

Advanced Database Management System Lab

Subject Code: MCAL13

A Practical Journal Submitted in Fulfillment

of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2022-2023

By

(Ravishankar Jaiswal)
(172047)

Semester- 1

Under the Guidance of

Prof. Dr. Sujatha Iyer



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

PCP Center

[Satish Pradhan Dyanasadhana College, Thane]



**Institute of Distance and Open Learning,
Vidyanagari, Kalina, Santacruz (E) -400098**

CERTIFICATE

This to certify that, (**Ravishankar Jaiswal**) appearing **Master in Computer Application (Semester I)** Application ID: 172047 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 2022-23

Teacher in charge

Examiners

Coordinator
IDOL, MCA
University of Mumbai

Date: -01/04/2023

Place: - THANE

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	<ol style="list-style-type: none">1. installation of R2. datatype in R programming3. Reading and Writing data to and from R.
6	Data preprocessing in R.
7	<ol style="list-style-type: none">1. To implement Simple linear regression2. To implement multiple linear regression3. To implement Logistic regression4. Performing K Nearest Neighbour on Dataset (KNN)
8	To implement K means clustering

PRACTICAL 1

DISTRIBUTED DATABASE

RANGE Partitioning in mysql

Aim: Implementation of Data partitioning through Range.

```
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.1.28-rc-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use mca;
Database changed
mysql> CREATE TABLE tr1 (id INT, name VARCHAR(50), purchased
-> DATE)
-> PARTITION BY RANGE( YEAR(purchased) ) (
-> PARTITION p0 VALUES LESS THAN (1990),
-> PARTITION p1 VALUES LESS THAN (1995),
-> PARTITION p2 VALUES LESS THAN (2000),
-> PARTITION p3 VALUES LESS THAN (2005),
-> PARTITION p4 VALUES LESS THAN (2010),
-> PARTITION p5 VALUES LESS THAN (2015)
-> );
Query OK, 0 rows affected (0.30 sec)
```

```
mysql>
mysql> INSERT INTO tr1 VALUES
-> (1, 'desk organiser', '2003-10-15'),
-> (2, 'alarm clock', '1997-11-05'),
-> (3, 'chair', '2009-03-10'),
-> (4, 'bookcase', '1989-01-10'),
-> (5, 'exercise bike', '2014-05-09'),
-> (6, 'sofa', '1987-06-05'),
-> (7, 'espresso maker', '2011-11-22'),
-> (8, 'aquarium', '1992-08-04'),
-> (9, 'study desk', '2006-09-16'),
-> (10, 'lava lamp', '1998-12-25');
Query OK, 10 rows affected (0.05 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> select * from tr1;
+---+-----+-----+
| id | name | purchased |
+---+-----+-----+
| 4 | bookcase | 1989-01-10 |
| 6 | sofa | 1987-06-05 |
| 8 | aquarium | 1992-08-04 |
| 2 | alarm clock | 1997-11-05 |
| 10 | lava lamp | 1998-12-25 |
| 1 | desk organiser | 2003-10-15 |
| 3 | chair | 2009-03-10 |
| 9 | study desk | 2006-09-16 |
| 5 | exercise bike | 2014-05-09 |
| 7 | espresso maker | 2011-11-22 |
+---+-----+-----+
10 rows in set (0.02 sec)
```

```
mysql> ■
```

```
1
mysql> SELECT PARTITION_NAME, TABLE_ROWS FROM
-> INFORMATION_SCHEMA.PARTITIONS WHERE
-> TABLE_NAME='tr1';
+-----+-----+
| PARTITION_NAME | TABLE_ROWS |
+-----+-----+
| p0 | 2 |
| p1 | 1 |
| p2 | 2 |
| p3 | 1 |
| p4 | 2 |
| p5 | 2 |
+-----+-----+
6 rows in set (0.27 sec)
```

```
mysql>
```

PRACTICAL 2

ANALYTICAL QUERIES

Aim: Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Rank AND Dense Rank.

```
SQL> CREATE TABLE emp(
 2  empno NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,
 3  ename VARCHAR2(10),
 4  job VARCHAR2(10),
 5  mgr NUMBER(4),
 6  hiredate DATE,
 7  sal NUMBER(7,2),
 8  comm NUMBER(7,2),
 9  deptno NUMBER(2));

Table created.

SQL> INSERT INTO emp VALUES(1,'Hema','Developer',2,'22-May-2020',25000,2000,4)
 2  ;

1 row created.

SQL> INSERT INTO emp VALUES(2,'Ram','Developer',2,'20-April-2019',45000,2300,4);

1 row created.

SQL> INSERT INTO emp VALUES(3,'Vrudhi','Tester',4,'20-April-2019',30000,8000,5);

1 row created.
```

```
SQL> INSERT INTO emp VALUES(4,'Rahul','Tester',4,'5-November-2018',50000,8000,5);

1 row created.
```

```
SQL> SELECT * FROM emp;

      EMPNO ENAME      JOB          MGR HIREDATE        SAL       COMM
----- -----
      DEPTNO
----- 
      1 Hema      Developer      2 22-MAY-20    25000     2000
      4                   

      2 Ram       Developer      2 20-APR-19    45000     2300
      4                   

      3 Vrudhi    Tester        4 20-APR-19    30000     8000
      5                   

      EMPNO ENAME      JOB          MGR HIREDATE        SAL       COMM
----- -----
      DEPTNO
----- 
      4 Rahul     Tester        4 05-NOV-18    50000     8000
      5
```

```

SQL> SELECT empno,sal,sum(sal) as Totalsal from emp group by rollup(empno,sal);

      EMPNO        SAL    TOTALSAL
-----  -----
          1      25000      25000
          1          25000
          2      45000      45000
          2          45000
          3      30000      30000
          3          30000
          4      50000      50000
          4          50000
                         150000

9 rows selected.

SQL> -

```

```

SQL> SELECT empno,sal,sum(sal) as Totalsal from emp group by cube(empno,sal);

      EMPNO        SAL    TOTALSAL
-----  -----
                         150000
          30000      30000
          50000      50000
          25000      25000
          45000      45000
          1          25000
          1      25000      25000
          2          45000
          2      45000      45000
          3          30000
          3      30000      30000

      EMPNO        SAL    TOTALSAL
-----  -----
          4          50000
          4      50000      50000

13 rows selected.

```

RANK

```

SQL> SELECT empno,deptno,sal,RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank FROM emp;

      EMPNO     DEPTNO        SAL    MYRANK
-----  -----
          1         4      25000      1
          2         4      45000      2
          3         5      30000      1
          4         5      50000      2

```

```
SQL> SELECT empno,deptno,sal, DENSE_RANK() OVER (Partition By deptno ORDER By sal) as myrank FROM emp;

```

EMPNO	DEPTNO	SAL	MYRANK
1	4	25000	1
2	4	45000	2
3	5	30000	1
4	5	50000	2

DENSE_RANK

```
SQL> SELECT * FROM (SELECT empno,deptno,sal, DENSE_RANK() OVER (Partition By deptno ORDER By sal DESC) as myrank FROM emp) WHERE myrank<=2;

```

EMPNO	DEPTNO	SAL	MYRANK
2	4	45000	1
1	4	25000	2
4	5	50000	1
3	5	30000	2

SQL>

FIRST AND LAST

```
SQL> SELECT empno,deptno,sal, MIN(sal) KEEP (DENSE_RANK FIRST ORDER By sal) OVER (Partition By deptno) as lowest,MAX(sal) KEEP (DENSE_RANK LAST ORDER By sal) OVER (Partition By deptno) As highest FROM emp ORDER By deptno,sal;
```

EMPNO	DEPTNO	SAL	LOWEST	HIGHEST
1	4	25000	25000	45000
2	4	45000	25000	45000
3	5	30000	30000	50000
4	5	50000	30000	50000

LAG

```
SQL> SELECT deptno,empno,ename,job,sal, LAG(sal,1,0) OVER (PARTITION By deptno ORDER BY sal) As sal_prev FROM emp;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	SAL_PREV
4	1	Hema	Developer	25000	0
4	2	Ram	Developer	45000	25000
5	3	Vrudhi	Tester	30000	0
5	4	Rahul	Tester	50000	30000

LEAD

```
SQL> SELECT empno,ename,job,sal, LEAD(sal,1,0) OVER (ORDER By sal) As sal_next, LEAD(sal,1,0) OVER (ORDER By sal)-sal As sal_diff FROM emp;
```

EMPNO	ENAME	JOB	SAL	SAL_NEXT	SAL_DIFF
1	Hema	Developer	25000	30000	5000
3	Vrudhi	Tester	30000	45000	15000
2	Ram	Developer	45000	50000	5000
4	Rahul	Tester	50000	0	-50000

```
SQL> SELECT deptno,empno,ename,job,sal, LEAD(sal,1,0) OVER (PARTITION By deptno ORDER BY sal) As sal_next FROM emp;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	SAL_NEXT
4	1	Hema	Developer	25000	45000
4	2	Ram	Developer	45000	0
5	3	Vrudhi	Tester	30000	50000
5	4	Rahul	Tester	50000	0

```
SQL>
```

PRACTICAL 3

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);
```

SQL> desc customer_reltab;		
Name	Null?	Type
CUSTNO	NOT NULL	NUMBER
CUSTNAME	NOT NULL	VARCHAR2(200)
STREET	NOT NULL	VARCHAR2(200)
CITY	NOT NULL	VARCHAR2(200)
STATE	NOT NULL	CHAR(2)
ZIP	NOT NULL	VARCHAR2(20)
PHONE1		VARCHAR2(20)
PHONE2		VARCHAR2(20)
PHONE3		VARCHAR2(20)

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));
```

```
SQL> CREATE TABLE PurchaseOrder_reltab (
  2    PONo      NUMBER,
  3    Custno   NUMBER REFERENCES Customer_reltab,
  4    OrderDate DATE,
  5    ShipDate DATE,
  6    ToStreet  VARCHAR2(200),
  7    ToCity   VARCHAR2(200),
  8    ToState   CHAR(2),
  9    ToZip    VARCHAR2(20),
10    PRIMARY KEY(PONo));
Table created.
```

Stock_reltab

The Stock_reltab table has the following definition:

```
CREATE TABLE Stock_reltab (
  StockNo  NUMBER PRIMARY KEY,
  Price    NUMBER,
  TaxRate  NUMBER);
```

```
SQL> CREATE TABLE Stock_reltab (
  2    StockNo    NUMBER PRIMARY KEY,
  3    Price     NUMBER,
  4    TaxRate   NUMBER);
Table created.
```

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));
```

```

SQL> CREATE TABLE LineItems_reltab (
 2   LineItemNo      NUMBER,
 3   PONo            NUMBER REFERENCES PurchaseOrder_reltab,
 4   StockNo         NUMBER REFERENCES Stock_reltab,
 5   Quantity        NUMBER,
 6   Discount        NUMBER,
 7   PRIMARY KEY (PONo, LineItemNo));
Table created.

```

Inserting Values Under the Relational Model

In our application, statements like these insert data into the tables:

```

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');
INSERT INTO LineItems_reltab VALUES(01, 1001, 1534, 12, 0);
INSERT INTO LineItems_reltab VALUES(02, 1001, 1535, 10, 10);
INSERT INTO LineItems_reltab VALUES(01, 2001, 1004, 1, 0);
INSERT INTO LineItems_reltab VALUES(02, 2001, 1011, 2, 1);

```

Querying Data Under the Relational Model

The application can execute queries like these:

```

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

```

```
WHERE C.CustNo = P.CustNo  
AND P.PONo = L.PONo AND  
P.PONo = 1001;
```

Get the Total Value of Purchase Orders

```
SELECT  
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 BY P.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. The application can execute statements like these to update the data:

```
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;
```

PRACTICAL 4

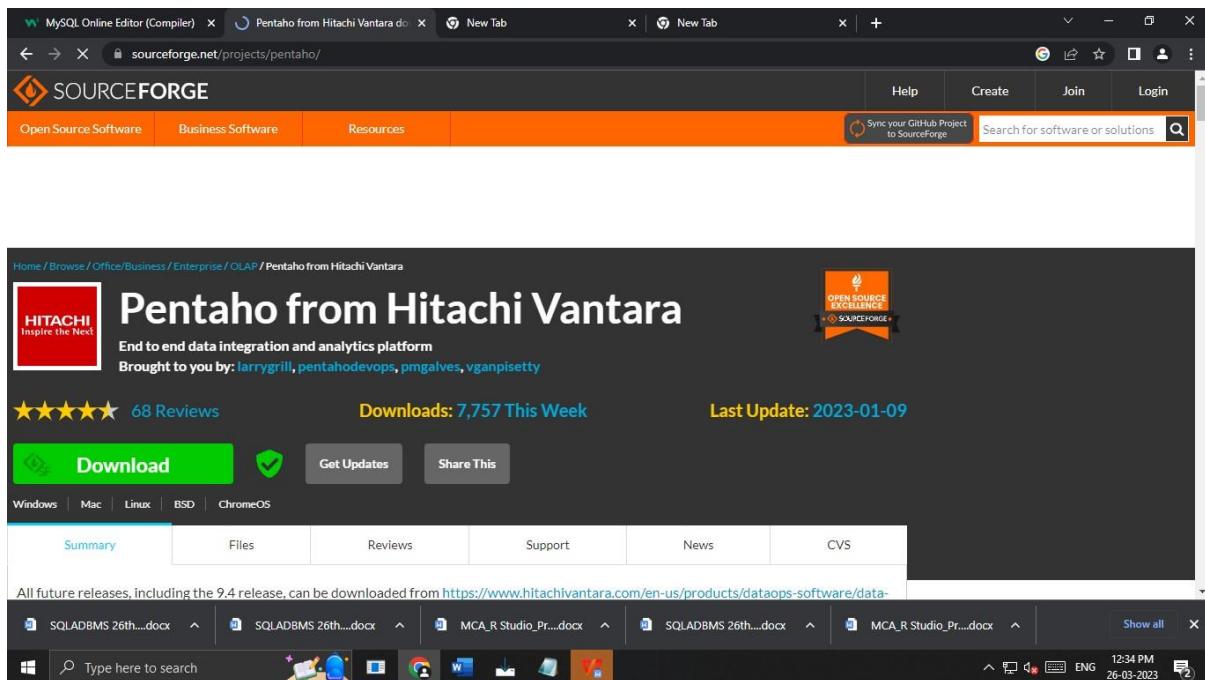
Aim: To study ETL process

* 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

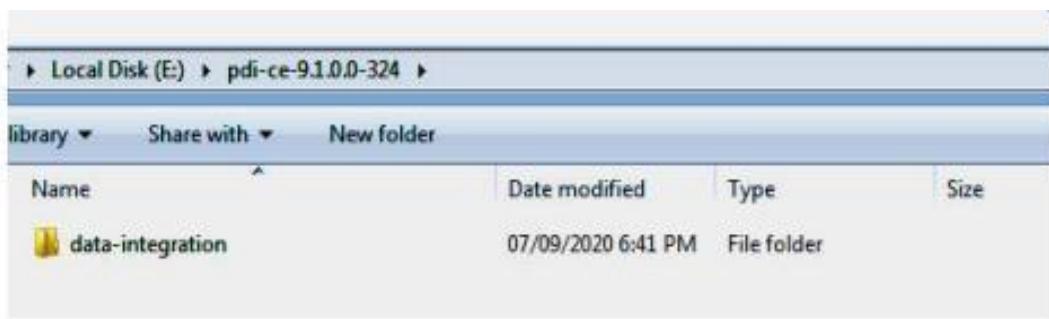
from link <https://sourceforge.net/projects/pentaho/>.



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:



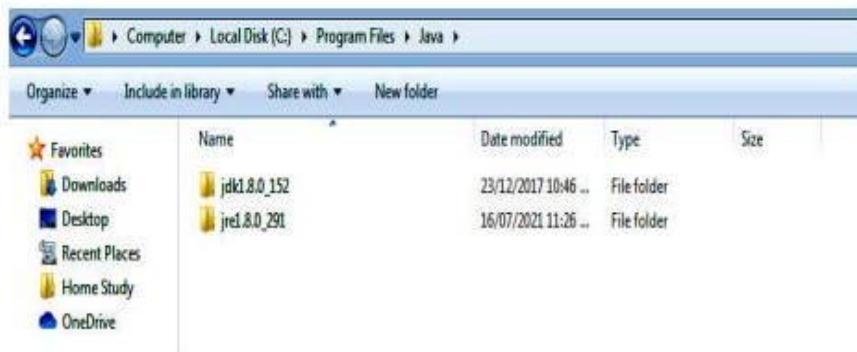
Step 2: Install the Java Dependencies, if Required.

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:

A screenshot of the Oracle Java download page for Windows. The left sidebar has 'Help Resources' with links: What is Java?, Remove Older Versions, Disable Java, Error Messages, Troubleshoot Java, and Other Help. The main content area is titled '64-bit Java for Windows' and says 'Recommended Version 8 Update 291 (filesize: 80.7 MB)' with a release date of April 20, 2021. A yellow box contains an important message: 'Important Oracle Java License Update' and 'The Oracle Java License has changed for releases starting April 16, 2019.' It explains the new Oracle Technology Network License Agreement for Oracle Java SE. Below this, it says 'Commercial license and support is available with a low cost Java SE Subscription.' and 'Oracle also provides the latest OpenJDK release under the open source GPL License at openjdk.java.net.' Another yellow box at the bottom left says 'We have detected you are using Google Chrome and might be unable to use the Java plugin from this browser. Starting with Version 42 (released April 2015), Chrome has disabled the standard way in which browsers support plugins. [More info](#)'.

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-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.

A screenshot of a web browser displaying the Oracle Java Downloads page. The URL in the address bar is "https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html". The page features the Oracle logo and navigation links for Products, Industries, Resources, Support, Events, and Developer. Below this, there is a table listing Java distributions:

It will open following window

X

You must accept the [Oracle Technology Network License Agreement](#) for Oracle Java SE to download this software.

I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

Required

You will be redirected to the login screen in order to download the file.

Download jdk-8u291-windows-i586.exe 

Press “Download”.

X

You must accept the [Oracle Technology Network License Agreement](#) for Oracle Java SE to download this software.

I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

Required

You will be redirected to the login screen in order to download the file.

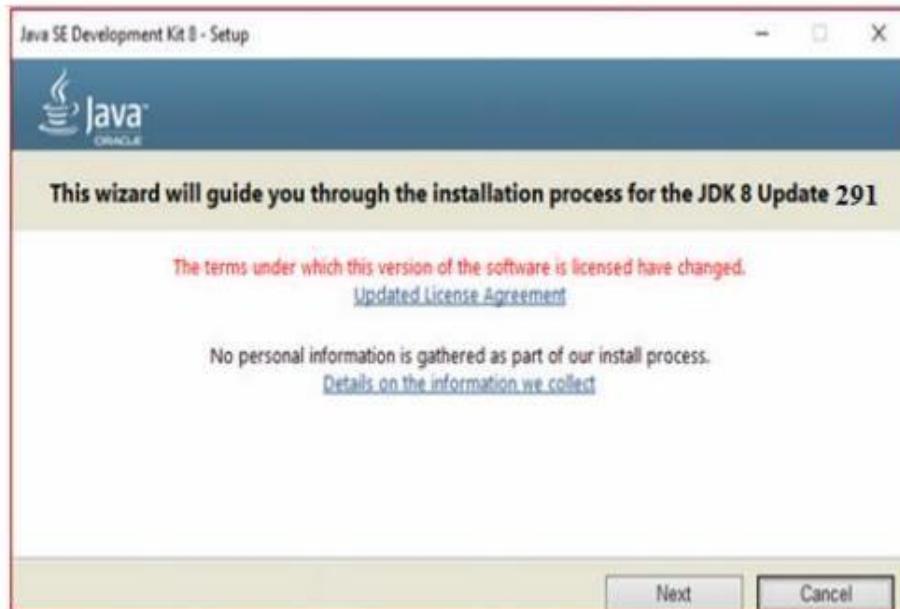
Download jdk-8u291-windows-i586.exe 

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.



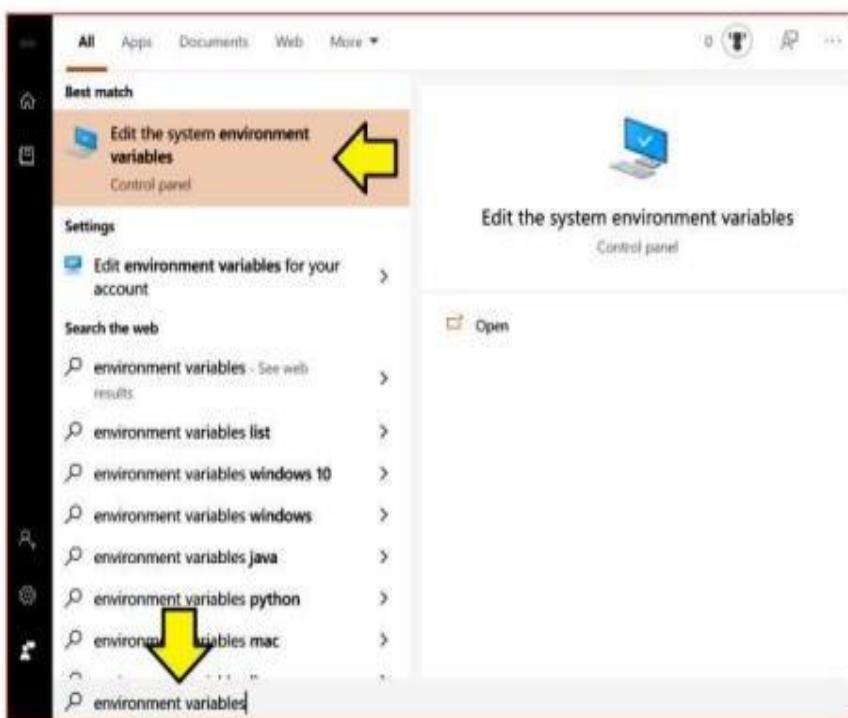
The installation setup will look like this:



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.

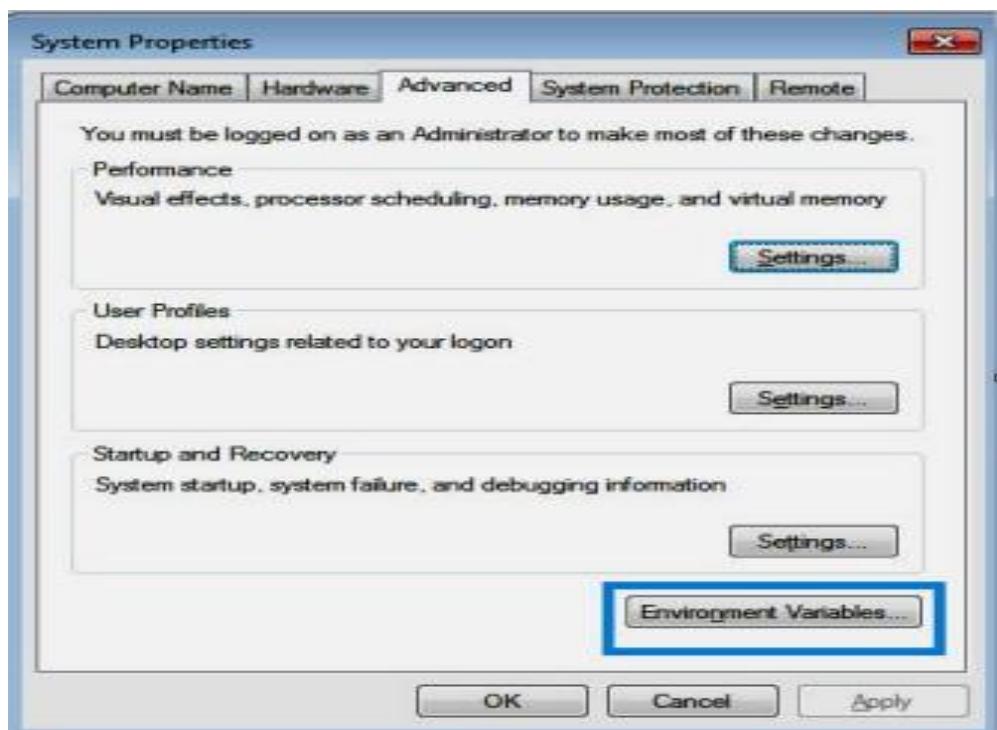
Step 3: Set Up the Environment Variables There are three environment variables that need to be set up. To open the environment variables menu type in “environment variables” in the Windows search bar like this:



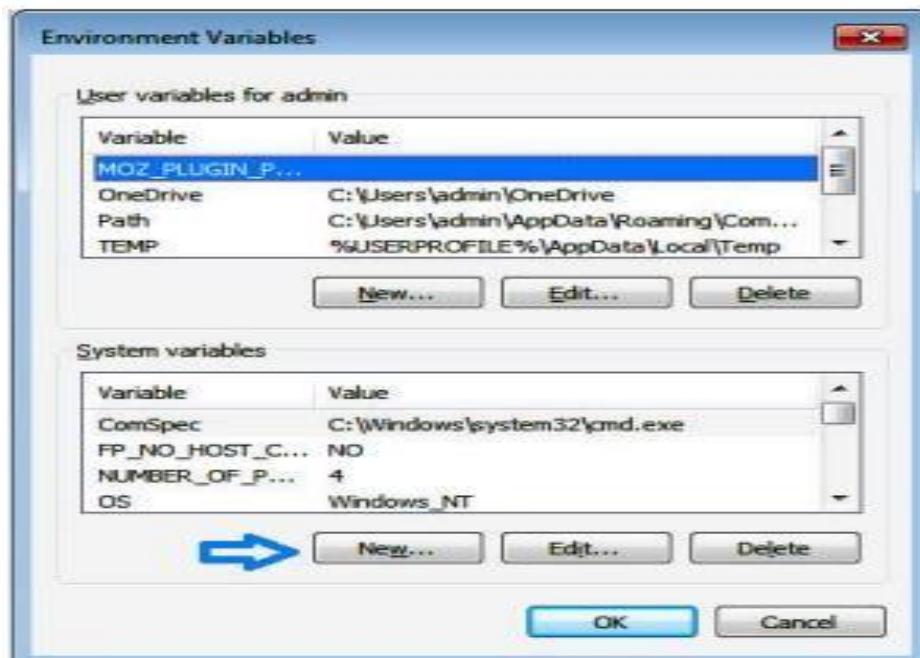
Click the “Edit the system environment variables” option.

That will open the “System Properties” window.

Under Advanced tab ...Click the “Environment Variables.” button at the bottom.

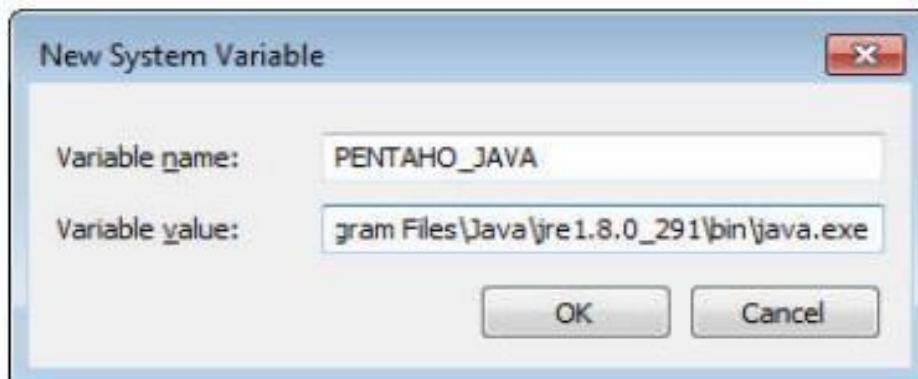


That will open a window that looks like this:



We need to add three new System variables.

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



Variable value: C:\Program Files\Java\jre1.8.0_251\bin\java.exe

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 Integration app.

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

Name	Date modified	Type	Size
ADDITIONAL-FILES	07/09/2020 6:41 PM	File folder	
classes	07/09/2020 4:52 PM	File folder	
Data Integration.app	07/09/2020 4:52 PM	File folder	
Data Service JDBC Driver	07/09/2020 6:42 PM	File folder	
docs	07/09/2020 4:52 PM	File folder	
drivers	07/09/2020 6:41 PM	File folder	
launcher	07/09/2020 4:52 PM	File folder	
lib	07/09/2020 6:40 PM	File folder	
libswt	07/09/2020 6:42 PM	File folder	
plugins	07/09/2020 4:52 PM	File folder	
pwd	07/09/2020 4:52 PM	File folder	
samples	07/09/2020 4:52 PM	File folder	
simple-jndi	07/09/2020 4:52 PM	File folder	
static	07/09/2020 4:52 PM	File folder	
system	07/09/2020 6:41 PM	File folder	
ui	07/09/2020 4:52 PM	File folder	
Carte.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
carte.sh	07/09/2020 4:52 PM	SH File	2 KB
Encr.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
encr.sh	07/09/2020 4:52 PM	SH File	2 KB
Import.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB

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

Local Disk (E:) > pdi-ce-9.1.0.0-324 > data-integration >				
		Date modified	Type	Size
	Pan.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
	pan.sh	07/09/2020 4:52 PM	SH File	2 KB
	PentahoDataIntegration_OSS_Licenses.htm...	07/09/2020 4:50 PM	HTML Document	3 KB
	purge-utility.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
	purge-utility.sh	07/09/2020 4:52 PM	SH File	2 KB
	README.txt	07/09/2020 4:52 PM	Text Document	2 KB
	README-spark-app-builder.txt	07/09/2020 4:52 PM	Text Document	3 KB
	runSamples.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
	runSamples.sh	07/09/2020 4:52 PM	SH File	2 KB
	set-pentaho-env.bat	07/09/2020 4:52 PM	Windows Batch File	6 KB
	set-pentaho-env.sh	07/09/2020 4:52 PM	SH File	5 KB
	Spark-app-builder.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
	spark-app-builder.sh	07/09/2020 4:52 PM	SH File	2 KB
	Spoon.bat	07/09/2020 4:52 PM	Windows Batch File	6 KB
	spoon.command	07/09/2020 4:52 PM	COMMAND File	2 KB
	spoon.ico	07/09/2020 4:52 PM	Icon	204 KB
	spoon.png	07/09/2020 4:52 PM	PNG image	1 KB
	spoon.sh	07/09/2020 4:52 PM	SH File	8 KB
	SpoonConsole.bat	07/09/2020 4:52 PM	Windows Batch File	2 KB
	SpoonDebug.bat	07/09/2020 4:52 PM	Windows Batch File	3 KB
	SpoonDebug.sh	07/09/2020 4:52 PM	SH File	2 KB

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.”

PRACTICAL 5

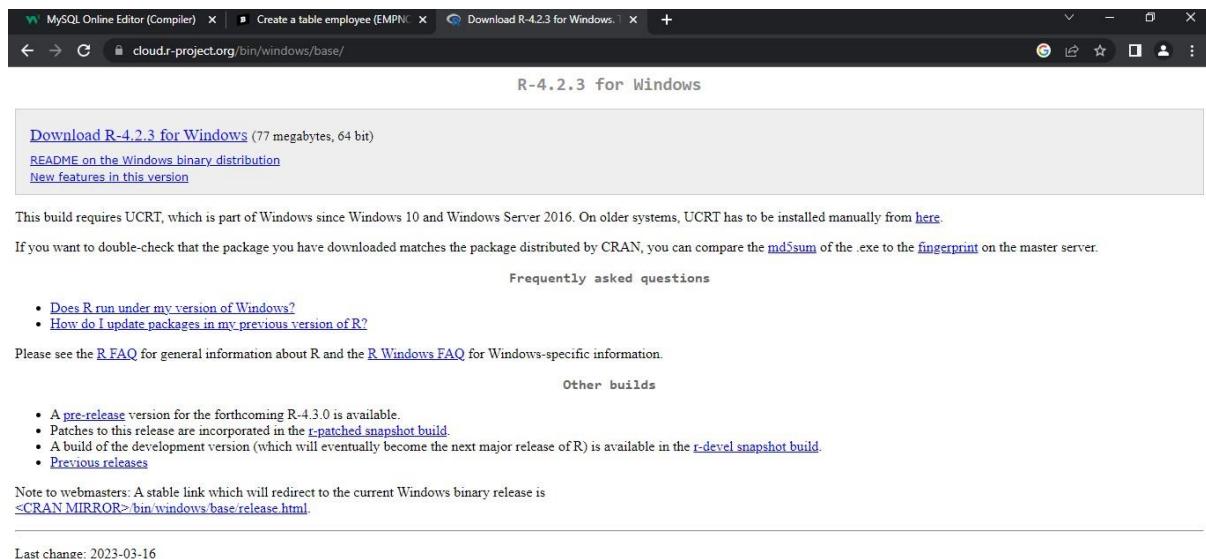
EXPERIMENT 1

Aim: installation of R

* R Installation in Windows

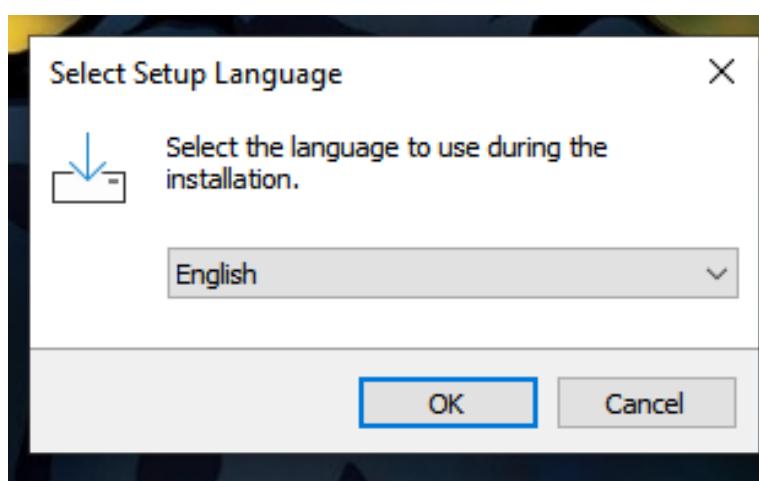
Steps used to install the R in Windows are as follows:

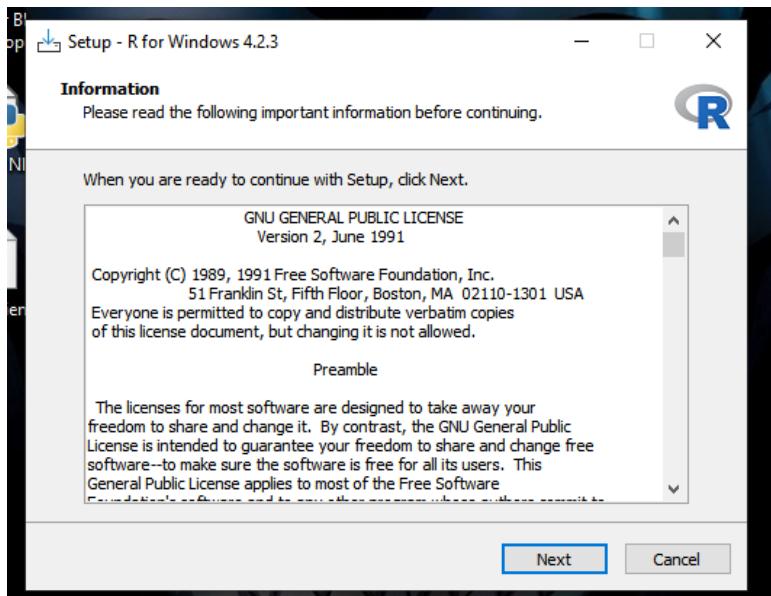
Step 1: First, we have to download the R setup from <https://cloud.r-project.org/bin/windows/base/>.



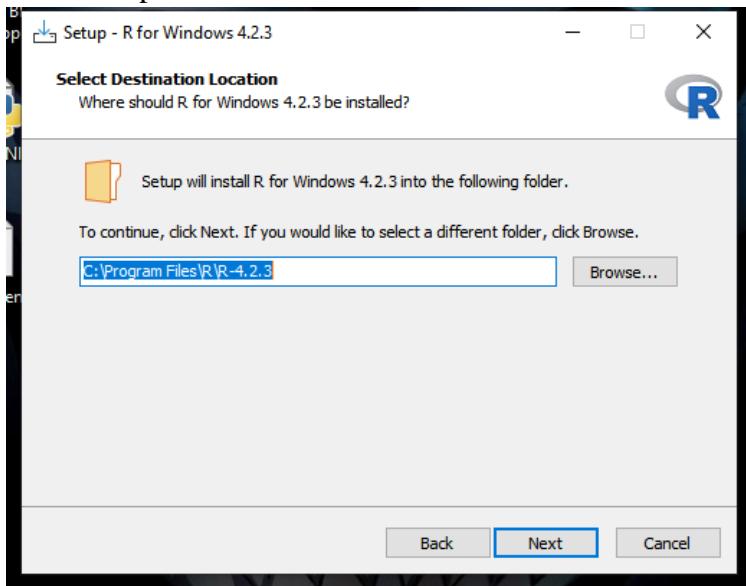
Step 2:

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:

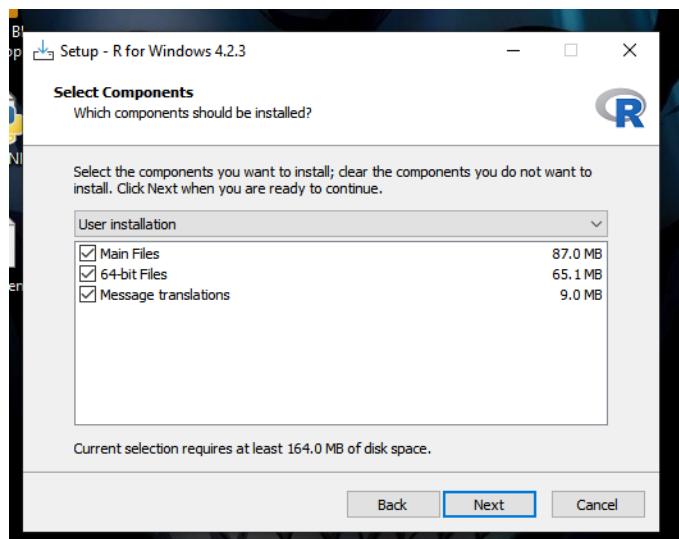




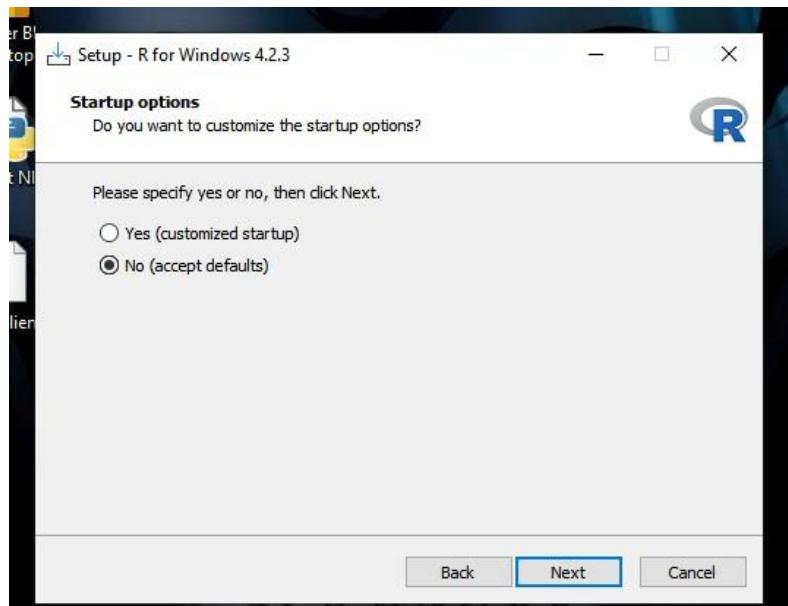
- 1) Select the path where we want to download the R and click Next



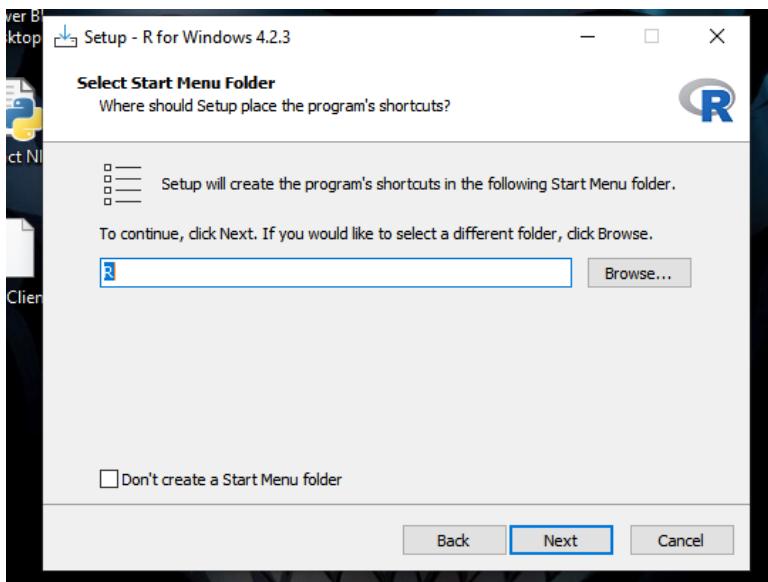
- 2) Select all components which we want to install, and then click Next.



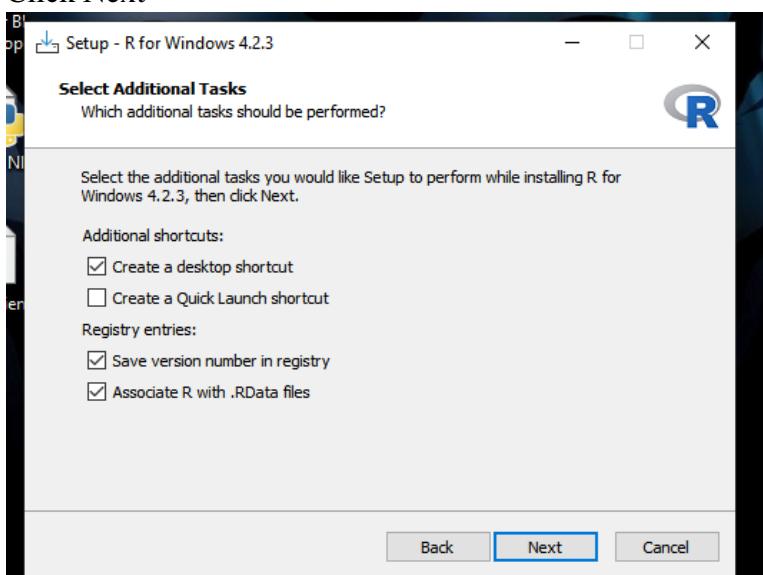
- 3) Now, we have to select either (customized startup) or (accept the default), and then click Next.



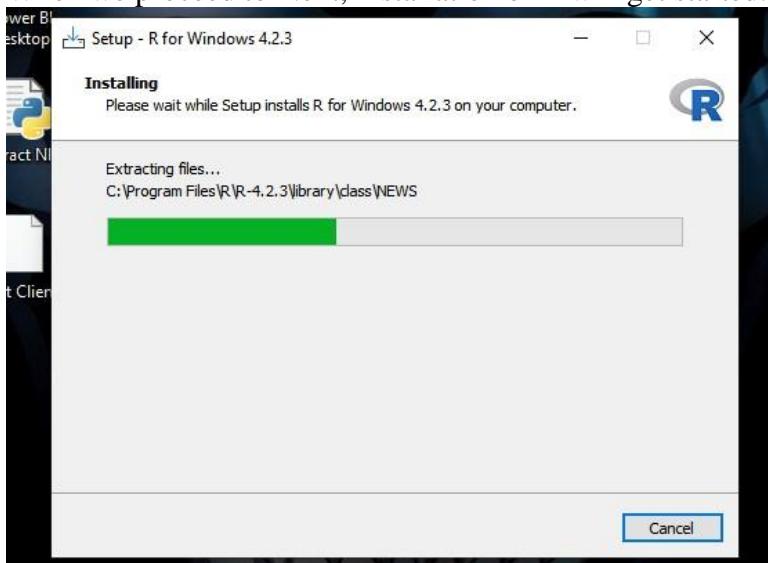
- 4) Now Select Start Menu Folder window will appear, click Next



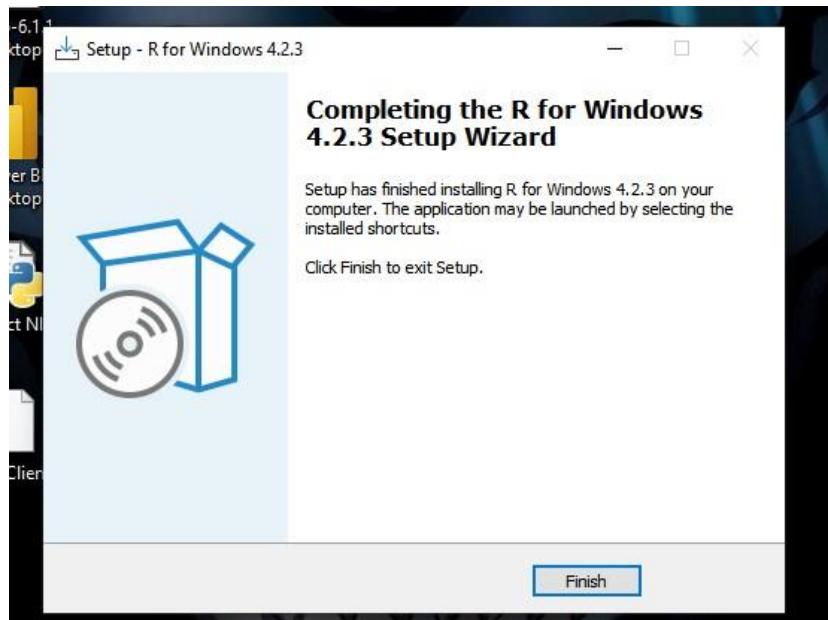
Click Next



5) When we proceed to Next, installation of R will get started:



6) Finally, we will click on Finish.



R has been successfully installed.

EXPERIMENT 2

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))
```

A screenshot of an R console window titled "R 4.1.3". The console shows the following R code and its output:

```
>  
> # Assign a decimal value to variable x  
> x = 5.6  
> # print the class name of variable x  
> print(class(x))  
[1] "numeric"  
> # print the type of variable x  
> print(typeof(x))  
[1] "double"  
>  
>
```

The console interface includes a menu bar with "File", "Edit", "View", "Help", and "Help Contents".

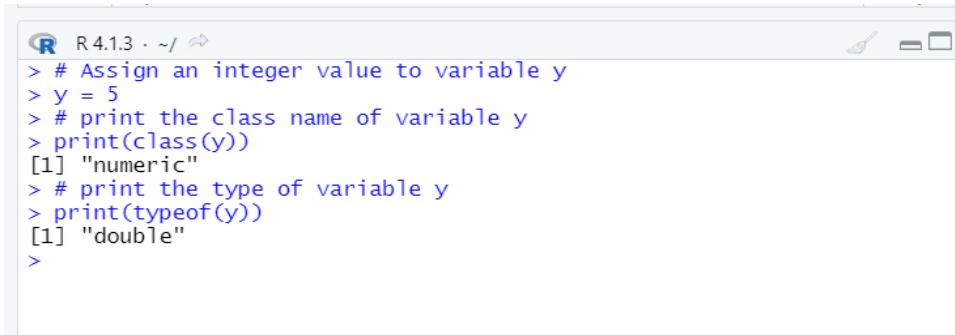
Output:

```
[1] "numeric"  
[1] "double"
```

2. R program to illustrate Numeric data type

```
# 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))
```



The screenshot shows the R 4.1.3 interface with a code editor window. The code is identical to the one above, but the output is visible below it:

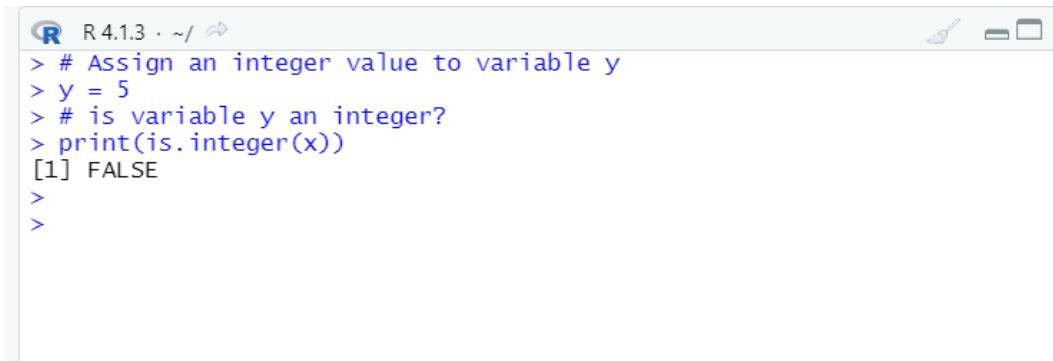
```
> # Assign an integer value to variable y
> y = 5
> # print the class name of variable y
> print(class(y))
[1] "numeric"
> # print the type of variable y
> print(typeof(y))
[1] "double"
>
```

Output:

```
[1] "numeric"
[1] "double"
```

3. R program to illustrate Numeric data type

```
# Assign an integer value to variable y
y = 5
# is variable y an integer?
print(is.integer(x))
```



The screenshot shows the R 4.1.3 interface with a code editor window. The code is identical to the one above, but the output is visible below it:

```
> # Assign an integer value to variable y
> y = 5
> # is variable y an integer?
> print(is.integer(x))
[1] FALSE
>
>
```

Output:

```
[1] FALSE
```

4. 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))

```

```

R 4.1.3 · ~/ ◁
> # Create an integer variable
> x = as.integer(5)
> # print the class name of variable x
> print(class(x))
[1] "integer"
> # print the type of variable x
> print(typeof(x))
[1] "integer"
> # Declare an integer by appending 'L' as suffix.
> y = 5L
> # print the class name of y
> print(class(y))
[1] "integer"
> # print the type of y
> print(typeof(y))
[1] "integer"
>
>

```

Output:

```

[1] "integer"
[1] "integer"
[1] "integer"
[1] "integer"
[1] "integer"

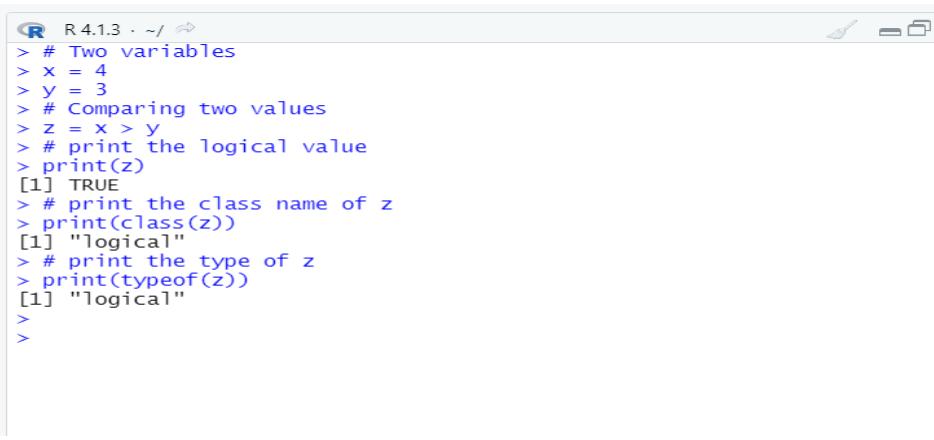
```

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))

```



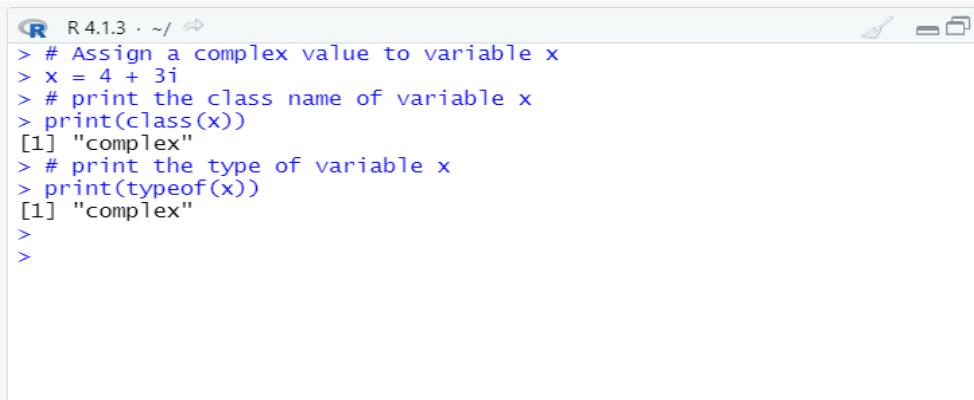
```
R 4.1.3 · ~/ ↗
> # Two variables
> x = 4
> y = 3
> # Comparing two values
> z = x > y
> # print the logical value
> print(z)
[1] TRUE
> # print the class name of z
> print(class(z))
[1] "logical"
> # print the type of z
> print(typeof(z))
[1] "logical"
>
>
```

Output:

```
[1] TRUE
[1] "logical"
[1] "logical"
```

6. R program to illustrate complex data type

```
# 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))
```



```
R 4.1.3 · ~/ ↗
> # Assign a complex value to variable x
> x = 4 + 3i
> # print the class name of variable x
> print(class(x))
[1] "complex"
> # print the type of variable x
> print(typeof(x))
[1] "complex"
>
>
```

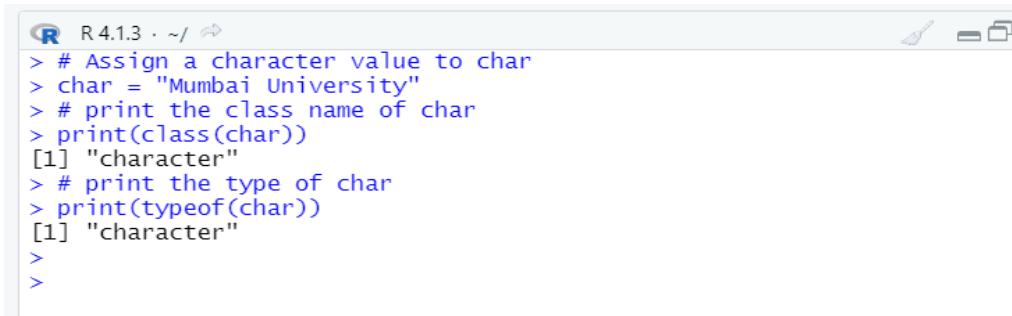
Output:

```
[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))
```



```
R 4.1.3 · ~/ 
> # Assign a character value to char
> char = "Mumbai University"
> # print the class name of char
> print(class(char))
[1] "character"
> # print the type of char
> print(typeof(char))
[1] "character"
>
>
```

Output:

```
[1] "character"
[1] "character"
```

Experiment 3

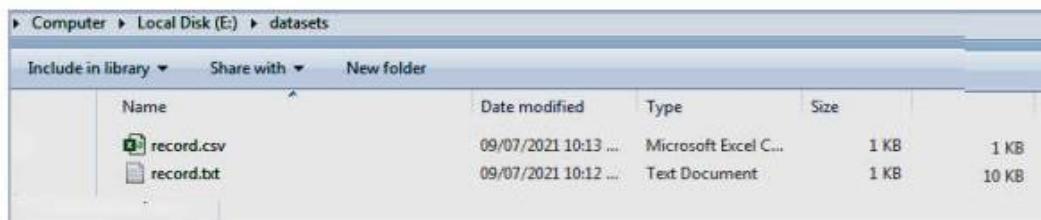
Aim: Reading and Writing data to and from R.

Reading data files with `read.table()`

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

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

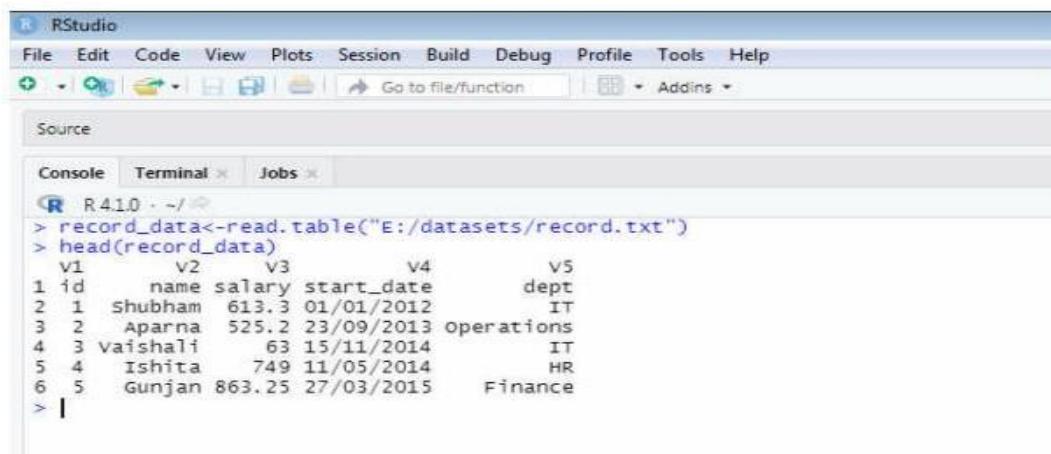
We have kept `record.txt` and `record.csv` files under datasets folder inside E: drive.



```
> record_data<- read.table("E:/datasets/record.txt")
```

```
> head(record_data)
```

```
#returns first n rows of the data
```

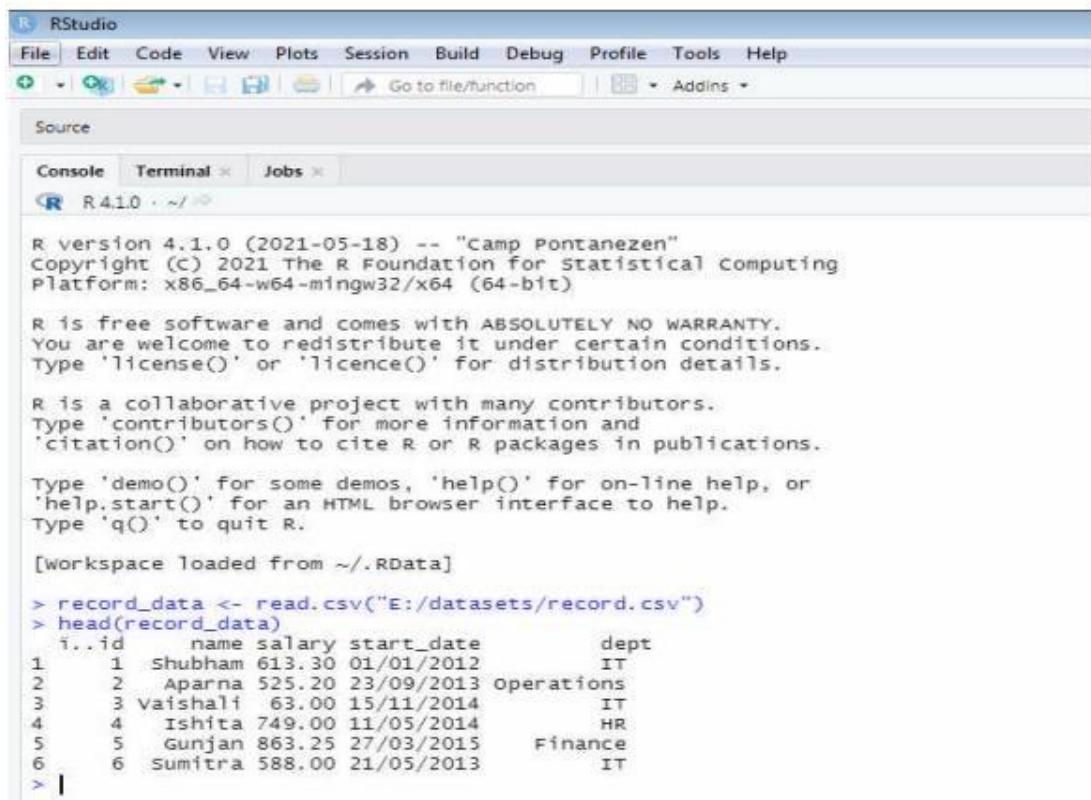


```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ - O G Go to file/function Addins

Source
Console Terminal Jobs
R 4.1.0 · ~/ 
> record_data<-read.table("E:/datasets/record.txt")
> head(record_data)
      v1      v2      v3      v4      v5
1  id    name  salary start_date   dept
2  1  Shubham  613.3  01/01/2012     IT
3  2  Aparna  525.2  23/09/2013 operations
4  3 Vaishali    63  15/11/2014     IT
5  4  Ishita   749  11/05/2014     HR
6  5  Gunjan  863.25 27/03/2015 Finance
> |
```

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 the data
```



The screenshot shows the RStudio interface with the R console tab selected. The console window displays the R startup message, followed by the command `> record_data <- read.csv("E:/datasets/record.csv")`, and the resulting output which is the head of the `record_data` dataset. The dataset contains information about employees, including their ID, name, salary, start date, and department.

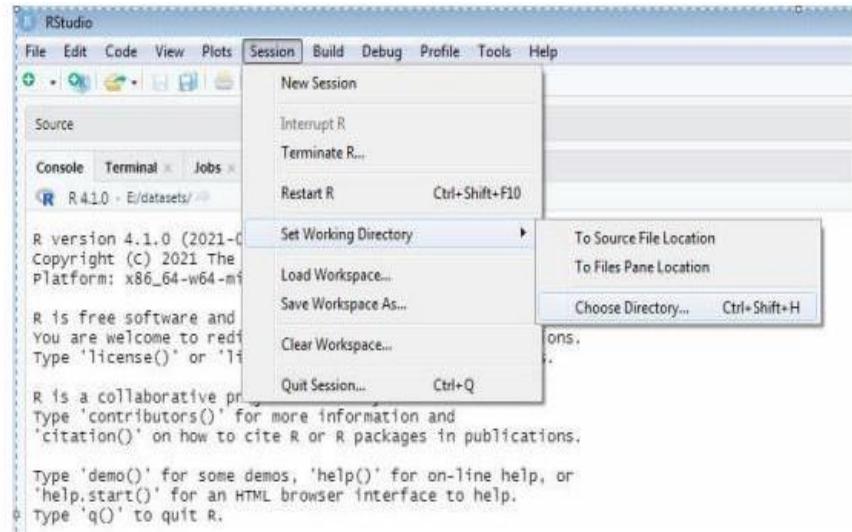
	i..id	name	salary	start_date	dept
1	1	Shubham	613.30	01/01/2012	IT
2	2	Aparna	525.20	23/09/2013	Operations
3	3	Vaishali	63.00	15/11/2014	IT
4	4	Ishita	749.00	11/05/2014	HR
5	5	Gunjan	863.25	27/03/2015	Finance
6	6	Sumitra	588.00	21/05/2013	IT

*Writing Data to a File

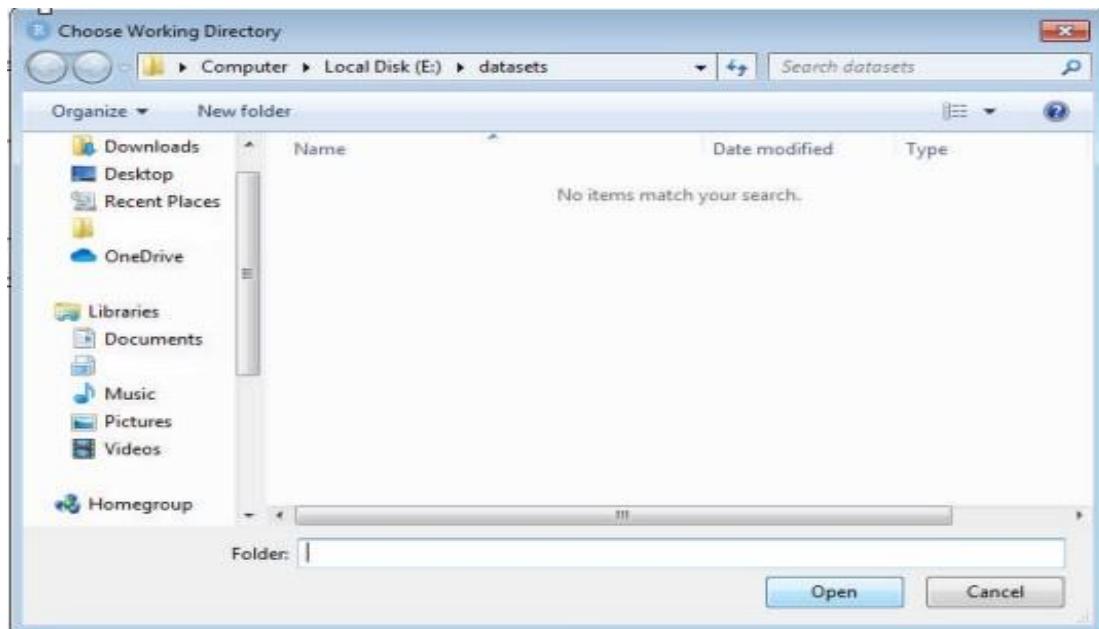
After working with a dataset, we might like to save it for future use. Before we do this, let's first set up a working directory so we know where we can find all our data sets and files later.

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

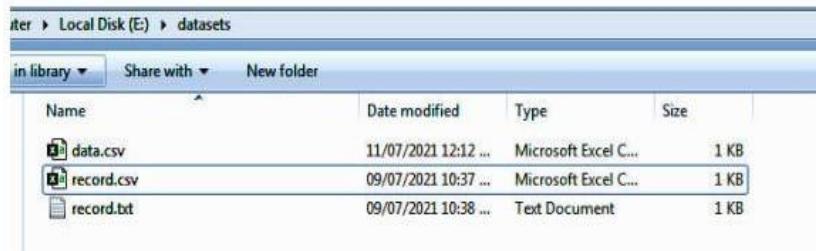
```
> getwd()
```

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 = ",")
```

RStudio interface showing the console tab with the following R code:

```
> 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.

Microsoft Excel spreadsheet titled "data.csv" showing the following data:

A	B	C	D	E	F	G	H
name							
Ishita		IT					

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

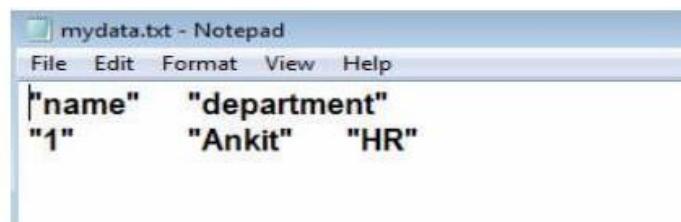
```
> y<-data.frame(name ="Ankit", department = "HR")
> write.table(y, "E:/datasets/mydata.txt", sep = "\t")
> |
```

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

Local Disk (E) > datasets

Name	Date modified	Type	Size
data.csv	11/07/2021 12:12 ...	Microsoft Excel C...	1 KB
mydata.txt	11/07/2021 12:17 ...	Text Document	1 KB
record.csv	09/07/2021 10:37 ...	Microsoft Excel C...	1 KB
record.txt	09/07/2021 10:38 ...	Text Document	1 KB

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")
```

R RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal × Jobs ×

R 4.1.0 · E:/datasets/ ↗

```
> 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.

Local Disk (E) > datasets

Name	Date modified	Type	Size
data.csv	11/07/2021 12:12 ...	Microsoft Excel C...	1 KB
mydata.txt	11/07/2021 12:17 ...	Text Document	1 KB
record.csv	09/07/2021 10:37 ...	Microsoft Excel C...	1 KB
record.txt	09/07/2021 10:38 ...	Text Document	1 KB
sample.csv	11/07/2021 12:49 ...	Microsoft Excel C...	1 KB

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

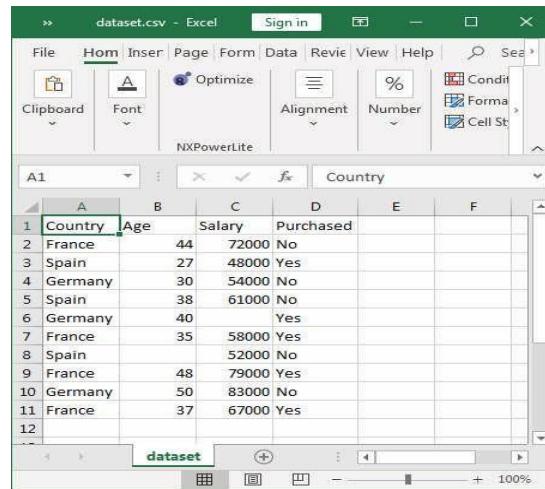
The screenshot shows a Microsoft Excel window titled "sample.csv - Excel". The ribbon menu is visible at the top, with "Home" selected. The "Font" group on the ribbon shows "Calibri" and "11". The "Number" group shows a percentage symbol (%). The "Alignment" group shows a center alignment icon. The formula bar shows "B2" and "10". The main worksheet area has columns A through H and rows 1 through 3. Row 1 contains cells A1 ("1"), B1 ("a"), C1 ("b"), D1 ("c"), E1 (" "), F1 (" "), G1 (" "), and H1 (" "). Row 2 contains cells A2 ("1"), B2 ("10"), C2 ("40"), D2 ("3.141593"), E2 (" "), F2 (" "), G2 (" "), and H2 (" "). Cell B2 is currently selected, indicated by a black border.

	A	B	C	D	E	F	G	H
1	1	a	b	c				
2	1	10	40	3.141593				

PRACTICAL 6

Aim: Data preprocessing in R.

◊ Data Preprocessing in R

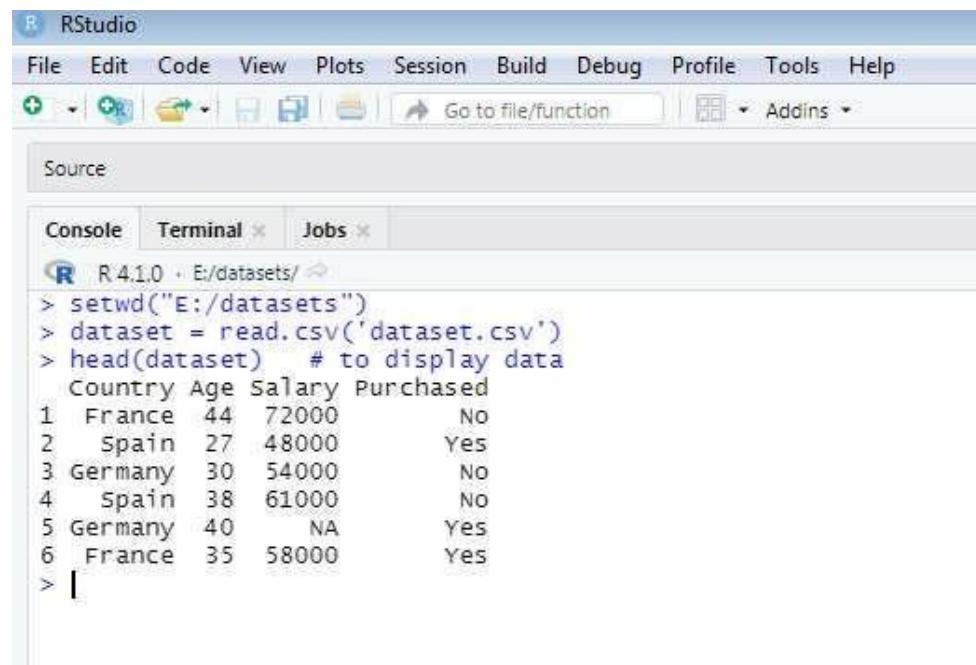


Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

dataset.csv file

◊ Importing the Dataset

Here, first we will change the working directory to E:/datasets (where dataset.csv is stored)



```
R 4.1.0 + E:/datasets/
> setwd("E:/datasets")
> dataset = read.csv('dataset.csv')
> head(dataset) # to display data
  Country Age Salary Purchased
1 France  44 72000      No
2 Spain   27 48000     Yes
3 Germany 30 54000      No
4 Spain   38 61000      No
5 Germany 40    NA     Yes
6 France  35 58000     Yes
> |
```

To display all 7 rows from csv file

```

> head(dataset,10)
  Country Age salary Purchased
1  France  44  72000      No
2  Spain   27  48000     Yes
3 Germany  30  54000      No
4  Spain   38  61000      No
5 Germany  40    NA     Yes
6  France  35  58000     Yes
7  Spain   NA  52000      No
8  France  48  79000     Yes
9 Germany  50  83000      No
10 France  37  67000    Yes
> |

```

↳ Dealing with Missing Values

```
dataset$Age = ifelse(is.na(dataset$Age),ave(dataset$Age, FUN = function(x) mean(x, na.rm = 'TRUE')),dataset$Age)
```

```
dataset$Salary = ifelse(is.na(dataset$Salary), ave(dataset$Salary, FUN = function(x) mean(x, na.rm = 'TRUE')), dataset$Salary)
```

The above code checks for missing values in the Age and Salary columns and update the missing cells with the column-wise average.

- *dataset\$column_header:*

Selects the column in the dataset specified after \$ (Age andSalary).

- *is.na(dataset\$column_header):*

This method returns true for all the cells in the specified columnwith no values.

- *ave(dataset\$column_header, FUN = function(x) mean(x, na.rm = 'TRUE')):*

This method calculates the average of the column passed asargument.

Output:

```

R RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Source
Console Terminal Jobs
R 4.1.0 · E/datasets/
> dataset$Age = ifelse(is.na(dataset$Age), ave(dataset$Age, FUN = function(x) mean(x, na.rm = 'TRUE')), dataset$Age)
> dataset$Salary = ifelse(is.na(dataset$Salary), ave(dataset$Salary, FUN = function(x) mean(x, na.rm = 'TRUE')), dataset$salary)
> head(dataset,10)
   Country     Age    Salary Purchased
1  France 44.00000 72000.00      No
2  Spain 27.00000 48000.00     Yes
3 Germany 30.00000 54000.00      No
4  Spain 38.00000 61000.00      No
5 Germany 40.00000 63777.78     Yes
6  France 35.00000 58000.00     Yes
7  Spain 38.77778 52000.00      No
8  France 48.00000 79000.00     Yes
9 Germany 50.00000 83000.00      No
10 France 37.00000 67000.00    Yes
>

```

Since we don't want decimal places for Age, we will round it up using the following code.

```
dataset$Age = as.numeric(format(round(dataset$Age,
```

The argument 0 in the round function means no decimal places.

After executing the above code block , the dataset would look like what's shown below :

```

> dataset$Age = as.numeric(format(round(dataset$Age, 0)))
> head(dataset,10)
   Country     Age    Salary Purchased
1  France 44.00000 72000.00      No
2  Spain 27.00000 48000.00     Yes
3 Germany 30.00000 54000.00      No
4  Spain 38.00000 61000.00      No
5 Germany 40.00000 63777.78     Yes
6  France 35.00000 58000.00     Yes
7  Spain 39.00000 52000.00      No
8  France 48.00000 79000.00     Yes
9 Germany 50.00000 83000.00      No
10 France 37.00000 67000.00    Yes
>

```

Dealing with Categorical Data

Categorical variables represent types of data which may be divided into groups. Examples of categorical variables are race, sex, age group, educational level etc.

In our dataset, we have categorical features ‘Purchased’. In R we can use the factor method to convert texts into numerical codes.

```
dataset$Purchased = factor(dataset$Purchased, levels = c('No','Yes'), labels = c(0,1))
```

- **factor(dataset\$column_header, levels = c(), labels = c()) :**
the factor method converts the categorical features in the specified column to factors or numerical codes.
- **levels:**
The categories in the column passed as a vector. Example
`c('No','Yes')`
- **labels:**
The numerical codes for the specified categories in the same order.Example
`c(0,1)`

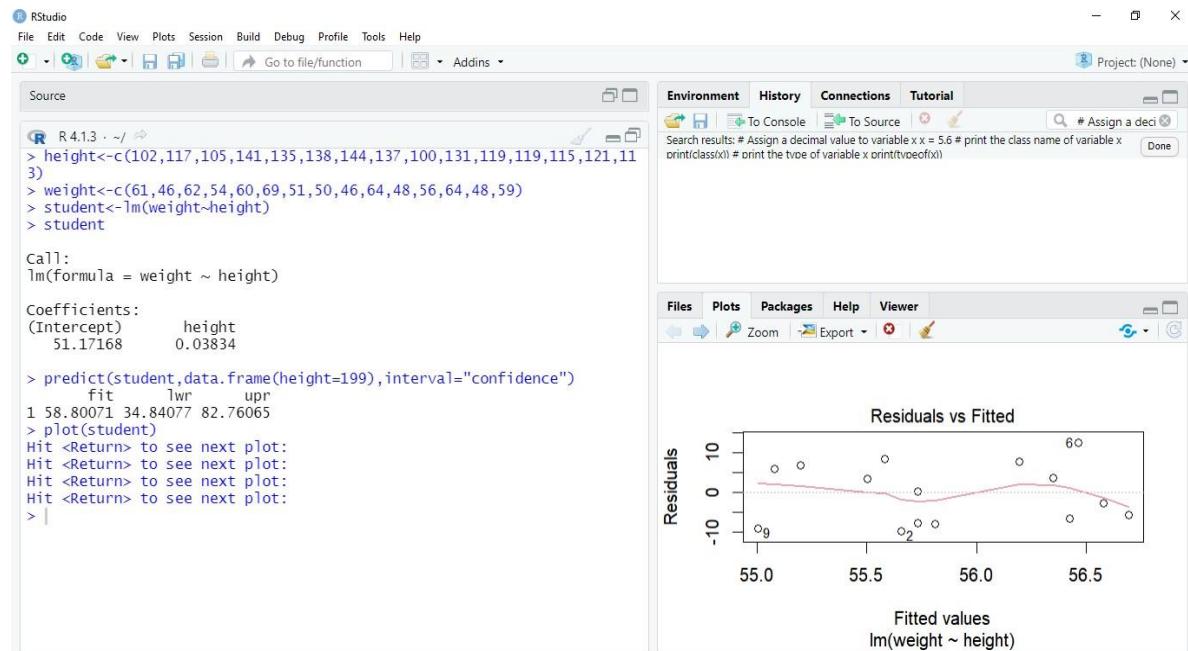
Output:

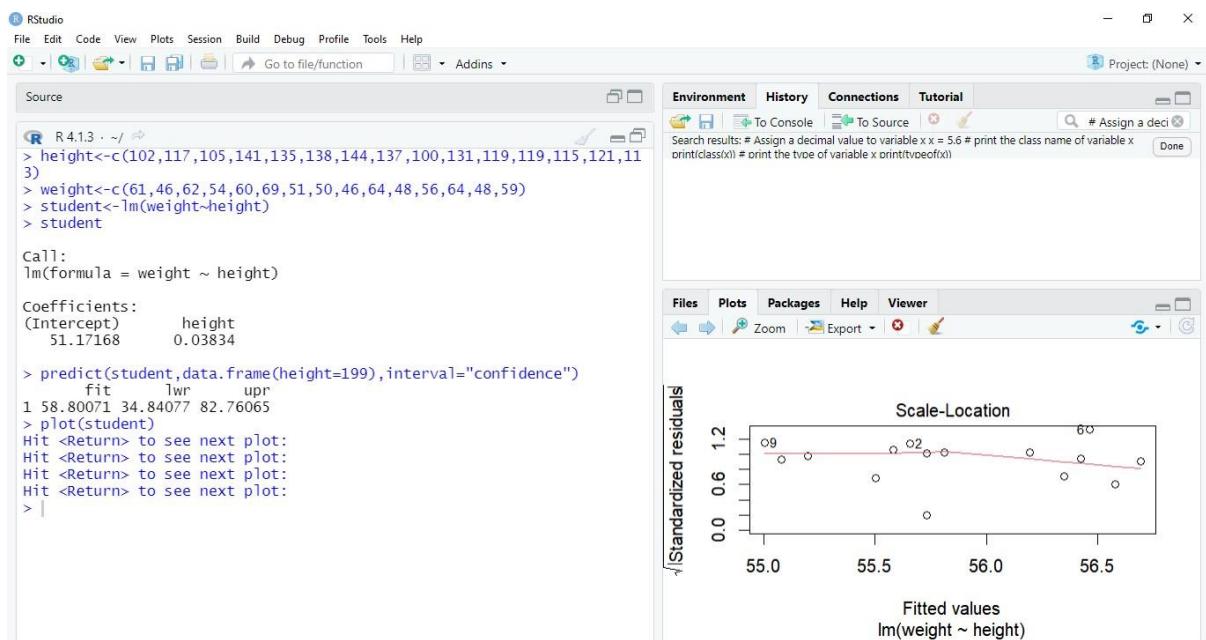
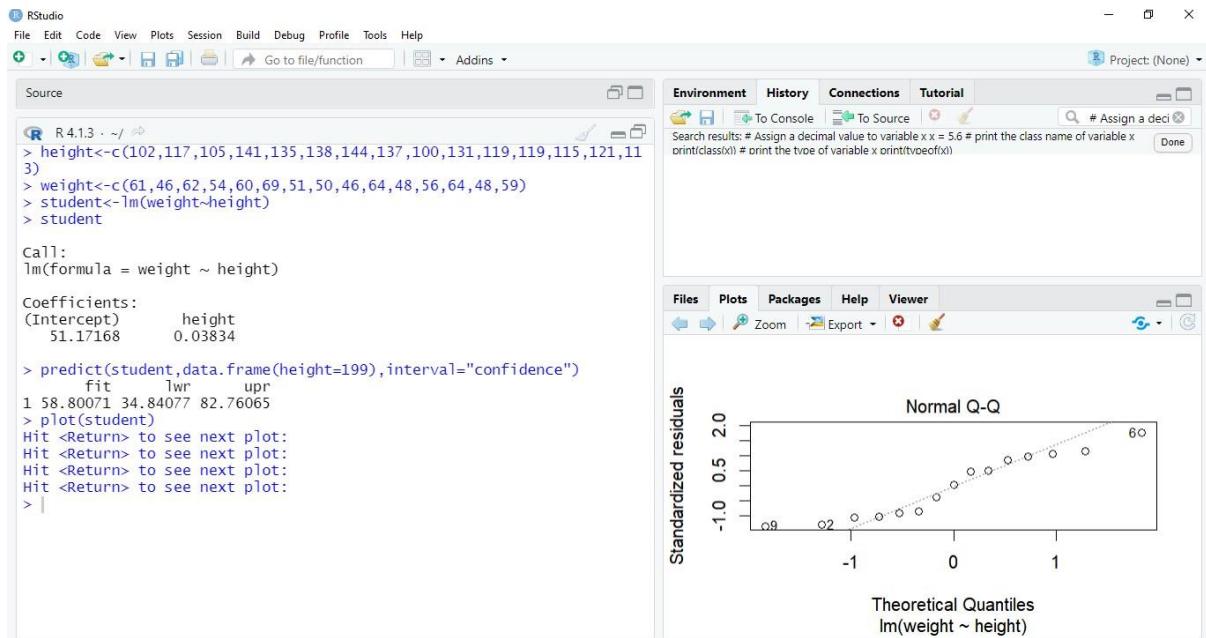
```
> dataset$Purchased = factor(dataset$Purchased, levels = c('No','Yes'), labels = c(0,1))
> head(dataset,10)
  Country Age   Salary Purchased
1  France  44 72000.00      0
2  Spain   27 48000.00      1
3 Germany  30 54000.00      0
4  Spain   38 61000.00      0
5 Germany  40 63777.78      1
6  France  35 58000.00      1
7  Spain   39 52000.00      0
8  France  48 79000.00      1
9 Germany  50 83000.00      0
10 France  37 67000.00      1
~ |
```

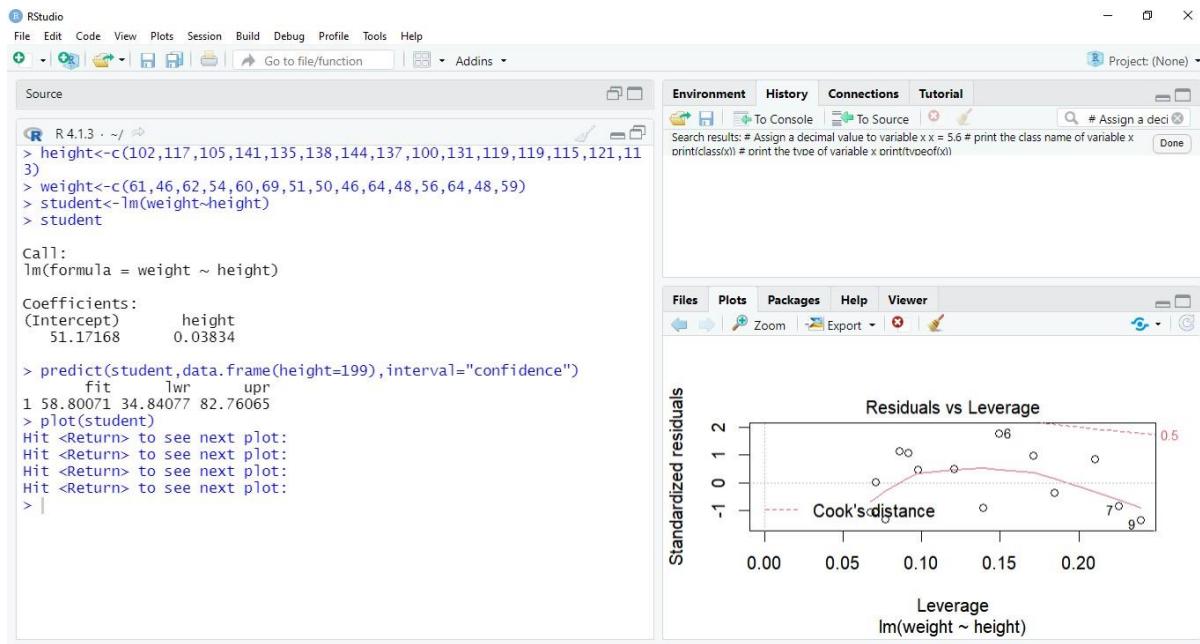
PRACTICAL 7

Aim: To implement Simple linear regression

```
height<-c(102,117,105,141,135,138,144,137,100,131,119,119,115,121,113)
weight<-c(61,46,62,54,60,69,51,50,46,64,48,56,64,48,59)
student<-lm(weight~height)
student
predict(student,data.frame(height=199),interval="confidence")
plot(student)
```

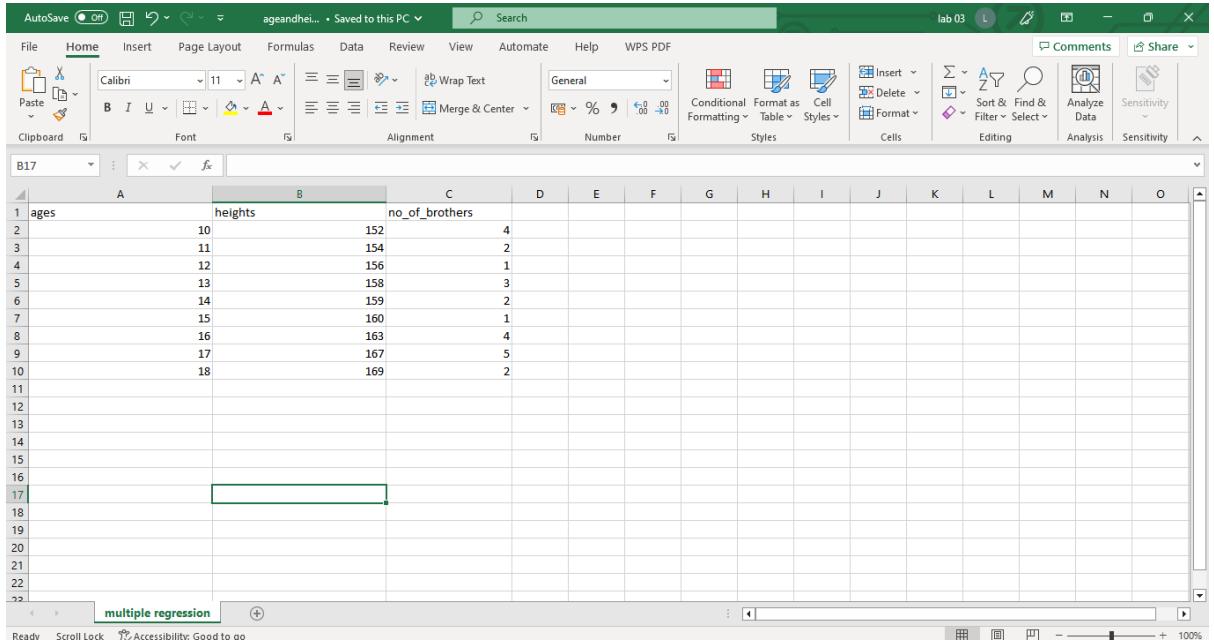






Aim: To implement multiple linear regression

Program: In this program, ages, number of brothers and heights of people are recorded in an excel file “ageandheight.xls” and the relationship between heights (dependent variable) and two independent variables – ages and number of brothers is studied. This relationship between heights and ages, number of brothers can be expressed as a linear equation: Heights = (m1*ages) + (m2* no_of_brothers) + c. M1 and m2 are the co-efficients and c is the intercept.



```

R 4.1.3 · ~/ ◀
> library("xlsx")
> ageheight<-read.xlsx("C:/Users/spdc/Documents/ageandheight.xlsx", sheetName="multiple regression")
> result<-lm(heights~ages+no_of_brothers, data=ageheight)
> summary(result)

Call:
lm(formula = heights ~ ages + no_of_brothers, data = ageheight)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3692 -0.6031  0.1630  0.5263  1.2842 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 130.6847   1.9285  67.764 6.95e-10 ***
ages         2.0282    0.1344  15.088 5.34e-06 ***
no_of_brothers 0.2620    0.2603   1.007   0.353  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.028 on 6 degrees of freedom
Multiple R-squared:  0.9756, Adjusted R-squared:  0.9675 
F-statistic: 119.9 on 2 and 6 DF,  p-value: 1.454e-05

>

```

Aim: To implement Logistic regression

```

> input<-mtcars[,c("am","hp","gear")]
> print(head(input))
> am.data = glm(formula = am~hp+gear,data=input,family = binomial)
> print(summary(am.data))

```

```

R 4.1.3 · ~/ ◀
> input<-mtcars[,c("am","hp","gear")]
> print(head(input))
      am  hp gear
Mazda RX4     1 110     4
Mazda RX4 Wag 1 110     4
Datsun 710    1  93     4
Hornet 4 Drive 0 110     3
Hornet Sportabout 0 175     3
Valiant       0 105     3
> am.data = glm(formula = am~hp+gear,data=input,family = binomial)
> print(summary(am.data))

Call:
glm(formula = am ~ hp + gear, family = binomial, data = input)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.8902  0.0000  0.0000   0.1321   1.1040 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -9.473e+01  1.866e+04 -0.005  0.996    
hp          -2.975e-02  2.779e-02 -1.070  0.284    
gear         2.454e+01  4.665e+03  0.005  0.996    
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.23 on 31 degrees of freedom
Residual deviance: 13.99 on 29 degrees of freedom
AIC: 19.00

```

```

R 4.1.3 . ~/ 
Datsun 710      1   93     4
Hornet 4 Drive  0  110     3
Hornet Sportabout 0  175     3
Valiant        0  105     3
> am.data = glm(formula = am~hp+gear,data=input,family = binomial)
> print(summary(am.data))

Call:
glm(formula = am ~ hp + gear, family = binomial, data = input)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.8902  0.0000  0.0000  0.1321  1.1040

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.473e+01  1.866e+04  -0.005  0.996
hp          -2.975e-02  2.779e-02  -1.070  0.284
gear         2.454e+01  4.665e+03   0.005  0.996

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.23  on 31  degrees of freedom
Residual deviance: 13.99  on 29  degrees of freedom
AIC: 19.99

Number of Fisher Scoring iterations: 20

```

Aim: Performing K Nearest Neighbour on Dataset (KNN)

Installing Packages

```
install.packages("e1071")
```

```
install.packages("caTools")
```

```
install.packages("class")
```

Loading package

```
library(e1071)
```

```
library(caTools)
```

```
library(class)
```

Loading data

```
data(iris)
```

```
head(iris)
```

Splitting data into train

and test data

```
split <- sample.split(iris, SplitRatio = 0.7)
```

```

train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_cl[, 1:4])
test_scale <- scale(test_cl[, 1:4])

# Fitting KNN Model
# to training dataset
classifier_knn <- knn(train = train_scale,
                        test = test_scale,
                        cl = train_cl$Species,
                        k = 1)

classifier_knn

# Confusion Matrix
cm <- table(test_cl$Species, classifier_knn)
cm

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 3
classifier_knn <- knn(train = train_scale,
                        test = test_scale,
                        cl = train_cl$Species,
                        k = 3)

misClassError <- mean(classifier_knn != test_cl$Species)

```

```
print(paste('Accuracy =', 1-misClassError))

# K = 5
classifier_knn <- knn(train = train_scale,
                       test = test_scale,
                       cl = train_cl$Species,
                       k = 5)

misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 7
classifier_knn <- knn(train = train_scale,
                       test = test_scale,
                       cl = train_cl$Species,
                       k = 7)

misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

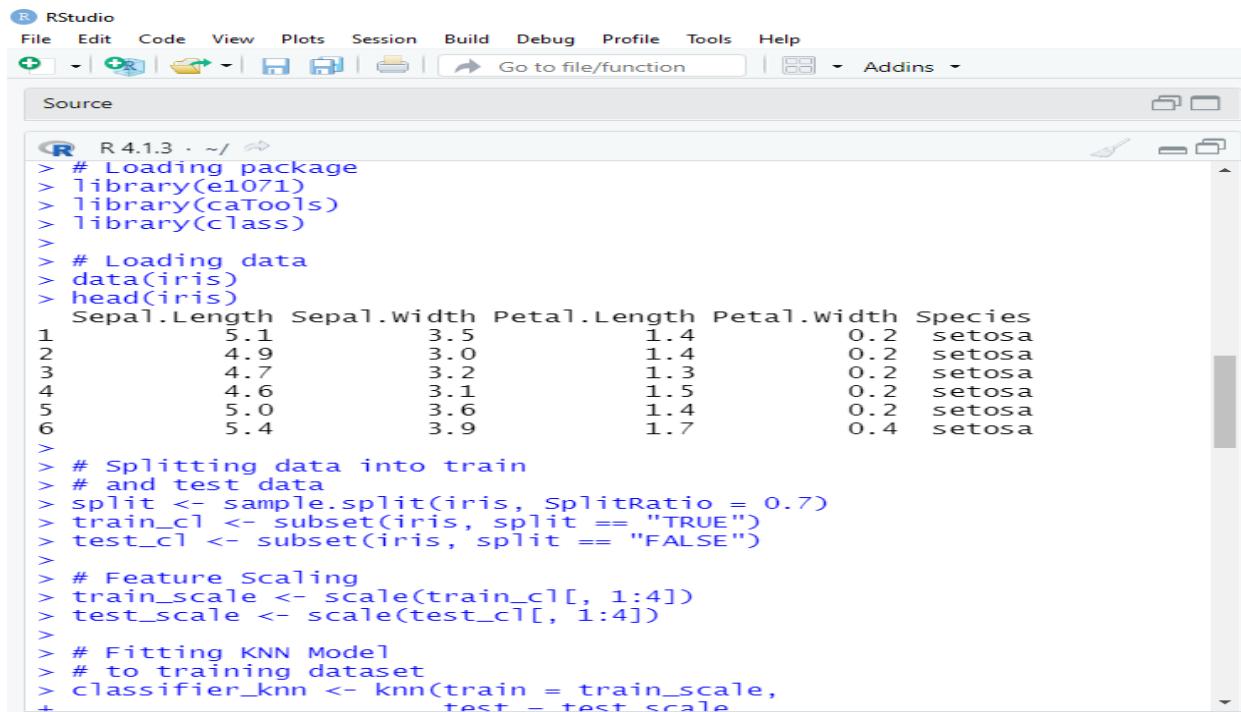
# K = 15
classifier_knn <- knn(train = train_scale,
                       test = test_scale,
                       cl = train_cl$Species,
                       k = 15)

misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 19
classifier_knn <- knn(train = train_scale,
                       test = test_scale,
                       cl = train_cl$Species,
```

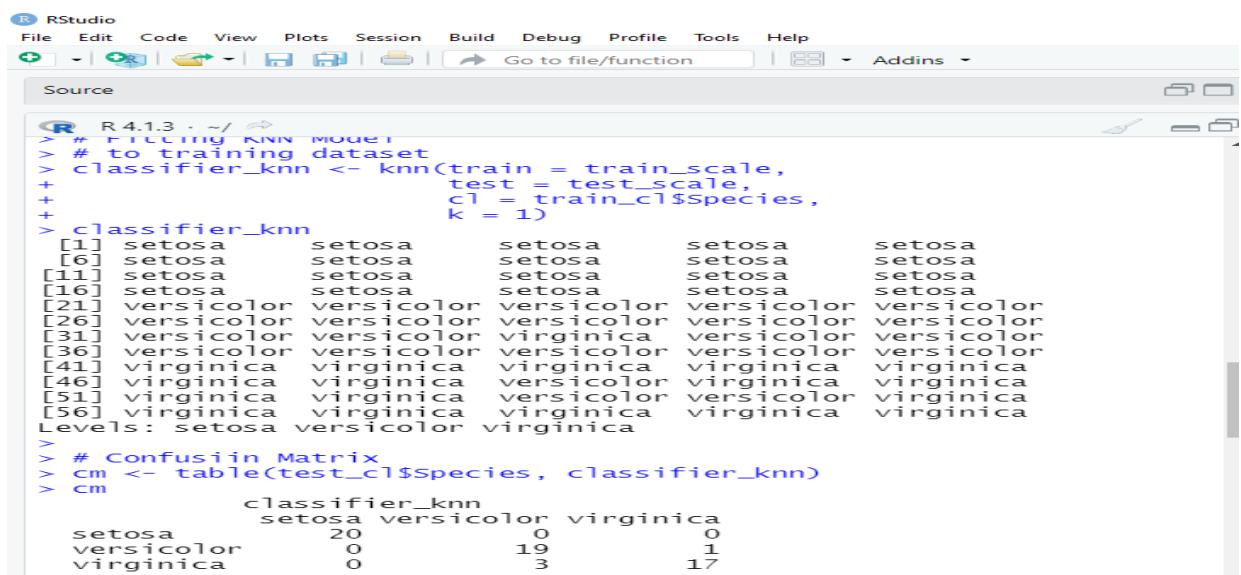
$k = 19)$

```
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))
```



RStudio interface showing the R console window with the following code and output:

```
R 4.1.3 · ~/ ~
> # Loading package
> library(e1071)
> library(caTools)
> library(class)
>
> # Loading data
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
> # Splitting data into train
> # and test data
> split <- sample.split(iris, SplitRatio = 0.7)
> train_cl <- subset(iris, split == "TRUE")
> test_cl <- subset(iris, split == "FALSE")
>
> # Feature Scaling
> train_scale <- scale(train_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
>
> # Fitting KNN Model
> # to training dataset
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
```



RStudio interface showing the R console window with the following code and output:

```
R 4.1.3 · ~/ ~
> # Fitting KNN Model
> # to training dataset
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 1)
> classifier_knn
 [1] setosa    setosa    setosa    setosa    setosa
 [6] setosa    setosa    setosa    setosa    setosa
[11] setosa    setosa    setosa    setosa    setosa
[16] setosa    setosa    setosa    setosa    setosa
[21] versicolor versicolor versicolor versicolor versicolor
[26] versicolor versicolor versicolor versicolor versicolor
[31] versicolor versicolor virginica  versicolor versicolor
[36] versicolor versicolor versicolor versicolor versicolor
[41] virginica  virginica  virginica  virginica  virginica
[46] virginica  virginica  versicolor virginica  virginica
[51] virginica  virginica  versicolor versicolor virginica
[56] virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
>
> # Confusion Matrix
> cm <- table(test_cl$Species, classifier_knn)
> cm
            classifier_knn
setosa      setosa versicolor virginica
setosa       20      0        0
versicolor     0      19        1
virginica     0       3       17
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

```
R 4.1.3 · ~/ ↗
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.933333333333333"
>
> # K = 3
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 3)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.95"
>
> # K = 5
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 5)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.95"
>
> # K = 7
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 7)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.95"
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

```
R 4.1.3 · ~/ ↗
>
> # K = 7
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 7)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.95"
>
> # K = 15
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 15)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.916666666666667"
>
> # K = 19
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$Species,
+                           k = 19)
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.933333333333333"
```

PRACTICAL 8

Aim: To implement K means clustering

```
install.packages("ggplot2")
library(ggplot2)
scatter <- ggplot(data=iris,aes(x=Sepal.Length,y=Sepal.Width))
scatter + geom_point(aes(color=Species,shape=Species))+
theme_bw()+
xlab("Sepal Length")+ylab("Sepal Width")+
ggtitle("Sepal Length-Width")
ggplot(data=iris,aes(Sepal.Length,fill=Species))+ 
theme_bw()+
geom_density(alpha=0.25)+ 
labs(x="Sepal.Length",title="Species vs Sepal Length")
vol <- ggplot(data=iris,aes(x=Sepal.Length))
vol + stat_density(aes(ymax=..density..,ymin=-
..density..,fill=Species,color=Species),geom="ribbon",position="identity")+
facet_grid(.~Species)+coord_flip()+theme_bw()+
labs(x="Sepal Length",title="Species vs Sepal Length")
vol <- ggplot(data=iris,aes(x=Sepal.Width))
vol + stat_density(aes(ymax=..density..,ymin=-
..density..,fill=Species,color=Species),geom="ribbon",position="identity")+
facet_grid(.~Species)+coord_flip()+theme_bw()+
labs(x="Sepal Width",title="Species vs Sepal Width")
irisData <- iris[,1:4]
totalwSS<-c()
for(i in 1:15)
{clusterIRIS<- kmeans(irisData,centers = i)
totalwSS[i] <-clusterIRIS$tot.withinss}
plot(x=1:15,y=totalwSS,type="b",xlab="Number of Clusters",ylab="Within groups sum-of-squares")
install.packages("NbClust")
```

```

library(NbClust)
par(mar=c(2,2,2,2))
nb<-NbClust(irisData,method="kmeans")
hist(nb$Best.nc[1,],breaks=15,main="Histogram for Number of Clusters")
install.packages("vegan")
library(vegan)
modelData<-cascadeKM(irisData,1,10,iter=100)
plot(modelData,sortg=TRUE)
modelData$results[2,]
which.max(modelData$results[2,])
library(cluster)
cl<-kmeans(iris[,-5],2)
dis<-dist(iris[,-5])^2
sil=silhouette(cl$cluster,dis)
plot(sil,main="Clustering Data with silhoutte plot using 2 Clusters",col=c("cyan","blue"))
library(cluster)
cl<-kmeans(iris[,-5],8)
dis<-dist(iris[,-5])^2
sil=silhouette(cl$cluster,dis)
plot(sil,main="Clustering Data with silhoutte plot using 8
Clusters",col=c("cyan","blue","orange","yellow","red","gray","green","maroon"))
install.packages("factoextra")
library(factoextra)
install.packages("clustertend")
library(clustertend)
genx<-function(x){
runif(length(x),min(x),(max(x)))}
random_df<-apply(iris[,-5],2,genx)
random_df<-as.data.frame(random_df)
iris[,-5]<-scale(iris[,-5])
random_df<-scale(random_df)

```

```

res<-get_clust_tendency(iris[,-5],n=nrow(iris)-1,graph=FALSE)
res$hopkins_stat

hopkins(iris[,-5],n=nrow(iris)-1)

res<-get_clust_tendency(random_df,n=nrow(random_df)-1,graph=FALSE)
res$hopkins_stat

```

