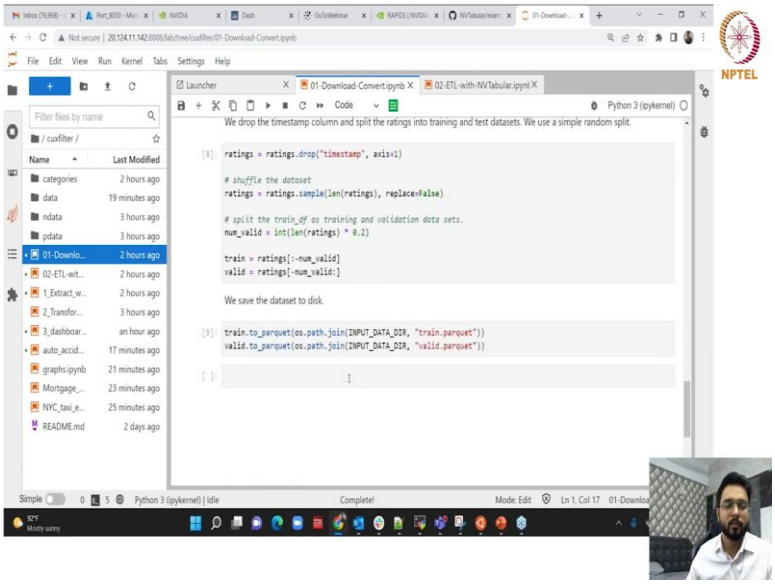
Then put the genres and all that stuff to basic pre-processing.

(Refer Slide Time: 22:05)



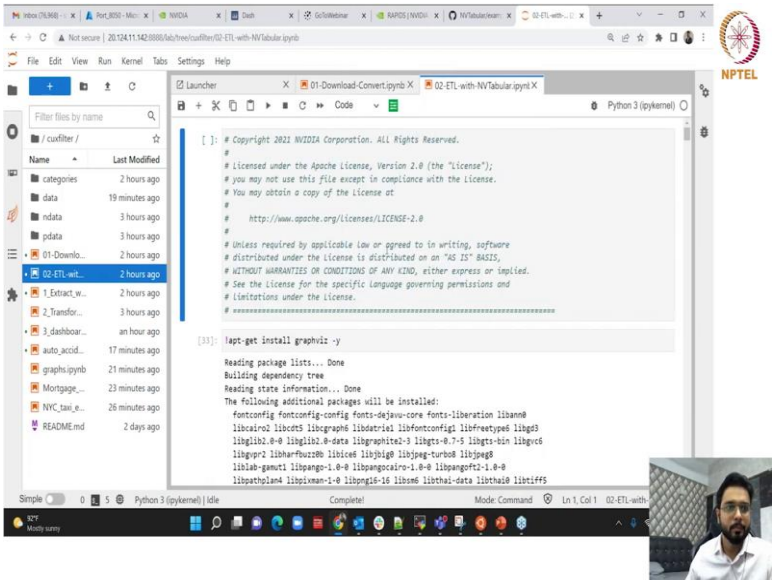
And then convert it to parquet format and write it back to the DASK.

(Refer Slide Time: 22:10)

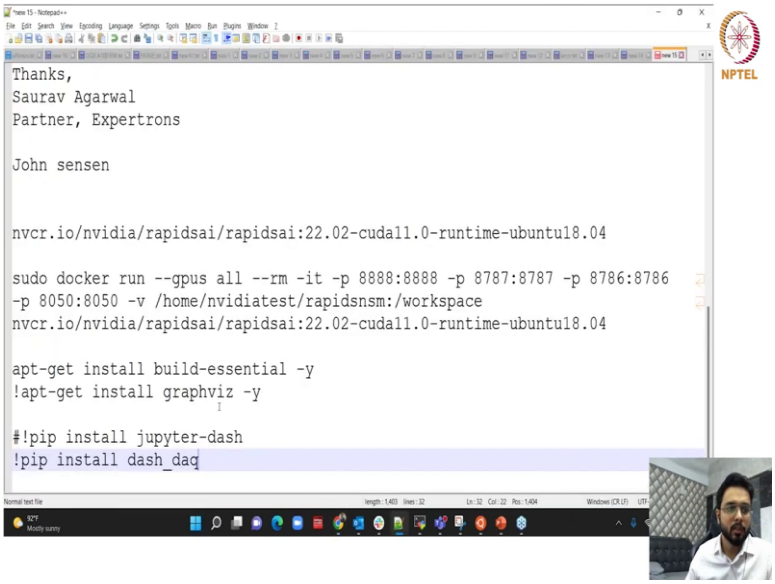


So, this is normal, this is now nowhere we are using NVTabular here it's just pandas. So, just to prepare the data, I mean download the data.

(Refer Slide Time: 22:17)

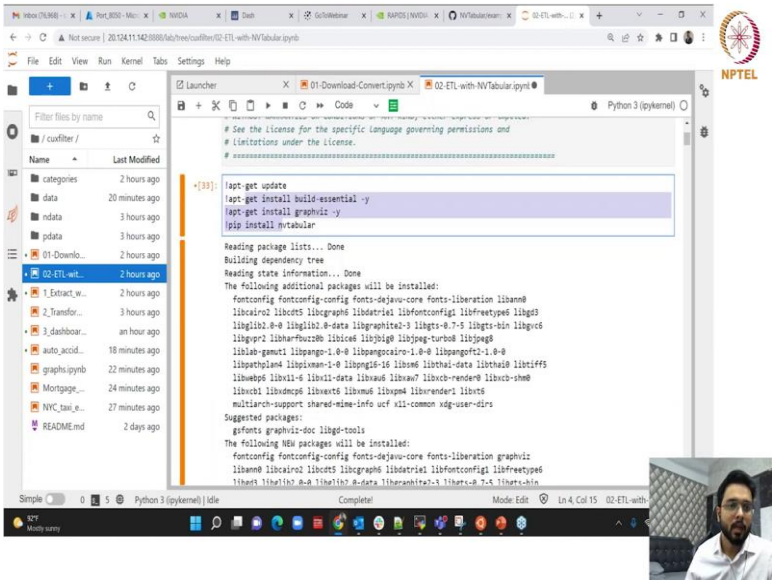


(Refer Slide Time: 22:23)



So, after that what we did is I installed NVTabular. So, using two formats, one is apt-get install build-essential -y.

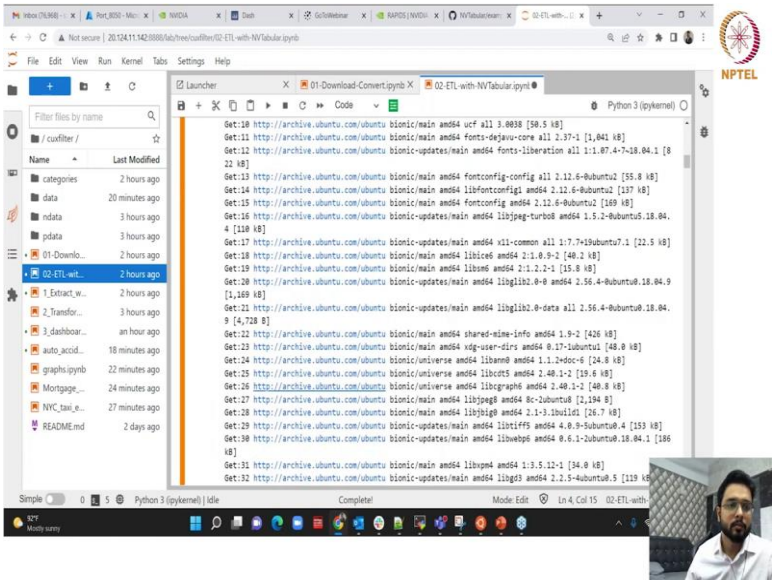
(Refer Slide Time: 22:46)



And then we have, I will just copy paste this, the 3 main methods, 3 main installation I have to do. So, first is apt-get install build-essential, second was Graphviz and third was pip install in nvtabular.

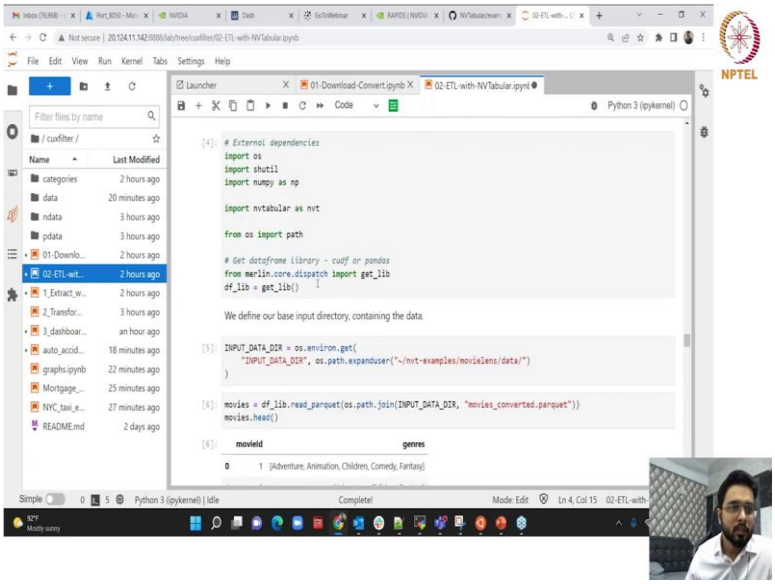
So, these three things I did and then everything got set up after this. If you want you can also add one more thing, that is apt get to update the repositories. So, yeah these are the four main installations I did.

(Refer Slide Time: 23:21)

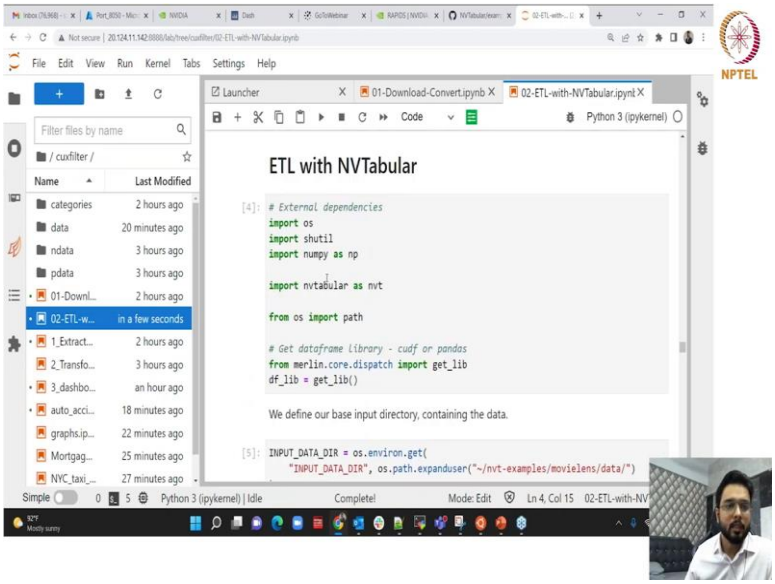


And then after that started ETL with NVTabular.

(Refer Slide Time: 23:31)

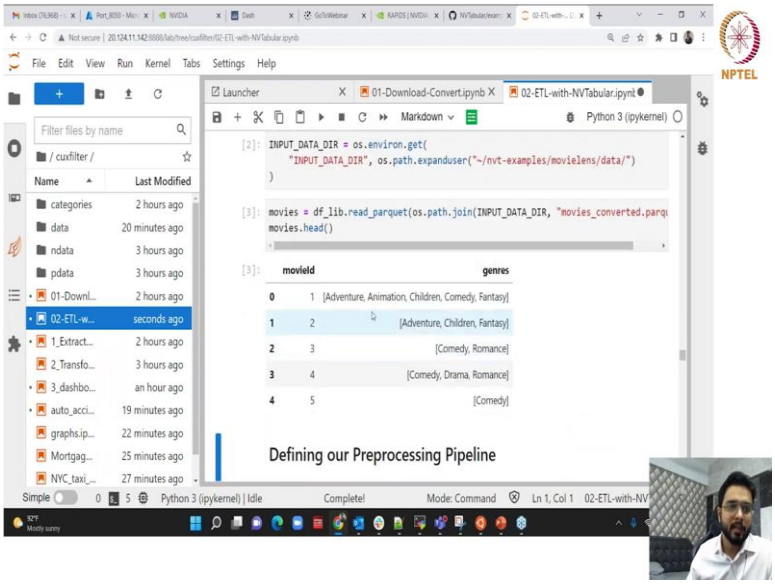


(Refer Slide Time: 23:38)



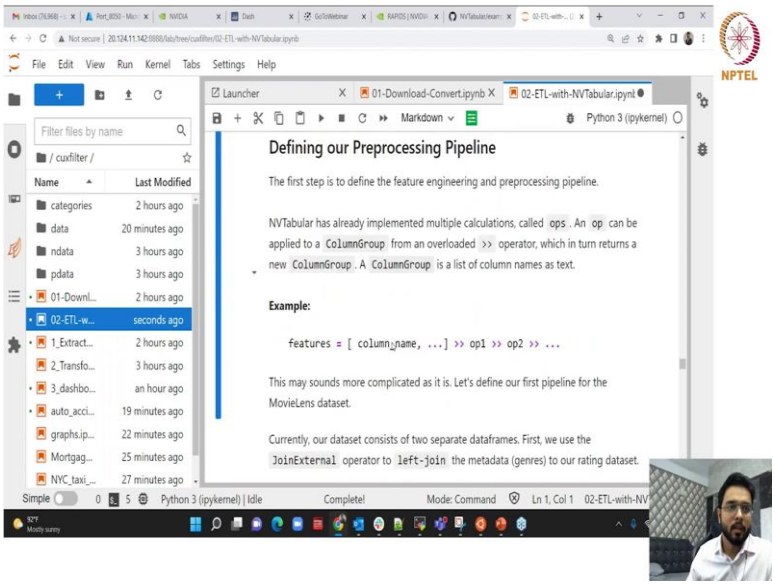
So, I will just magnify it so that is visible clearly, detail with NVTabular, import os import shutil, numpy nvtabular, imported everything.

(Refer Slide Time: 23:47)

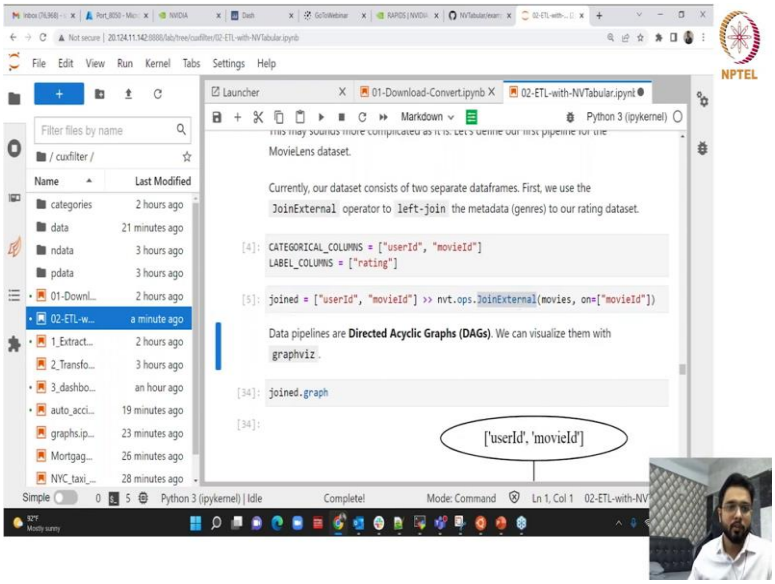


Then set up the input data directory, then set up the movies data frame, using df lib parquet. So, this movie id and genres are the columns here.

(Refer Slide Time: 24:01)

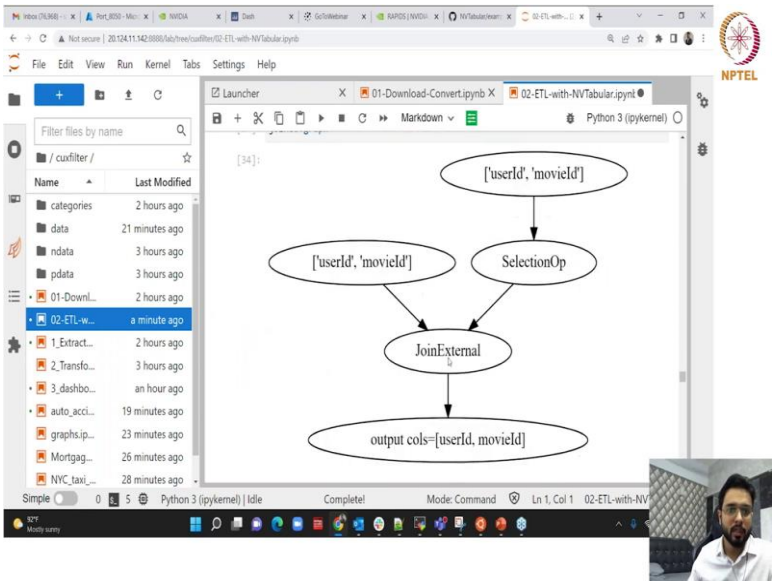


(Refer Slide Time: 24:06)



And then defining our pre-processing pipeline. Categorical columns are user Id and movie Id; label columns; that means, the target columns are rating and then joined.

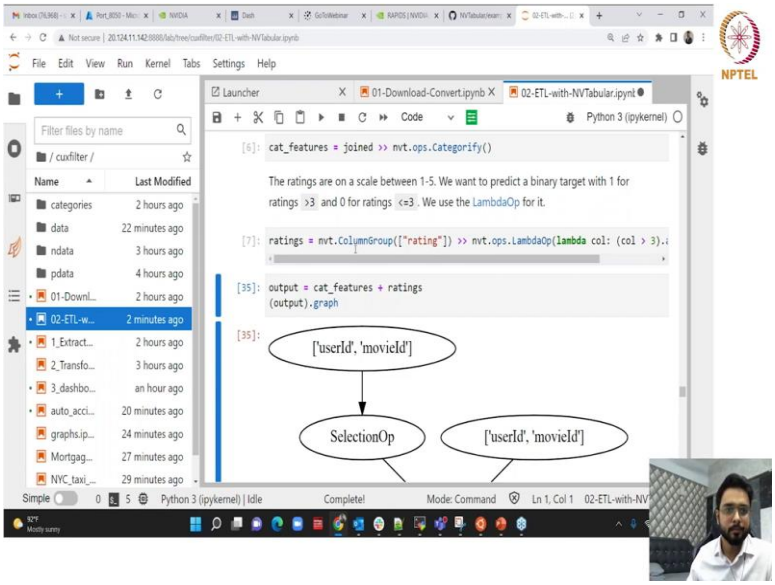
(Refer Slide Time: 24:32)



So, how we joined two columns, user Id and movie Id together. So, and we opt so this is the transformation which I was telling the shortcut transformation nvt ops JoinExternal movies with movie Id. So, now, it is joined.

You want to see how the how it worked. So, you can just run joined.graph. So, you will see that user Id and movie Id was there, then selected operation, then join operation, then output columns, user Id movie Id came.

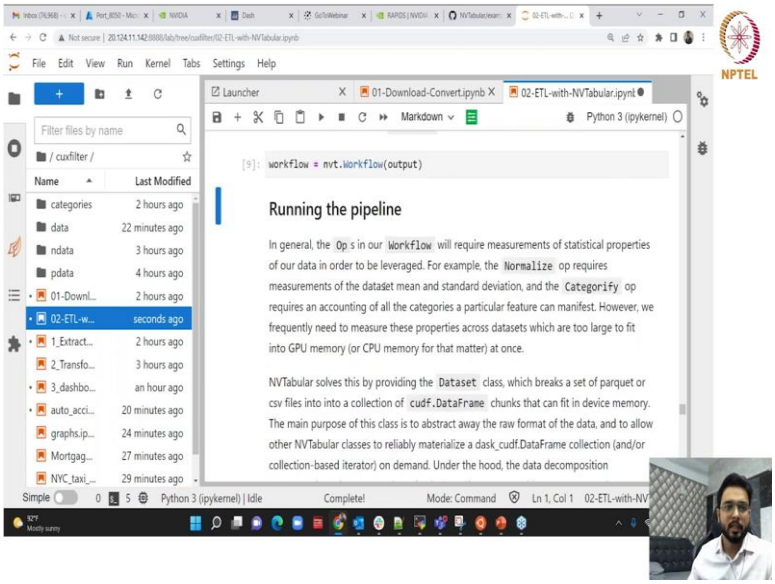
(Refer Slide Time: 24:45)



And then if you want to categorify, because these are numerical values if you want to create categorical values, just uncatagorify you will create categorical values in that column. And then if you want to do some lambda operations so; that means, custom operation. So, col greater than 3, if it is values greater than 3 it becomes a. So, ratings are on scale of 1 to 5, we want to predict a binary target with 1 ratings greater than 3 and 0 for ratings less than 3.

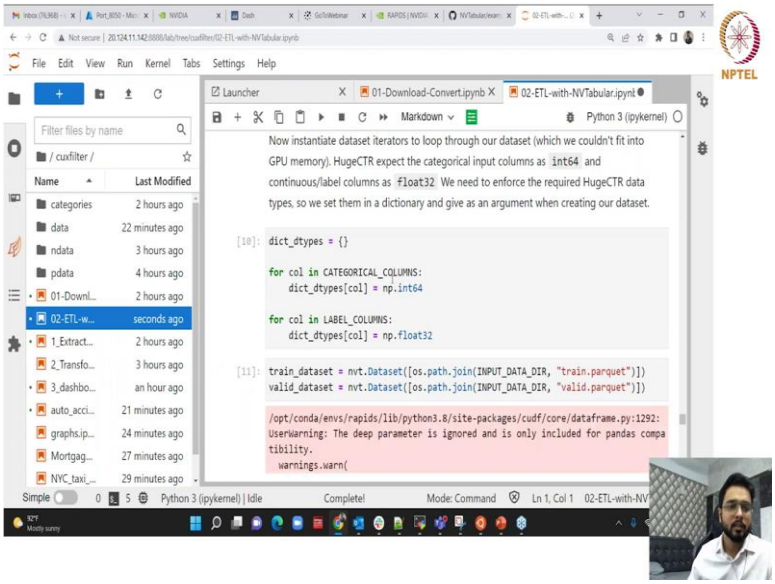
So, again we are again creating type of two categories for 5 values ok. So, based on this you have to create lambda functions. So, simple function to converts 1 to 5 to 1 or 0, based on less than 3 or greater than 3. So, here it is then create workflow nvt.workflow (output).

(Refer Slide Time: 25:41)



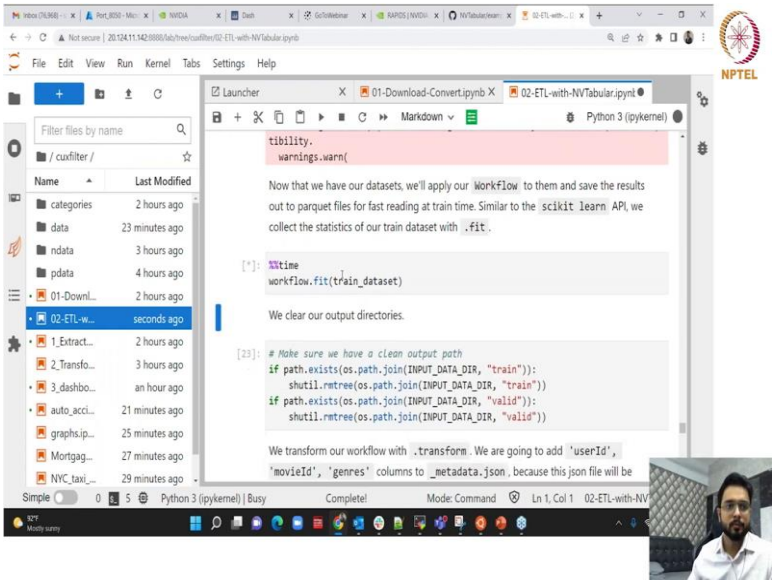
So, this workflow is created and these are all lazy operations again.

(Refer Slide Time: 25:46)



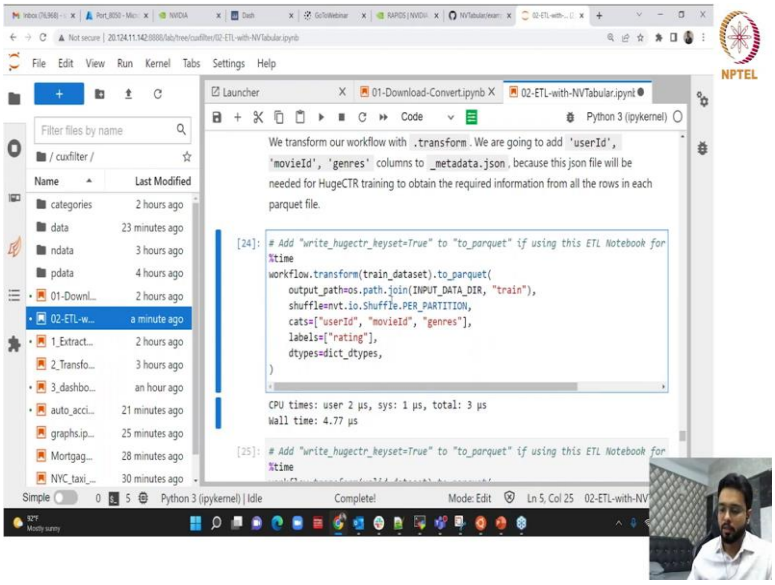
So, the pipeline will run unless until we create the data set and do the fit. So, for column of categorical columns, integer, label columns, float32, then creating the training data set, creating the validation data set.

(Refer Slide Time: 26:04)



I just ignore the warning, then workflow dot fit.

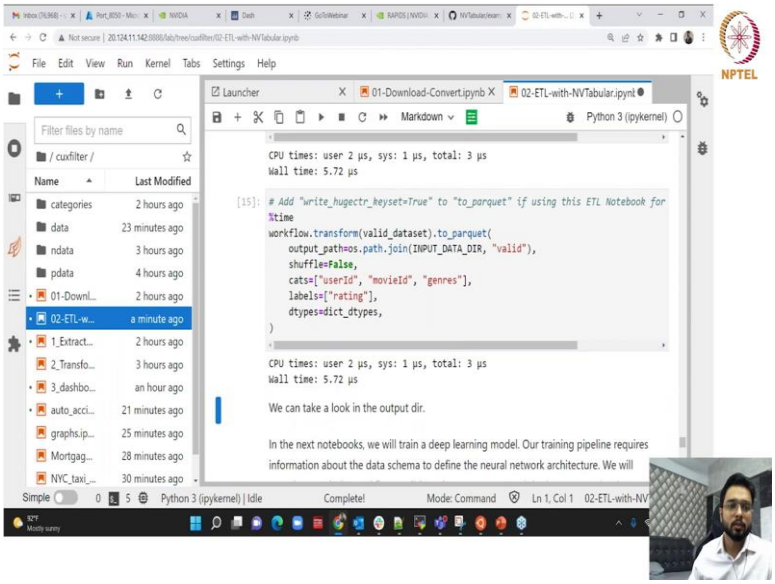
(Refer Slide Time: 26:09)



If path exists then you can create basic clean output path in the local and then add right HugeCTR key set true, to parquet, if this ETL notebook is there for training with HugeCTR.

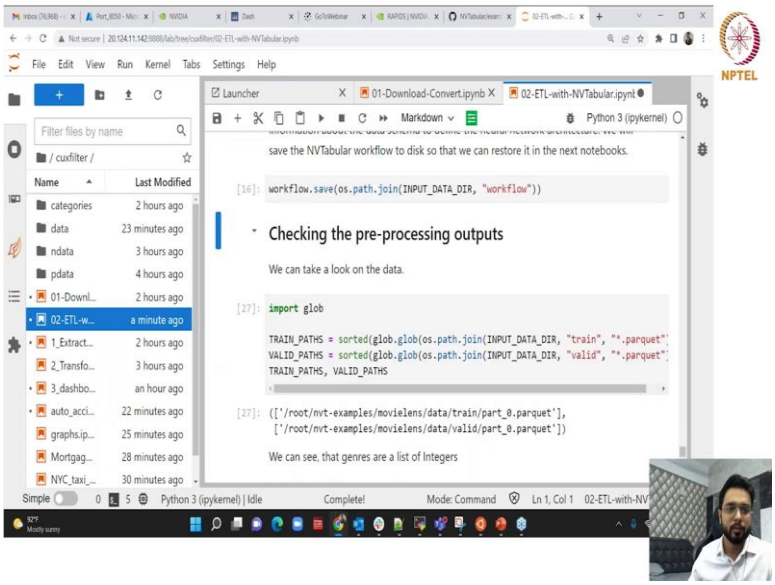
So, this is basically preparing the model for HugeCTR model, though it is not required today because we will mostly focus on the pre-processing part of it using NVTabular.

(Refer Slide Time: 26:31)



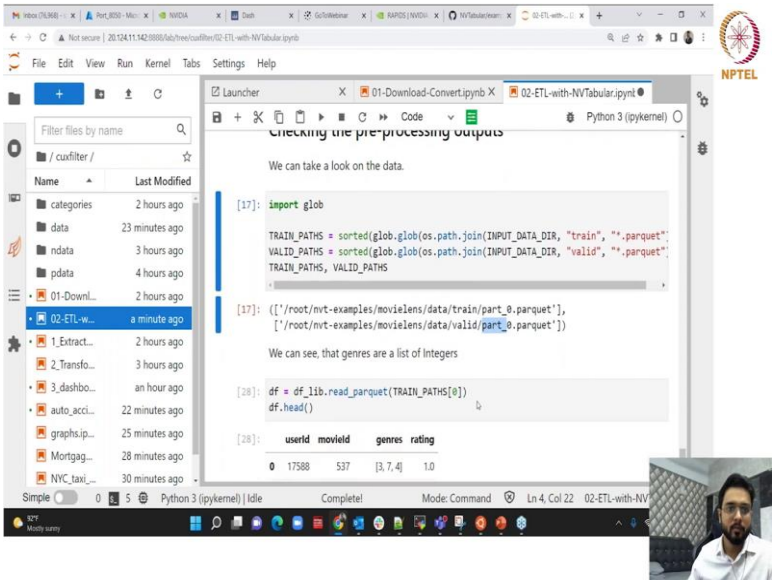
And then finally, transform to parquet output path, the directory you can give, categorical columns, label columns and done.

(Refer Slide Time: 26:47)



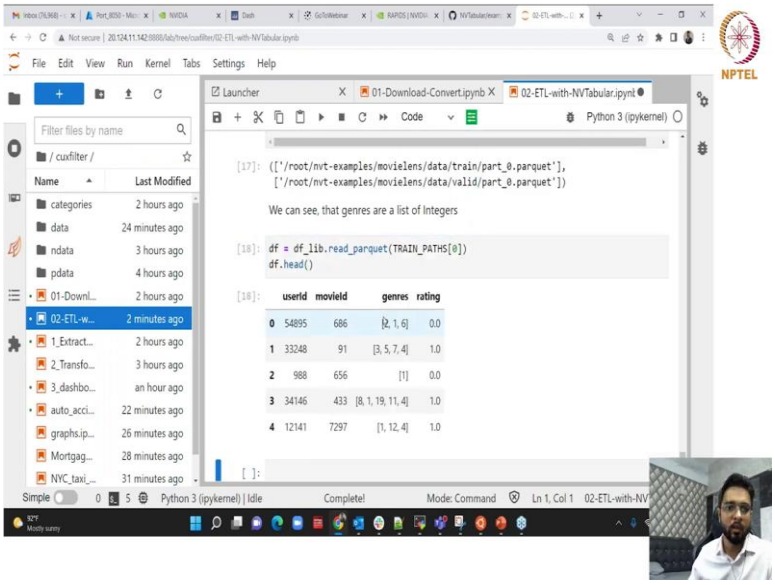
So, entire thing got done in 5.2 microsecond. You can save the workflow, if you want to use it later for different data sets.

(Refer Slide Time: 26:28)



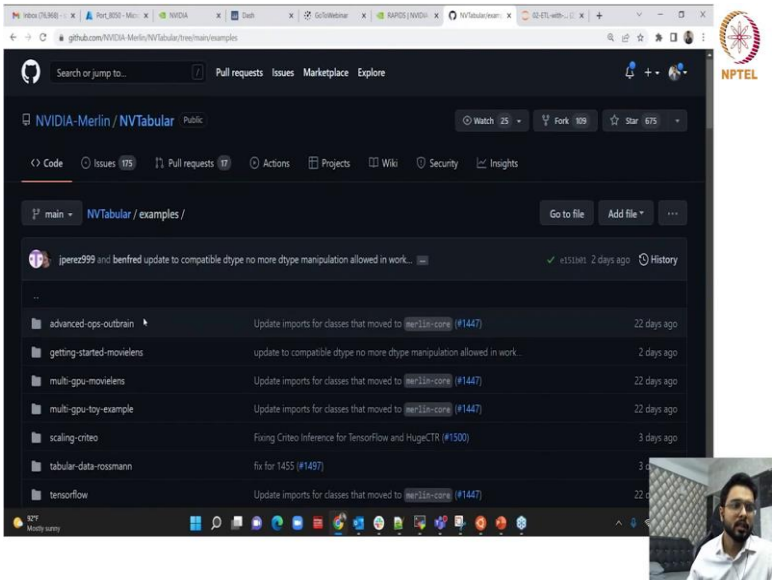
These are the files, if you see parquet part 0 dot parquet being created.

(Refer Slide Time: 27:05)

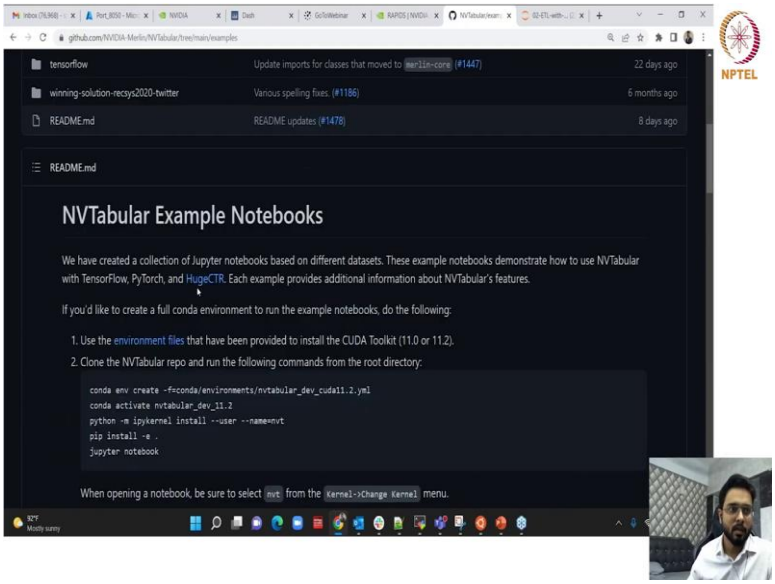


And if you want to see the output so this is the output. User id, movie id, genres, 0 and so on and so forth. Basically, this is the shortcut pre-processing done using NVTabular on GPUs.

(Refer Slide Time: 27:31)

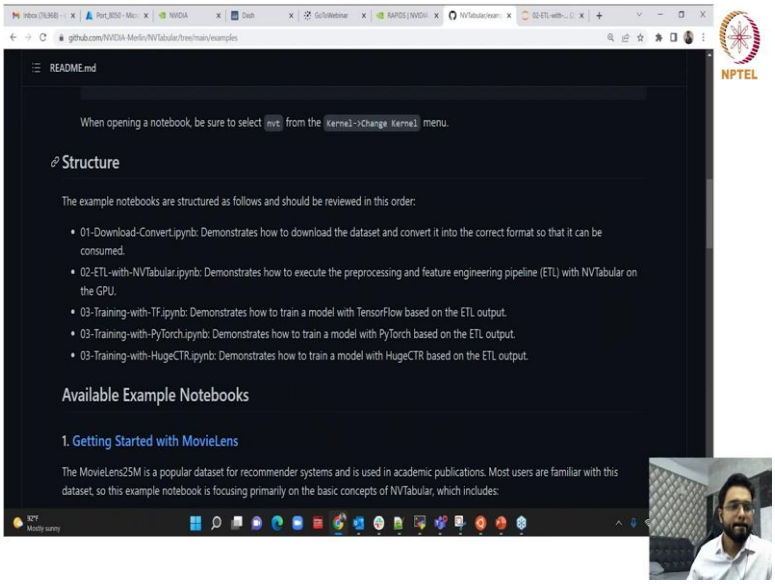


(Refer Slide Time: 27:32)



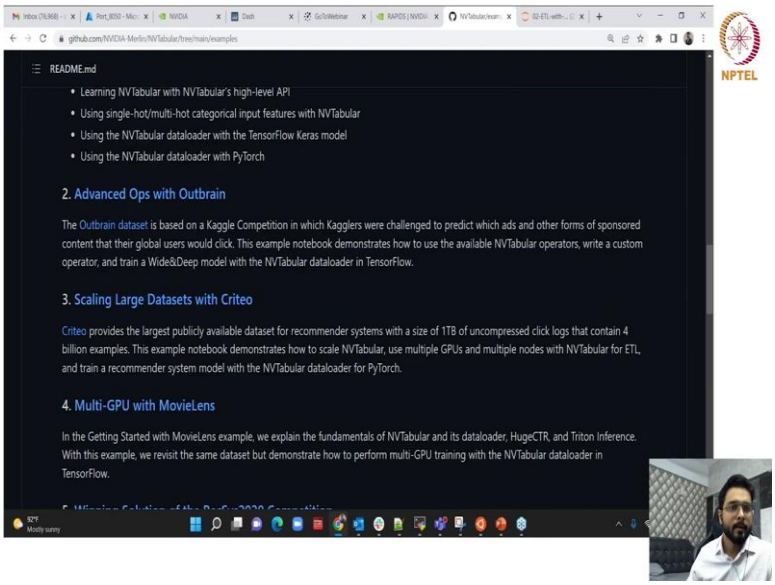
So, there are some more notebooks you can try in the example repositories. So, there is movie lens, apart of movie lens there is advance operations outbrain, outbrain data, rossmann data, criteo data you can also scale using DASK. So, the DASK intrigrated versions are also there.

(Refer Slide Time: 27:45)

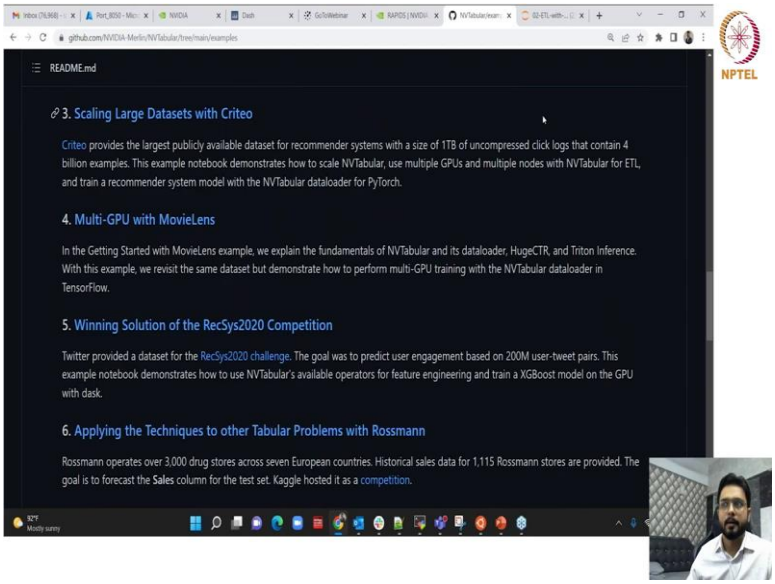


So, right now I just showed you the simple one node cuDF based, but it can be also integrated with DASK or multi node as well ok.

(Refer Slide Time: 27:57)

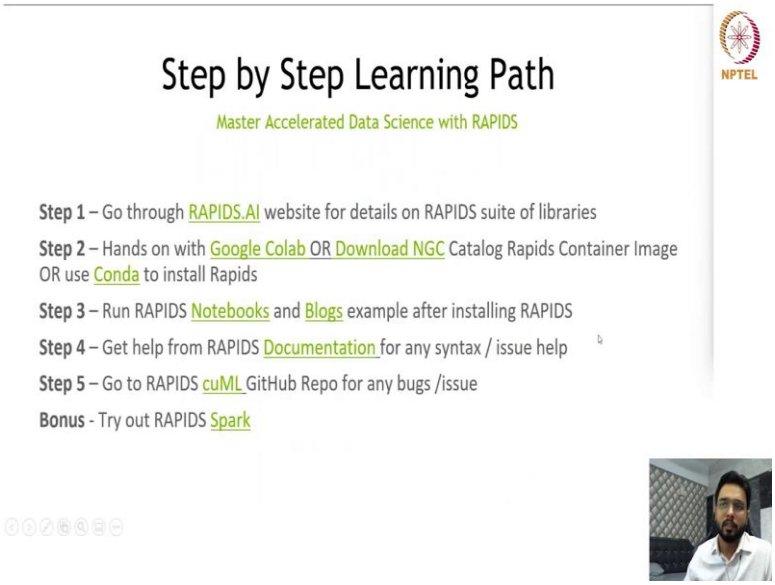


(Refer Slide Time: 27:58)



So, yeah I think this is how NVTabular works and this is the, this is how fast it is as compared to pandas or as compared to Spark or even it is how fast it is to develop code of NVTabular as compared to even DASK on GPU.

(Refer Slide Time: 28:32)



Coming to the learning next steps and learning paths. So, how you can get hands on and you can do something on what you learn today.

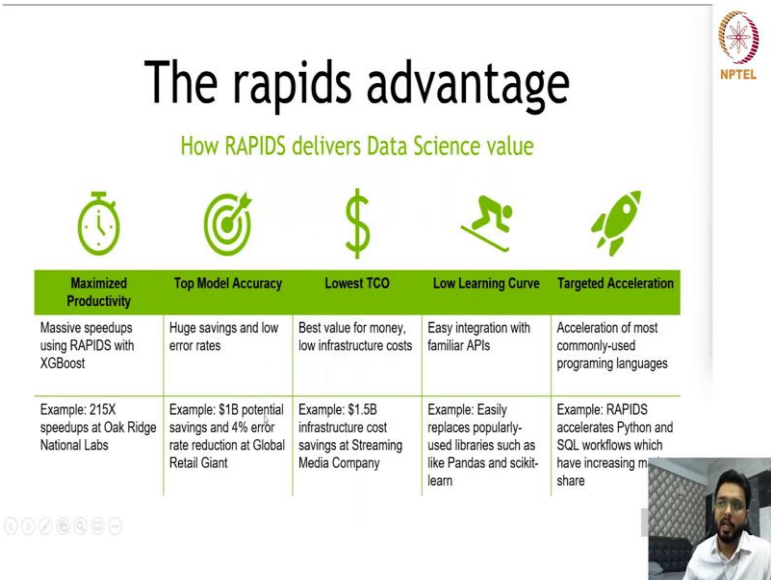
So, if there is RAPIDS.AI website you can go and understand in detail, there is Google Colab or NGC links have given in the slides or you can use Conda as well to install. The

Conda commands are also available in rapid AI website which the link to which I have given here.

Then you can go to rapids notebooks, some default notebooks are there. So, I will particularly link the NVTabular GitHub repository here as well and some of the blogs which you can refer, there are a lot of blogs published every day in day out you can refer to that and you can get help from rapids documentation for any syntax or issue help.

And you can also file GitHub report bugs or issue if you see some there are some bugs or issue with our team works day in, day out to resolve those. Spark is something which we will cover in the next session.

(Refer Slide Time: 29:34)



So, just summarizing that ok what is the advantage you can maximize your data scientists productivity, you can get top model accuracy because you spent more time on optimizing the model rather than pre-processing.

You had a very low total cost of ownership your leaning learning curve was very less, because you do not have to learn anything new, you just used python to learn like your existing knowledge of pandas or sklearn or Python based interfaces. And then using Python or SQL based workflows you did all the things.