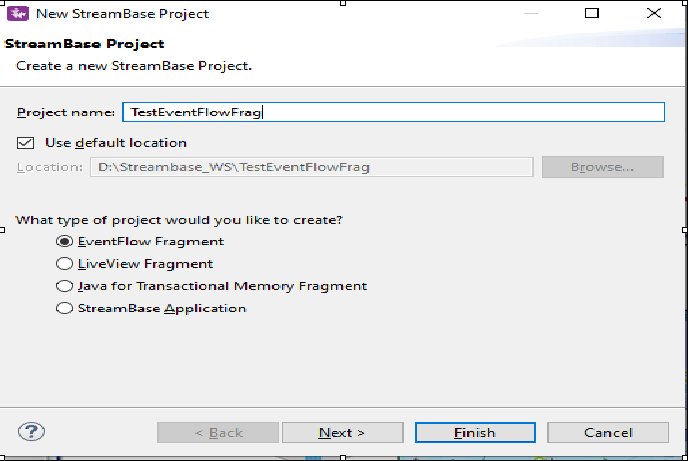
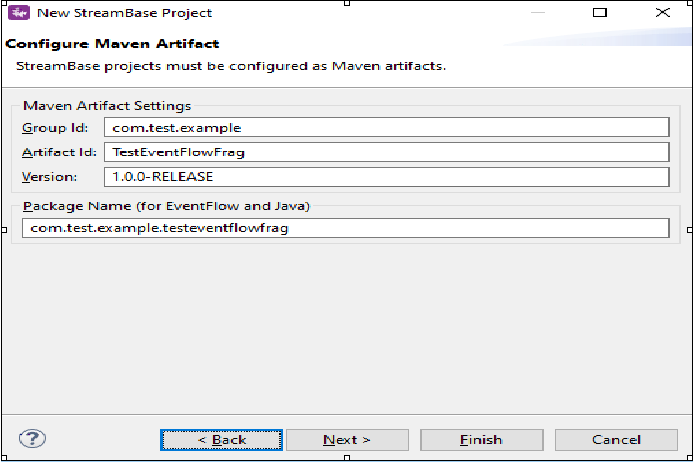
**Create Streambase Studio Project, Create Application Module and Configurations:**

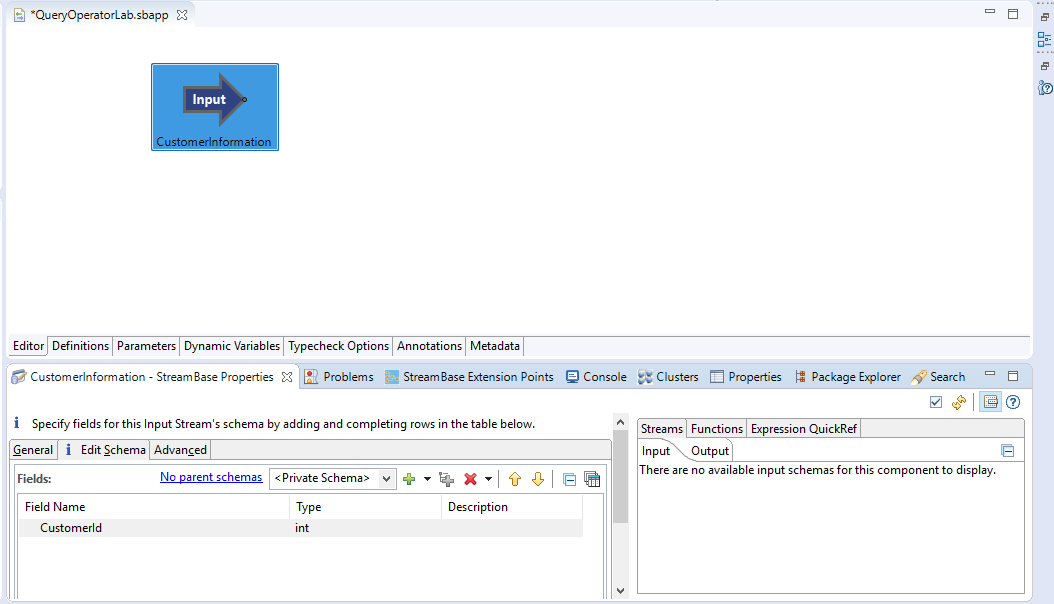
⦁ Create the EventFlow fragment and place all your logic inside this fragment. Select the Radio button as **EventFlow Fragment** as shown in below Screenshot.



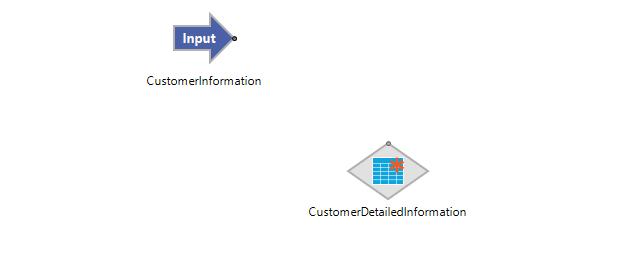
⦁ Click Next and provide the value for **Group Id**, **Artifact Id** and **Version**. Provide the packagename as <<Group Id>>.<<any name>> as shown in below screenshot. Please note down these three parameters and click finish.



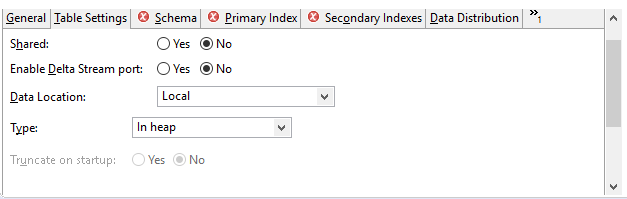
* Create a new event flow fragment and name it as QueryOperatorLab.
* Now go to the folder src/maineventflow and click on QueryOperatorLab.sbapp
* From the palette view click on Input Stream and Drag it into the canvas.
* Name the Input Stream as Customer Information.
* In the Edit Schema tab Give the Field Name as CustomerId and Select the type as int.



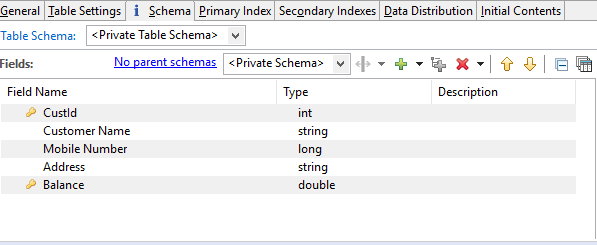
* Now Drag the Query table to the canvas.
* In the General Tab Give the name as CustomerDetailedInformation.



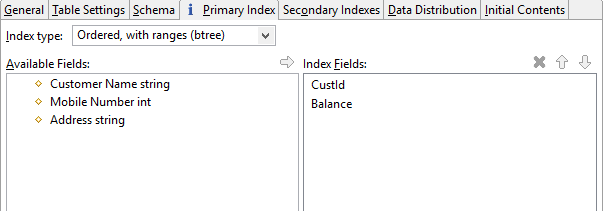
⦁ In the Table Settings leave the values as Default as mentioned in the screenshot below.



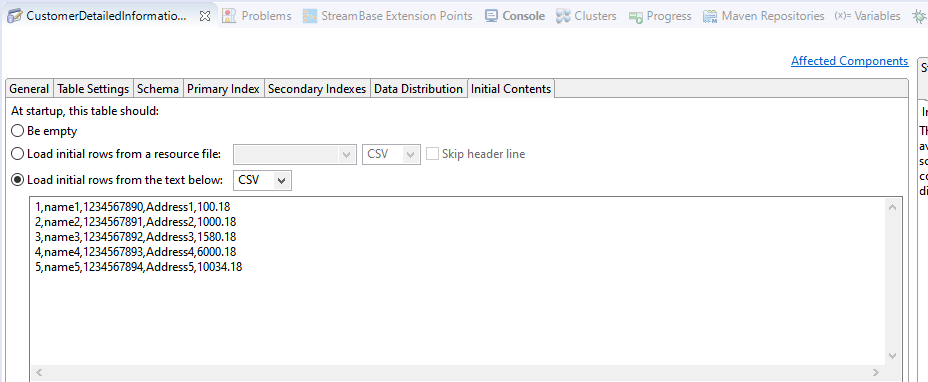
* Create a Private schema which contains values CustId, Customer Name, Mobile Number, Address, Balance as shown in the below screenshot.



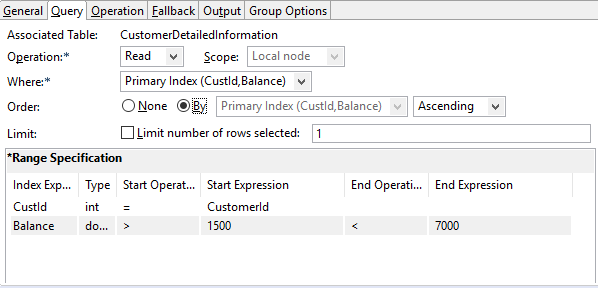
⦁ Now go to the primary Index tab and select the CustId and Balance as primary key from the available fields and leave the secondary Indexes and Data Distribution as Default.

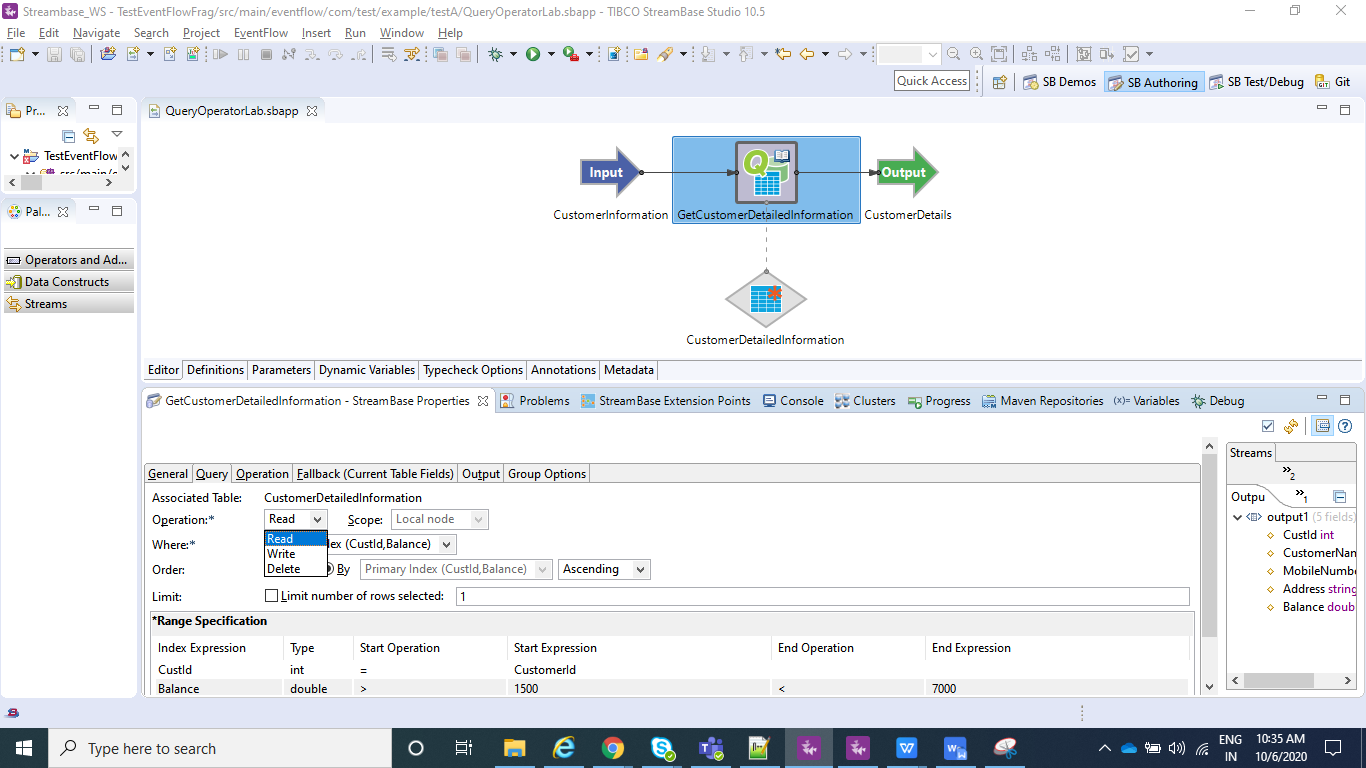


⦁ Load initial row from the text as shown below. This are just the sample values that consist of all the above parameters defined in schema.

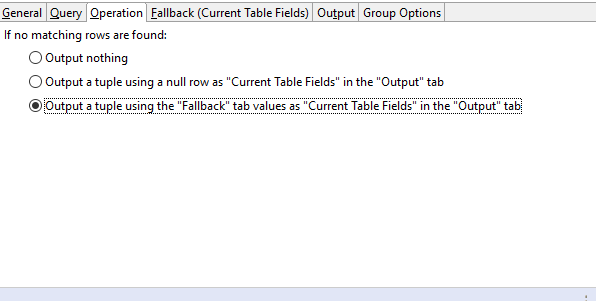


⦁ Now Drag the Query Operator onto the canvas and name it as GetCustomerDetailedInformation.Now connect the input port to the Query Operator port and the Query operator port to the Query table. In the Query Operator settings go to Query and Perform a read operation.Like Read we can also select Write or Delete as per the requirement. CustId is the input from the Associated Table (Query Table) and the CustomerId is the input from the input table.

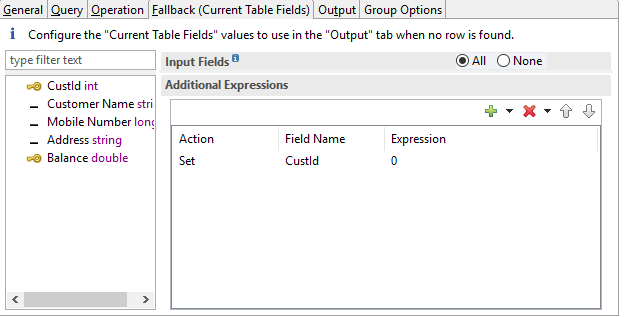




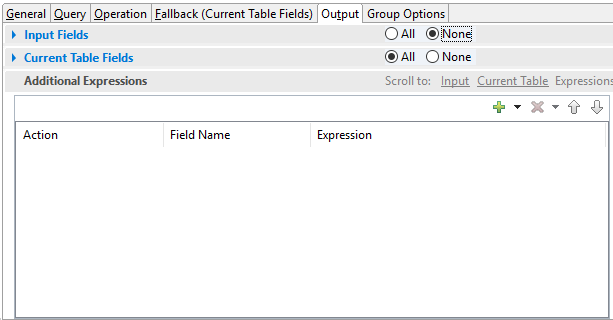
⦁ Now go to operation and click on the third radio button (using Fall back tab).



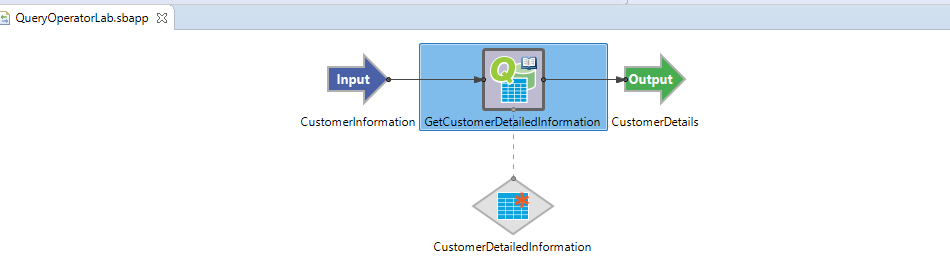
⦁ Here we set CustId as 0(Here 0 means does not exist)



⦁ In the Output tab select All in the Current Table Fields and None in the Input Fields.



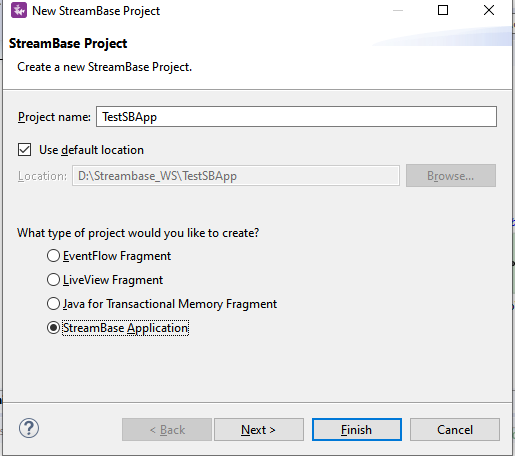
⦁ Now Drag the Output to the canvas and name it as CustomerDetails.



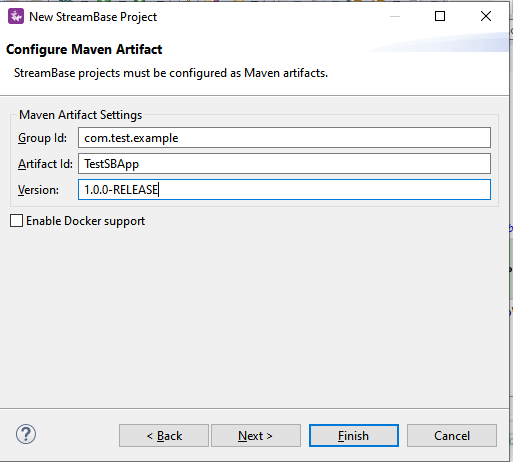
⦁ Now In the Project Explorer view, select and right-click the Streambase Application Project **(TestEventFlowFrag in my case)--> Run As>Maven Build...**

⦁ After that In the **Goals** field, enter **clean install** and Select the **Skip Tests** check box. Click Apply and Run.

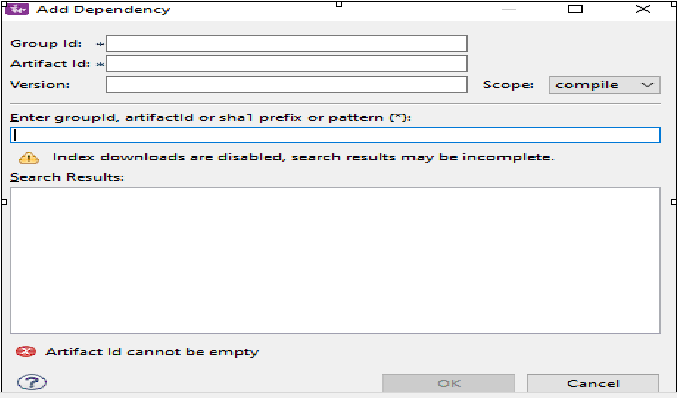
⦁ Now create a new Streambase Project. Select Radio button as **StreamBase Application**.

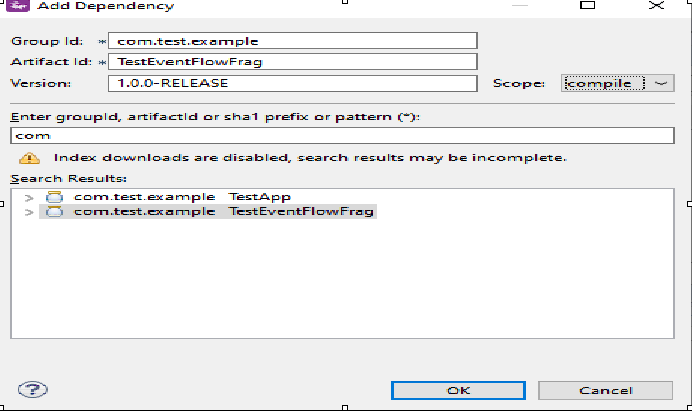


⦁ Now click next and make sure you provide the same **Group Id** and **Version** that you specified for the EventFlow Fragment created Earlier and click **Finish**.



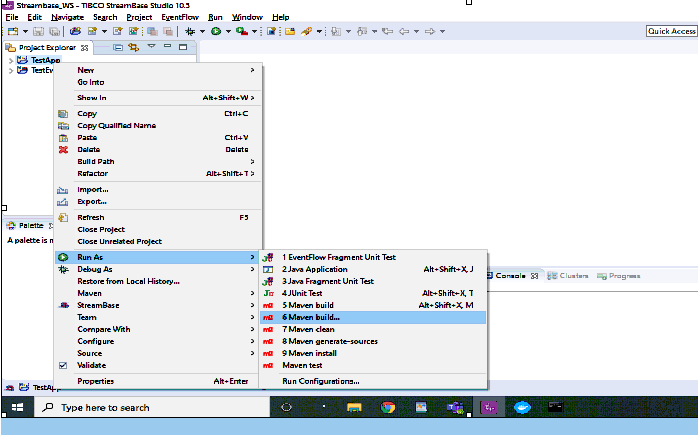
⦁ Add the EventFlow Fragment Project depedency to our main Streambase Application. Right click on Streambase Application Project **(TestSBApp in my case) --> StreamBase>Manage Project Dependencies**. Add the depedency for the EventFlow Fragment and click ok.

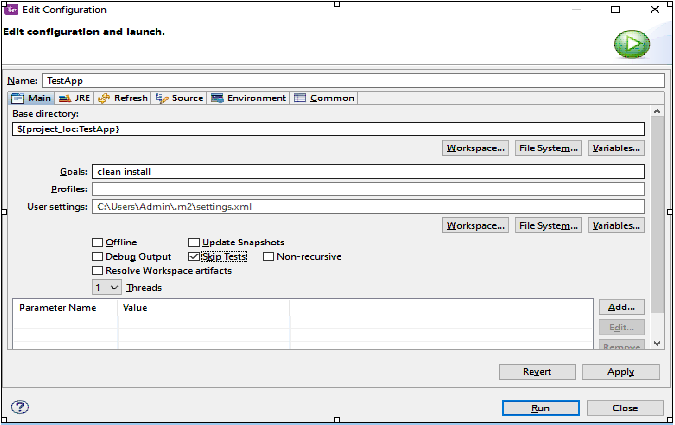


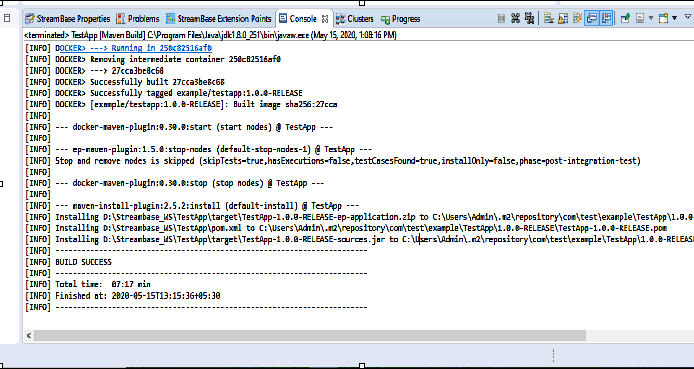


⦁ Now In the Project Explorer view, select and right-click the Streambase Application Project **(TestSBApp in my case)--> Run As>Maven Build...**

⦁ After that In the **Goals** field, enter **clean install** and Select the **Skip Tests** check box. Click Apply and Run.

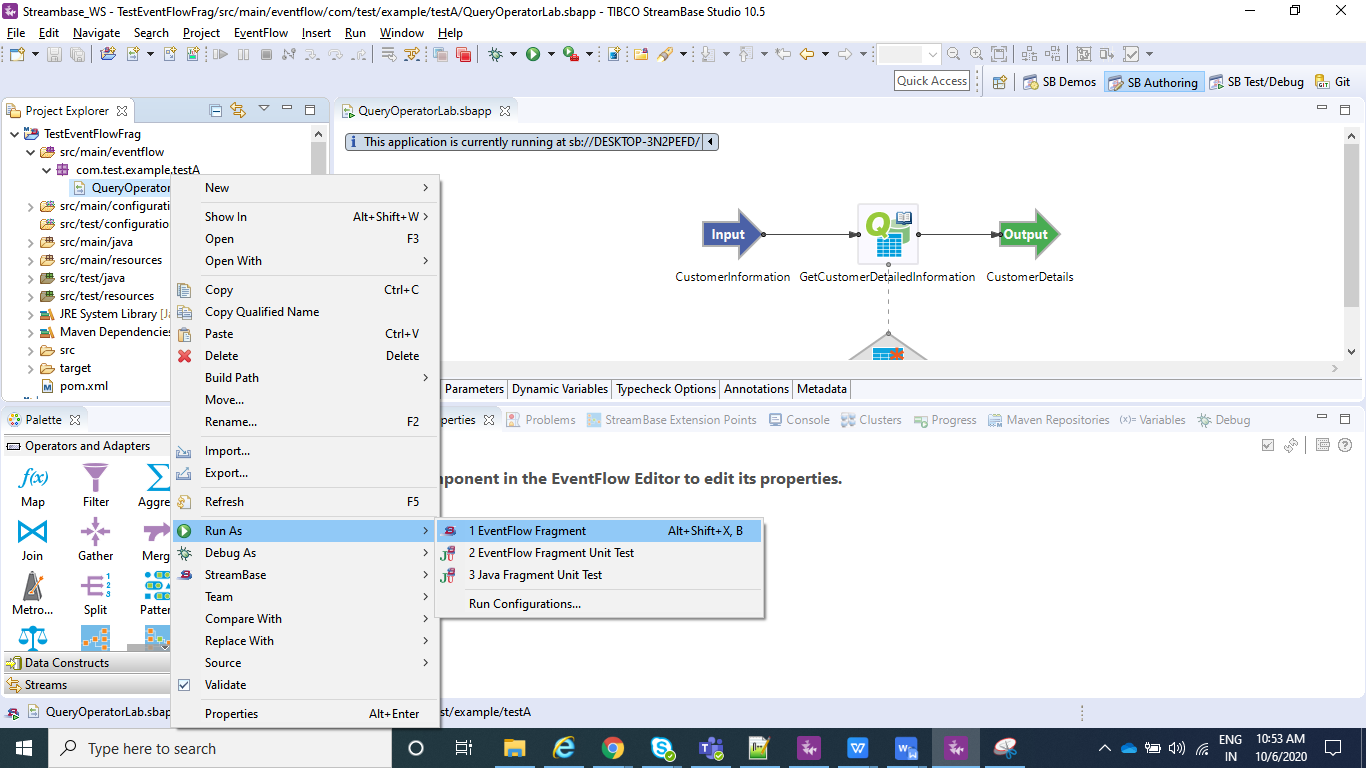




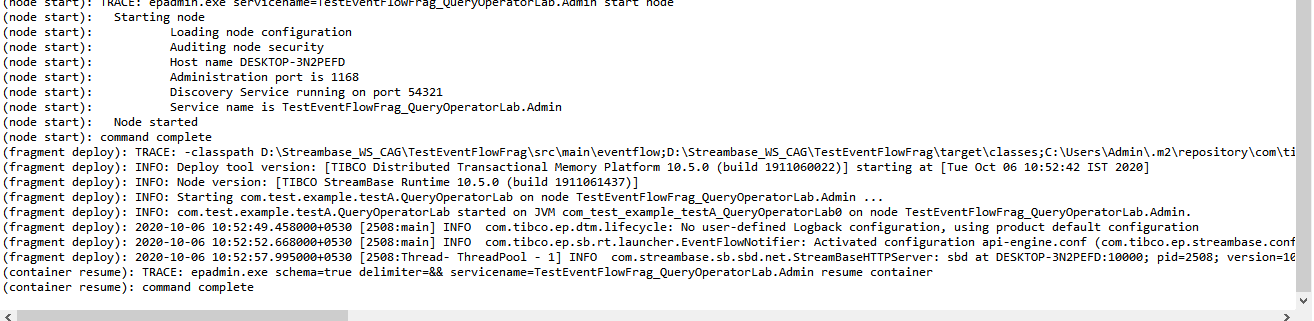
⦁ After the above step check the console and it should be showing BUILD SUCCESS message.

**Run the application in Studio:**

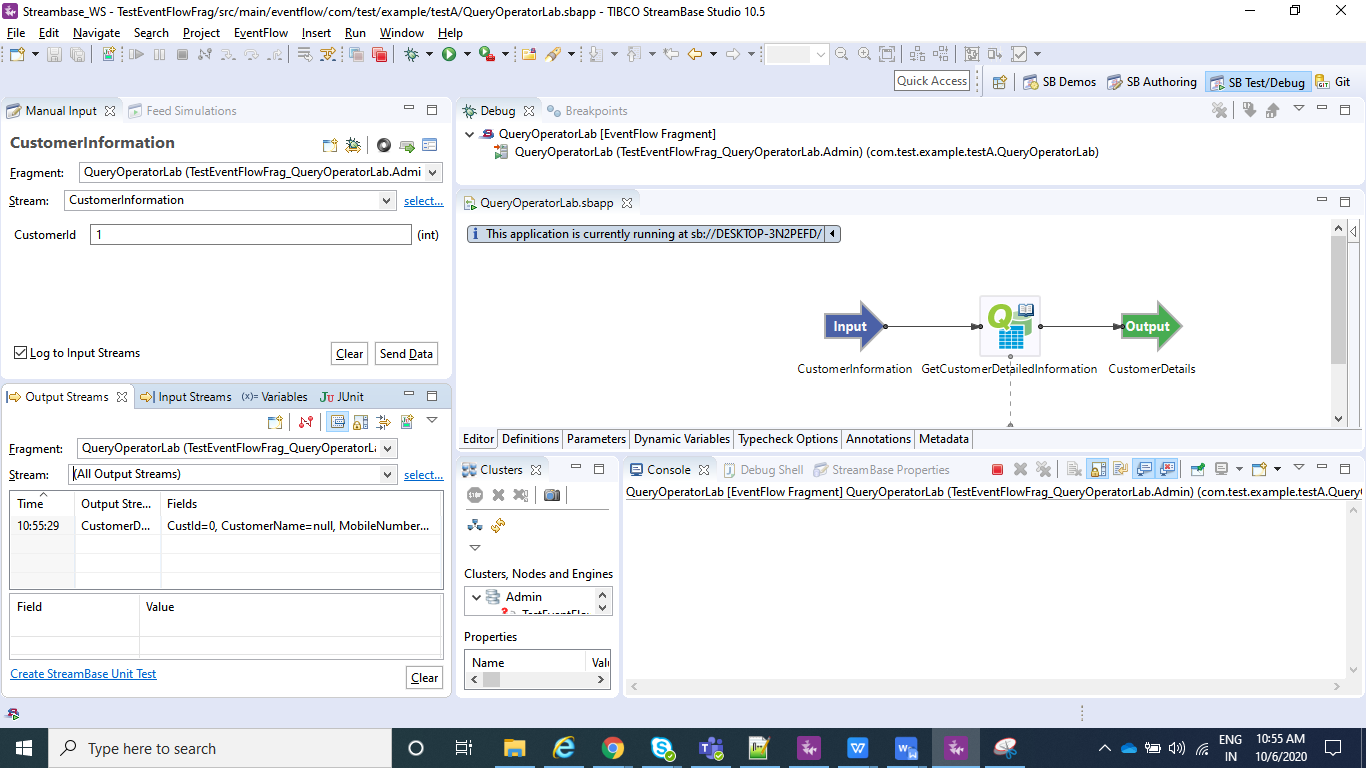
⦁ Right click on your .sbapp and select Run as EventFlowFragment as shown in below screenshot:

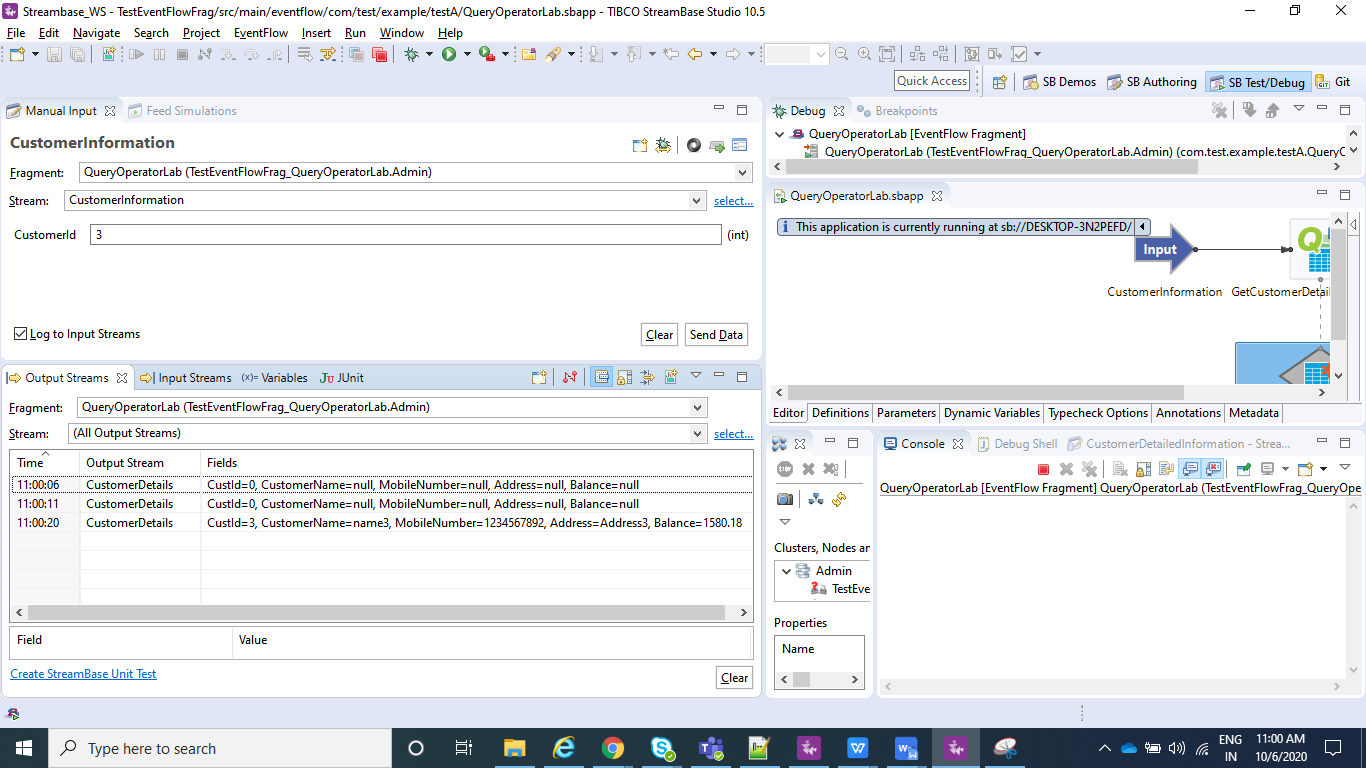


⦁ After you application is running you can see to the console logs as shown below:



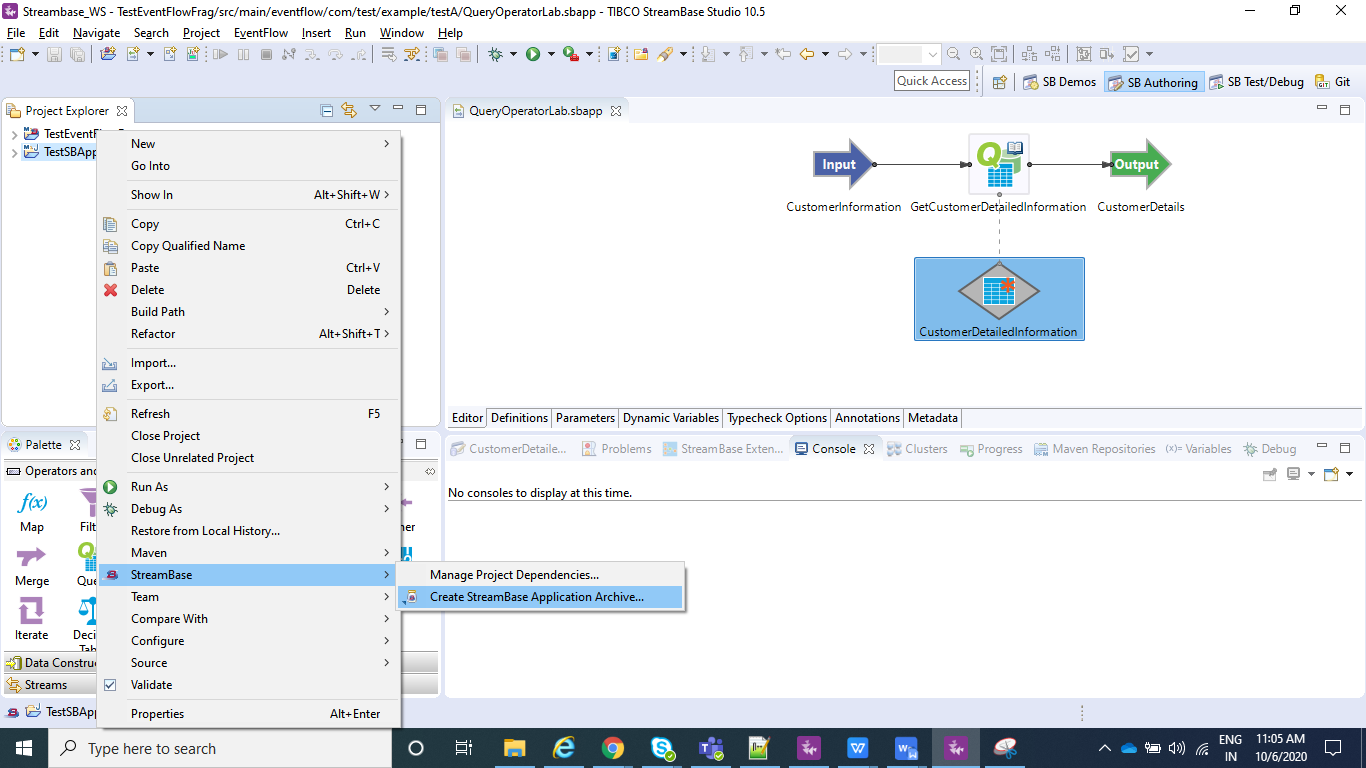
⦁ Now your application is running and same can be tested using Manual Input as shown in below screenshot. The value for the employee id 1 and 2 will be CustId=0 and rest of the fields as null, but when you provide the employee id as 3 you can see "CustId":3,"CustomerName":"name3","MobileNumber":1234567892,"Address":"Address3","Balance":1580.18. This is because in the above steps we have configured the query operator as display the data only for the customer whose Balance is >=1500 and <7000 and for the rest of the customers display CustId=0 as we have configured in fallback tab.

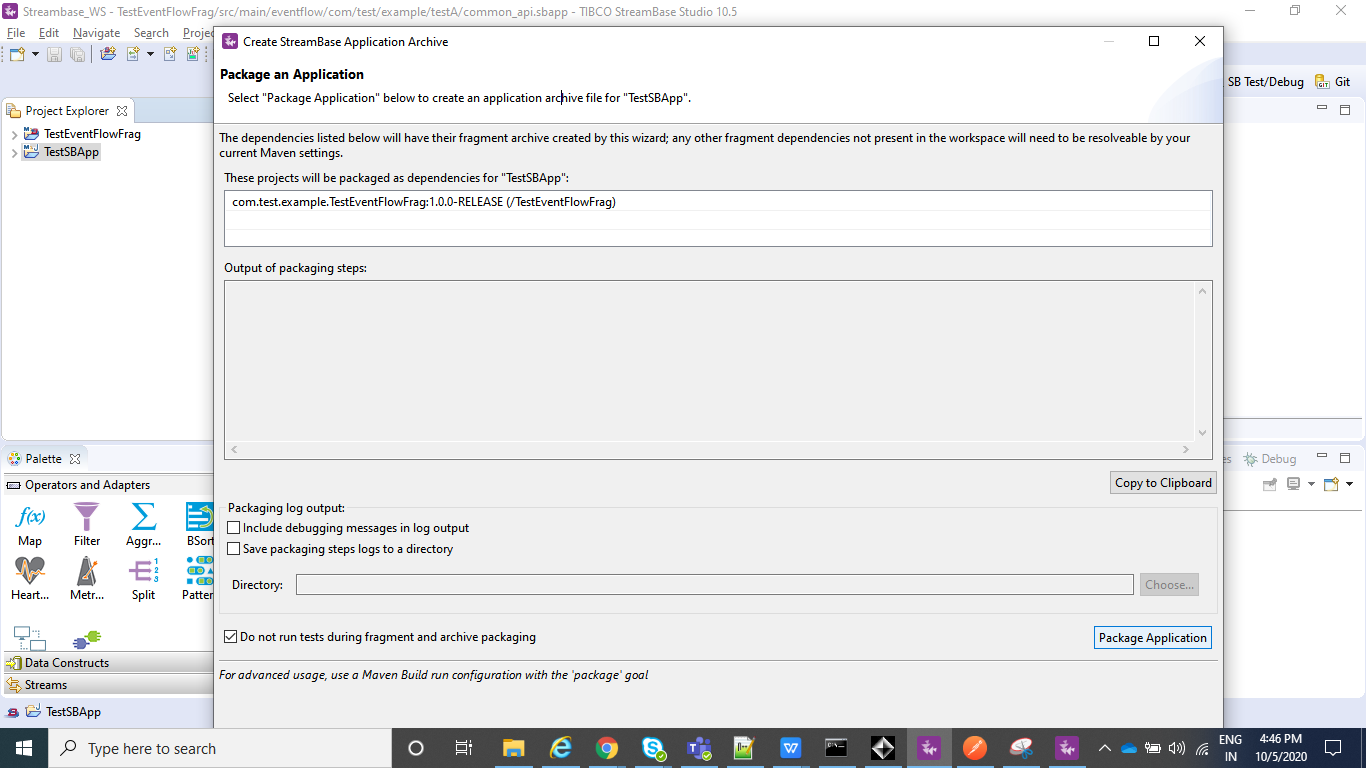




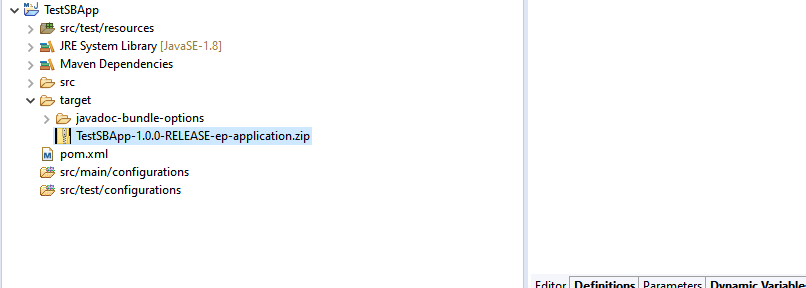
**Create the Application Archive:**

⦁ Right click on the Streambase Application(**TestSBApp** in my case) and than select Create Streambase Application Archive and than select Package Application**.**





⦁ After the above step is completed we can see a .zip file under our Streambase Application under target folder.



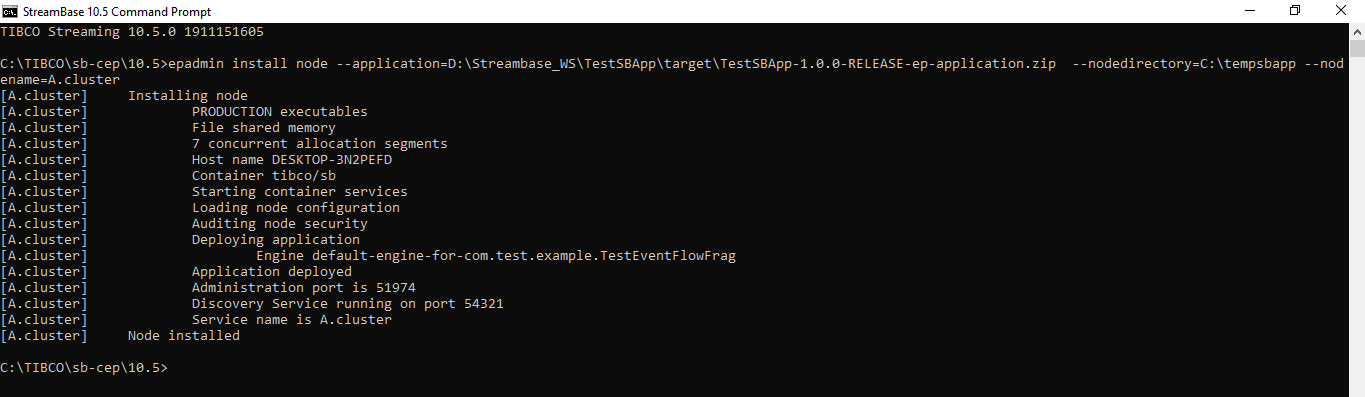
**Deploy and Run the Application:**

⦁ Open the Streambase CommandPrompt to Deploy and Run the application locally as shown in below screenshot.



⦁ Open the Streambase CommandPrompt to Deploy and Run the application locally as shown in below screenshot and run the below command where --**application=D:\Streambase\_WS\TestSBApp\target\TestSBApp-1.0.0-RELEASE-ep-application.zip** is the path of the zip file we just created above,  **--nodedirectory=C:\tempsbapp** is any temp directory created in C drive and **nodename=A.cluster** is the any random name assigned to the node which will be running.

**epadmin install node --application=D:\Streambase\_WS\TestSBApp\target\TestSBApp-1.0.0-RELEASE-ep-application.zip --nodedirectory=C:\tempsbapp --nodename=A.cluster**

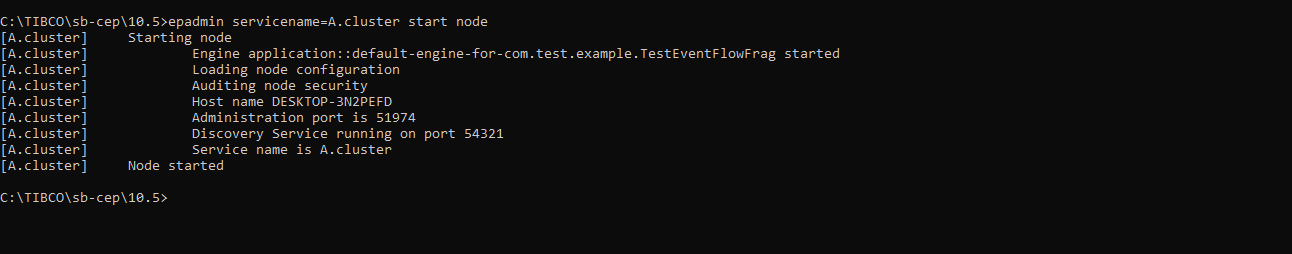


⦁ Now after your Node is installed, we can **start**, **stop** or **remove** node using below commands and once you have started the node you can test your application using postman in the same way we did before:

**epadmin servicename=A.cluster start node**

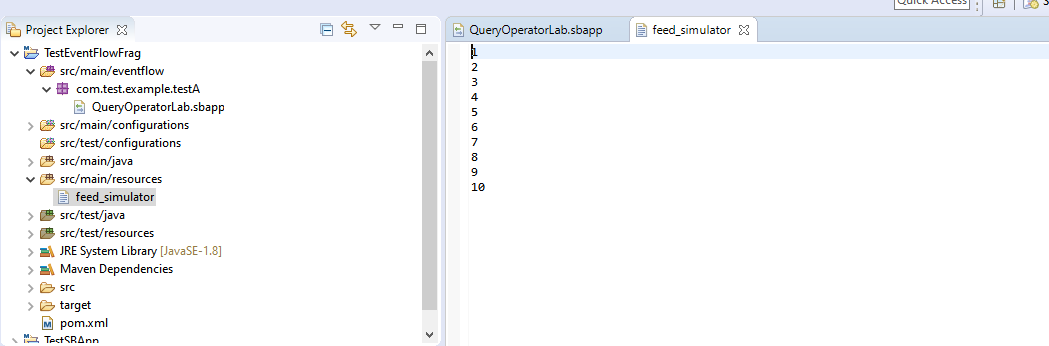
**epadmin servicename=A.cluster stop node**

**epadmin servicename=A.cluster remove node**

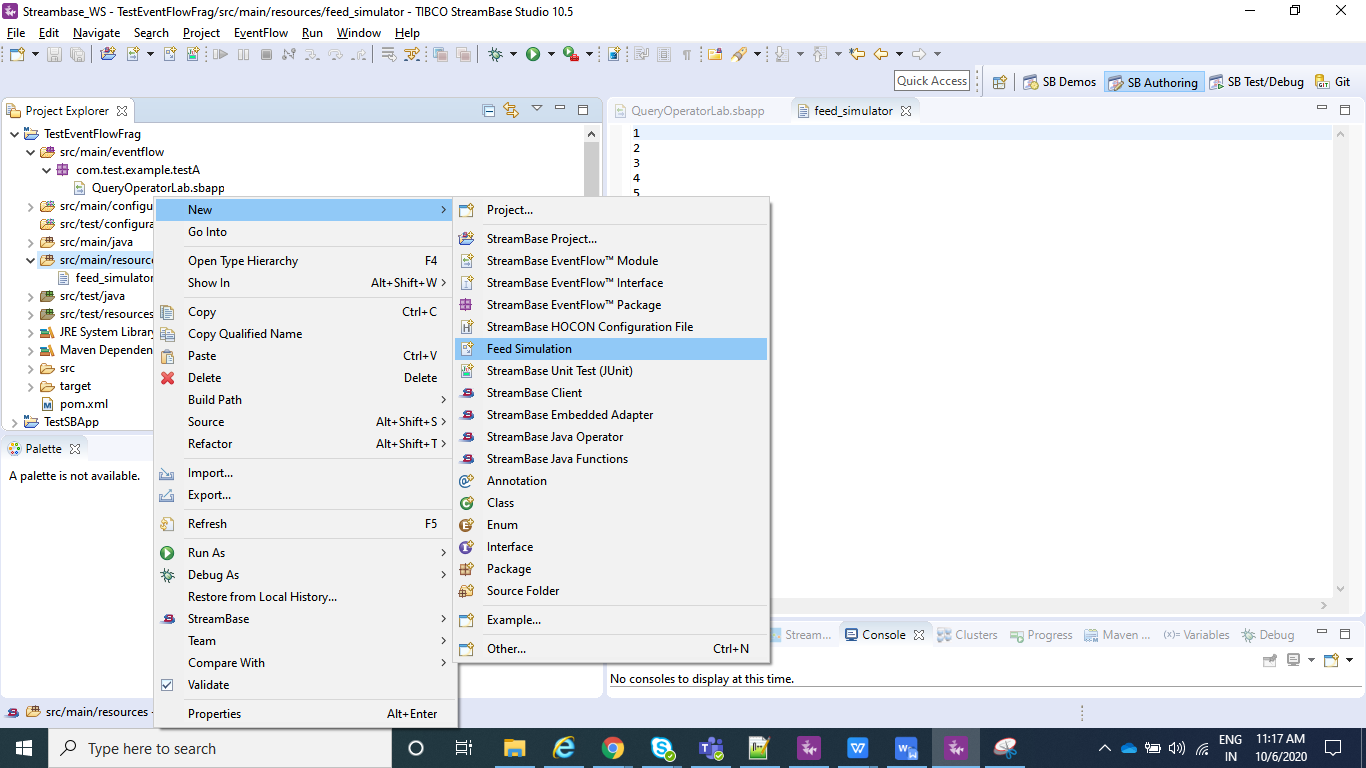


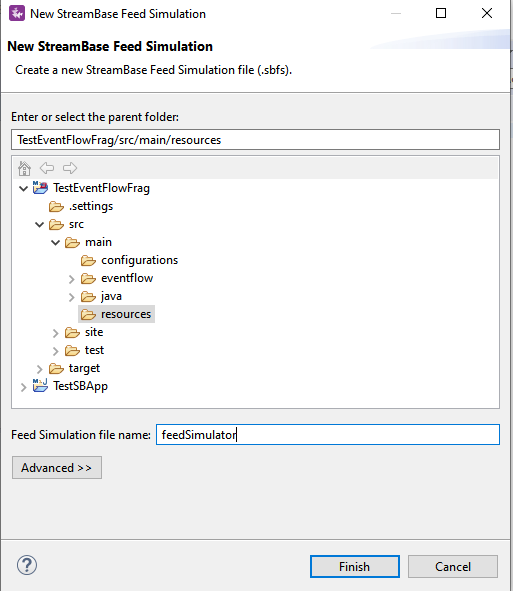
**Test the application (Using Feed Simulator file from command line):**

* Import a feed\_simulator.txt file with below entries under src/main/resources.

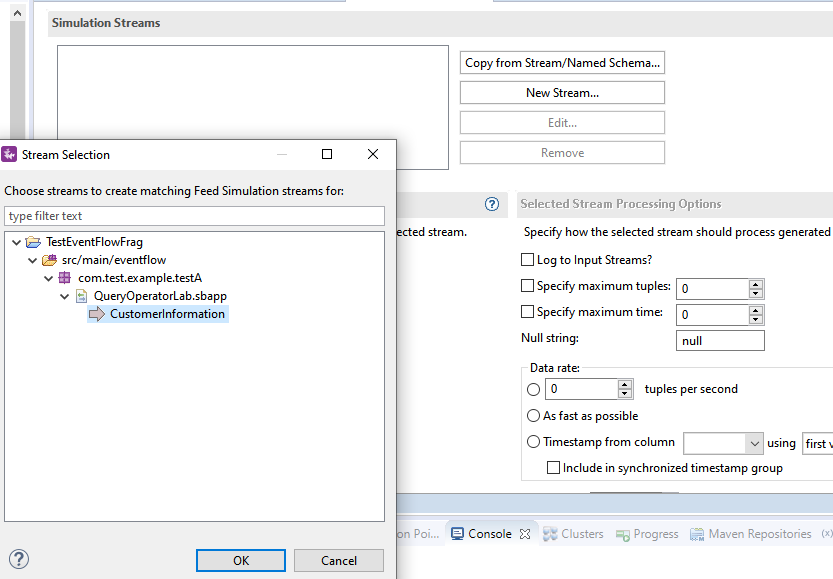


* To Configure feed simulator please follow below screenshots :

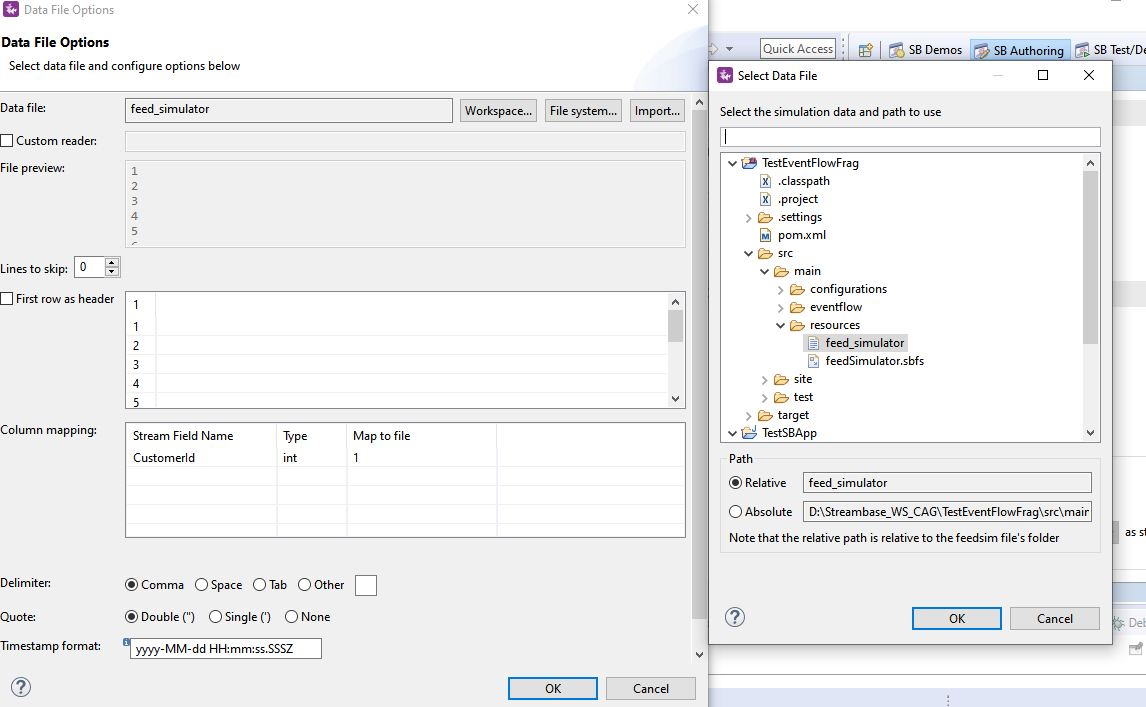




* Click on Copy from Stream/Named Schema and go to your Input Stream(CustomerInformation in my case) and click OK.



* Now select Data File option and select the file you just created , i.e, feed\_simulator.txt file where you have your data loaded.



* Create a configuration file which will contains information on which of your .sbapp need to run when an streambase application is deployed.Sample for the same is :

name = "sample-StreamBaseEngine-document"

type = "com.tibco.ep.streambase.configuration.sbengine"

version = "1.0.0"

configuration = {

StreamBaseEngine = {

jvmArgs = [

"-Xmx8g"

"-Xms512m"

"-XX:+UseG1GC"

"-XX:MaxGCPauseMillis=500"

"-XX:ConcGCThreads=1"

]

}

EventFlowDeployment = {

modules = [

{

moduleName="com.test.example.testA.QueryOperatorLab"

containerName="QueryOperatorLab"

}

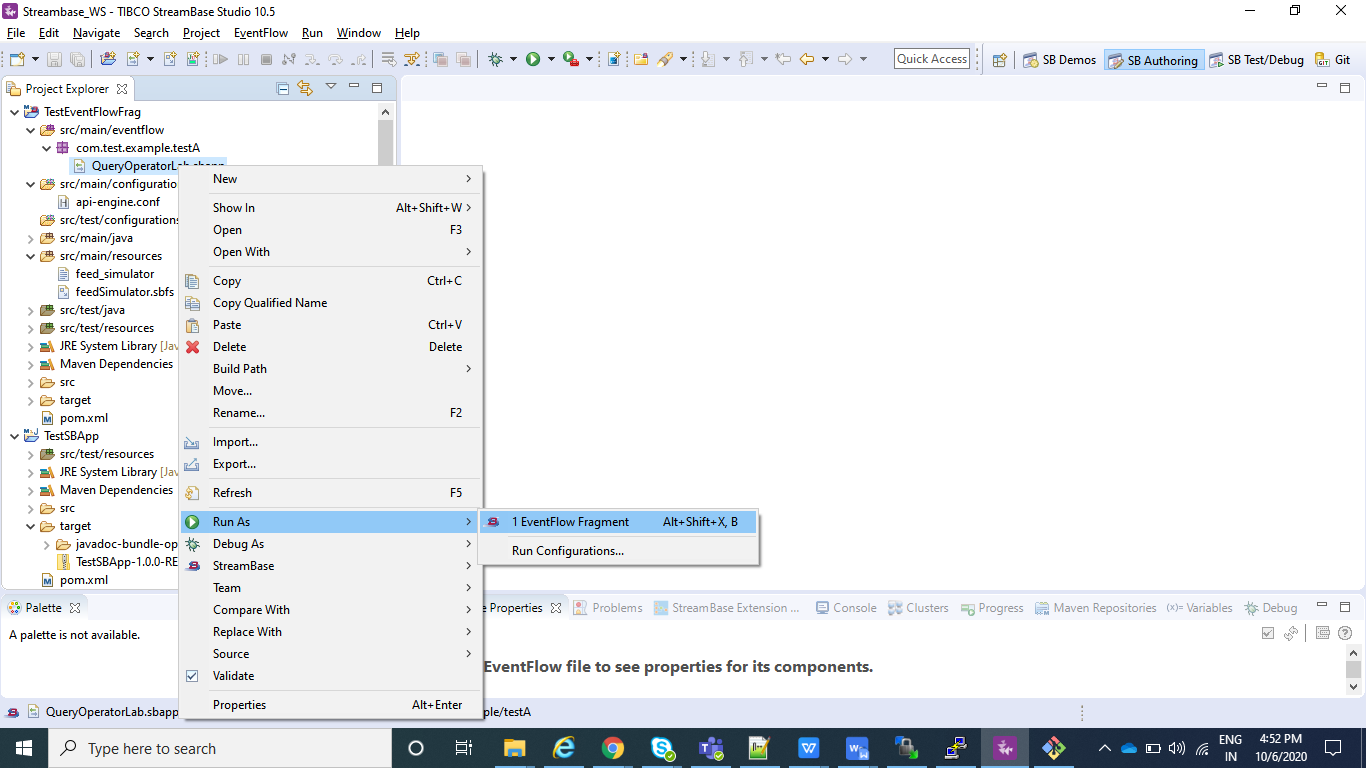
] }

}

Note : moduleName="com.test.example.testA.QueryOperatorLab" is nothing but my .sbapp which I wish to run when my application is deployed and containerName is just any name that you wish to give.



* Now make sure you do a clean install as explained above for the EventFlowFragment(TestEventFlowFrag in my case) and than for your Streambase Application(TestSBApp in my case) and than create Archive and Deploy the node as explained in above steps.
* In StreamBase command prompt navigate to TesteventFlowFrag project workspace location where your **feedSimulator.sbfs** is located and use **sbfeedsim** utility command and enter as below as **sbfeedsim feedSimulator.sbfs** and we can see Data from feed simulator.
* Also using the Streambase Studio we can run Feed Simulation.Just Right click on your .sbapp and run as eventFlowFragment.



* After your application is running select Feed Simulations instead of manual input as shown in below screenshot and create a new Feed Simulation and provide any name and after it is created configure in the same way we configured in above steps for feedSimulator.sbfs and save and than just do Run.

