**Advanced Database Management System Lab**

**Subject Code: MCAL13**

A Practical Journal Submitted in Fulfilment of Degree of

**Master in Computer Application**

**Year 2023-2024**

By

**Mr. Pratik Shantaram Kajare**

**(Application Id: - 81206)**

Semester – 1

Under the Guidance of



Institute of Distance and Open Learning Vidya Nagari, Kalina, Santacruz East – 400098,

University of Mumbai

**PCP Center**

[Vidyavardhini’s College of Engineering & Technology, Vasai Road]



**Institute of Distance and Open Learning,**

**Vidya Nagari, Kalina, Santacruz (E) -400098**

# CERTIFICATE

This to certify that, **Mr. Pratik Shantaram Kajare** appearing **Master in Computer Application (Semester I) Application ID: 81206** has satisfactory completed the prescribed practical of **MCAL13-Advanced Database Management System Lab** as laid down by the University of Mumbai for the academic year 2023-24

Teacher in charge Examiners Coordinator

IDOL, MCA

University of Mumbai

Date: -31/01/2024

Place: - Vasai

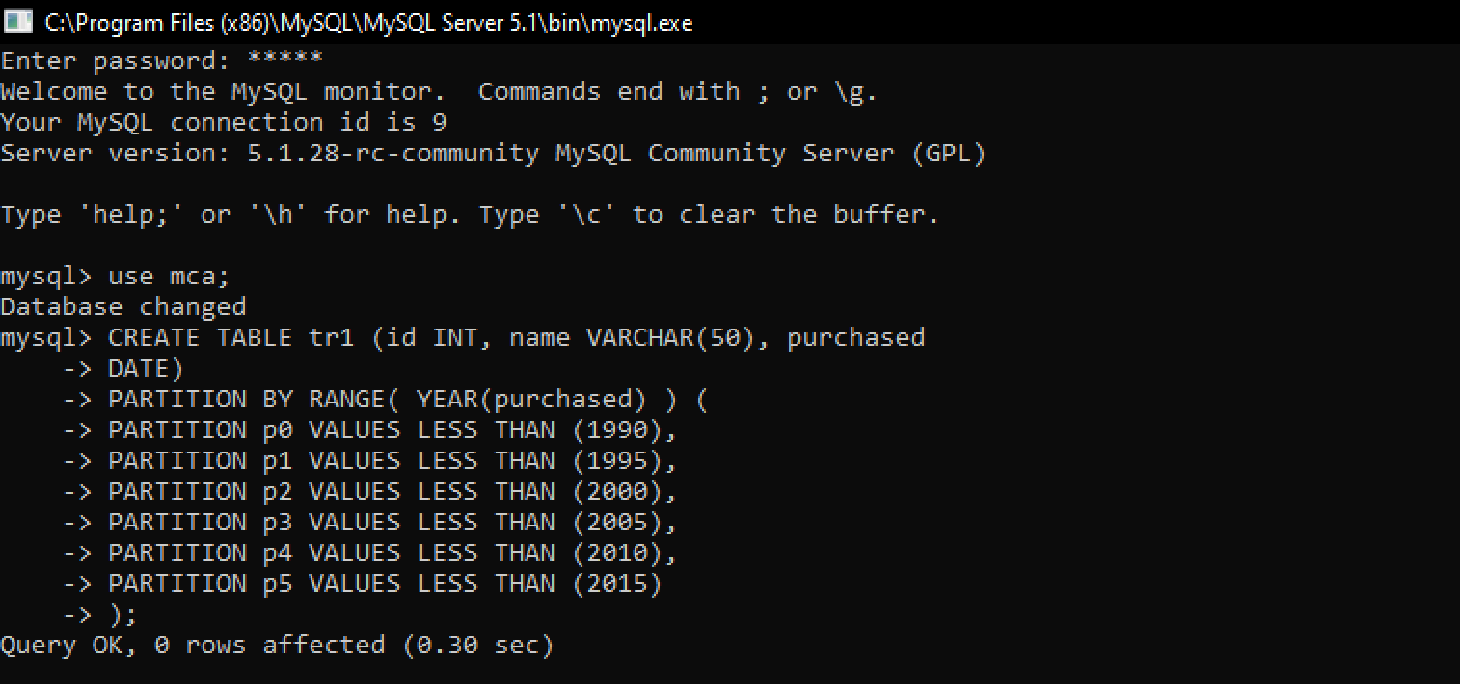
**INDEX**

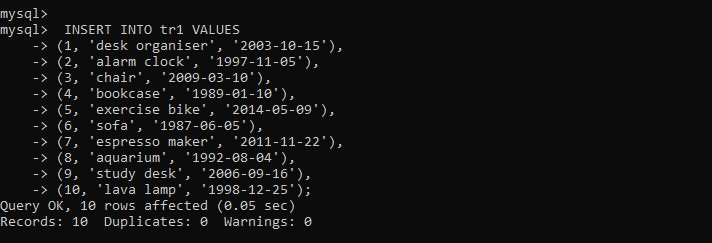
|  |  |
| --- | --- |
| **Practical No.** | **Practical** |
| 1. | Implementation of Data partitioning through Range |
| 2. | Implementation of Analytical queries like Roll\_UP, CUBE, First, Last, Rank AND Dense Rank |
| 3. | Implementation of Abstract Data Type & Reference |
| 4. | To study ETL process |
| 5. | Installation of R  datatype in R programming  Reading and Writing data to and from R |
| 6. | To study Linear Regression. |
| 7. | To study Analysis of Regression |
| 8. | To study Logistic Regression |
| 9. | To study support Vector Machine |
| 10 | To study varied Algorithm |

**EXPERIMENT NO: 01**

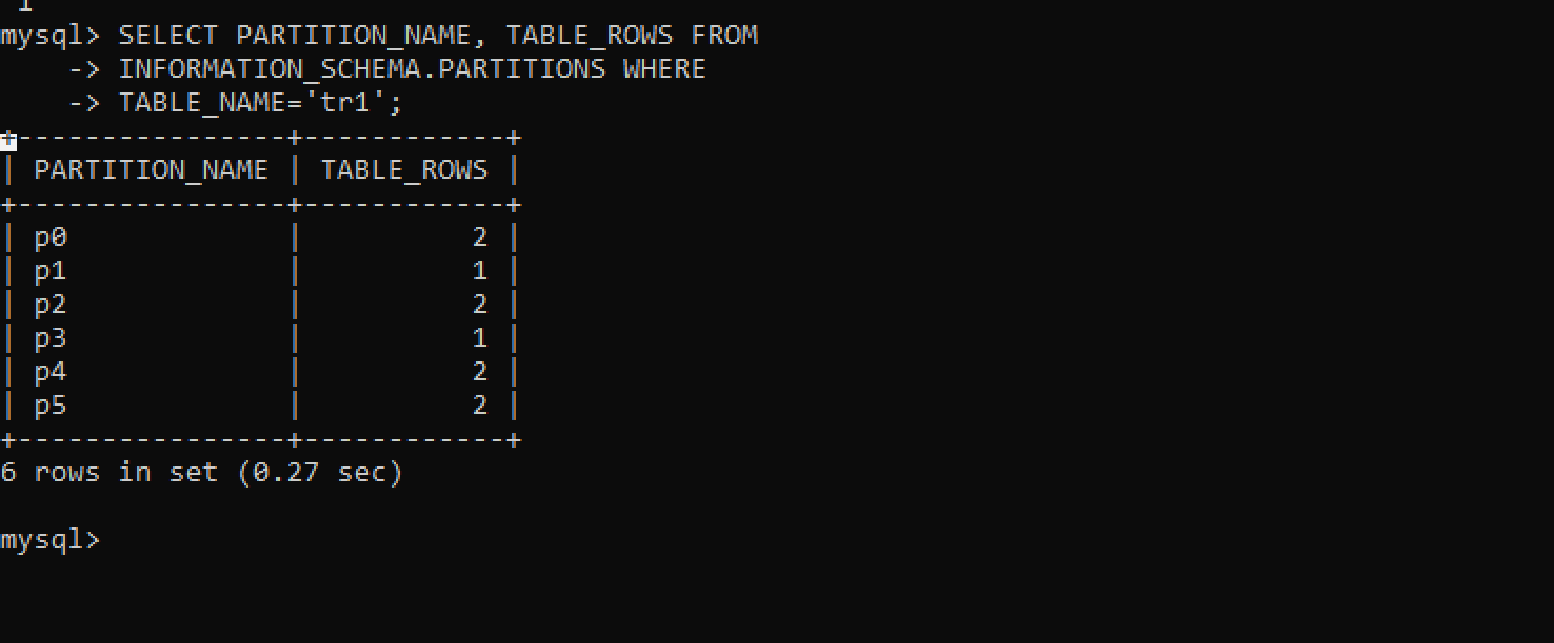
**RANGE Partitioning in MySQL**

**Aim**: Implementation of Data partitioning through Range

.



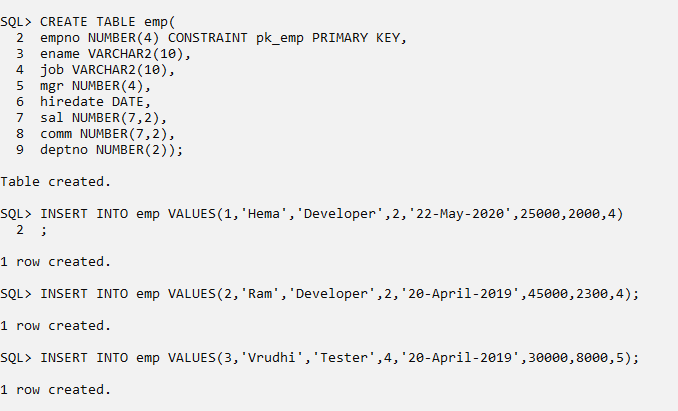


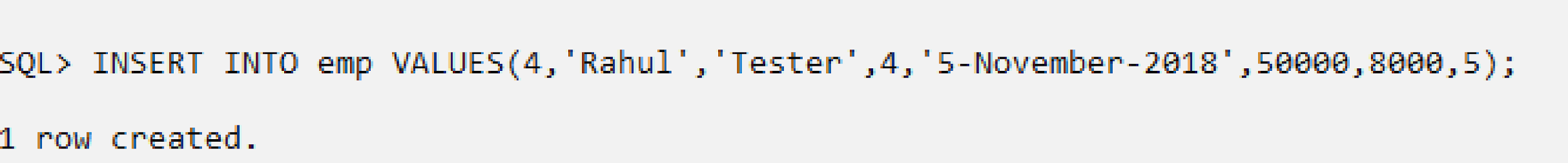


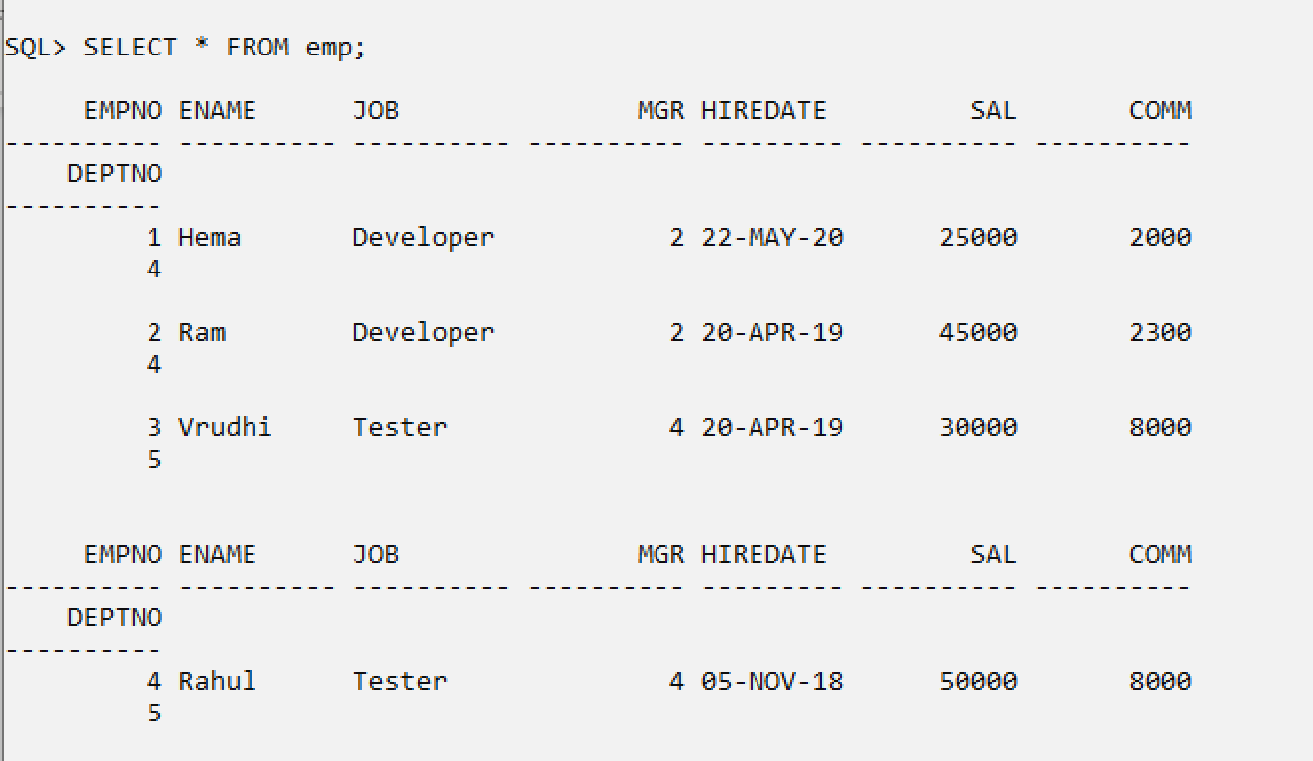
**EXPERIMENT NO: 02**

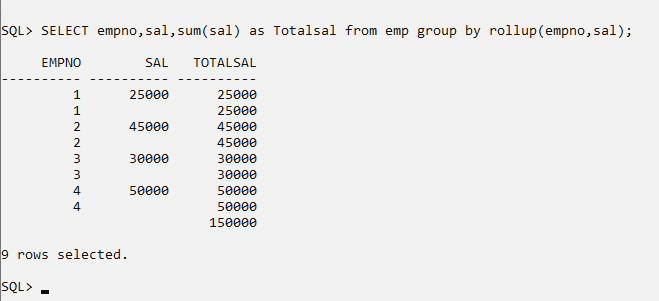
**ANALYTICAL QUERIES**

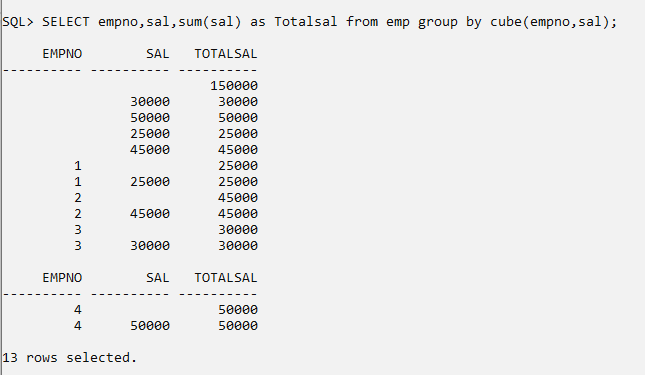
**Aim**: Implementation of Analytical queries like Roll\_UP, CUBE, First, Last, Rank AND Dense Rank.



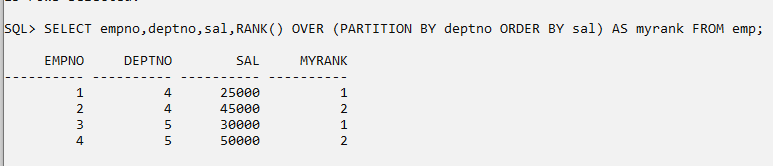


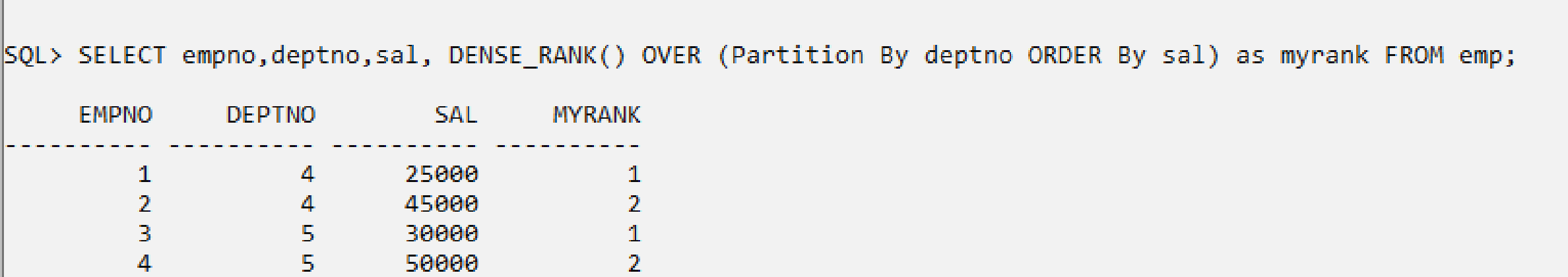




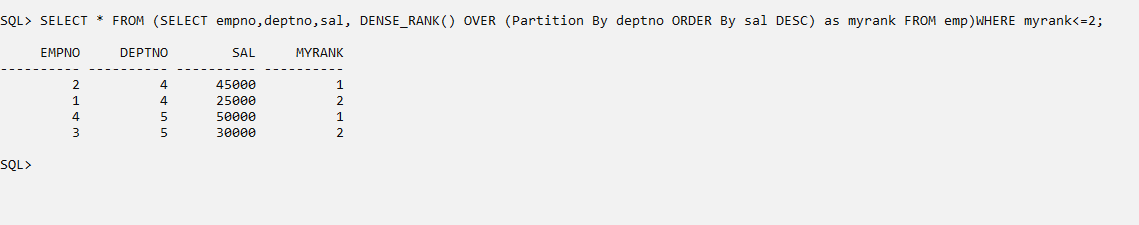


**RANK**

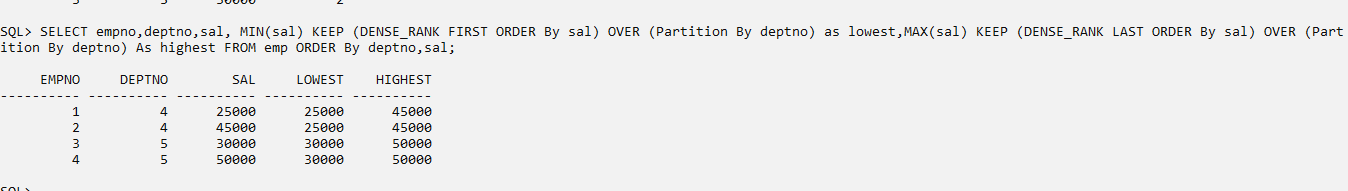




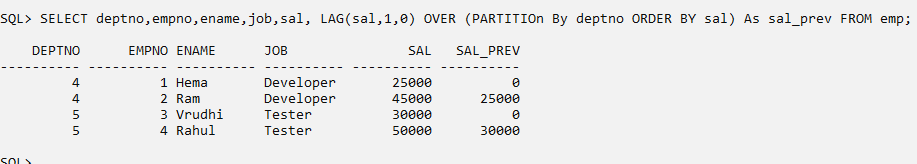
**DENSE\_RANK**



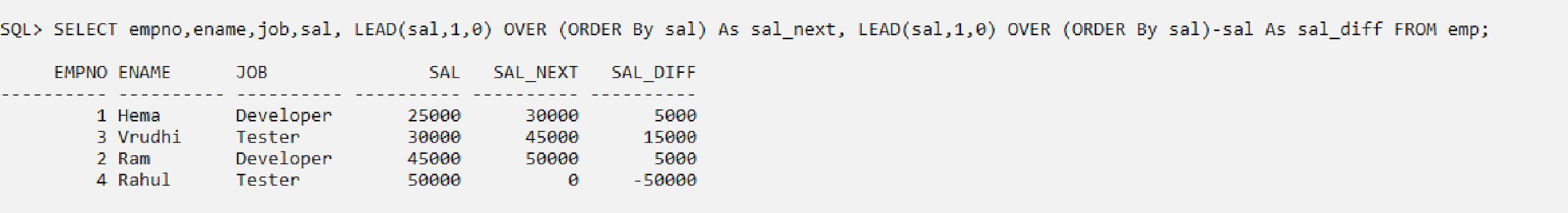
**FIRST AND LAST**

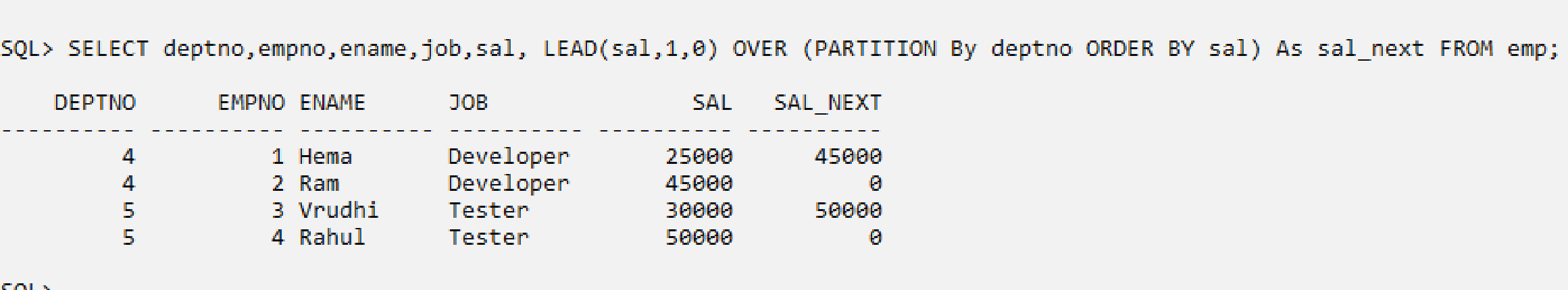


**LAG**



**LEAD**





**EXPERIMENT NO: 03**

Aim: Implementation of Abstract Data Type & Reference

**Customer\_reltab :**

The Customer\_reltab table has the following definition:

CREATE TABLE Customer\_reltab (

CustNo NUMBER NOT NULL,

CustName VARCHAR2(200) NOT NULL,

Street VARCHAR2(200) NOT NULL,

City VARCHAR2(200) NOT NULL,

State CHAR(2) NOT NULL,

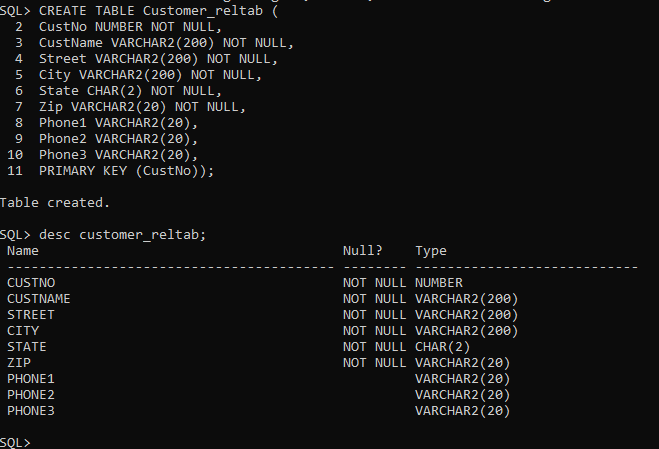
Zip VARCHAR2(20) NOT NULL,

Phone1 VARCHAR2(20),

Phone2 VARCHAR2(20),

Phone3 VARCHAR2(20),

PRIMARY KEY (CustNo));



**PurchaseOrder\_reltab:**

The PurchaseOrder\_reltab table has the following definition:

CREATE TABLE PurchaseOrder\_reltab (

PONo NUMBER, /\* purchase order no \*/

Custno NUMBER references Customer\_reltab, /\* Foreign KEY referencing

customer \*/ OrderDate DATE, /\* date of order \*/ ShipDate DATE, /\* date to be shipped \*/

ToStreet VARCHAR2(200), /\* shipto address \*/ ToCity VARCHAR2(200),

ToState CHAR(2),

ToZip VARCHAR2(20),

PRIMARY KEY(PONo));

A computer screen with white text

Description automatically generated

**Stock\_reltab:**

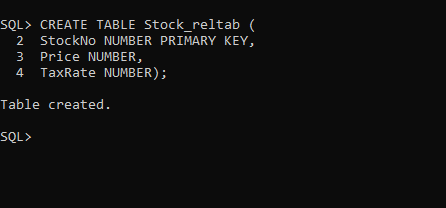
The Stock\_reltab table has the following definition:

CREATE TABLE Stock\_reltab (

StockNo NUMBER PRIMARY KEY,

Price NUMBER,

TaxRate NUMBER);



**LineItems\_reltab**

The LineItems\_reltab table has the following definition:

CREATE TABLE LineItems\_reltab (

LineItemNo NUMBER,

PONo NUMBER REFERENCES PurchaseOrder\_reltab,

StockNo NUMBER REFERENCES Stock\_reltab,

Quantity NUMBER,

Discount NUMBER,

PRIMARY KEY (PONo, LineItemNo));

A screen shot of a computer

Description automatically generated

**Inserting Values Under the Relational Model**:

INSERT INTO Stock\_reltab VALUES(1004, 6750.00, 2);

INSERT INTO Stock\_reltab VALUES(1011, 4500.23, 2);

INSERT INTO Stock\_reltab VALUES(1534, 2234.00, 2);

INSERT INTO Stock\_reltab VALUES(1535, 3456.23, 2);

INSERT INTO Customer\_reltab VALUES

(1, 'Jean Nance', '2 Avocet Drive',

'Redwood Shores', 'CA', '95054',

'415-555-1212', NULL, NULL);

INSERT INTO Customer\_reltab

VALUES (2, 'John Nike', '323 College Drive',

'Edison', 'NJ', '08820',

'609-555-1212', '201-555-1212', NULL);

INSERT INTO PurchaseOrder\_reltab

VALUES (1001, 1, SYSDATE, '10-MAY-1997',

NULL, NULL, NULL, NULL);

INSERT INTO PurchaseOrder\_reltab

VALUES (2001, 2, SYSDATE, '20-MAY-1997',

'55 Madison Ave', 'Madison', 'WI', '53715');

**Querying Data Under the Relational Model:**

SELECT C.CustNo, C.CustName, C.Street, C.City, C.State, C.Zip, C.phone1, C.phone2, C.phone3,

P.PONo, P.OrderDate,

L.StockNo, L.LineItemNo, L.Quantity, L.Discount FROM Customer\_reltab C,

PurchaseOrder\_reltab P, LineItems\_reltab L Advanced Database Management System Lab

WHERE C.CustNo = P.CustNo AND P.PONo = L.PONo AND

P.PONo = 1001;

Get the Total Value of Purchase OrdersSELECT P.PONo, SUM(S.Price \* L.Quantity) FROM PurchaseOrder\_reltab P,

LineItems\_reltab L, Stock\_reltab S

WHERE P.PONo= L.PONo

AND L.StockNo = S.StockNo GROUP BYP.PONo;

Get the Purchase Order and Line Item Data for Stock Item 1004

SELECT P.PONo, P.CustNo,

L.StockNo, L.LineItemNo, L.Quantity, L.Discount FROM PurchaseOrder\_reltab P,

LineItems\_reltab L WHERE P.PONo = L.PONo AND L.StockNo = 1004;

**Updating Data Under the Relational Model:**

UPDATE LineItems\_reltab SET Quantity = 20 WHERE PONo = 1001

AND StockNo = 1534;

**Deleting Data Under the Relational Model:**

DELETE

FROM LineItems\_reltab WHERE PONo = 1001;

DELETE

FROM PurchaseOrder\_reltab WHERE PONo = 1001;

**EXPERIMENT NO: 04**

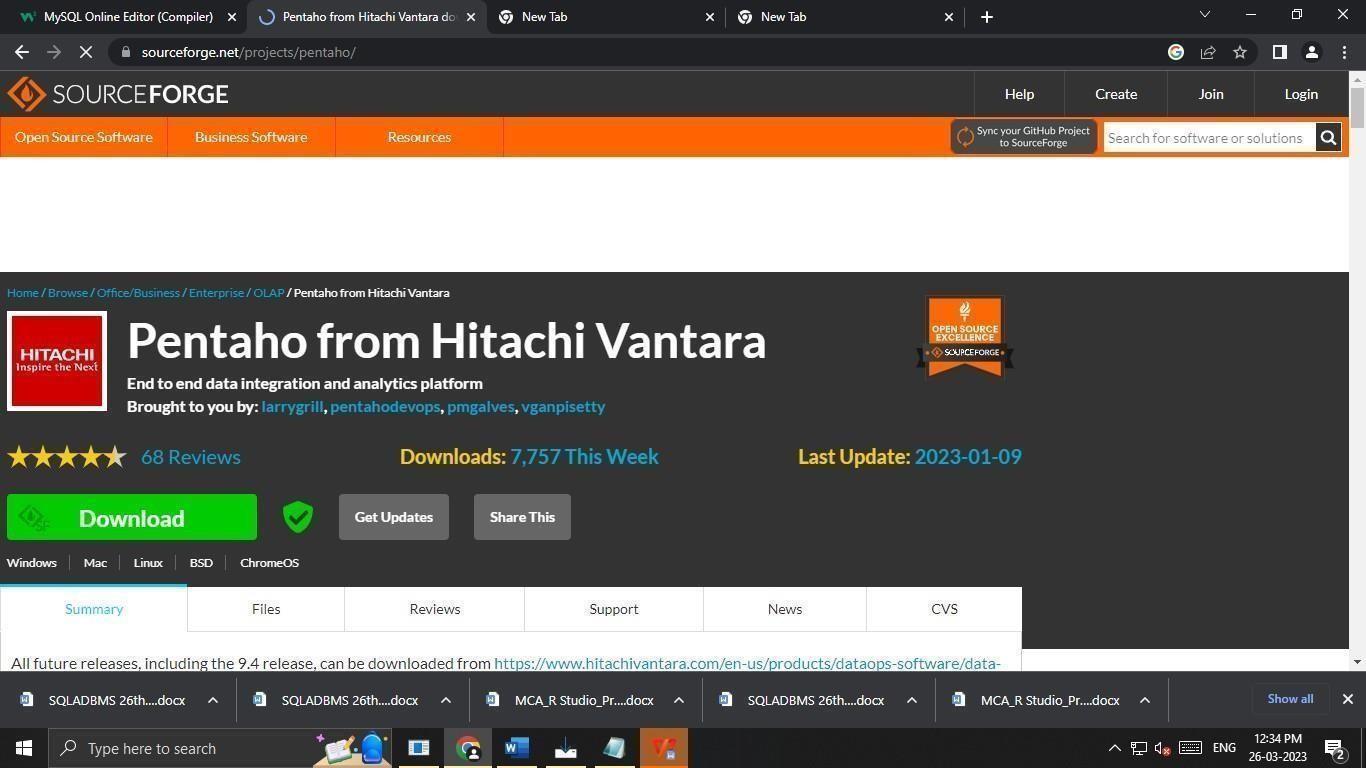
**Aim**: To study ETL process

\* Installation steps for Pentaho Data Integration Software

Step 1: Download Pentaho Data Integration Software. The first thing we need is the Pentaho

Data Integration software that we’ll be working with

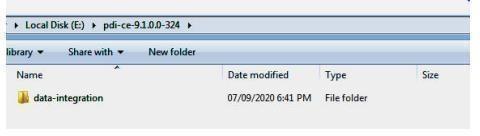
You can download the set up file from link [https://sourceforge.net/projects/pentaho/.](https://sourceforge.net/projects/pentaho/)



Press the “Download” button.

It will start downloading zip file on your computer. Once the downloading is finished, extract the files into a folder you want to.

Your folder should look something like this:

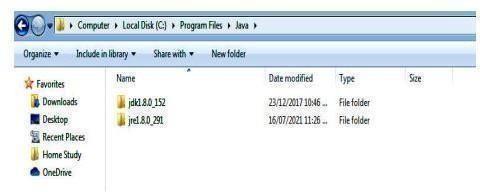


Step 2: Install the Java Dependencies, if Required.

To run Pentaho Data Integration, Java Runtime Environment and Java Development Kit are required. To check if you already have these installed, go to this path in your file explorer:

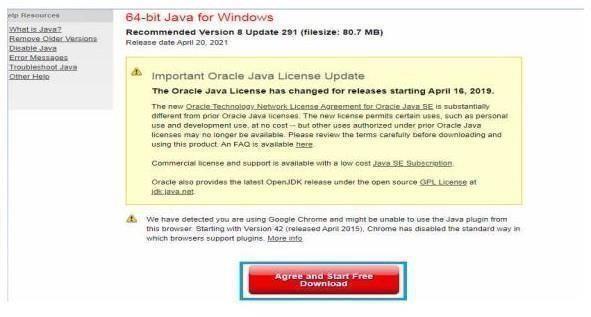
C:\Program Files\Java

Or: C:\Program Files (x86)\Java If this folder exists and you see folders that look like:



Then you have the required files. If this folder doesn’t exist or you don’t see one or both of these folders, then you need to download JRE and/or JDK. To download JRE, go to this link https://java.com/en/download/ and press “Download.”

Your page should look like this:



The installation window will look something like this:



Follow the instructions until finished.

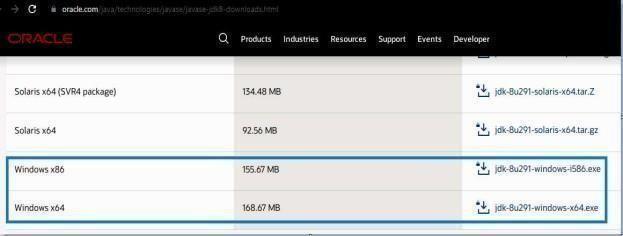
Next, download the JDK from this link

https:[//www.oracle.com/java/technologies/javase/javase-jdk8-](http://www.oracle.com/java/technologies/javase/javase-jdk8-) downloads.html.

Please note that there have been substantial changes to the Oracle JDK licensing agreement. Details are available at Oracle Technology Network License Agreement for Oracle Java SE.

There will be a list of different operating systems to choose from. Scroll until you find Windows.

If you’re unsure about which version (x64 or x86) your Windows is, select x86.



It will open following window



Press “Download”.



If you’re not logged in to Oracle, then you will be prompted to log in.

If you don’t have an Oracle account, you need to create one in order to download the JDK.



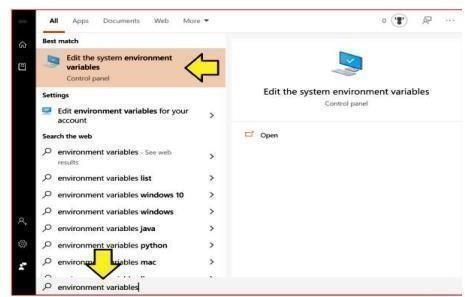
The installation setup will look like this:



Graphics:

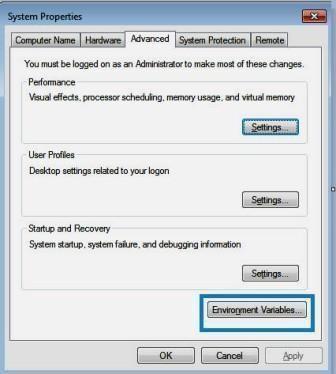
Hitachi Video Management Platform (VMP) has been designed from the ground up to meet the challenges of data storage and processing that new video systems present.

Step 3: Set Up the Environment Variables There are three environment variables that need to be set up. To open the environment variables menu type in “environment variables” in the Windows search bar like this:

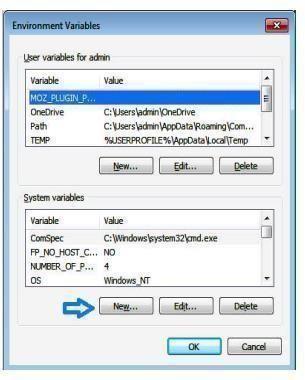


Click the “Edit the system environment variables” option.

That will open the “System Properties” window. Under Advanced tab …Click the “Environment Variables.” button at the bottom.



That will open a window that looks like this:



We need to add three new System variables.

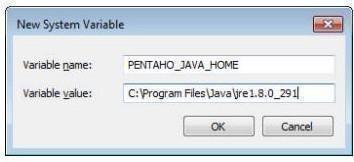
Click the “New…” button under “System variables” and enter the following:



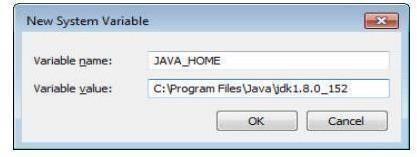


Make sure your variable value file path is the same one on your computer.

Press “OK” and then enter two more.



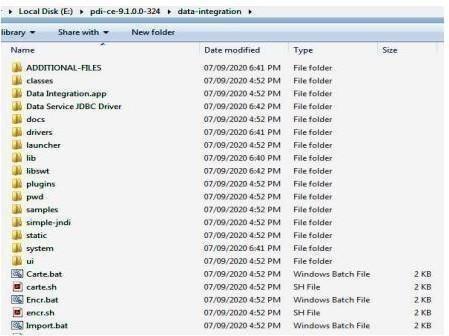
Press ‘’OK’’.



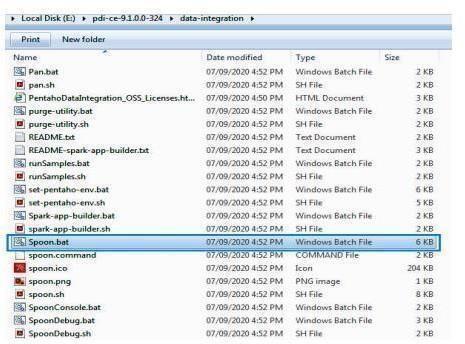
Press ‘’OK’’ and close all the previous windows by pressing “OK.”

Step 4: Open the Pentaho Data Integration App Now that Java is installed successfully and the environment variables are also set, we can start running the Pentaho Data Integrationapp.

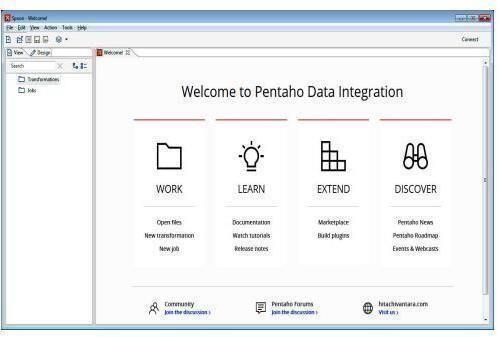
The data integration folder that you downloaded earlier will looklike this:



The file that runs the app is called “Spoon.bat”.



Double click this file to open the Pentaho Data Integration app.



Now you can start using this app by pressing “New transformation” or “New job.”

**EXPERIMENT NO: 05**

**Aim**:

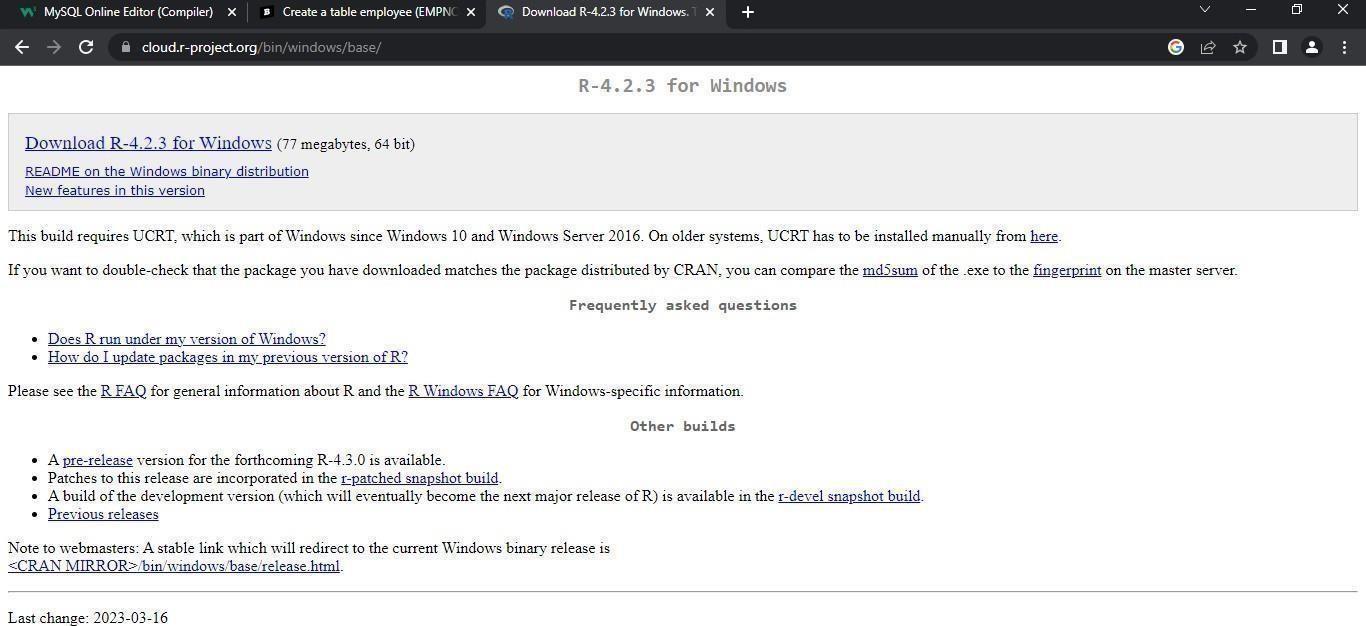
1. installation of R
2. datatype in R programming
3. Reading and Writing data to and from R.

**Aim: installation of R**

\* R Installation in Windows

Steps used to install the R in Windows are as follows:

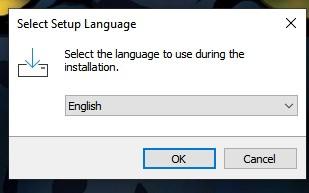
Step 1: First, we have to download the R setup from https://cloud.rproject.org/bin/windows/base/.

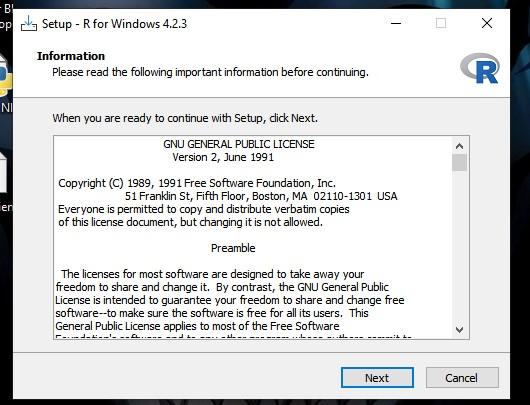


Step 2:

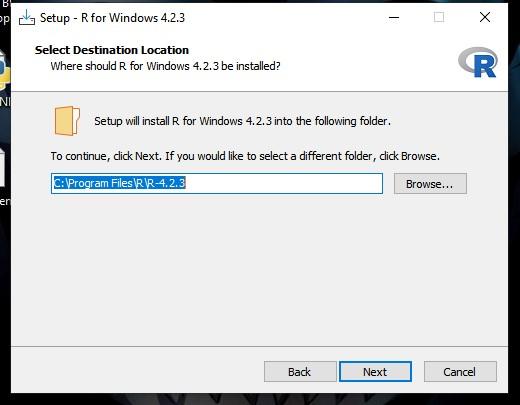
When we click on Download R- 4.1.0 for windows, our downloading will start. Once the

downloading is finished, we have to run the setup of R as follows:

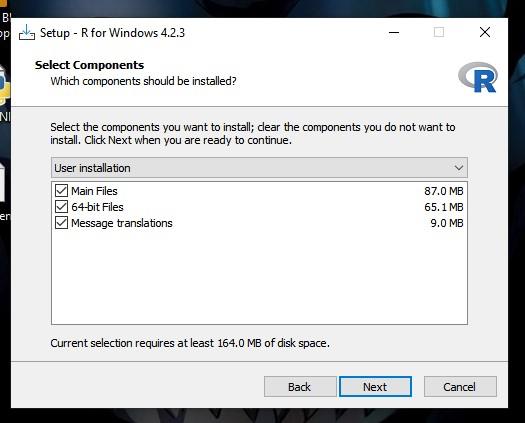




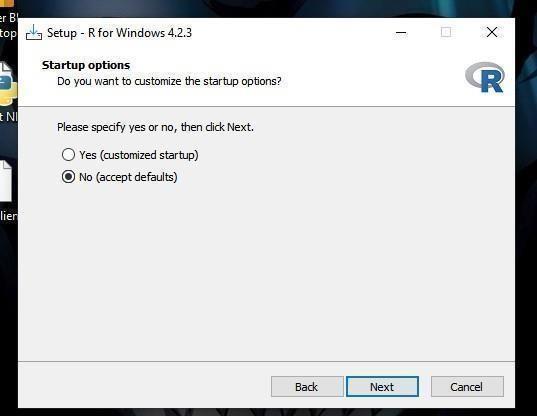
1. Select the path where we want to download the R and click Next



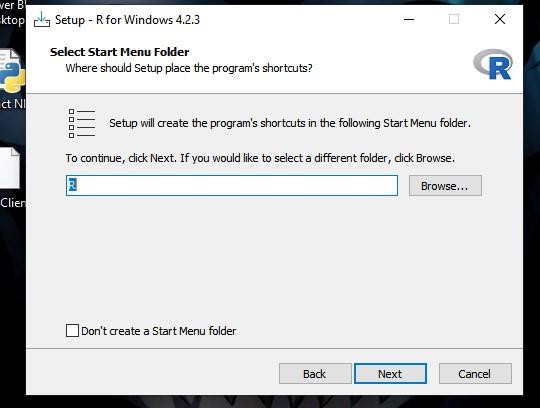
1. Select all components which we want to install, and then click Next.



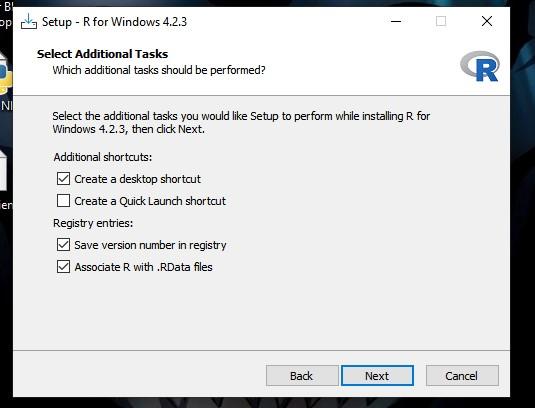
1. Now, we have to select either (customized startup) or (accept the default), and then click Next.



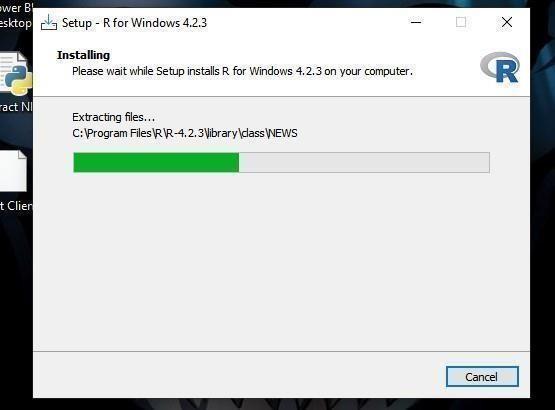
1. Now Select Start Menu Folder window will appear, click Next



1. Click Next



1. When we proceed to Next, installation of R will get started:



1. Finally, we will click on Finish.



R has been successfully installed.

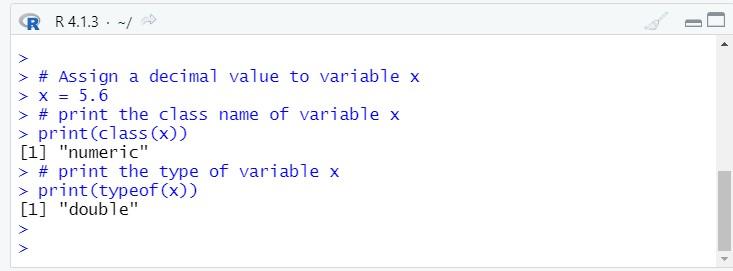
**Aim: datatype in R programming**

**1. R program to illustrate Numeric data** # Assign a decimal value to variable x x

= 5.6

# print the class name of variable x print(class(x))

# print the type of variable x print(typeof(x))



Output:

[1] "numeric"

[1] "double"

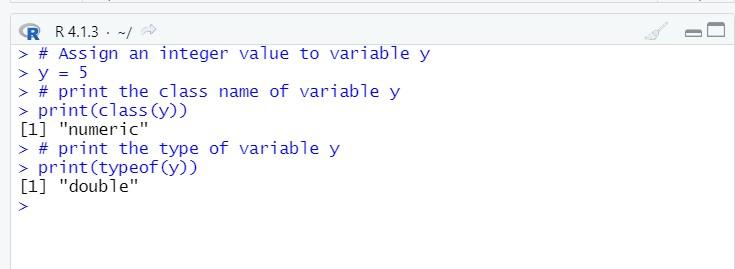
**2. R program to illustrate Numeric datatype**

# Assign an integer value to variable y

y = 5

# print the class name of variable y print(class(y))

# print the type of variable y print(typeof(y))



Output:

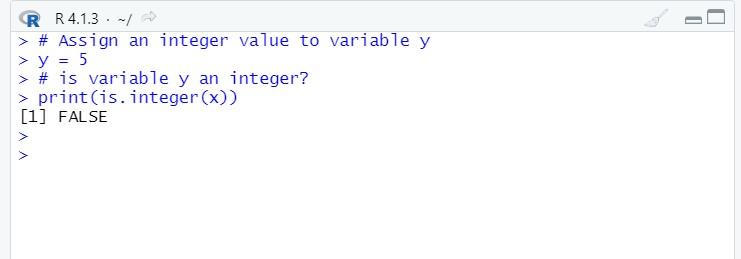
[1] "numeric"

[1] "double"

1. **R program to illustrate Numeric datatype**

# Assign an integer value to variable y y = 5

# is variable y an integer? print(is.integer(x))



Output:

[1] FALSE

1. **R program to illustrate integer data type** # Create an integer variable x = as.integer(5)

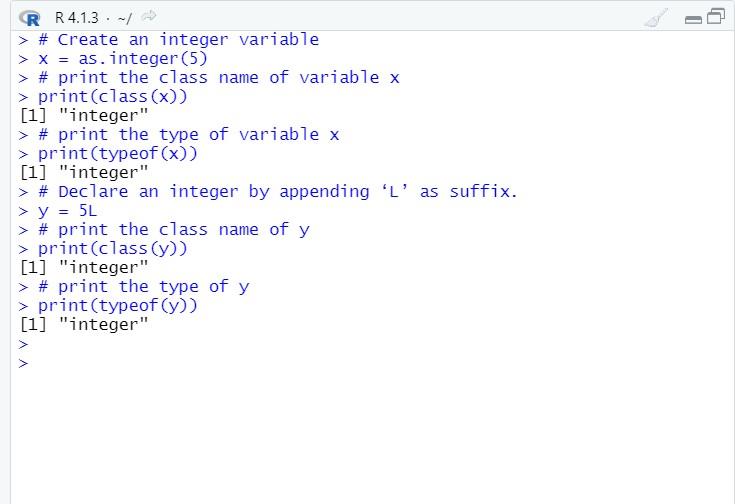
# print the class name of variable x print(class(x))

# print the type of variable x print(typeof(x))

# Declare an integer by appending ‘L’ as suffix. y = 5L

# print the class name of y

print(class(y)) # print the type of y print(typeof(y))



Output:

[1] "integer"

[1] "integer"

[1] "integer"

[1] "integer"

**5. R program to illustrate logical data type** # Two variables x = 4

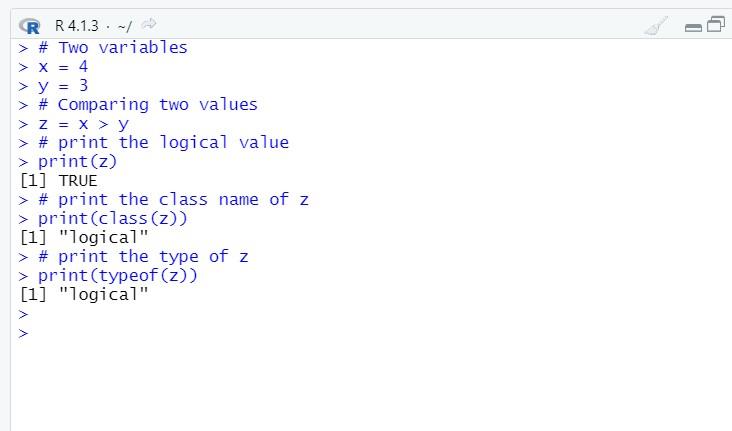
y = 3

# Comparing two values z = x > y # print the logical value print(z)

# print the class name of z print(class(z))

# print the type of z

print(typeof(z))



Output:

[1] TRUE

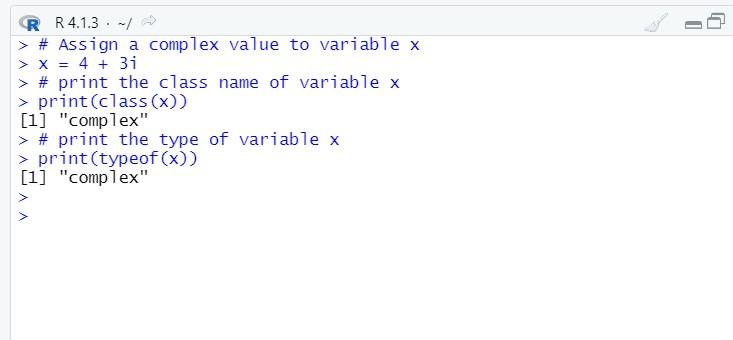
[1] "logical"

[1] "logical"

**6. R program to illustrate complex datatype** # Assign a complex value to variable x x = 4 + 3i

# print the class name of variable x print(class(x))

# print the type of variable x print(typeof(x))



Output:

[1] "complex"

[1] "complex"

**7. R program to illustrate character data type**

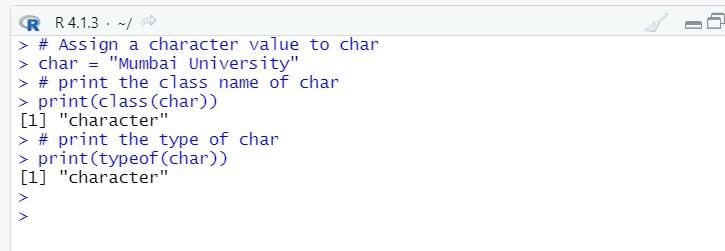
# Assign a character value to char char = "Mumbai University"

# print the class name of char

print(class(char))

# print the type of char

print(typeof(char))



Output:

[1] "character"

[1] "character"

**Aim: Reading and Writing data to and from R.**

Reading data files with read.table()

The read.table() function is one of the most common used functions for readingdata into R. It has following arguments.

The function read.table() can be used to read the data frame.

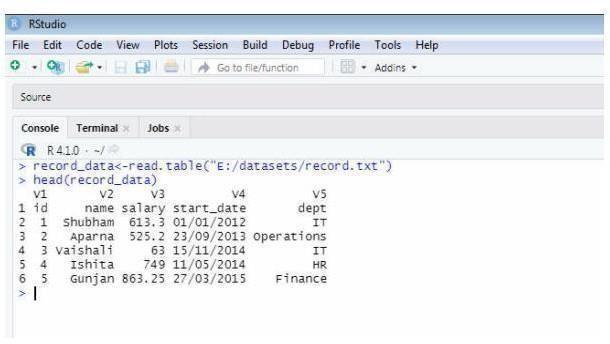
We have kept record.txt and record.csv files under datasets folder inside E: drive.



>record\_data<- read.table("E:/datasets/record.txt")

>head(record\_data)

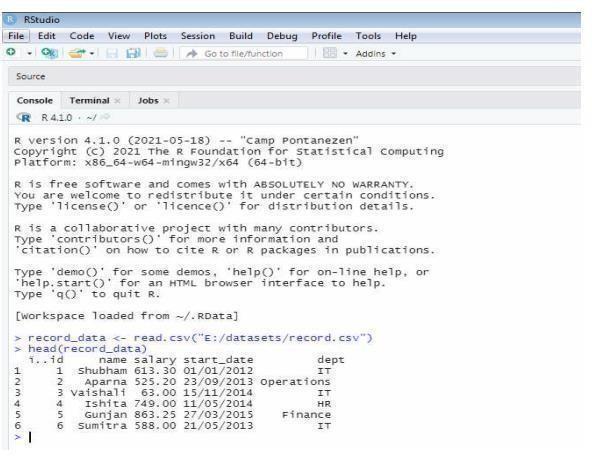
#returns first n rows of the data



Similarly, read.csv() function can be used to read data from csv files.

>record\_data<- read.csv("E:/datasets/record.csv")

>head(record\_data) #returns first n rows of thedata

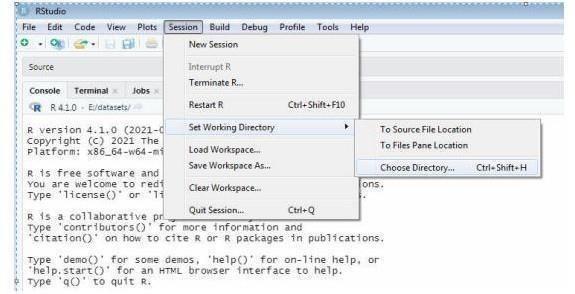


**\*Writing Data to a File**

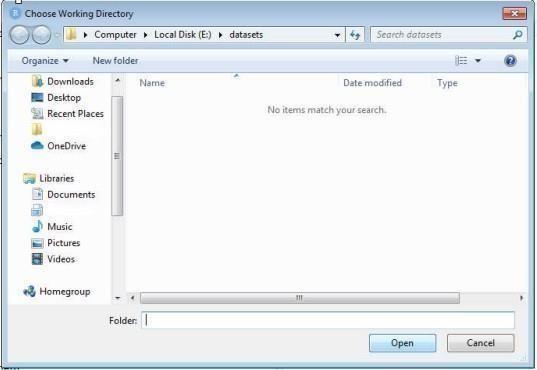
After working with a dataset, we might like to save it for future use. Before we do this, let's first set up a working directory so we know where we can find all our data sets and files later.

**Setting up a Directory**

From RStudio, use the menu to change your working directory under Session > Set Working Directory > Choose Directory



Click Open.



Alternatively, you can use the setwd() function to assign working directory.

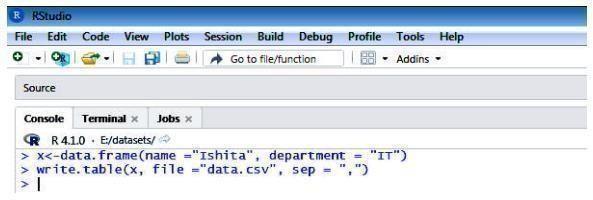
> setwd("E:/datasets")

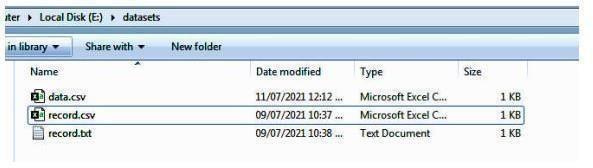
To check your current working directory, type

> getwd()

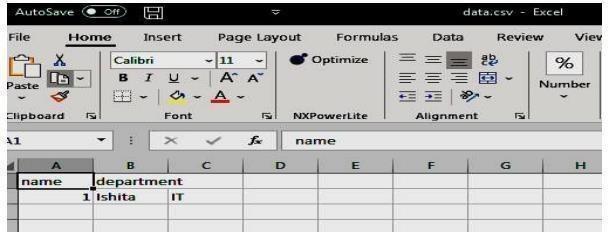
In R, we can write data easily to a file, using the write.table() command.

**x<-data.frame(name ="Ishita", department = "IT") write.table(x, file ="data.csv", sep = ",")**





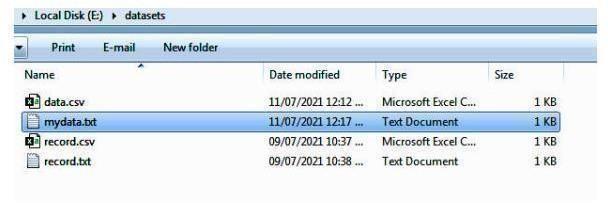
By going to this location E:/datasets,you should see a data.csv file.



**y<-data.frame(name ="Ankit", department = "HR") write.table(y,"E:/datasets/mydata.txt", sep = "\t")**



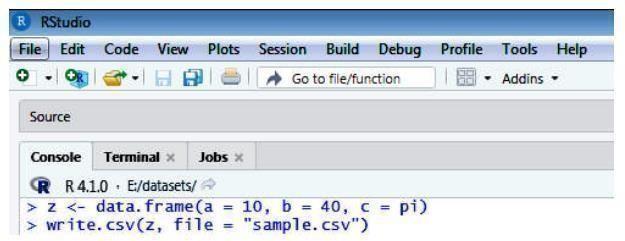
Now, let's check whether R created the file mydata.txt under E:/datasets folder or not.



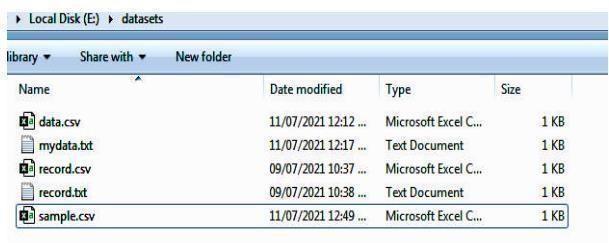
By going to this location E:/datasets, you should see a mydata.txt file.



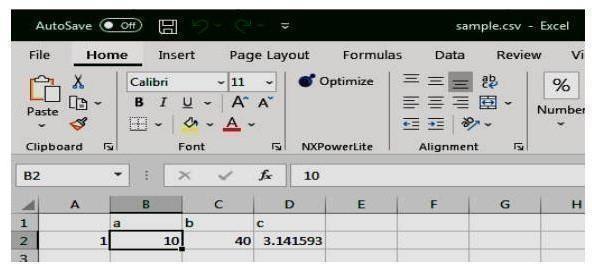
**z <- data.frame(a = 10, b = 40, c = pi) write.csv(z, file = "sample.csv")**



Now, let's check whether R created the file sample.csv under E:/datasets folder or not.



By going to this location E:/datasets,you should see a sample.csv file.



**EXPERIMENT NO: 06**

**Aim**: To study Linear Regression

**Code:** -   
  
import numpy as np

class LinearRegression:

def \_\_init\_\_ (self):

self.b0 = 0

self.b1 = 0

def fit (self, X, y):

X\_mean = np.mean (X)

y\_mean = np.mean (y)

numerator, denomintor = 0, 0

for \_ in range (len (X)):

numerator += (X[\_]-X\_mean)\*(y[\_]-y\_mean)

denomintor += (X[\_]-X\_mean)\*\*2

self.b1 = numerator / denomintor

self.b0 = y\_mean - (X\_mean\*self.b1)

return self.b0, self.b1

def predict (self, X):

y\_hat = self.b0 + (self.b1 \* X)

return y\_hat

if \_\_name\_\_ == '\_\_main\_\_':

X = np.array ([173, 160, 154, 188, 168], ndmin=2)

X = X.reshape (5, 1)

y = np.array ([73, 65, 54, 80, 70])

model = LinearRegression ()

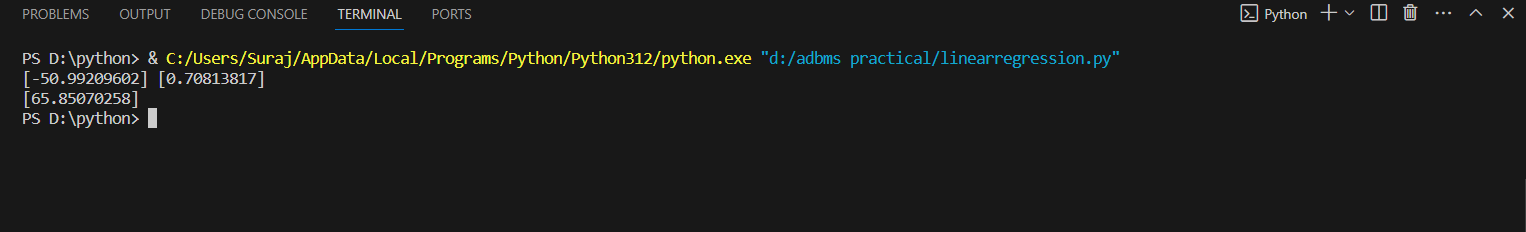
b0, b1 = model.fit (X, y)

print (b0, b1)

y\_pred = model.predict ([165])

print (y\_pred)

**Output** :



**EXPERIMENT NO: 7**

**Aim**: To study Analysis of Regression

**Code**:

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

if \_\_name\_\_ == '\_\_main\_\_':

X = np.array ([173, 160, 154, 188, 168], ndmin=2)

X = X.reshape (5, 1)

y = np.array ([73, 65, 54, 80, 70])

model = LinearRegression ()

model.fit (X, y)

y\_hat = model.predict (X)

print (y\_hat)

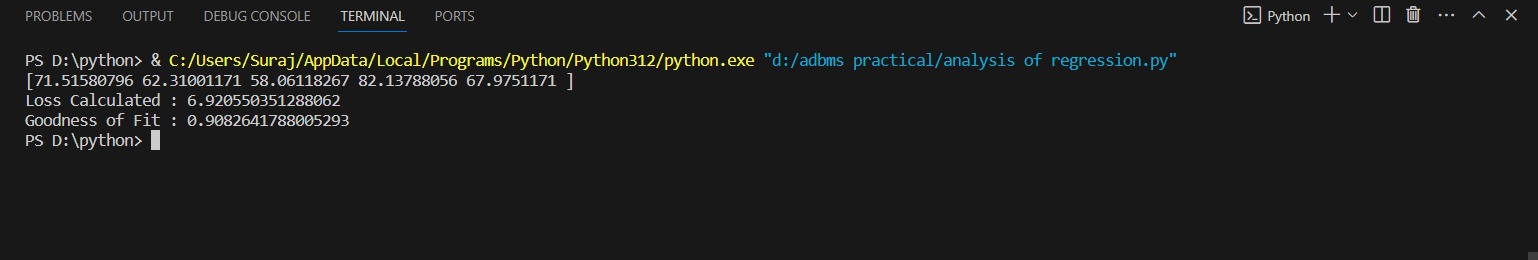
mse = mean\_squared\_error (y\_true=y, y\_pred=y\_hat)

print (f'Loss Calculated : {mse}')

r2 = r2\_score (y\_true=y, y\_pred=y\_hat)

print (f'Goodness of Fit : {r2}')

**Output**:



**EXPERIMENT NO: 08**

**Aim**: To study Logistic Regression

**Code**:

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import log\_loss

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score

if \_\_name\_\_ == '\_\_main\_\_':

X = np.array ([6, 2, 5, 9, 1], ndmin=2)

X = X.reshape (5, 1)

y = np.array ([1, 0, 1, 1, 0])

model = LogisticRegression ()

model.fit (X, y)

y\_hat = model.predict (X)

print (y\_hat)

loss = log\_loss (y\_true=y, y\_pred=y\_hat)

print (f'Logarithimic Log : {loss}')

cm = confusion\_matrix (y\_true=y, y\_pred=y\_hat)

precision = precision\_score (y\_true=y, y\_pred=y\_hat)

recall = recall\_score (y\_true=y, y\_pred=y\_hat)

f1 = f1\_score (y\_true=y, y\_pred=y\_hat)

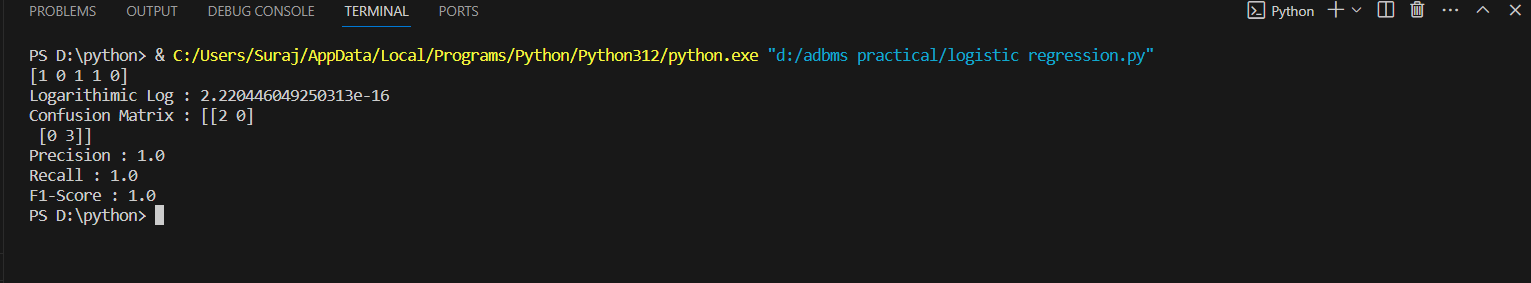
print (f'Confusion Matrix : {cm}')

print (f'Precision : {precision}')

print (f'Recall : {recall}')

print (f'F1-Score : {f1}')

**Output**:



**EXPERIMENT NO: 09**

**Aim**: To study support Vector Machine

**Code** –

from sklearn.datasets import load\_iris

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import precision\_score, recall\_score, f1\_score

if \_\_name\_\_ == '\_\_main\_\_':

iris = load\_iris ()

X = np.array (iris.data)

y = np.array (iris.target)

X\_train, X\_test, y\_train, y\_test = train\_test\_split (X, y, shuffle=True, test\_size=0.1)

model = SVC ()

model.fit (X\_train, y\_train)

y\_hat = model.predict (X\_test)

precision = precision\_score (y\_true=y\_test, y\_pred=y\_hat, average='micro')

recall = recall\_score (y\_true=y\_test, y\_pred=y\_hat, average='micro')

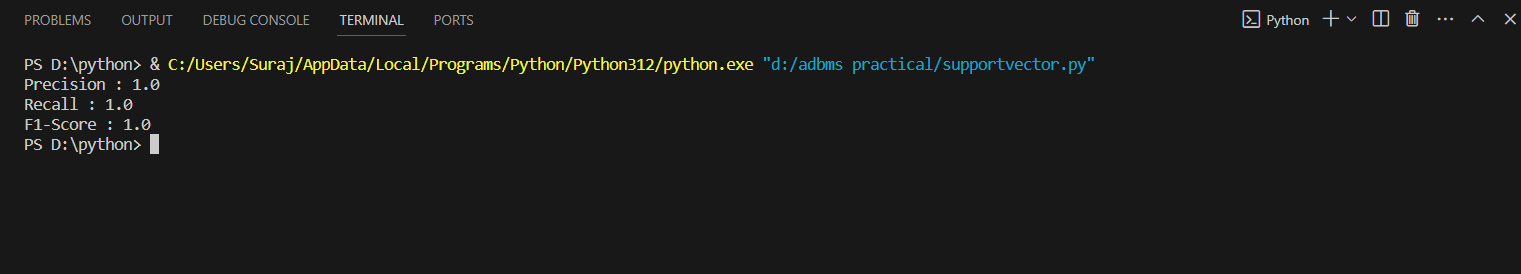
f1 = f1\_score (y\_true=y\_test, y\_pred=y\_hat, average='micro')

print (f'Precision : {precision}')

print (f'Recall : {recall}')

print (f'F1-Score : {f1}')

**Output**:



**EXPERIMENT NO: 10**

**Aim**: To study varied Algorithm

**Code** –

from sklearn.datasets import load\_iris

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

if \_\_name\_\_ == '\_\_main\_\_':

iris = load\_iris ()

X = np.array (iris.data)

y = np.array (iris.target)

X\_train, X\_test, y\_train, y\_test = train\_test\_split (X, y, shuffle=True, test\_size=0.1)

dt = DecisionTreeClassifier ()

dt.fit (X\_train, y\_train)

dt\_pred = dt.predict ([[1, 2, 3, 4]])

print (f'Decision Tree : {dt\_pred}')

rf = RandomForestClassifier ()

rf.fit (X\_train, y\_train)

rf\_pred = rf.predict ([[1, 2, 3, 4]])

print (f'Random Forest : {rf\_pred}')

knn = KNeighborsClassifier ()

knn.fit (X\_train, y\_train)

knn\_pred = knn.predict ([[1, 2, 3, 4]])

print (f'KNN : {knn\_pred}')

nb = GaussianNB ()

nb.fit (X\_train, y\_train)

nb\_pred = nb.predict ([[1, 2, 3, 4]])

print (f'Naive Bayes : {nb\_pred}')

**Output** –

