LAB MANUAL

INDEX

S.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO.	STAFF SIGNATURE
1.	19.12.22	CREATE A LIST IN R		
2.	26.12.22	CREATE A DATAFRAME IN R		
3.	2.1.23	CREATE PROJECT IN WATSON STUDIO		
4.	9.1.23	DATA REFINERY-PREPARING THE DATA		
5.	23.1.23	VISUALIZATION DATA IN WATSON STUDIO		
6.	30.1.23	NATURAL LANGUAGE PROCESSING USING WATSON STUDIO		
7.	13.2.23	LINNER REGRESSION IN R		
8.	20.2.23	DECISION TREE IN R PROGRAMMING		

1. CREATE A LIST IN R

Aim :

To create the program for List by using R programming

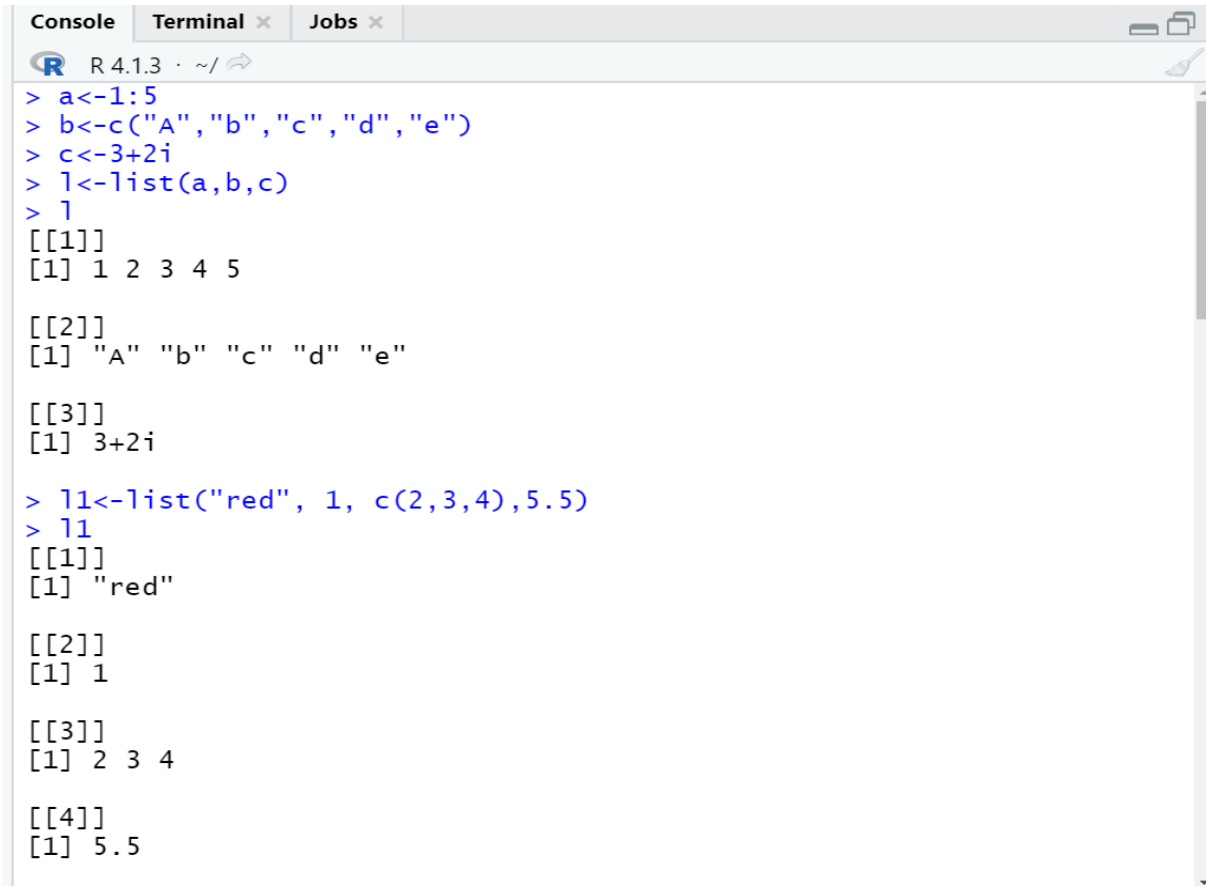
Procedure :

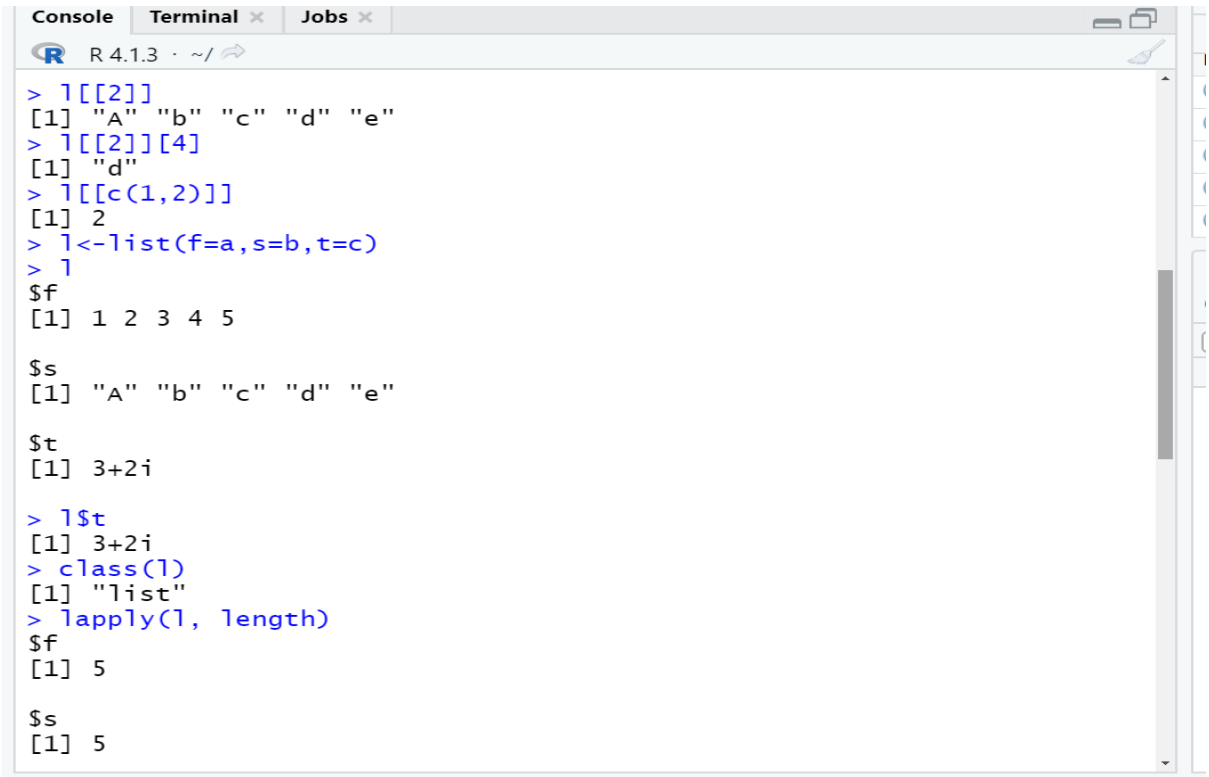
Step 1 : Open R studio – New- New script and R programming work area will get created with console and environment .
Step 2 : Create list by declaring variable in vector
Step 3: By using list () , data will get created inside List .
Step 4 : Select the element from the list by index value .
Step 5 : To find the length use Len() and more list function to atchive List operations.
Step 6: To merge two list we use merge.List
Step 7: Save the program
Step 8 : Select the entire code and click run .

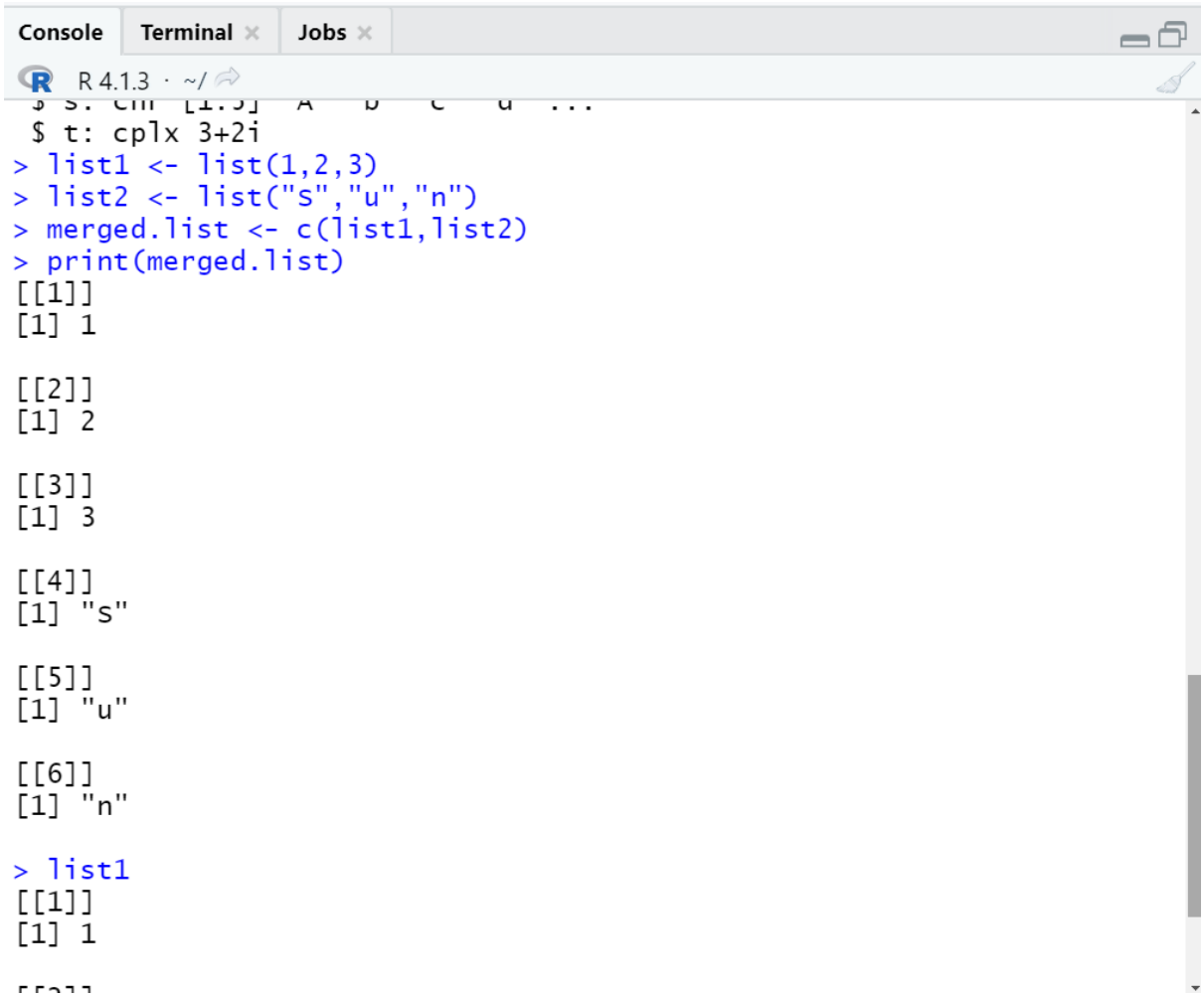
Program:

```
# Creating list
a<-1:5
b<-c("A","b","c","d","e")
c<-3+2i
l<-list(a,b,c)
l
l1<-list("red", 1, c(2,3,4),5.5)
l1
#selecting Elements from list
l[[2]]
l[[2]][4]
l[[c(1,2)]]
l<-list(f=a,s=b,t=c)
l
class(l)
lapply(l, length)
mode(l)
str(l)
list1 <- list(1,2,3)
list2 <- list("S","u","n")
merged.list<- c(list1,list2)
print(merged.list)
list1
```

Output:


Console
Terminal
Jobs
R 4.1.3 · ~/
> a<-1:5
> b<-c("A","b","c","d","e")
> c<-3+2i
> l<-list(a,b,c)
> l
[[1]]
[1] 1 2 3 4 5
[[2]]
[1] "A" "b" "c" "d" "e"
[[3]]
[1] 3+2i
> l1<-list("red", 1, c(2,3,4),5.5)
> l1
[[1]]
[1] "red"
[[2]]
[1] 1
[[3]]
[1] 2 3 4
[[4]]
[1] 5.5


Console
Terminal
Jobs
R 4.1.3 · ~/
> l[[2]]
[1] "A" "b" "c" "d" "e"
> l[[2]][4]
[1] "d"
> l[[c(1,2)]]
[1] 2
> l<-list(f=a,s=b,t=c)
> l
$f
[1] 1 2 3 4 5
$s
[1] "A" "b" "c" "d" "e"
$t
[1] 3+2i
> l$t
[1] 3+2i
> class(l)
[1] "list"
> lapply(l, length)
$f
[1] 5
$s
[1] 5



Result :

Thus, the above program was successfully executed.

2.CREATE A DATAFRAME IN R

Aim :

To create the program for Data frame by using R programming

Procedure :

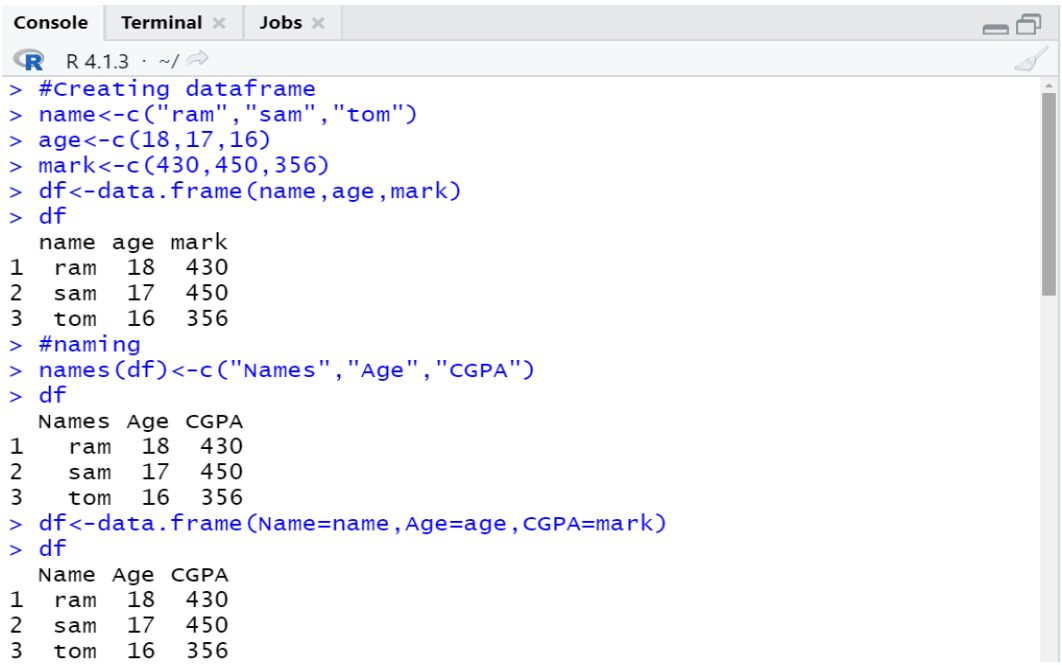
Step 1 : Open R studio – New- New script and R programming work area will get created with console and environment .
Step 2 : Create vector by c( ), pass the value of same datatype
Step 3: By using data. frame() , can convert the List to Dataframe with rows and column .
Step 4 : To find the mid value use the index [] value that comes in column .
Step 5 : rbind() , we can add the row of the dataframe
Step 6: Adding and sorting operations can also be done
Step 7: Save the program
Step 8 : Select the entire code and click run .

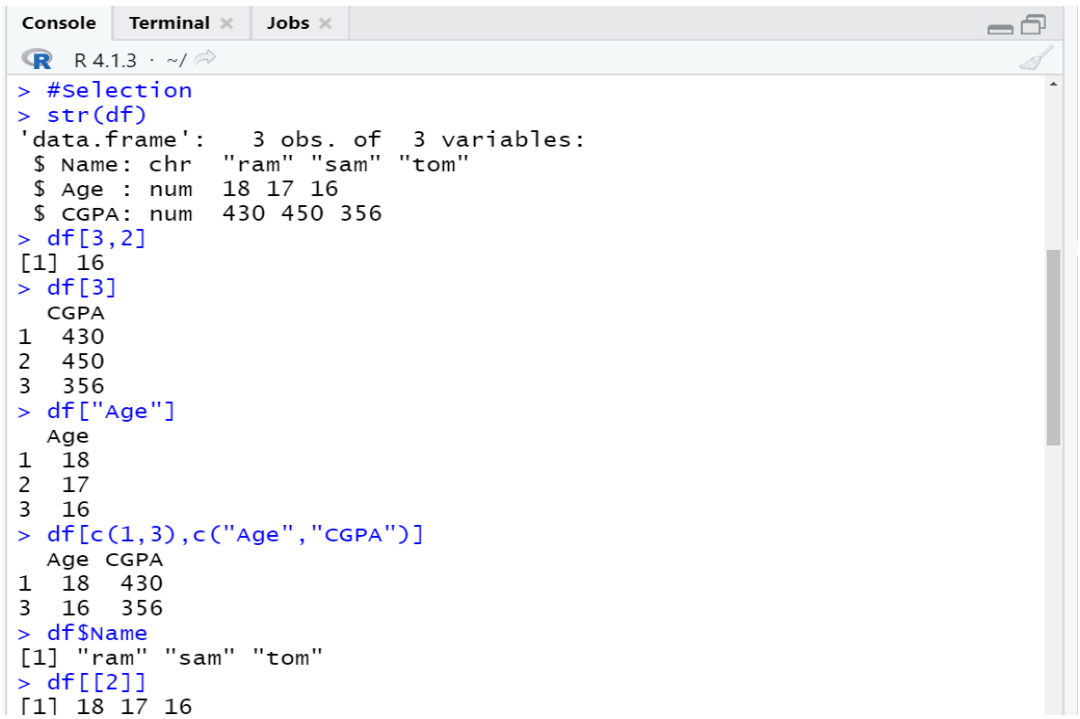
Program code:

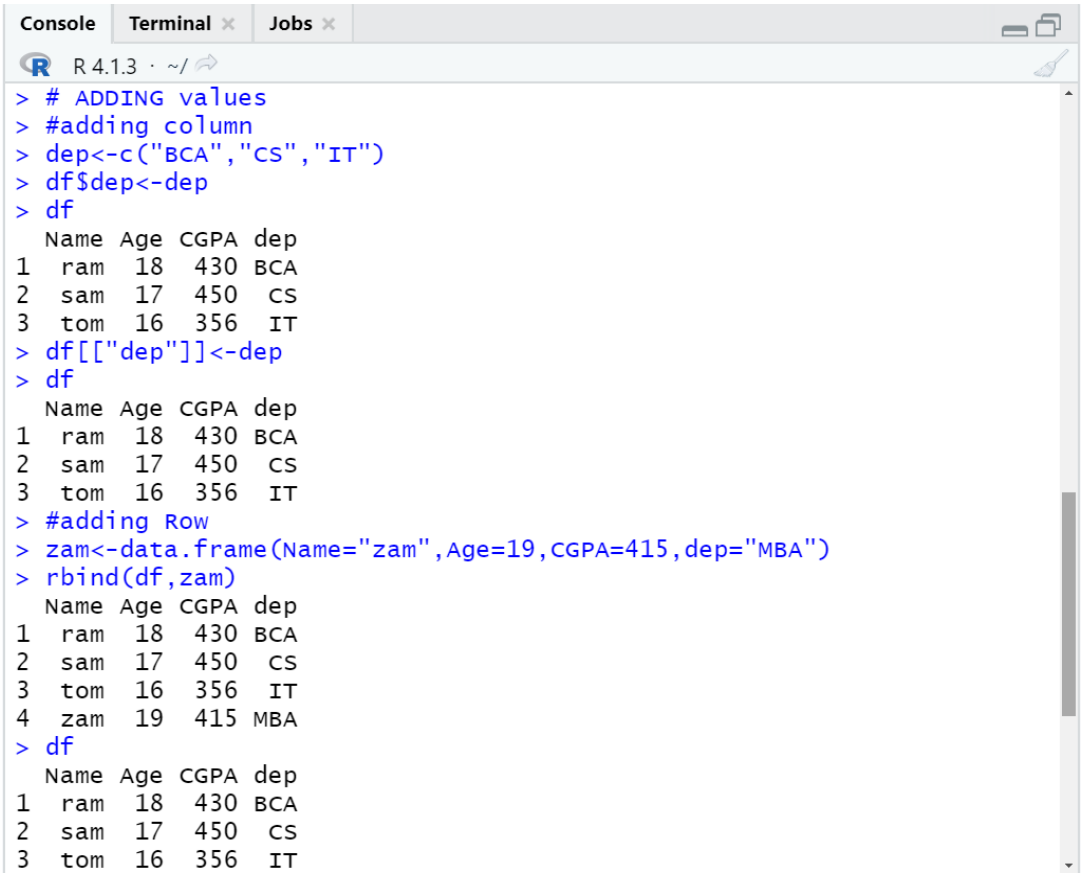
```
#Creating dataframe
name<-c("ram","sam","tom")
age<-c(18,17,16)
mark<-c(430,450,356)
df<-data.frame(name,age,mark)
df
#naming
names(df)<-c("Names","Age","CGPA")
df
df<-data.frame(Name=name,Age=age,CGPA=mark)
df
#Selection
str(df)
df[3,2]
df[3]
df["Age"]
df[c(1,3),c("Age","CGPA")]
df$Name
df[[2]]
# ADDING values
#adding column
dep<-c("BCA","CS","IT")
df$dep<-dep
df
df[["dep"]]<-dep
```

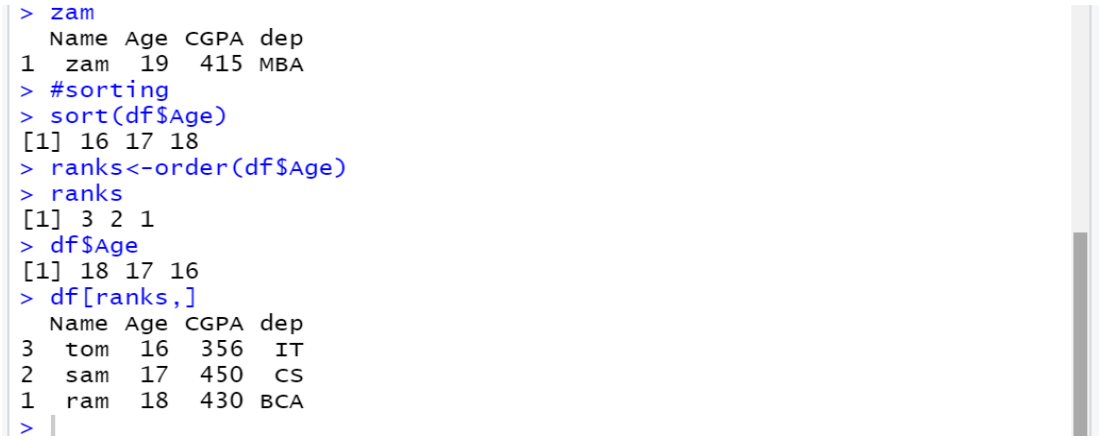
```
df
#adding Row
zam<-data.frame(Name="zam",Age=19,CGPA=415,dep="MBA")
rbind(df,zam)
df
zam
#sorting
sort(df$Age)
ranks<-order(df$Age)
ranks
df$Age
df[ranks,]
```

Output:


Console
Terminal
Jobs
R 4.1.3 · ~/
> #Creating dataframe
> name<-c("ram","sam","tom")
> age<-c(18,17,16)
> mark<-c(430,450,356)
> df<-data.frame(name,age,mark)
> df
name age mark
1 ram 18 430
2 sam 17 450
3 tom 16 356
> #naming
> names(df)<-c("Names","Age","CGPA")
> df
Names Age CGPA
1 ram 18 430
2 sam 17 450
3 tom 16 356
> df<-data.frame(Name=name,Age=age,CGPA=mark)
> df
Name Age CGPA
1 ram 18 430
2 sam 17 450
3 tom 16 356


Console
Terminal
Jobs
R 4.1.3 · ~/
> #Selection
> str(df)
'data.frame': 3 obs. of 3 variables:
$ Name: chr "ram" "sam" "tom"
$ Age : num 18 17 16
$ CGPA: num 430 450 356
> df[3,2]
[1] 16
> df[3]
CGPA
1 430
2 450
3 356
> df["Age"]
Age
1 18
2 17
3 16
> df[c(1,3),c("Age","CGPA")]
Age CGPA
1 18 430
3 16 356
> df$Name
[1] "ram" "sam" "tom"
> df[[2]]
[1] 18 17 16





Result :

Thus, the above program was successfully executed

3.CREATE PROJECT IN WATSON STUDIO

Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

Step 2: Click **New project** on the home page or on your **Projects** page.

Step 3: Choose whether to create an empty project or to create a project based on an exported project file or a sample project.

Step 4: On the New project screen, add a name and optional description for the project.

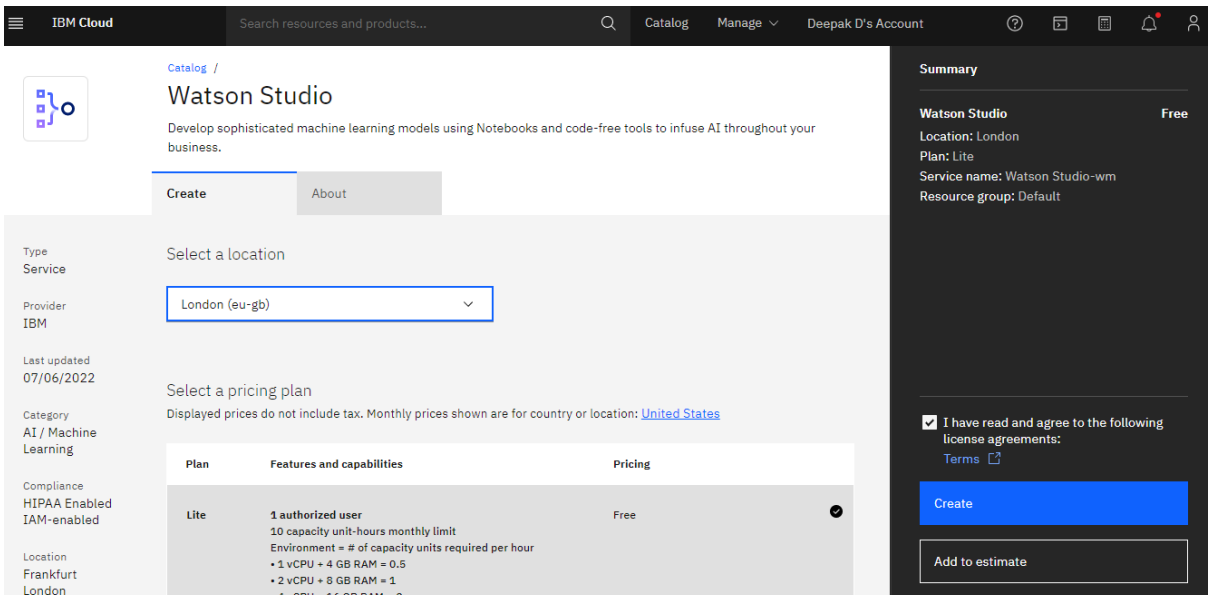
Step 5: After giving name , create the storage by clicking Define storage .

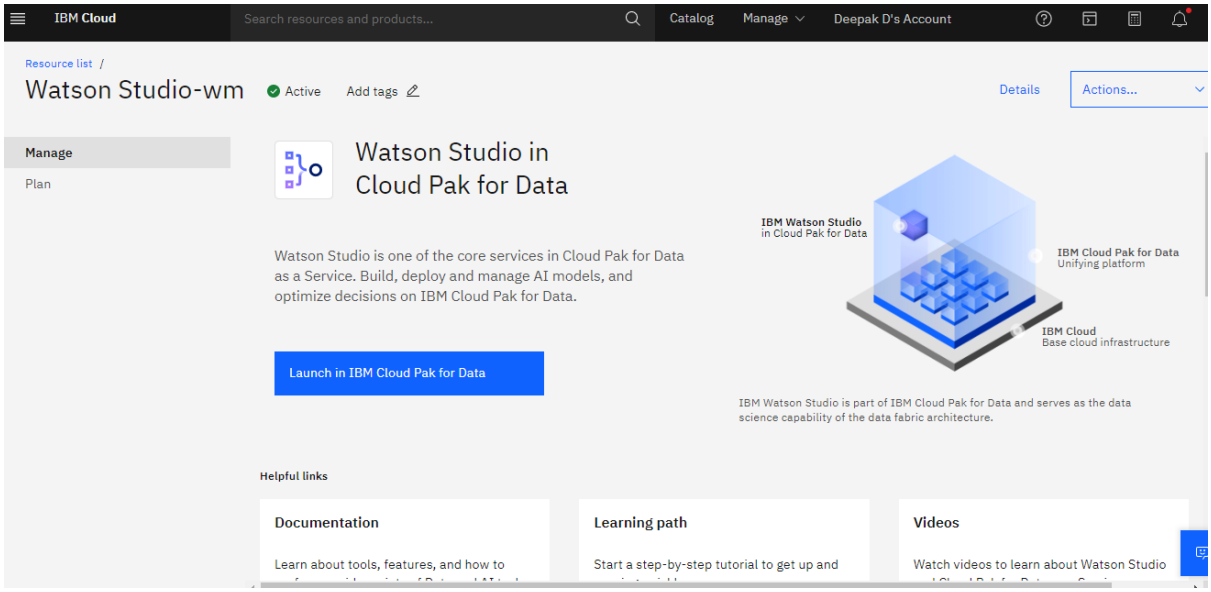
Step 6: Click Lite plan as it is free

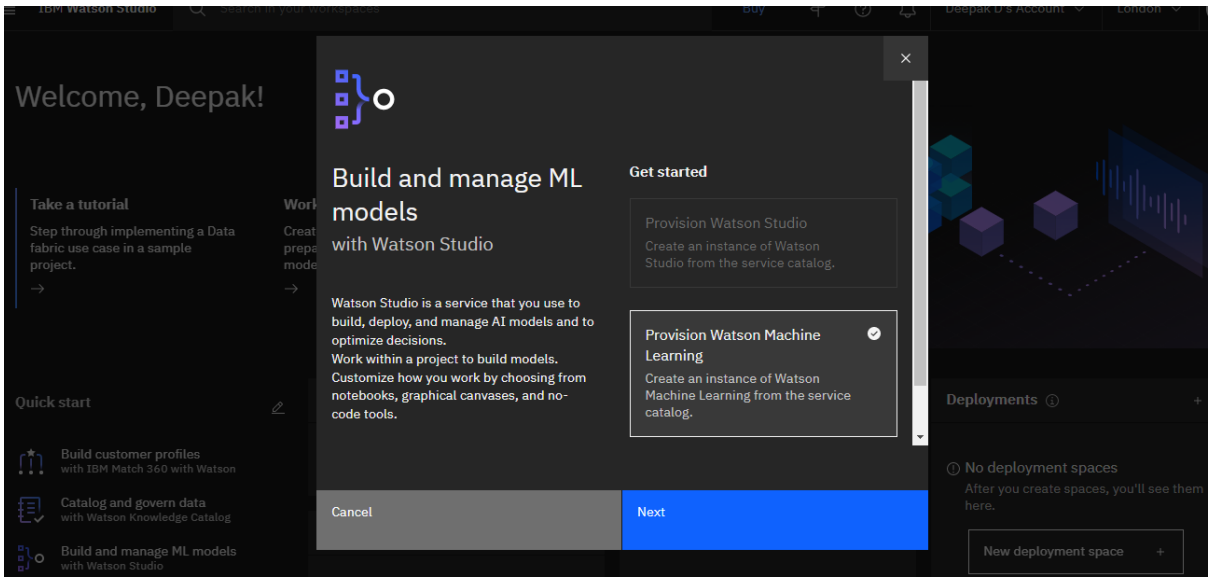
Step 7 : click create at the bottom of the page

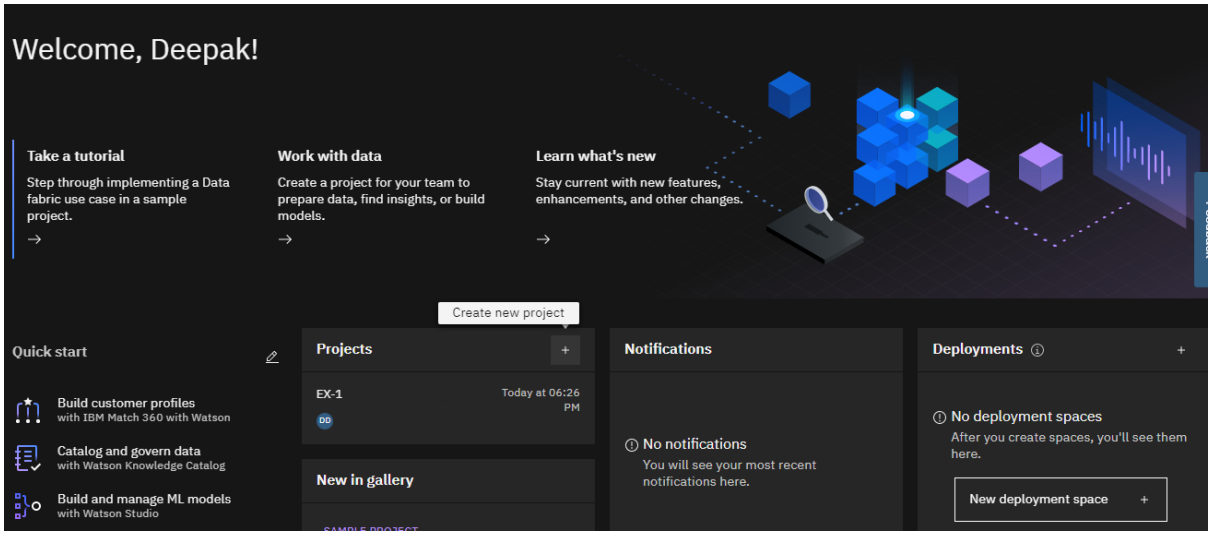
Step 8: Finally , Your project was created .

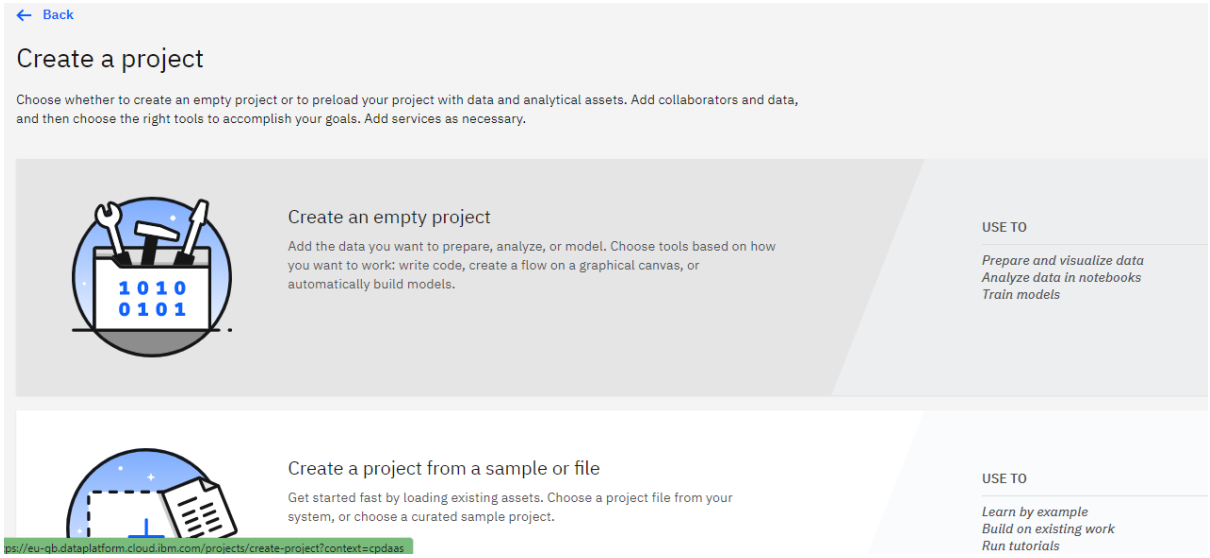
Output screen short:

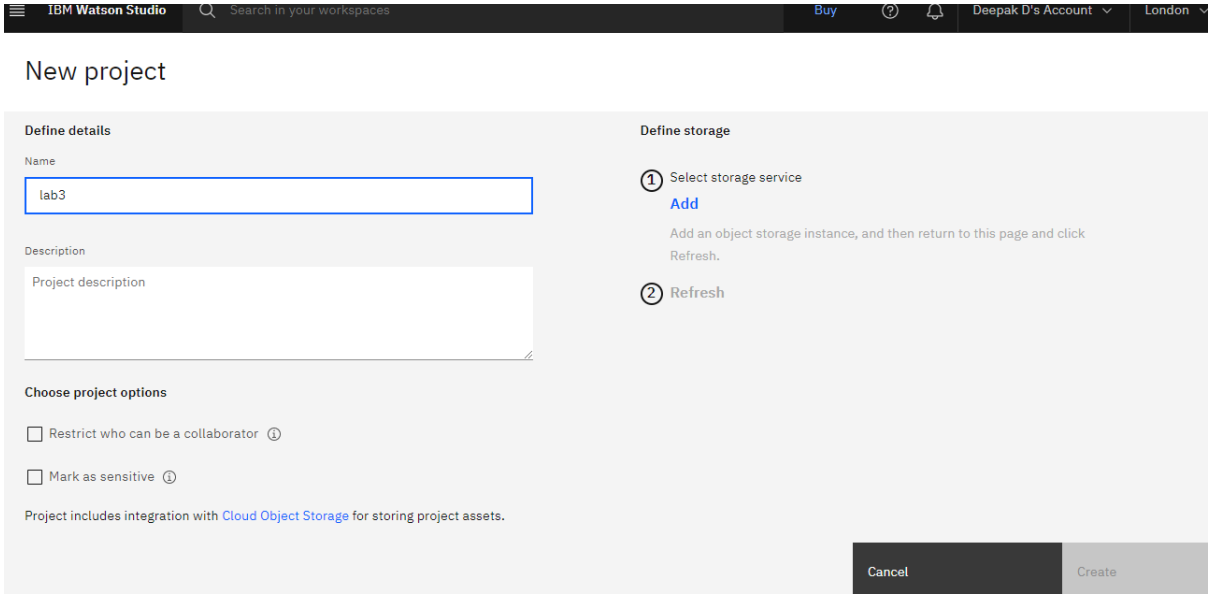


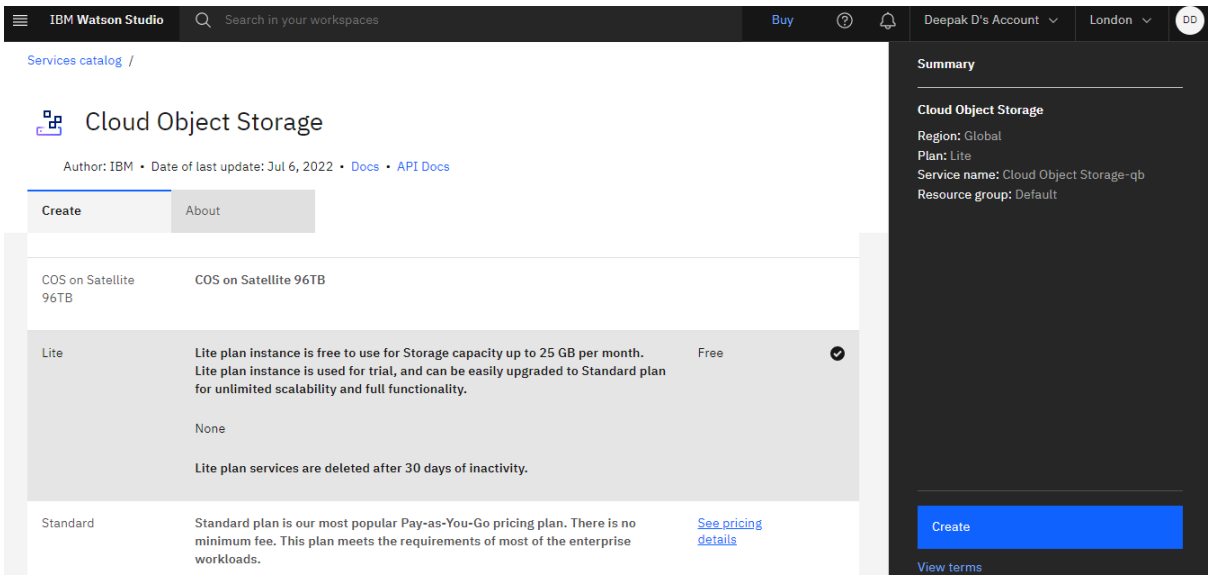

IBM Cloud
Search resources and products...
Catalog
Manage
Deepak D's Account
Resource list /
Watson Studio-wm
Active
Add tags
Details
Actions...
Manage
Plan
Watson Studio in Cloud Pak for Data
Watson Studio is one of the core services in Cloud Pak for Data as a Service. Build, deploy and manage AI models, and optimize decisions on IBM Cloud Pak for Data.
Launch in IBM Cloud Pak for Data
IBM Watson Studio in Cloud Pak for Data
IBM Cloud Pak for Data Unifying platform
IBM Cloud Base cloud infrastructure
IBM Watson Studio is part of IBM Cloud Pak for Data and serves as the data science capability of the data fabric architecture.
Helpful links
Documentation
Learning path
Videos

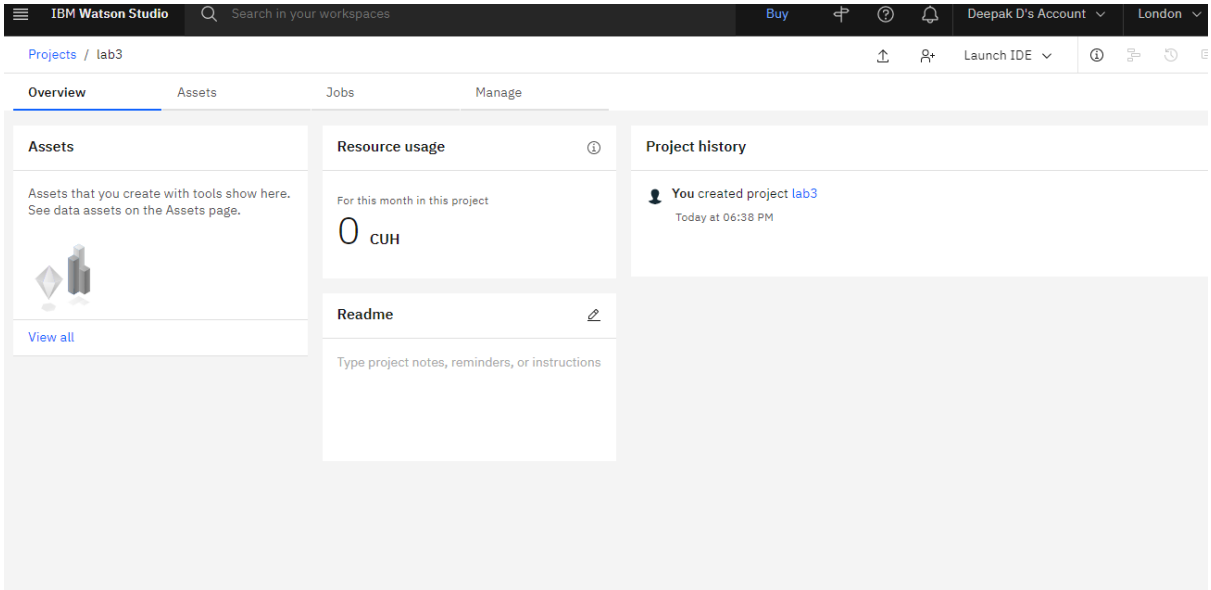

Welcome, Deepak!
Take a tutorial
Step through implementing a Data fabric use case in a sample project.
Quick start
Build customer profiles with IBM Match 360 with Watson
Catalog and govern data with Watson Knowledge Catalog
Build and manage ML models with Watson Studio
Build and manage ML models with Watson Studio
Watson Studio is a service that you use to build, deploy, and manage AI models and to optimize decisions.
Work within a project to build models.
Customize how you work by choosing from notebooks, graphical canvases, and no-code tools.
Get started
Provision Watson Studio
Create an instance of Watson Studio from the service catalog.
Provision Watson Machine Learning
Create an instance of Watson Machine Learning from the service catalog.
Cancel
Next
Deployments
No deployment spaces
After you create spaces, you'll see them here.
New deployment space


Welcome, Deepak!
Take a tutorial
Step through implementing a Data fabric use case in a sample project.
Work with data
Create a project for your team to prepare data, find insights, or build models.
Learn what's new
Stay current with new features, enhancements, and other changes.
Create new project
Quick start
Build customer profiles
with IBM Match 360 with Watson
Catalog and govern data
with Watson Knowledge Catalog
Build and manage ML models
with Watson Studio
Projects
EX-1
Today at 06:26 PM
DD
New in gallery
Notifications
No notifications
You will see your most recent notifications here.
Deployments
No deployment spaces
After you create spaces, you'll see them here.
New deployment space


Back
Create a project
Choose whether to create an empty project or to preload your project with data and analytical assets. Add collaborators and data, and then choose the right tools to accomplish your goals. Add services as necessary.
1010
0101
Create an empty project
Add the data you want to prepare, analyze, or model. Choose tools based on how you want to work: write code, create a flow on a graphical canvas, or automatically build models.
USE TO
Prepare and visualize data
Analyze data in notebooks
Train models
Create a project from a sample or file
Get started fast by loading existing assets. Choose a project file from your system, or choose a curated sample project.
USE TO
Learn by example
Build on existing work
Run tutorials
ps://eu-gb.dataplatform.cloud.ibm.com/projects/create-project?context=cpdaas


IBM Watson Studio
Search in your workspaces
Buy
Deepak D's Account
London
New project
Define details
Name
lab3
Description
Project description
Choose project options
Restrict who can be a collaborator
Mark as sensitive
Project includes integration with Cloud Object Storage for storing project assets.
Define storage
1 Select storage service
Add
Add an object storage instance, and then return to this page and click Refresh.
2 Refresh
Cancel
Create


IBM Watson Studio
Search in your workspaces
Buy
Deepak D's Account
London
DD
Services catalog /
Cloud Object Storage
Author: IBM • Date of last update: Jul 6, 2022 • Docs • API Docs
Create
About
COS on Satellite 96TB
COS on Satellite 96TB
Lite
Lite plan instance is free to use for Storage capacity up to 25 GB per month. Lite plan instance is used for trial, and can be easily upgraded to Standard plan for unlimited scalability and full functionality.
None
Lite plan services are deleted after 30 days of inactivity.
Free
Standard
Standard plan is our most popular Pay-as-You-Go pricing plan. There is no minimum fee. This plan meets the requirements of most of the enterprise workloads.
See pricing details
Summary
Cloud Object Storage
Region: Global
Plan: Lite
Service name: Cloud Object Storage-qb
Resource group: Default
Create
View terms



Result :

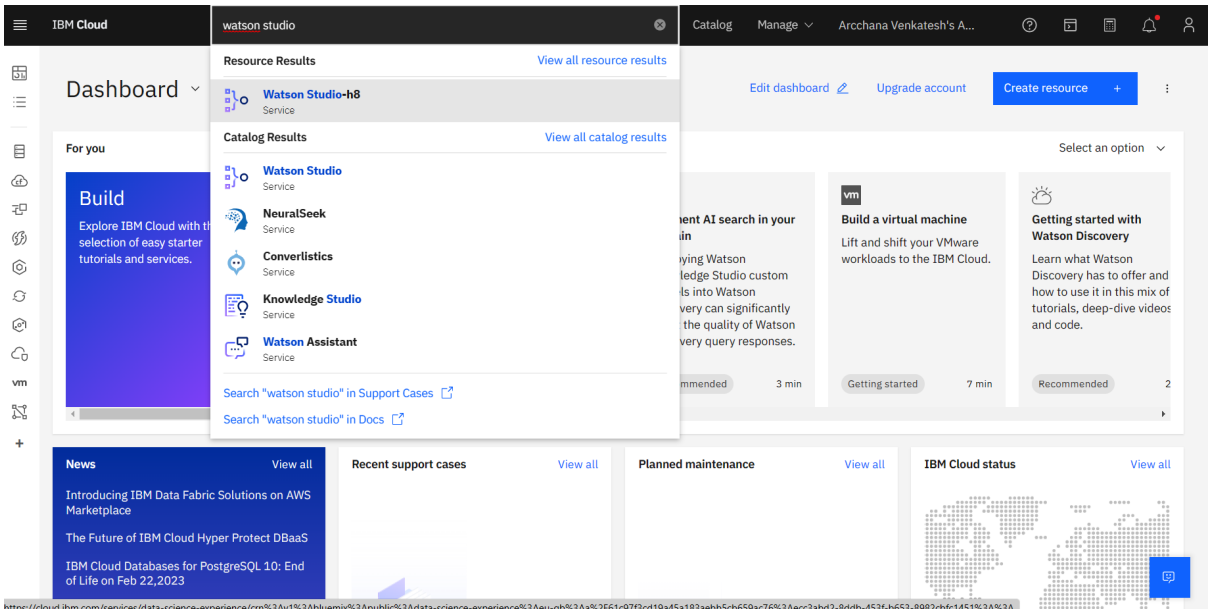
Thus, the above experiment was successfully executed

4.DATA REFINERY-PREPARING THE DATA

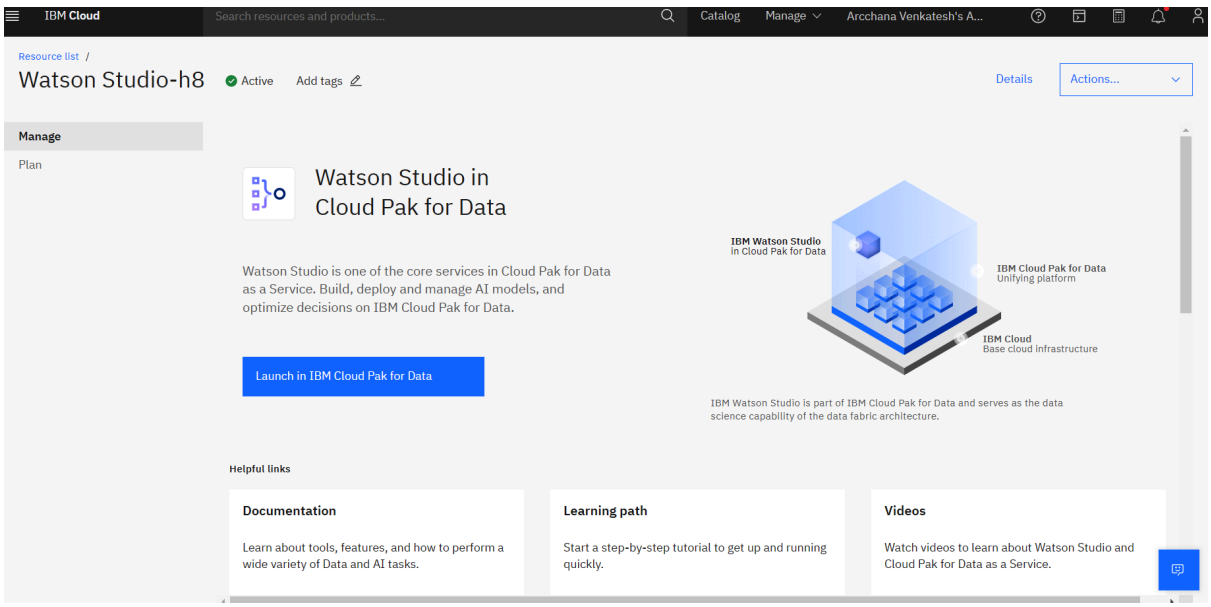
Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

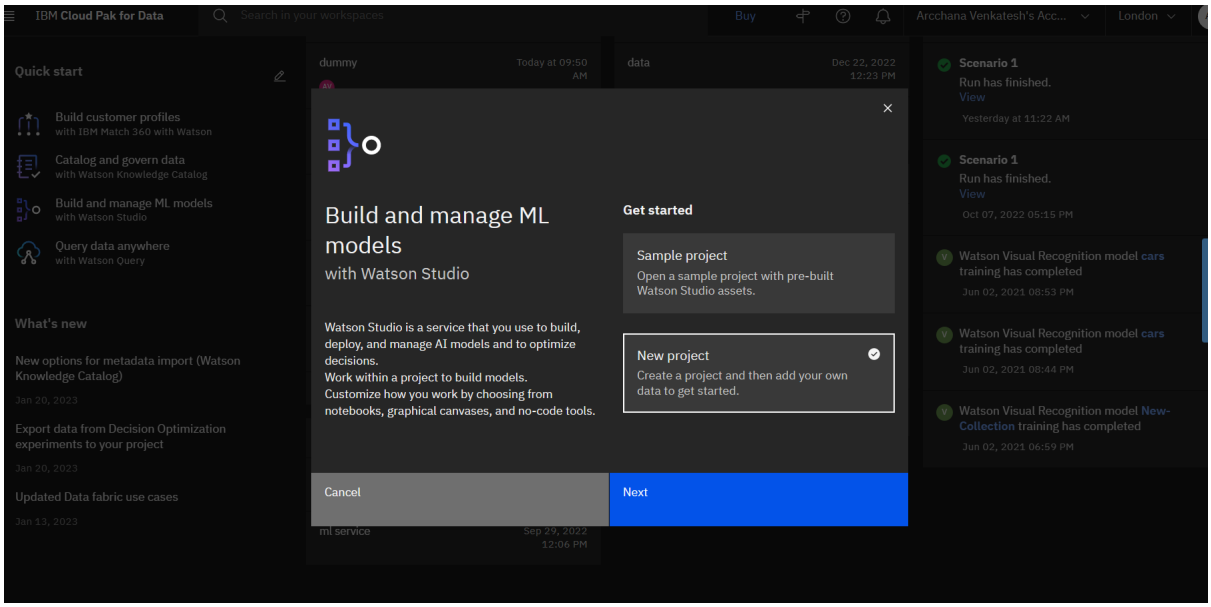
Step 1: click Watson studio h8 .



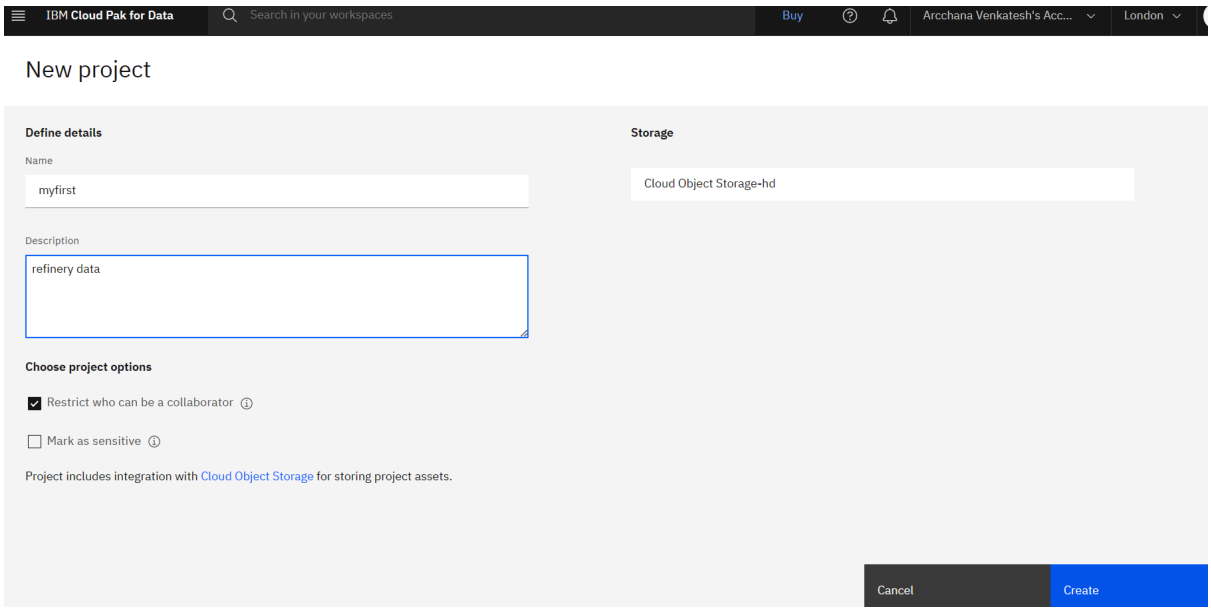
Step 3: click launch in IBM cloud park data



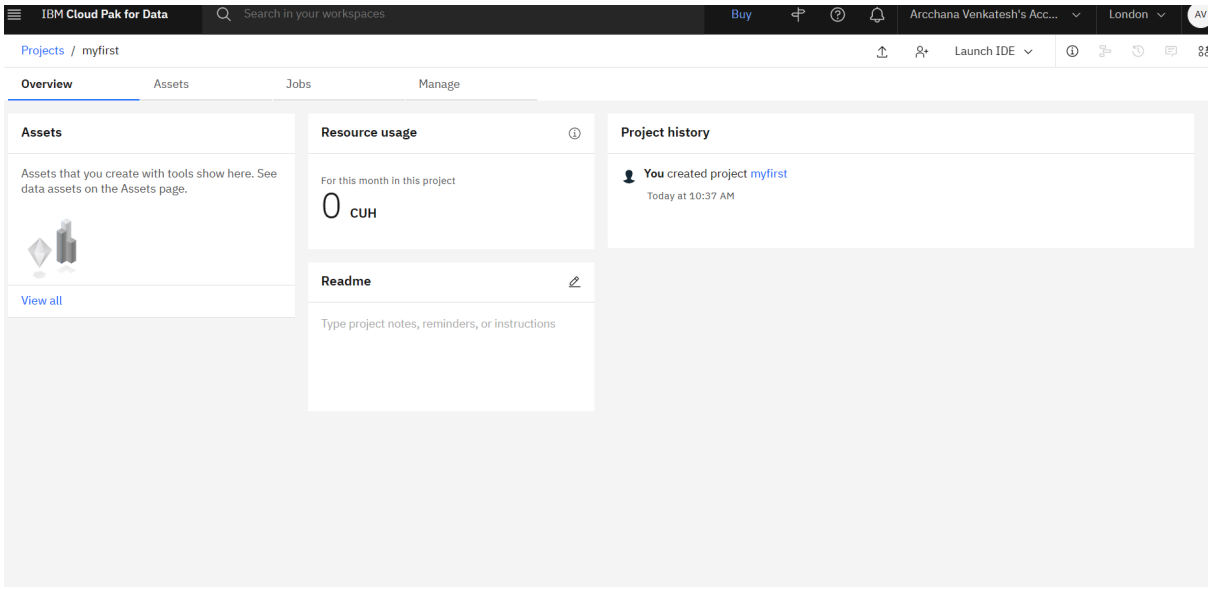
Step 4: create new project and give next



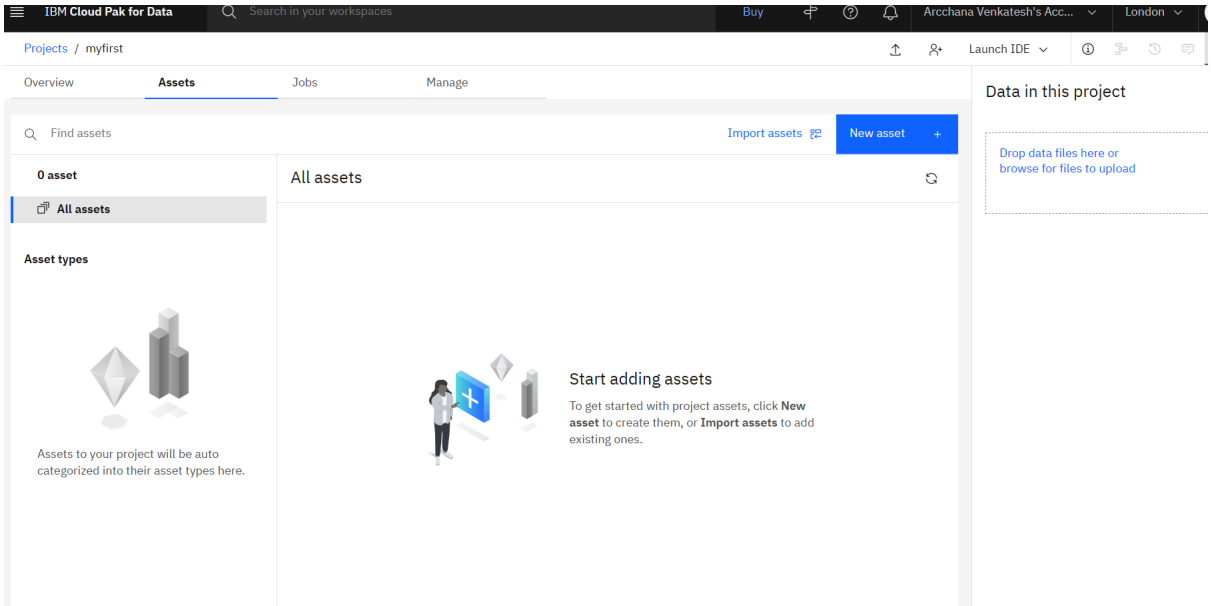
Step 5: Give project name and description and storage for your project and click create



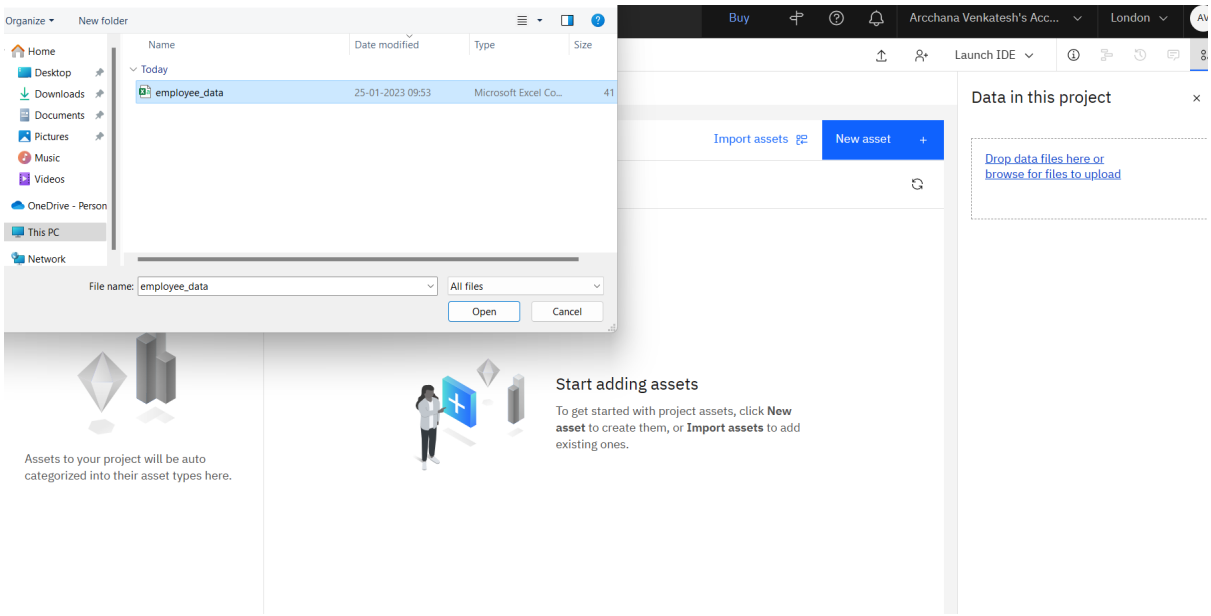
Step 6: Project created successfully .



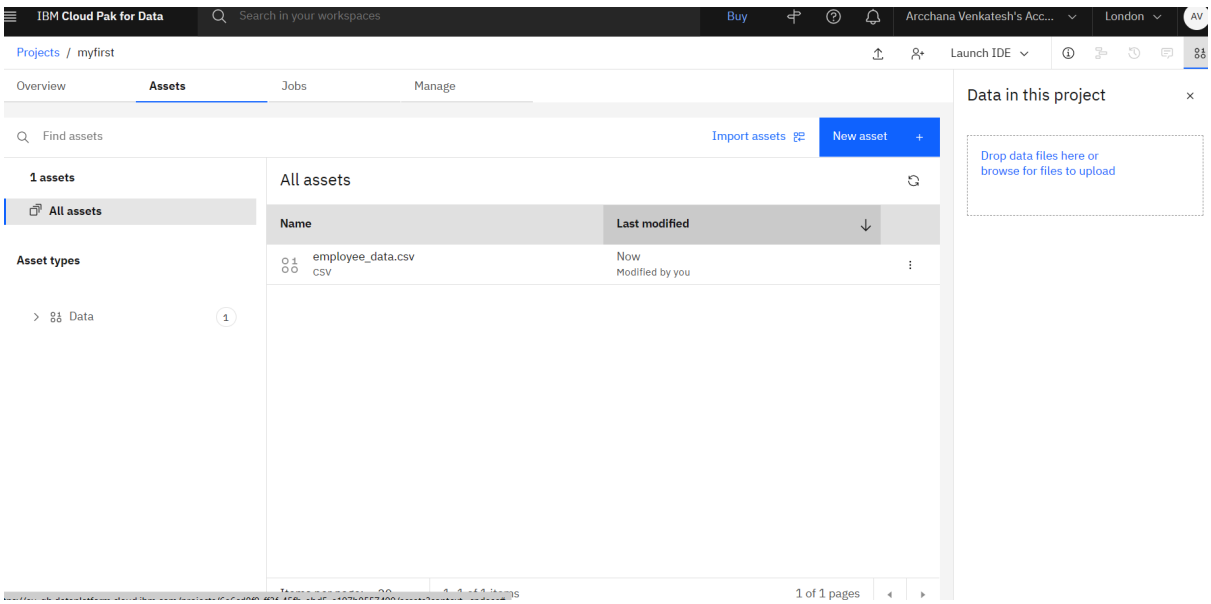
Step 7: Click assets and inset data set



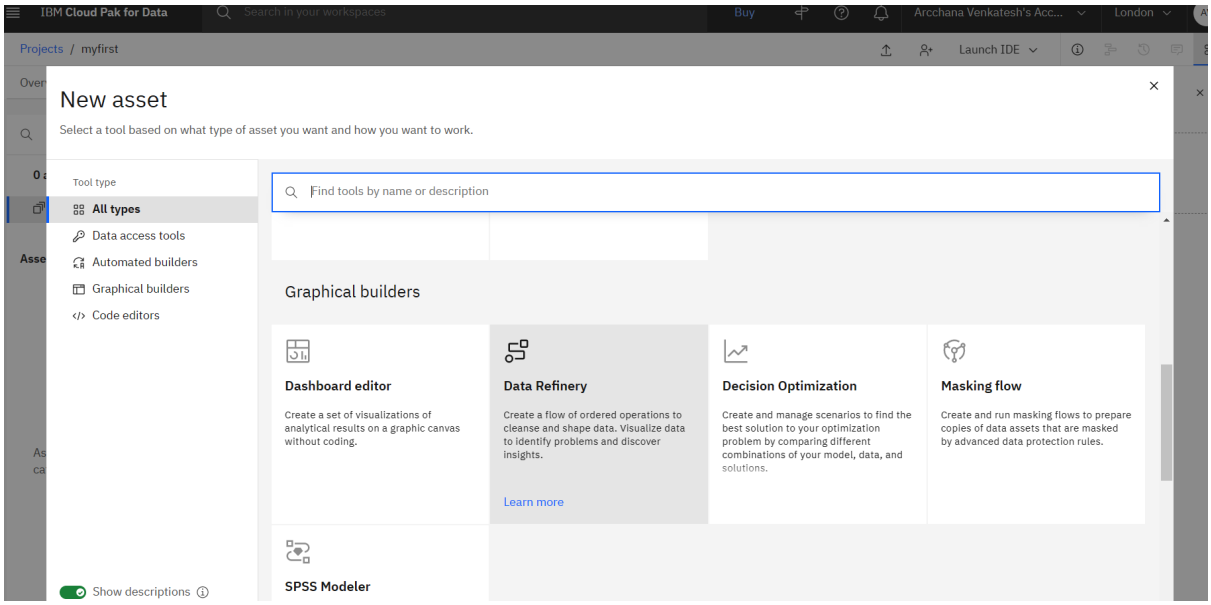
Step 8: import dataset -Employee_data with the help of link

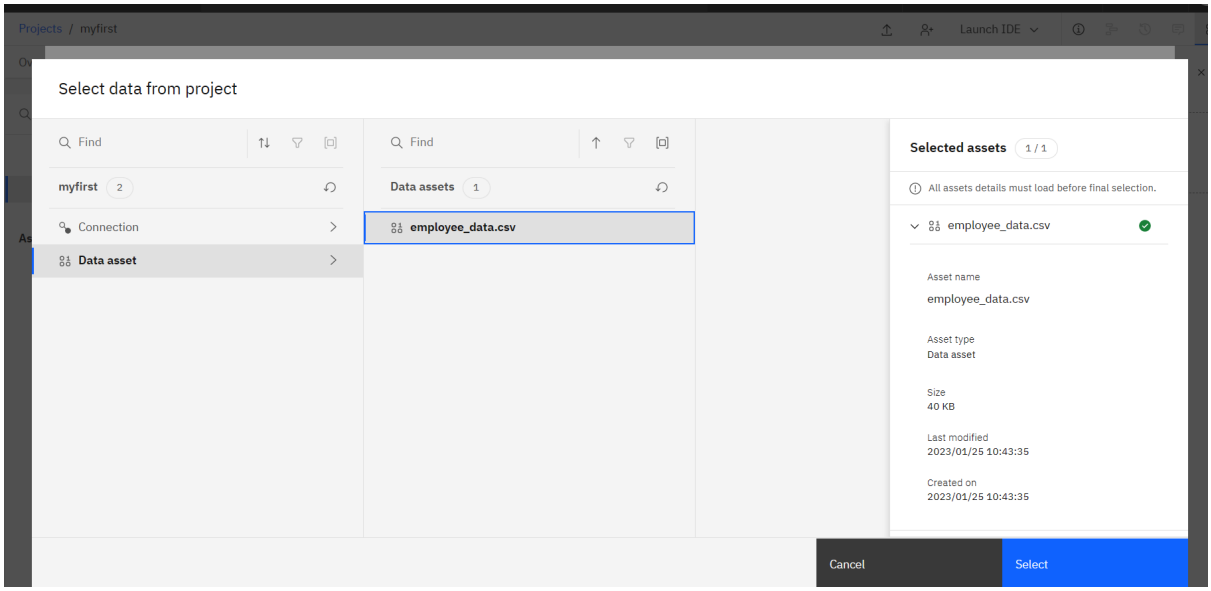


Step 9: Data set was imported successfully and click new asset .

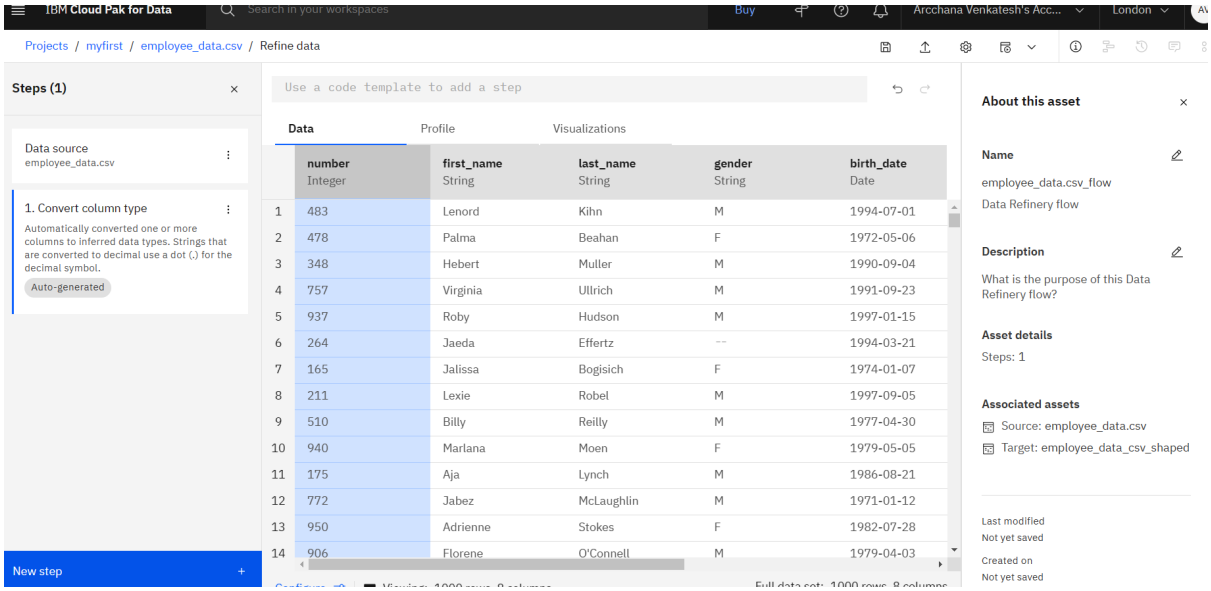


Step 10: After cliking New asset click Data Refinery and give select .

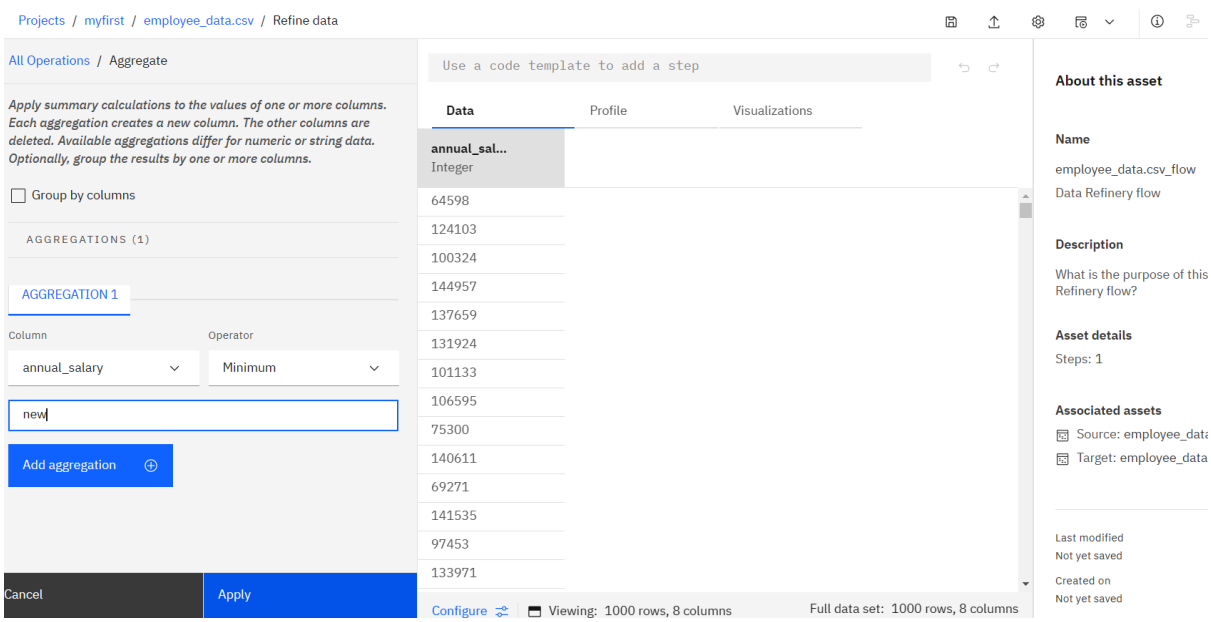




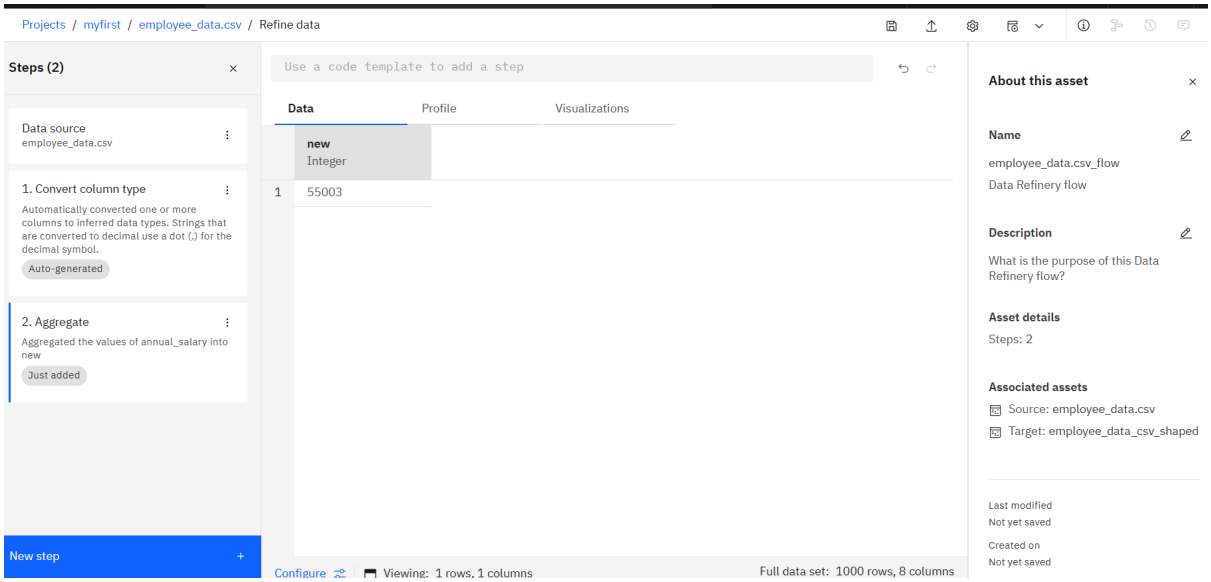
Step 11: Refinery process will start working .



Step 12: Click on a new step and aggricate and select the column and operation and create a new colunm name and give Apply .



Step 13: After Aggregating new data was refined successfully .



Result :

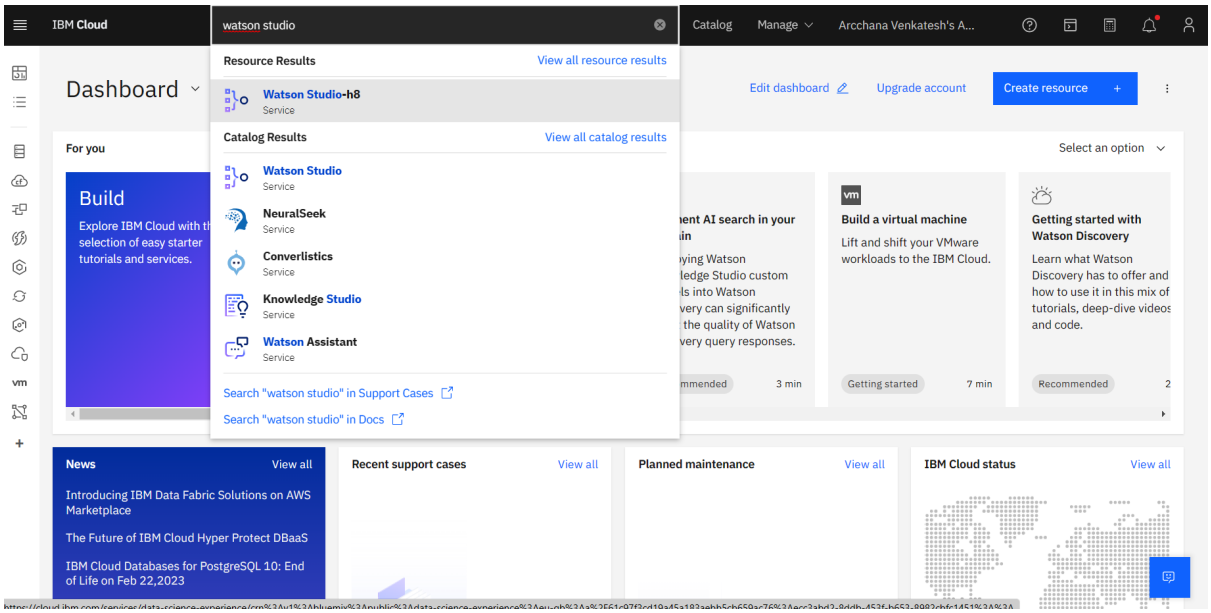
Thus, the above experiment was successfully executed

5.Visualization data in Watson studio

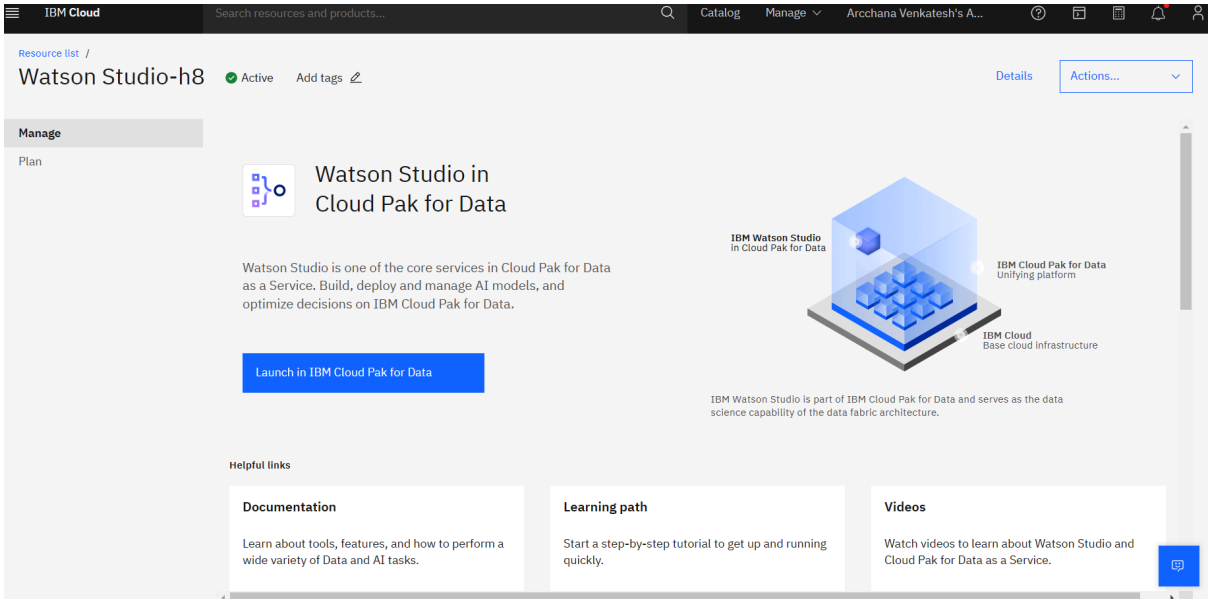
Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

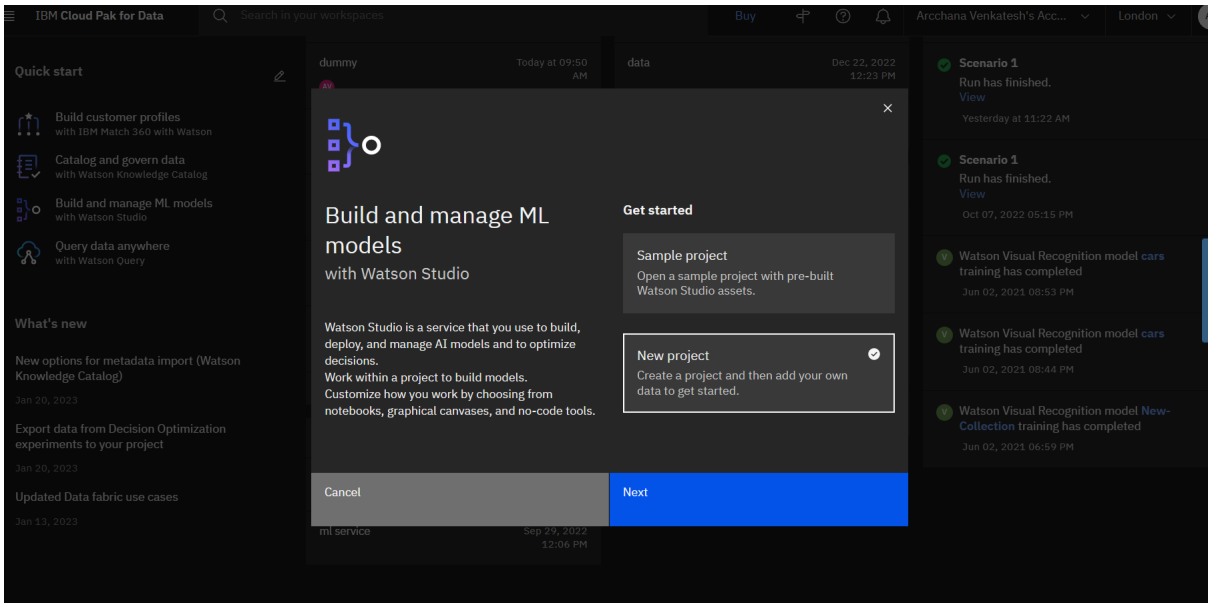
Step 1: click Watson studio h8 .



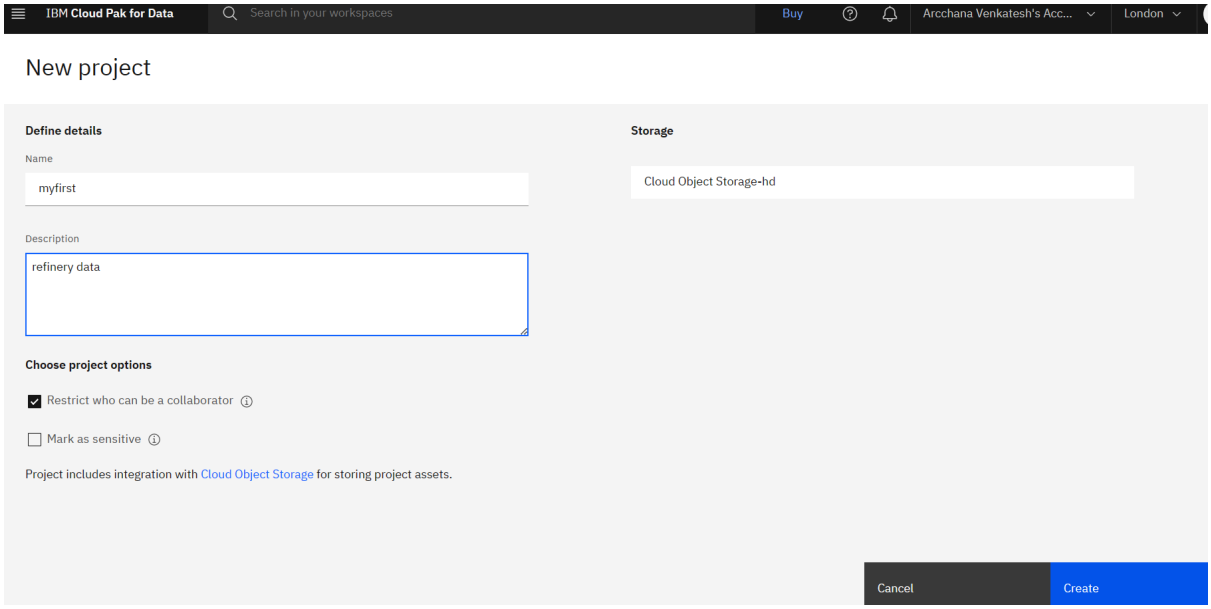
Step 3: click launch in IBM cloud park data



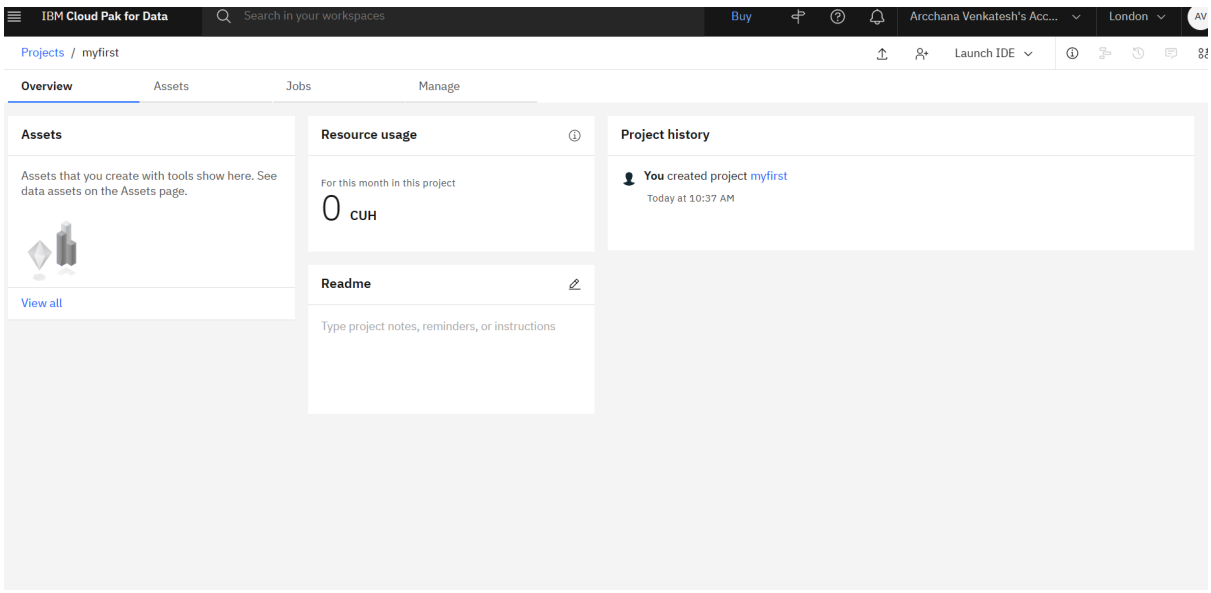
Step 4: create new project and give next



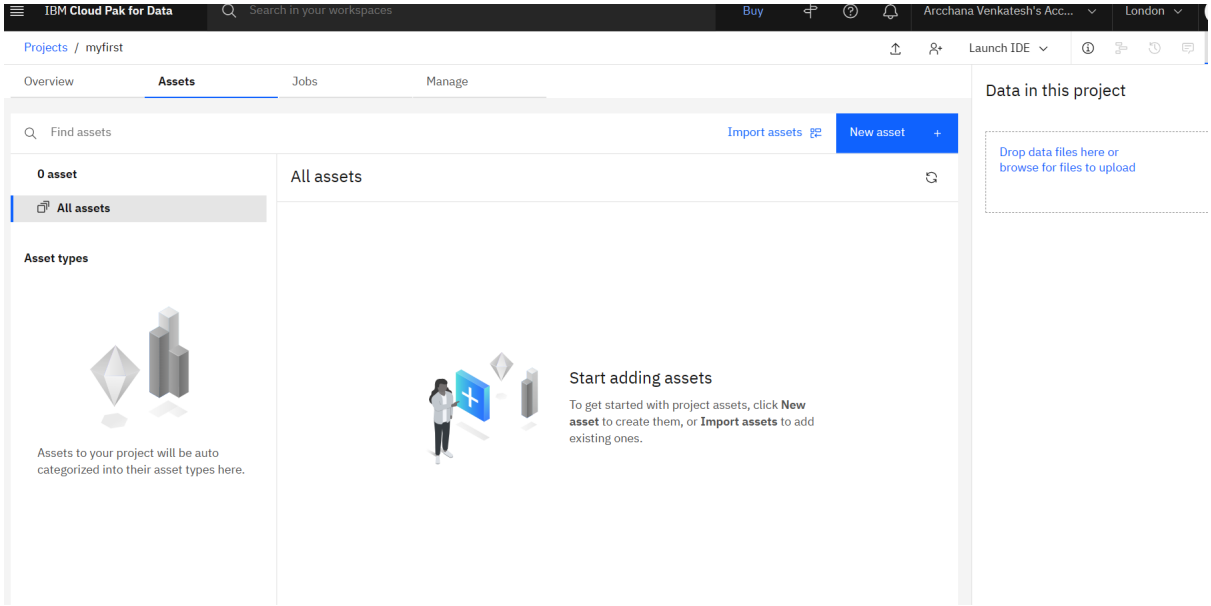
Step 5: Give project name and description and storage for your project and click create



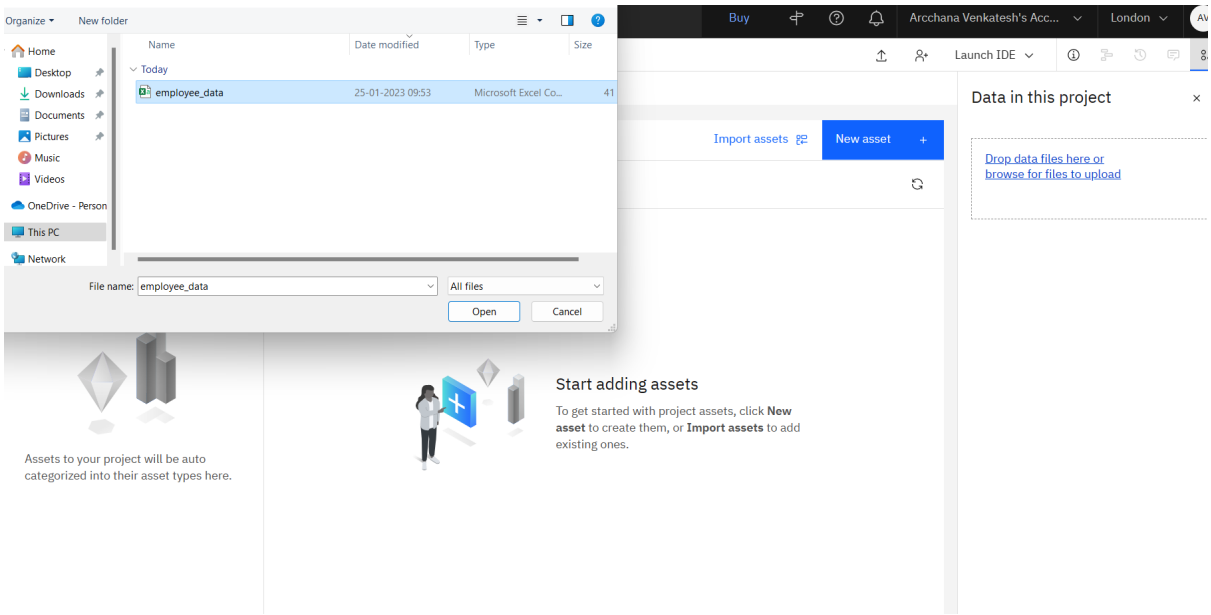
Step 6: Project created successfully .



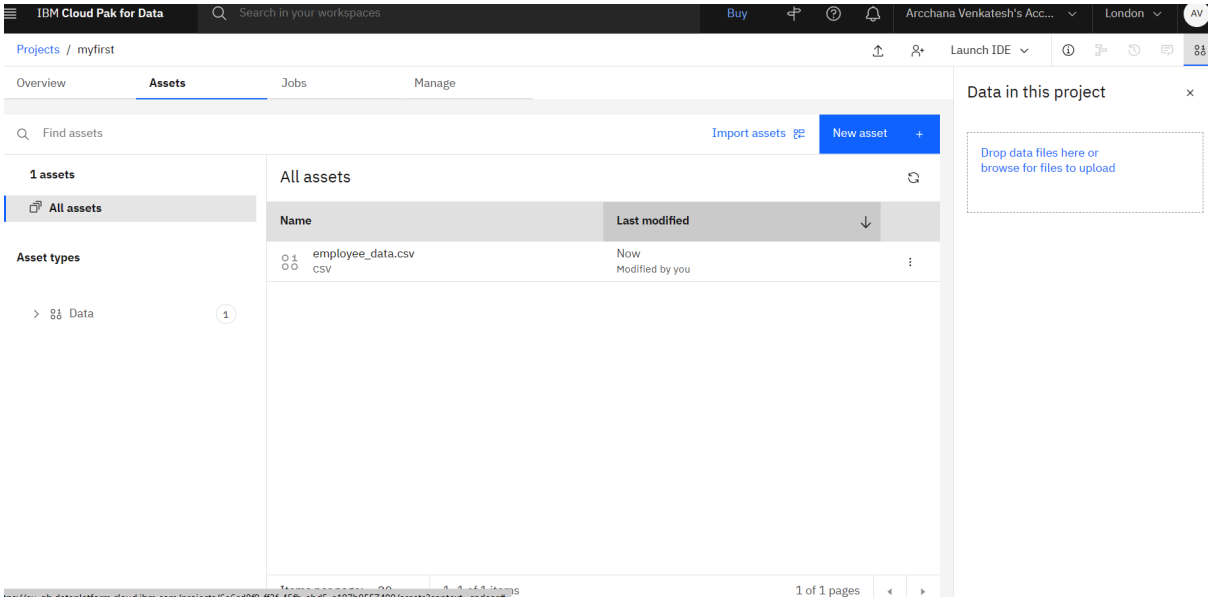
Step 7: Click assets and inset data set



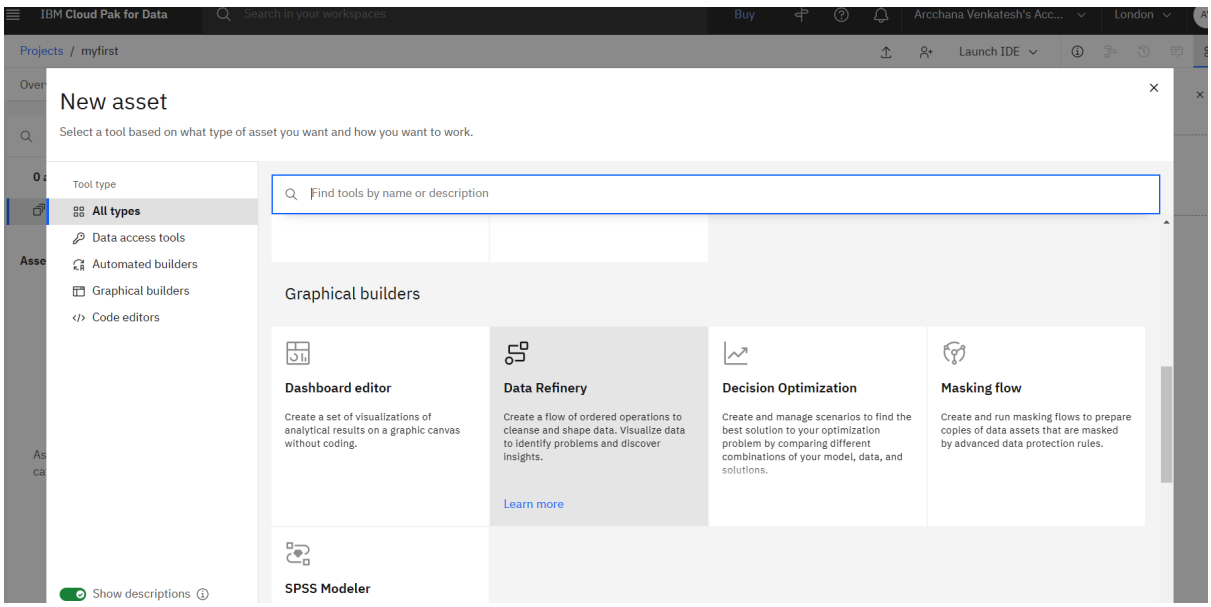
Step 8: import dataset -Employee_data with the help of link

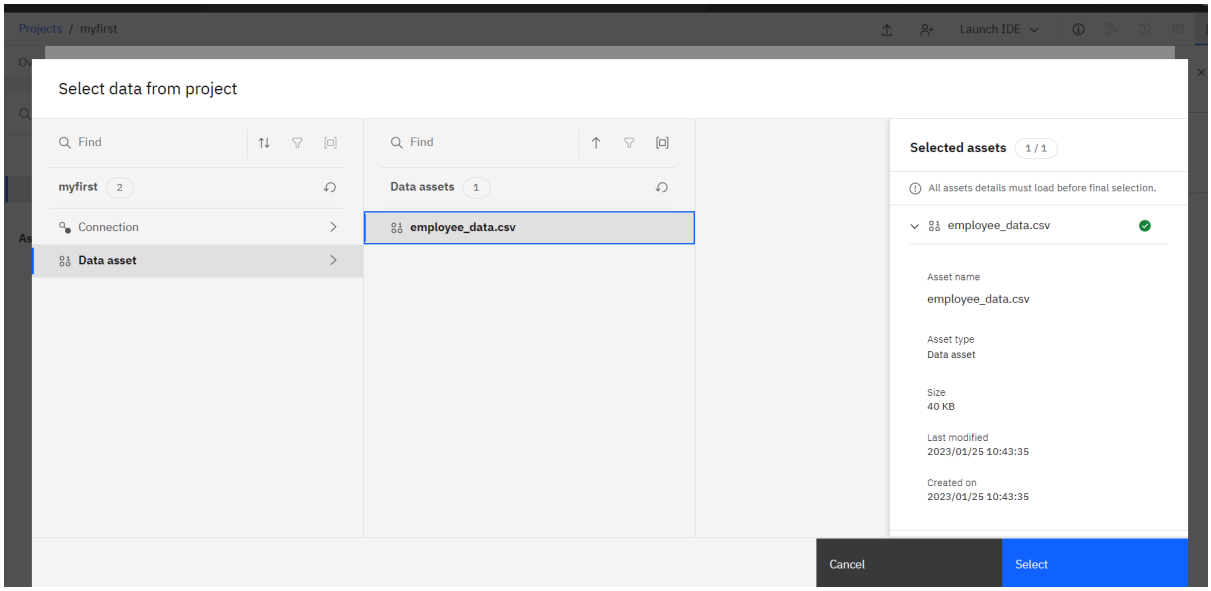


Step 9: Data set was imported successfully and click new asset .

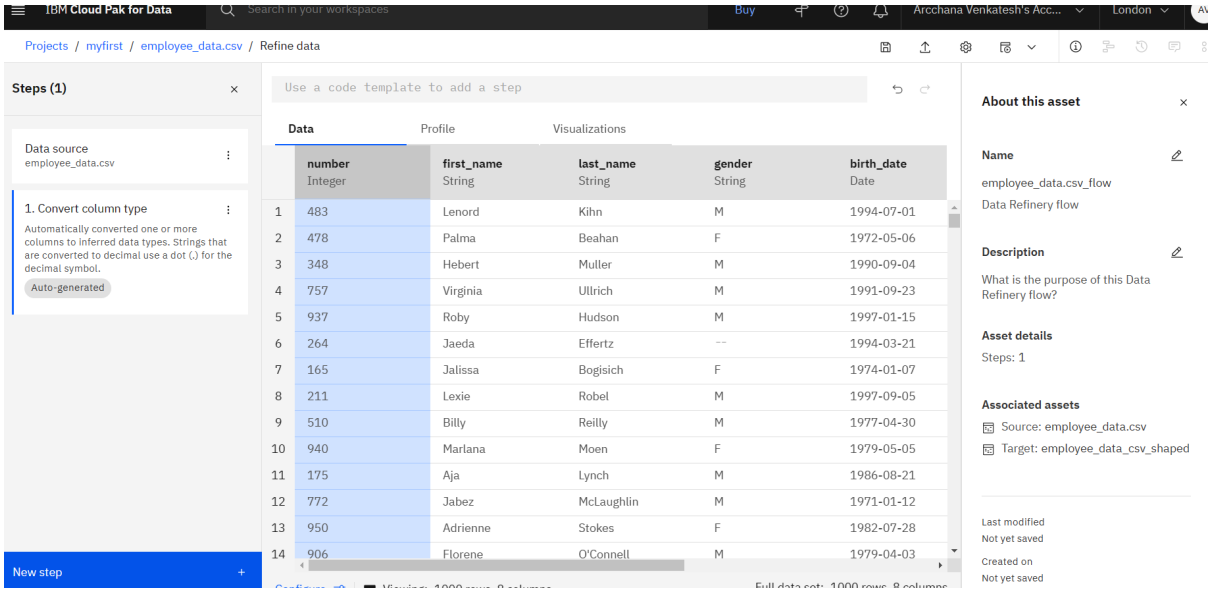


Step 10: After cliking New asset click Data Refinery and give select .

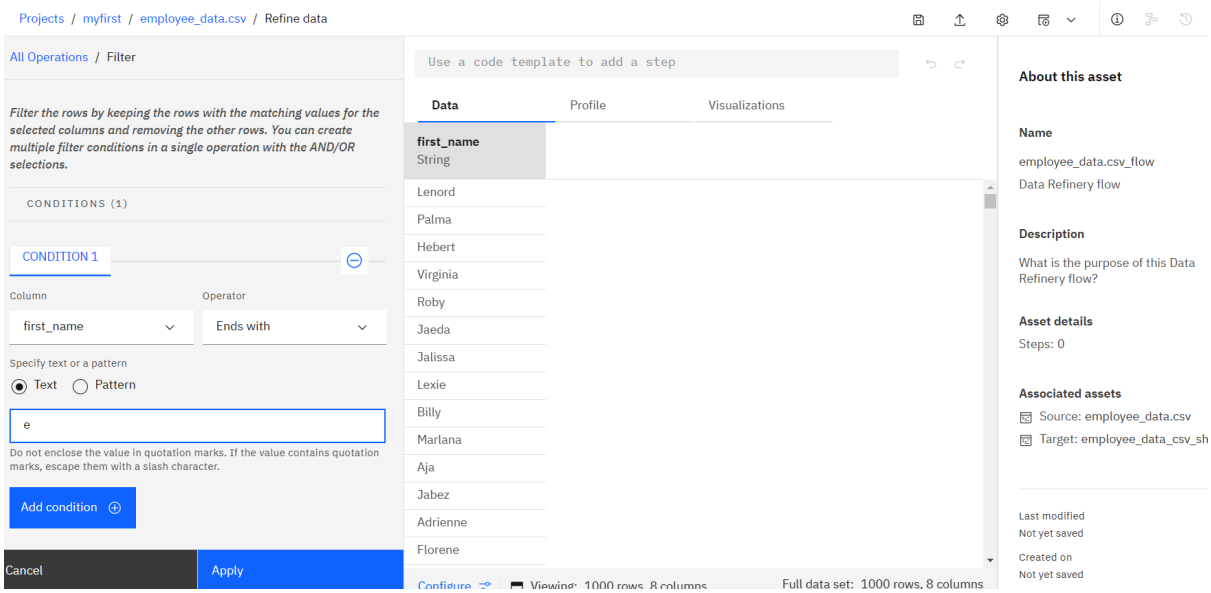




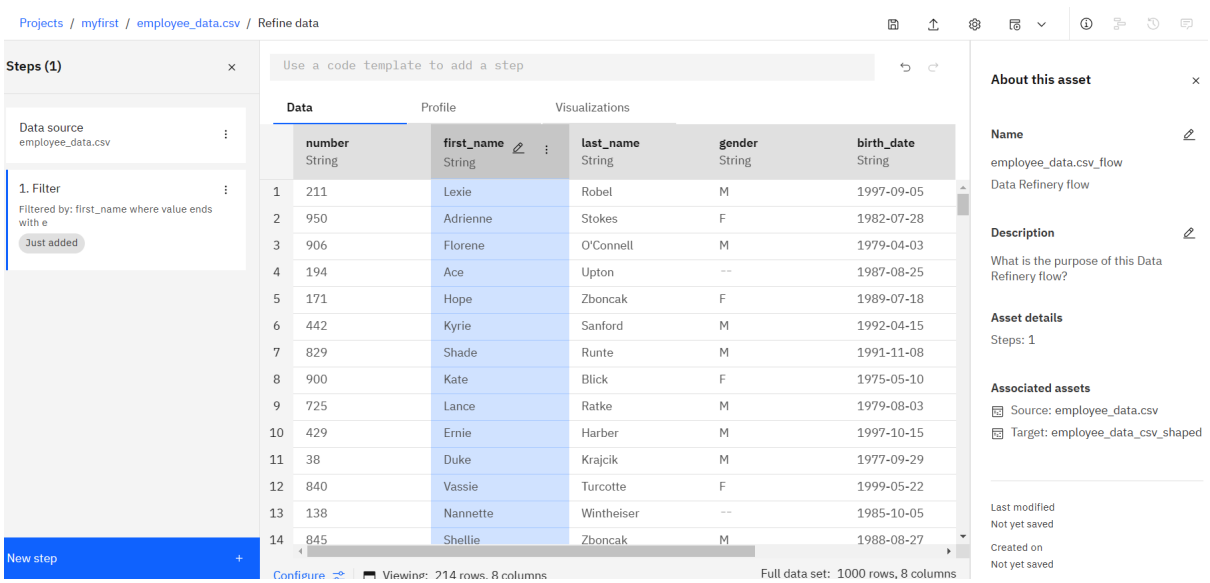
Step 11: Refinery process will start working .



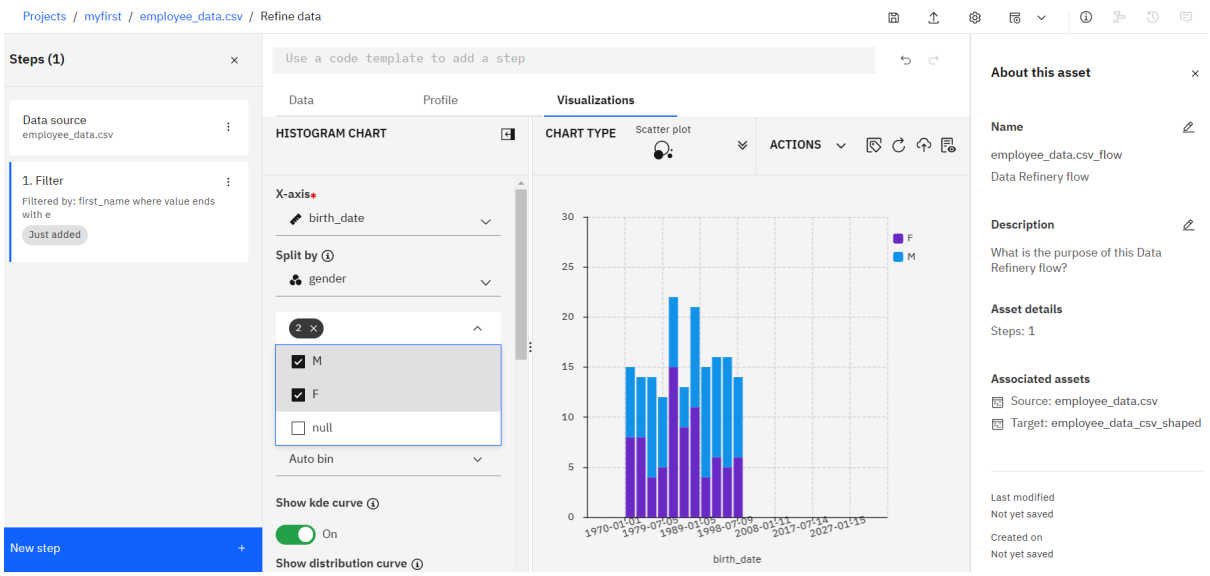
Step 12: Click on a new step and do filtering ,select column name as first name and select the operation ends with and select text and give “e” and click Apply .



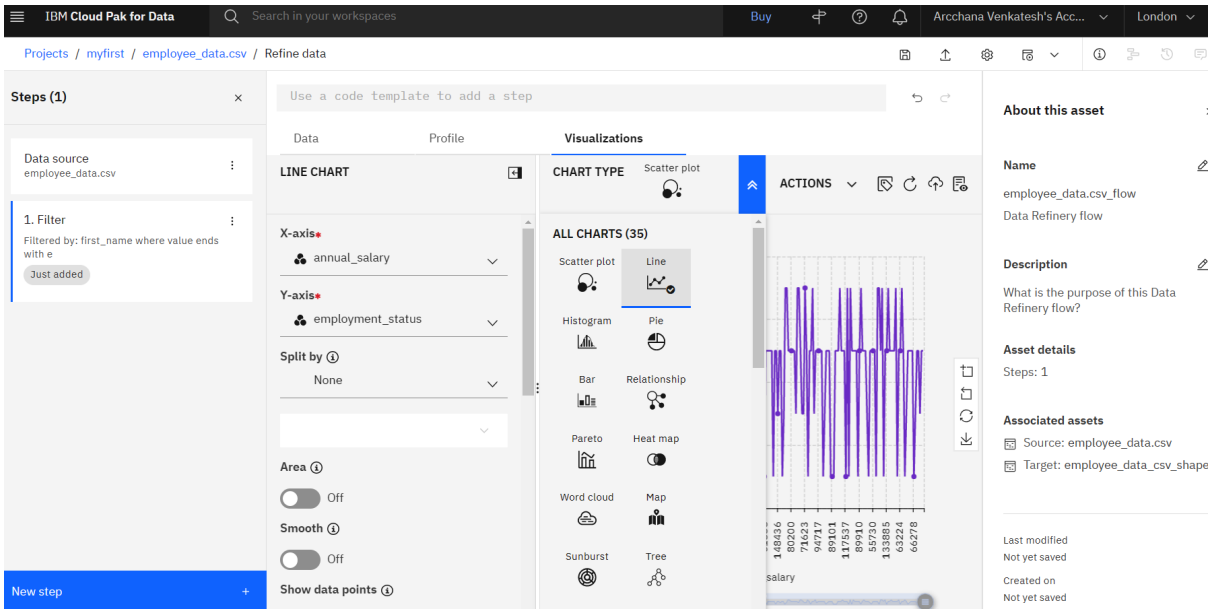
Step 13: first name was end up with “e” successfully and click visualization .



Step 14: Give x axis and split by and visualization will exist successfully .



Step 14: Select the chart as per requirement and do visualization in watson studio .



Result :

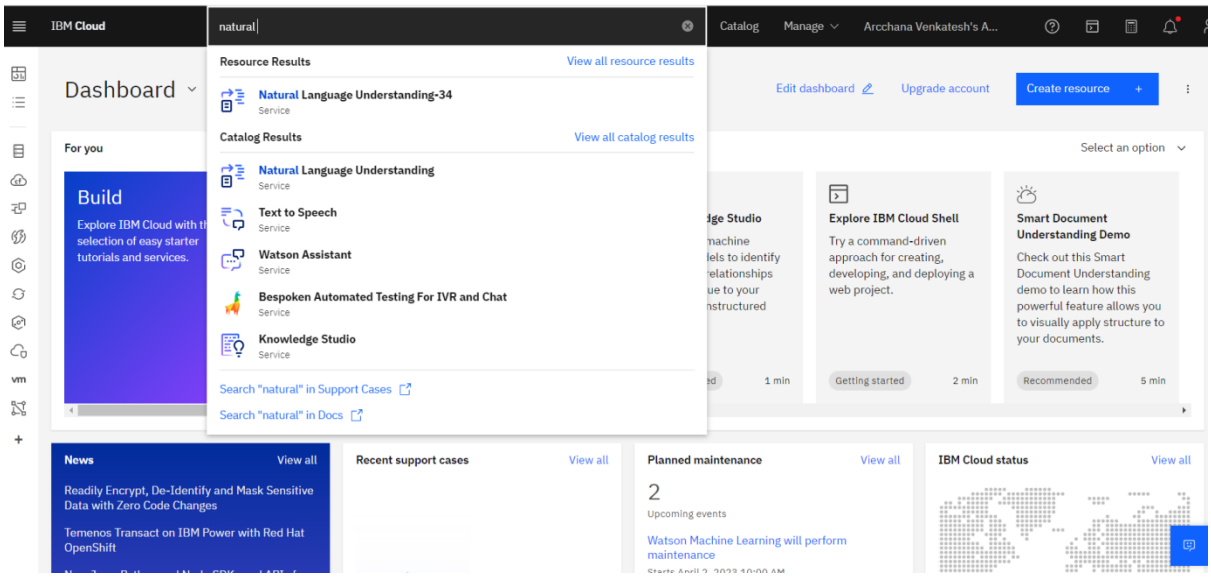
Thus, the above experiment was successfully executed

6.Natural Language Processing using Watson studio.

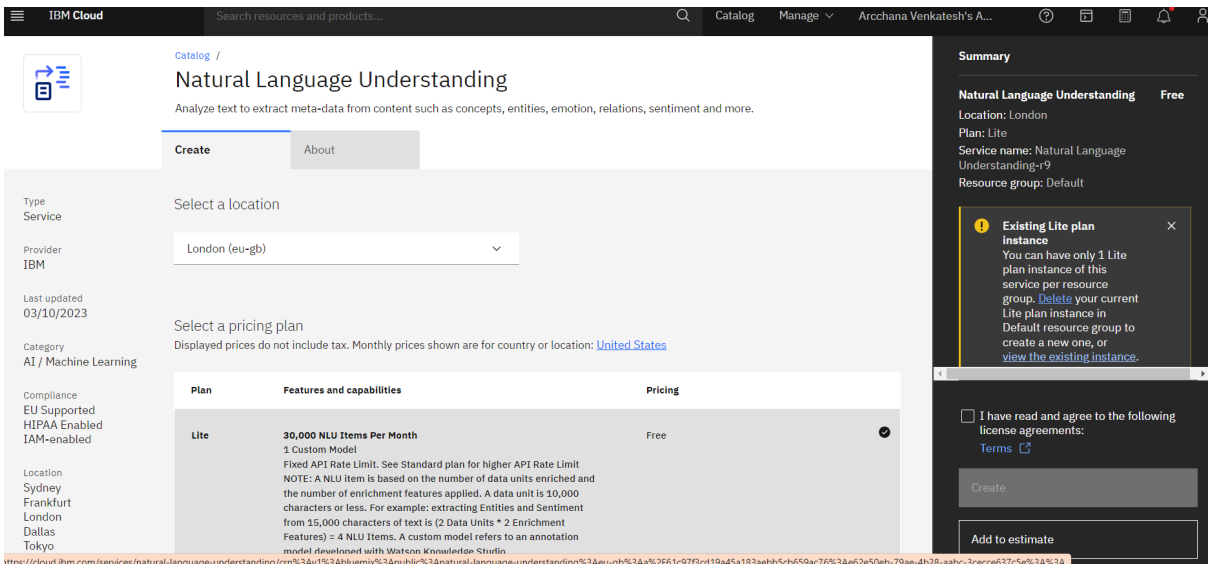
Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

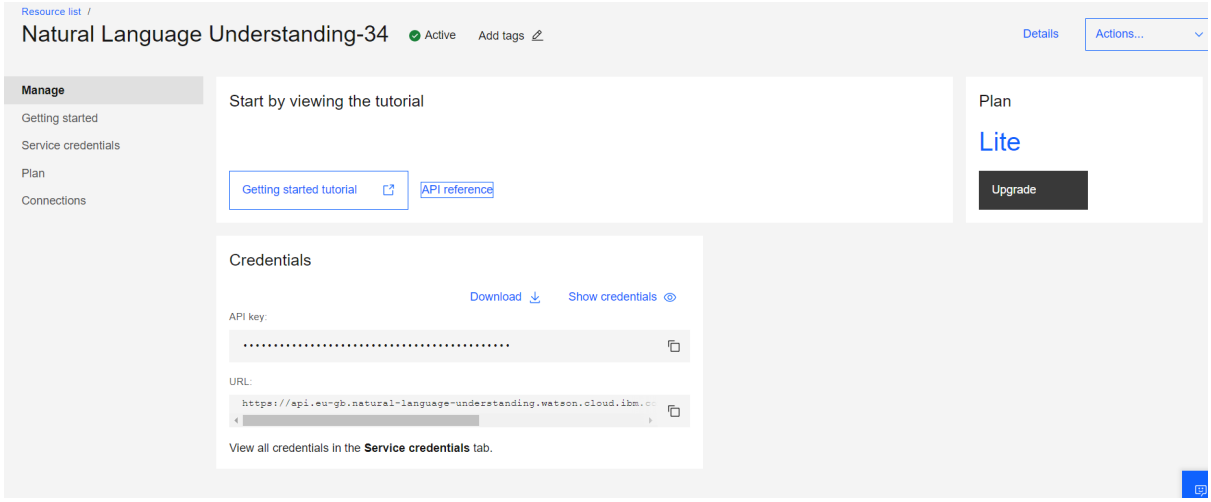
Step 2: click Natural language understanding



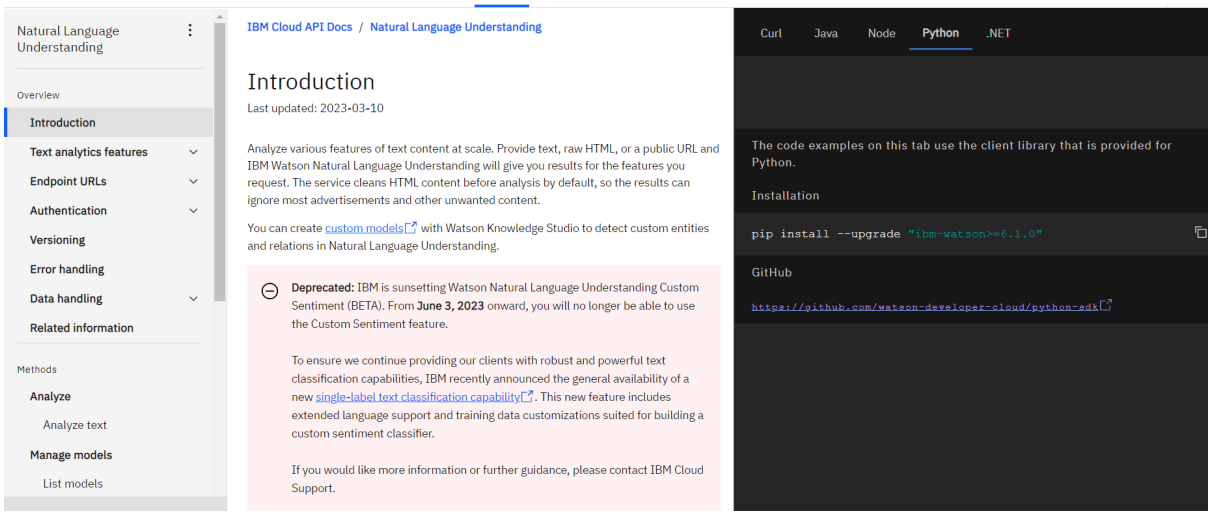
Step 3: click the link for view the existing instance link .

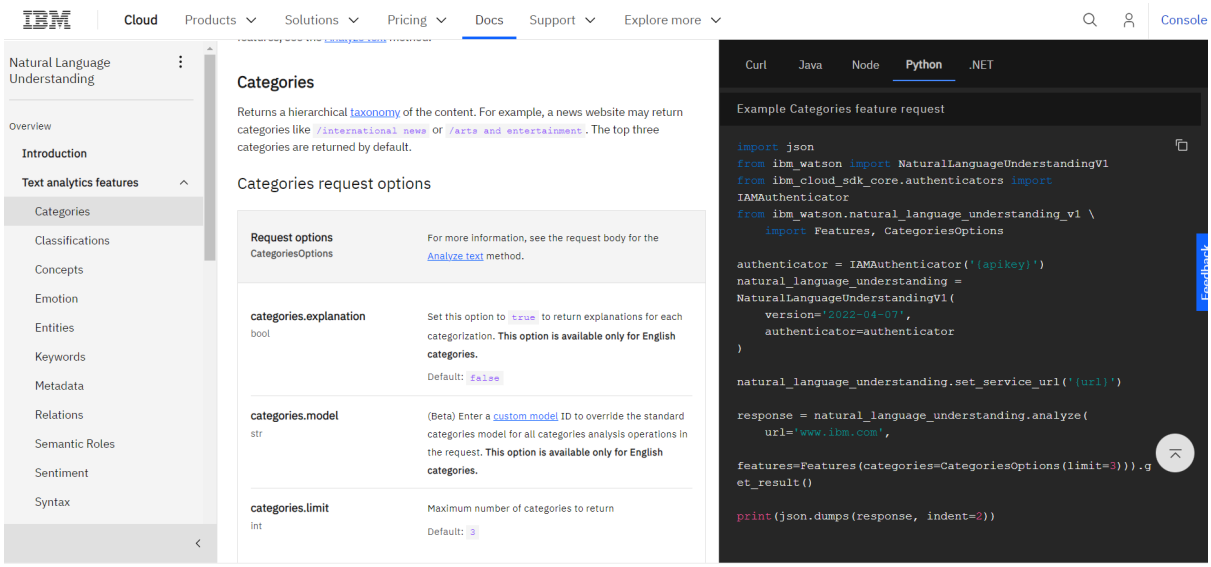


Step 4: Click on to API reference .



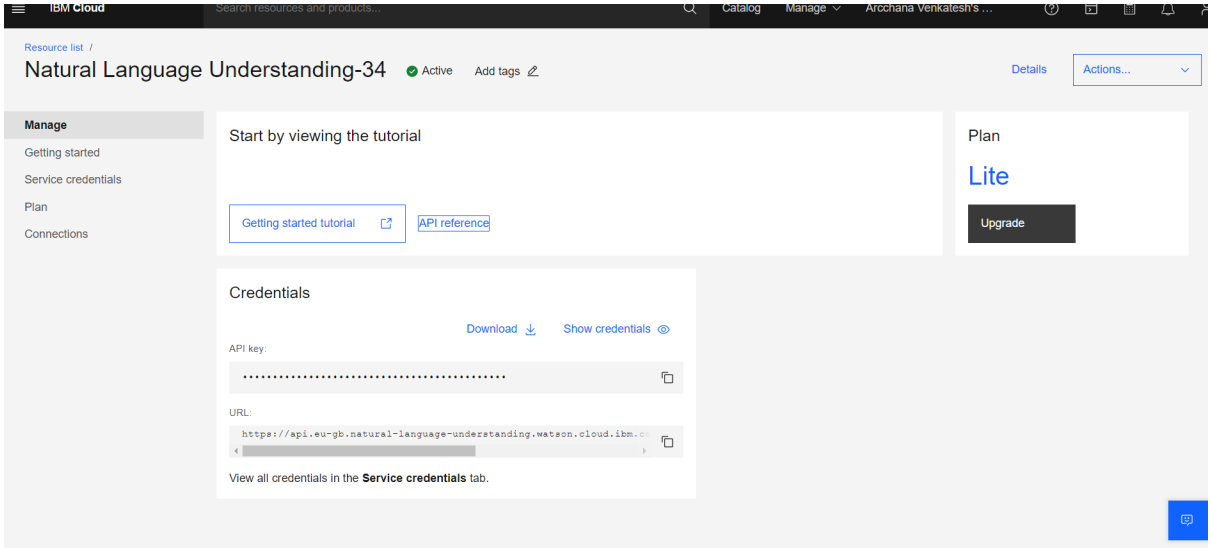
Step 5: In the right hand side under python the code will already exist , Make use of the code in Python note book inside Watson studio



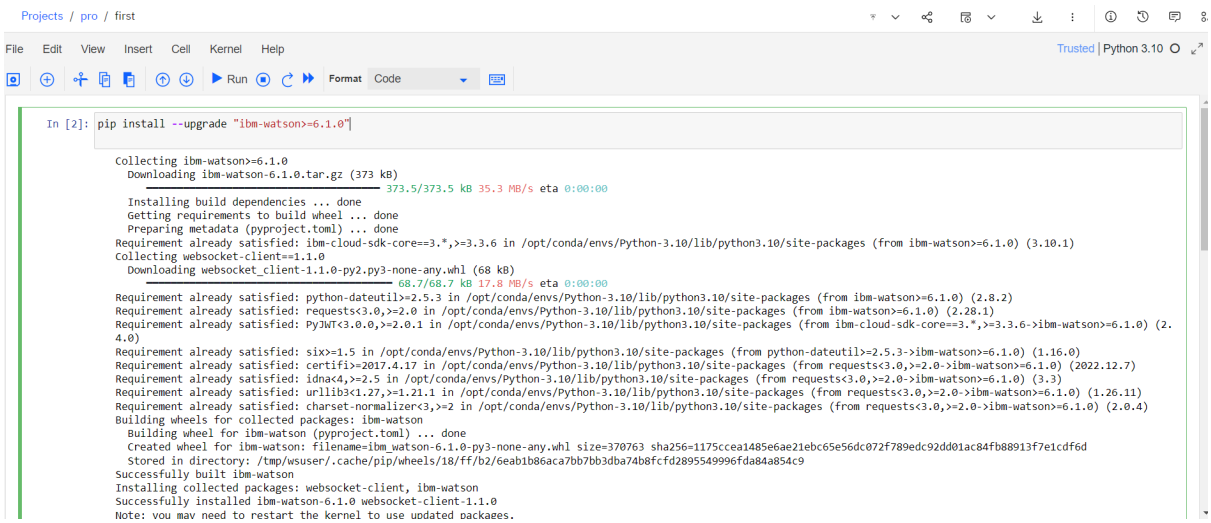


```
import json
from ibm_watson import NaturalLanguageUnderstandingV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.natural_language_understanding_v1 \
    import Features, CategoriesOptions

authenticator = IAMAuthenticator('{apikey}')
natural_language_understanding = NaturalLanguageUnderstandingV1(
    version='2022-04-07',
    authenticator=authenticator
)

natural_language_understanding.set_service_url('{url}')

response = natural_language_understanding.analyze(
    url='www.ibm.com',
    features=Features(categories=CategoriesOptions(limit=3))).get_result()

print(json.dumps(response, indent=2))
```

Step 6: Click the URL and paste in the code



Step 7: After pip install and write the code in note book and run



In [4]:
```
import json
from ibm_watson import NaturalLanguageUnderstandingV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.natural_language_understanding_v1 \
    import Features, CategoriesOptions

authenticator = IAMAuthenticator('gqQ9O1RE0wcc4s0lI86tgUFhHN5jqlXPkRnCYceOd3Js')
natural_language_understanding = NaturalLanguageUnderstandingV1(
    version='2022-04-07',
    authenticator=authenticator
)

natural_language_understanding.set_service_url('https://api.eu-gb.natural-language-understanding.watson.cloud.ibm.com/instances/e62e50eb-79ae-4b28-aabc-3cecce637c5e')

response = natural_language_understanding.analyze(
    url='www.ibm.com',
    features=Features(categories=CategoriesOptions(limit=3))).get_result()

print(json.dumps(response, indent=2))
```

Output :


```
{
  "usage": {
    "text_units": 1,
    "text_characters": 1138,
    "features": 1
  },
  "retrieved_url": "https://www.ibm.com/uk-en",
  "language": "en",
  "categories": [
    {
      "score": 0.952923,
      "label": "/business and finance/industries/technology industry"
    },
    {
      "score": 0.936069,
      "label": "/technology & computing/computing"
    },
    {
      "score": 0.917225,
      "label": "/education/online education"
    }
  ]
}
```


Result :

Thus, the above experiment was successfully executed

7.Linner Regression in R

Procedure:

Step:1 Create the predictor and response variable.

Step:2 Give the chart file a name

Step:3 Plot the chart and save the chart

Step:4 Predict the values of testing data.

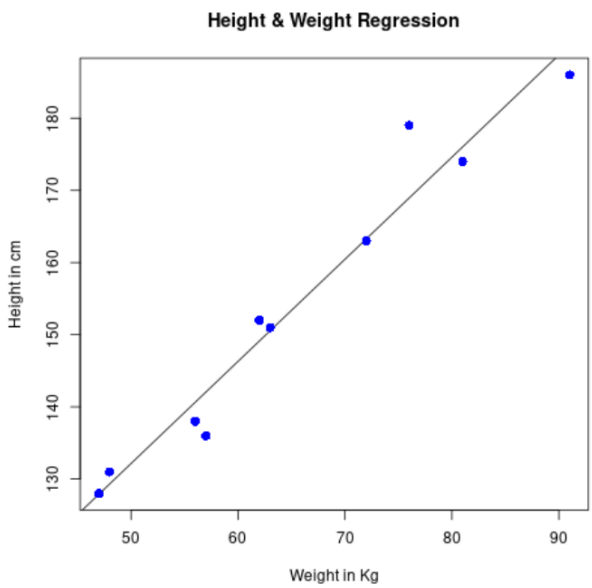
Step 5: save the code by dev.off() and run the code Linner regression plot will get conformed in console

Program code:

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)

# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.
dev.off()
```

```r
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)

# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.
dev.off()
```

Output:



Result :

Thus, the above experiment was successfully execute.

8.Decision tree in R programming

Procedure:

Step:1 install the needed library – datasets, caTools, Party, Dplyr , mgrittr.

Step:2 Split the dataset into training and testing data.

Step:3 Create a model for decision tree algorithm using R programming language.

Step:4 Predict the values of testing data.

Step 5: find the accuracy of the decision tree model.

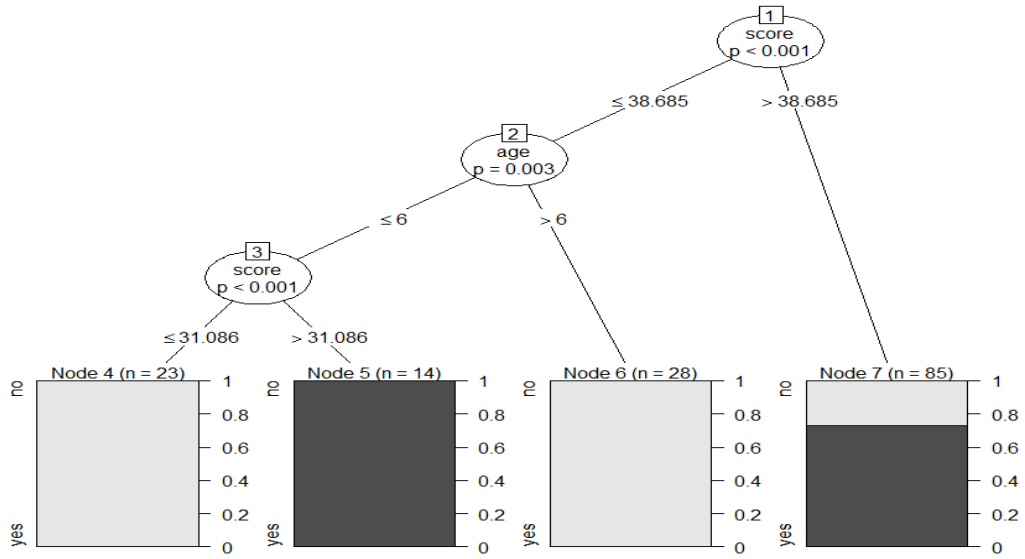
Step 6: Save the code and run the code Decision tree will get conformed in console.

Program:

```
library(datasets)
library(caTools)
library(party)
library(dplyr)
library(magrittr)
data("readingSkills")
head(readingSkills)
```

```
> data("readingSkills")
> head(readingSkills)
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
```

```
sample_data = sample.split(readingSkills, SplitRatio = 0.8)
train_data <- subset(readingSkills, sample_data == TRUE)
test_data <- subset(readingSkills, sample_data == FALSE)
model<- ctree(nativeSpeaker ~ ., train_data)
plot(model)
```



predict_model<-predict(ctree_, test_data)

m_at <- table(test_data$nativeSpeaker, predict_model)

m_at

```
> predict_model<-predict(ctree_,test_data)
> predict_model<-predict(ctree_,test_data)
> m_at <- table(test_data$nativeSpeaker,predict_model)
> m_at
     predict_model
      no yes
  no  13  13
  yes  0  24
```

ac_Test < - sum(diag(table_mat)) / sum(table_mat)

print(paste('Accuracy for test is found to be', ac_Test))

```
> ac_Test <- sum(diag(table_mat)) / sum(table_mat)
> print(paste('Accuracy for test is found to be', ac_Test))
[1] "Accuracy for test is found to be 0.74"
>
```

Result :

Thus, the above experiment was successfully executed