F.Y. MCA

SEMESTER - I(CBCS)

ADVANCED DATABASE MANAGEMENT SYSTEM LAB

SUBJECT CODE : MCAL13

Prof. Suhas Pednekar
Vice-Chancellor,
University of Mumbai,

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai,

Prof. Prakash Mahanwar
Director,
IDOL, University of Mumbai,

Programme Co-ordinator : **Shri Mandar Bhanushe**
Asst. Prof. cum Asst. Director in Mathematics,
IDOL, University of Mumbai, Mumbai

Course Co-ordinator **:** **Mrs. Reshama Kurkute**
Asst. Professor, Dept. of MCA
IDOL, University of Mumbai, Mumbai

Course Writers **:** **Prof. Kashafurrehman Ansari**
Assistant Professor
SCCT College, Sanpada, Navi Mumbai

: **Prof. Vandana Maurya**
Assistant Professor
B.K. Birla College(Autonomous), Kalyan

: **Prof. Prema Satav**
Assistant Professor
KLE College, Navi Mumbai

DECEMBER 2021, Print - I

Published by : Director
Institute of Distance and Open Learning ,
University of Mumbai,
Vidyanagari, Mumbai - 400 098.

DTP Composed : Mumbai University Press
Printed by Vidyanagari, Santacruz (E), Mumbai,

CONTENTS

F.Y. MCA

SEMESTER - I(CBCS)

ADVANCED DATABASE MANAGEMENT SYSTEM LAB

Syllabus

Module No	Detailed Contents	Hrs
1	**Distributed Database** : Implementation of Partitions: Range, List. Self-Learning Topics : Hash Partition, Composite partition	2
2	**OLAP with Oracle** : Analytical Queries Self-Learning Topics: Cume_list, Percent_rank	4
3	**ORDBMS** : Implementation of, • Abstract Data Type • Reference Self-Learning Topics: Nested ADT, Inheritance	2
4	**ETL through Pentaho**: ETL Transformation with Pentaho Self-Learning Topics: Any two more transformation operation in Pentaho beyond the syllabus	4
5	**Basics Of R and Data Acquisition** : Introduction to R, Data Types and Objects, Reading and writing data, Reading data from the console Packages, Loading packages, Attach, and detaching data. Loading Data from different Data Source Self-Learning Topics: Operators, Conditional Statements and Loops, Functions, Loading data from Relational Databases, XML	2
6	**Preprocessing in R** : Data preprocessing techniques in R Self-Learning Topics: Sorting, Date Conversion	2

7	**Data Mining - Classification using R-Programming** : Implementation and Analysis of -Regression, Classification Models Self-Learning Topics: Implement One classification algorithm in weka	6
8	**Data Mining - Clustering and Association using R-Programming** : Implementation of Market Basket Analysis and Clustering. Self-Learning Topics: Implementation clustering, association in Weka	4

Module I

DISTRIBUTED DATABASE

RANGE Partitioning in mysql

Problem Statement:

1a. Implementation of Data partitioning through Range.

Concept:

A table that is partitioned by range is partitioned in such a way that each partition contains rows for which the partitioning expression value lies within a given range.

Ranges should be contiguous but not overlapping, and are defined using the VALUES LESS THAN operator.

Below example demonstrate the partitioning in range into mysql database.

Program:

Mysql>create database mca;
Mysql>use mca;

```
mysql> use mca;
Database changed
mysql> show tables;
+---------------+
| Tables_in_mca |
+---------------+
| employees     |
+---------------+
```

mysql> CREATE TABLE tr (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)
);
```

```
mysql> CREATE TABLE tr (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 (2.18 sec)
```

Above code simply create a table named as tr with three columns and 6 partitions on which values are defined inside brackets i.e ().

Now insert data as rows into employees table with following queries.

Mysql> INSERT INTO tr 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');

```
mysql> INSERT INTO tr 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 (0.08 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Now check the inserted data with select statement.
Mysql> select * from tr;

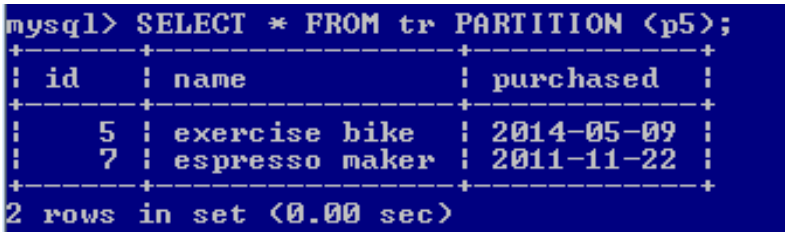
```
mysql> select * from tr;
+------+-----------------+------------+
| id   | name            | purchased  |
+------+-----------------+------------+
|    4 | bookcase        | 1989-01-10 |
|    6 | sofa            | 1987-06-05 |
|    8 | aquarium        | 1992-08-04 |
|    2 | alarm clock     | 1997-11-05 |
|   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 (0.00 sec)
```

We can also check the inserted data into table tr with partition with range created with following code.

Mysql> SELECT * FROM tr PARTITION (p2);

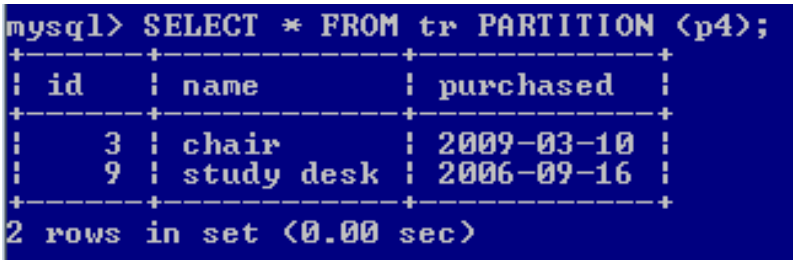
```
mysql> SELECT * FROM tr PARTITION (p2);
+------+-------------+------------+
| id   | name        | purchased  |
+------+-------------+------------+
|    2 | alarm clock | 1997-11-05 |
|   10 | lava lamp   | 1998-12-25 |
+------+-------------+------------+
2 rows in set (0.00 sec)
```

Mysql> SELECT * FROM tr PARTITION (p5);



```
mysql> SELECT * FROM tr PARTITION (p5);
+------+----------------+------------+
| id   | name           | purchased  |
+------+----------------+------------+
|    5 | exercise bike  | 2014-05-09 |
|    7 | espresso maker | 2011-11-22 |
+------+----------------+------------+
2 rows in set (0.00 sec)
```

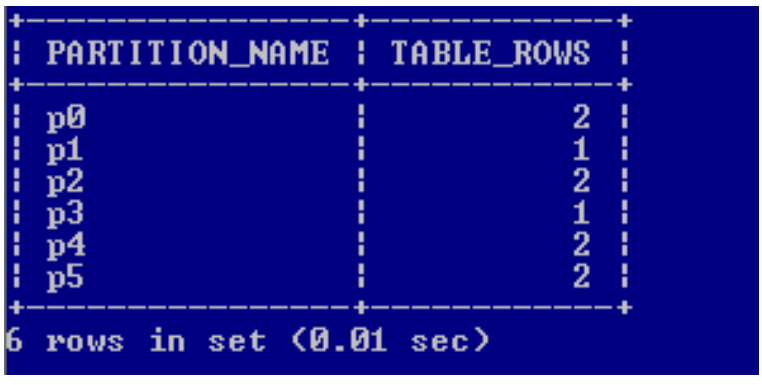
Mysql> SELECT * FROM tr PARTITION (p4);



```
mysql> SELECT * FROM tr PARTITION (p4);
+------+------------+------------+
| id   | name       | purchased  |
+------+------------+------------+
|    3 | chair      | 2009-03-10 |
|    9 | study desk | 2006-09-16 |
+------+------------+------------+
2 rows in set (0.00 sec)
```

We can also check the data with below select statement.

Mysql> SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME='tr';



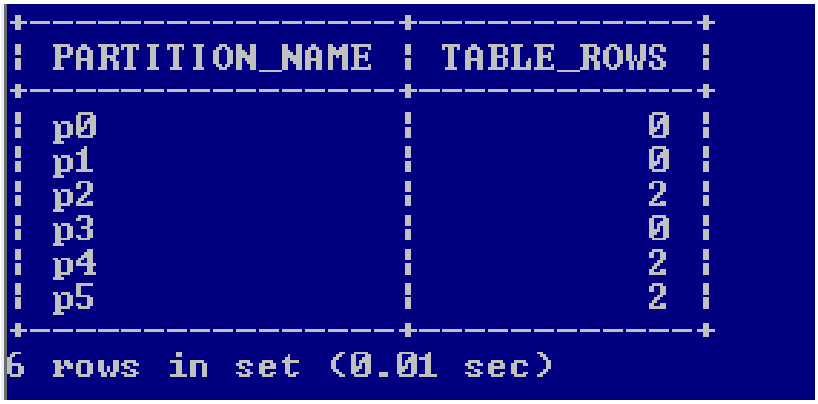
```
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          2 |
| p1             |          1 |
| p2             |          2 |
| p3             |          1 |
| p4             |          2 |
| p5             |          2 |
+----------------+------------+
6 rows in set (0.01 sec)
```

Drop a MySQL partition

If you feel some data are useless in a partitioned table you can drop one or more partition(s). To delete all rows from partition p0 of tr, you can use the following statement:

MySQL> ALTER TABLE tr TRUNCATE PARTITION p0;

Mysql> SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME='tr';



PARTITION_NAME	TABLE_ROWS
p0	0
p1	0
p2	2
p3	0
p4	2
p5	2

6 rows in set (0.01 sec)

1b. Implementation of Data partitioning through List partition.

MySQL LIST Partitioning

It is the same as Range Partitioning. Here, the partition is defined and selected based on columns matching one of a set of discrete value lists rather than a set of a contiguous range of values. It is performed by the **PARTITION BY LIST(exp)** clause. The exp is an expression or column value that returns an integer value. The VALUES IN(value_lists) statement will be used to define each partition.

In the below example, suppose we have 12 stores distributed among four franchises based on their region. The table explains it more clearly:

Region	**Store ID Number**
East	101, 103, 105
West	102, 104, 106
North	107, 109, 111
South	108, 110, 112

We can partition the above table where rows for stores belonging to the same region and will be stored in the same partition. The following

statement arranges the stores in the same region using LIST partitioning, as shown below:

```
Mysql>CREATE TABLE Stores (
    cust_name VARCHAR(40),
    bill_no VARCHAR(20) NOT NULL,
    store_id INT PRIMARY KEY NOT NULL,
    bill_date DATE NOT NULL,
    amount DECIMAL(8,2) NOT NULL
 )
  PARTITION BY LIST(store_id) (
  PARTITION pEast VALUES IN (101, 103, 105),
  PARTITION pWest VALUES IN (102, 104, 106),
  PARTITION pNorth VALUES IN (107, 109, 111),
PARTITION pSouth VALUES IN (108, 110, 112));
```

Module II

OLAP WITH ORACLE

Aim :ANALYTICAL QUERIES

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

Concept:

The last decade has seen a tremendous increase in the use of query, reporting, and on-line analytical processing (OLAP) tools, often in conjunction with data warehouses and data marts. Enterprises exploring new markets and facing greater competition expect these tools to provide the maximum possible decision-making value from their data resources.

Oracle expands its long-standing support for analytical applications in Oracle8i release 8.1.5 with the CUBE and ROLLUP extensions to SQL. Oracle also provides optimized performance and simplified syntax for Top-N queries. These enhancements make important calculations significantly easier and more efficient, enhancing database performance, scalability and simplicity.

ROLLUP and CUBE are simple extensions to the SELECT statement's GROUP BY clause. ROLLUP creates subtotals at any level of aggregation needed, from the most detailed up to a grand total. CUBE is an extension similar to ROLLUP, enabling a single statement to calculate all possible combinations of subtotals.

Syntax-

ROLLUP appears in the GROUP BY clause in a SELECT statement. Its form is:

```
SELECT ... GROUP BY
  ROLLUP(grouping_column_reference_list)
```

Example:

```
select empno,sal, sum(sal) as Totalsal from emp group by rollup(sal);
```

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

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
      7900        950        950
      7900                   950
      7369        800        800
      7369                   800
      7499       1600       1600
      7499                  1600
      7521       1250       1250
      7521                  1250
      7566       2975       2975
      7566                  2975
      7654       1250       1250

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
      7654                  1250
      7698       2850       2850
      7698                  2850
      7782       2450       2450
      7782                  2450
      7788       3000       3000
      7788                  3000
      7839       5000       5000
      7839                  5000
      7844       1500       1500
      7844                  1500

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
      7876       1100       1100
      7876                  1100
      7902       3000       3000
      7902                  3000
      7934       1300       1300
      7934                  1300
                           29025

29 rows selected.
```

CUBE can generate the information needed in cross-tab reports with a single query. To enhance performance, both CUBE and ROLLUP are parallelized: multiple processes can simultaneously execute both types of statements.

CUBE appears in the GROUP BY clause in a SELECT statement. Its form is:

Syntax-

SELECT ... GROUP BY

CUBE (grouping_column_reference_list)

Example:

select empno,sal, sum(sal) as Totalsal from emp group by cube(sal);

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

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
                           29025
                  800        800
                 1100       1100
                 1300       1300
                 1500       1500
                 1600       1600
                 3000       6000
                 5000       5000
                  950        950
                 1250       2500
                 2450       2450

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
                 2850       2850
                 2975       2975
      7900                   950
      7900        950        950
      7369                   800
      7369        800        800
      7499                  1600
      7499       1600       1600
      7521                  1250
      7521       1250       1250
      7566                  2975

     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
      7566       2975       2975
      7654                  1250
      7654       1250       1250
      7698                  2850
      7698       2850       2850
      7782                  2450
      7782       2450       2450
      7788                  3000
      7788       3000       3000
      7839                  5000
      7839       5000       5000
```

```
     EMPNO        SAL   TOTALSAL
---------- ---------- ----------
      7844                  1500
      7844       1500       1500
      7876                  1100
      7876       1100       1100
      7902                  3000
      7902       3000       3000
      7934                  1300
      7934       1300       1300

41 rows selected.
```

With analytic queries, **we can combine data from multiple queries from the same or differing data sources into one result set**.

In some situations, we may need to draw data from several different sets of data, some of which might be stored in different data sources.

An analytic function **computes values over a group of rows and returns a single result for each row**.

Example: we will create a tables named as EMP as follows.

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

```
SQL> conn
Enter user-name: sys as sysdba
Enter password:
Connected.
SQL> CREATE TABLE emp (
  2    empno    NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,
  3    ename    VARCHAR2(10),
  4    job      VARCHAR2(9),
  5    mgr      NUMBER(4),
  6    hiredate DATE,
  7    sal      NUMBER(7,2),
  8    comm     NUMBER(7,2),
  9    deptno   NUMBER(2)
 10  );

Table created.
```

RANK

```
RANK() OVER ([ query_partition_clause ] order_by_clause)
```

Let's assume we want to assign a sequential order, or rank, to people within a department based on salary, we might use the RANK function like this.

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

```
SQL> SELECT empno,
  2         deptno,
  3         sal,
  4         RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
  5  FROM   emp;

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7934         10       1300          1
      7782         10       2450          2
      7839         10       5000          3
      7369         20        800          1
      7876         20       1100          2
      7566         20       2975          3
      7788         20       3000          4
      7902         20       3000          4
      7900         30        950          1
      7654         30       1250          2
      7521         30       1250          2

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7844         30       1500          4
      7499         30       1600          5
      7698         30       2850          6

14 rows selected.
```

What we see here is where two people have the same salary they are assigned the same rank. When multiple rows share the same rank the next rank in the sequence is not consecutive. This is like olympic medaling in that if two people share the gold, there is no silver medal etc.

The fact we can rank the rows in the department means we are able to do a Top-N query on a per-department basis. The example below assigns the rank in the inline view, then uses that rank to restrict the rows to the bottom 2 (worst paid) employees in each department.

```
SELECT *
FROM   (SELECT empno,
               deptno,
               sal,
               RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
```

```
      FROM   emp)
WHERE  myrank <= 2;
```

```
SQL> SELECT *
  2  FROM   (SELECT empno,
  3                 deptno,
  4                 sal,
  5                 RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
  6          FROM   emp)
  7  WHERE  myrank <= 2;

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7934         10       1300          1
      7782         10       2450          2
      7369         20        800          1
      7876         20       1100          2
      7900         30        950          1
      7521         30       1250          2
      7654         30       1250          2

7 rows selected.
```

DENSE_RANK

The basic description for the DENSE_RANK analytic function is shown below. The analytic clause is described below.

```
DENSE_RANK() OVER([ query_partition_clause ] order_by_clause)
```

The DENSE_RANK function acts like the RANK function except that it assigns consecutive ranks, so this is not like olympic medaling.

```
SELECT empno,
       deptno,
       sal,
       DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal)
AS myrank
FROM   emp;
```

```
SQL> SELECT empno,
  2         deptno,
  3         sal,
  4         DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
  5  FROM   emp;

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7934         10       1300          1
      7782         10       2450          2
      7839         10       5000          3
      7369         20        800          1
      7876         20       1100          2
      7566         20       2975          3
      7788         20       3000          4
      7902         20       3000          4
      7900         30        950          1
      7654         30       1250          2
      7521         30       1250          2

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7844         30       1500          3
      7499         30       1600          4
      7698         30       2850          5

14 rows selected.
```

As with the RANK analytic function, we can do a Top-N query on a per-department basis. The example below assigns the dense rank in the inline view, then uses that rank to restrict the rows to the top 2 (best paid) employees in each department.

```
SELECT *
FROM   (SELECT empno,
               deptno,
               sal,
               DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC)
AS myrank
        FROM   emp)
WHERE  myrank <= 2;
```

```
SQL> SELECT *
  2  FROM   (SELECT empno,
  3                 deptno,
  4                 sal,
  5                 DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS myrank
  6          FROM   emp)
  7  WHERE  myrank <= 2;

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7839         10       5000          1
      7782         10       2450          2
      7788         20       3000          1
      7902         20       3000          1
      7566         20       2975          2
      7698         30       2850          1
      7499         30       1600          2

7 rows selected.
```

FIRST and LAST

Most of the time I find myself using FIRST_VALUE and LAST_VALUE Analytic Functions in preference to FIRST and LAST. Pick which feels best for your use case.

The FIRST and LAST functions can be used to return the first or last value from an ordered sequence. Say we want to display the salary of each employee, along with the lowest and highest within their department we may use something like.

```
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;
```

```
SQL> SELECT empno,
  2         deptno,
  3         sal,
  4         MIN(sal) KEEP (DENSE_RANK FIRST ORDER BY sal) OVER (PARTITION BY deptno) AS lowest,
  5         MAX(sal) KEEP (DENSE_RANK LAST ORDER BY sal) OVER (PARTITION BY deptno) AS highest
  6  FROM   emp
  7  ORDER BY deptno, sal;

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7934         10       1300       1300       5000
      7782         10       2450       1300       5000
      7839         10       5000       1300       5000
      7369         20        800        800       3000
      7876         20       1100        800       3000
      7566         20       2975        800       3000
      7788         20       3000        800       3000
      7902         20       3000        800       3000
      7900         30        950        950       2850
      7654         30       1250        950       2850
      7521         30       1250        950       2850

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7844         30       1500        950       2850
      7499         30       1600        950       2850
      7698         30       2850        950       2850

14 rows selected.
```

The MIN and MAX functions are almost irrelevant here as it's FIRST, LAST and KEEP that are picking the row whose value will be used. We can demonstrate this by using MIN for both the high and low value.

```
SELECT empno,
       deptno,
       sal,
       MIN(sal) KEEP (DENSE_RANK FIRST ORDER BY sal) OVER (PARTITION BY deptno) AS lowest,
       MIN(sal) KEEP (DENSE_RANK LAST ORDER BY sal) OVER (PARTITION BY deptno) AS highest
FROM   emp
ORDER BY deptno, sal;
```

```
SQL> SELECT empno,
  2         deptno,
  3         sal,
  4         MIN(sal) KEEP (DENSE_RANK FIRST ORDER BY sal) OVER (PARTITION BY deptno) AS lowest,
  5         MIN(sal) KEEP (DENSE_RANK LAST ORDER BY sal) OVER (PARTITION BY deptno) AS highest
  6  FROM   emp
  7  ORDER BY deptno, sal;

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7934         10       1300       1300       5000
      7782         10       2450       1300       5000
      7839         10       5000       1300       5000
      7369         20        800        800       3000
      7876         20       1100        800       3000
      7566         20       2975        800       3000
      7788         20       3000        800       3000
      7902         20       3000        800       3000
      7900         30        950        950       2850
      7654         30       1250        950       2850
      7521         30       1250        950       2850

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7844         30       1500        950       2850
      7499         30       1600        950       2850
      7698         30       2850        950       2850

14 rows selected.
```

We get the same result.

We could also achieve the same result using FIRST_VALUE and LAST_VALUE, or MIN and MAX as basic analytic functions. In practice I don't use FIRST and LAST very often.

LAG

The **LAG** function is used to access data from a previous row. The following query returns the salary from the previous row to calculate the difference between the salary of the current row and that of the previous row. Notice that the **ORDER BY** of the **LAG** function is used to order the data by salary.

```
SELECT empno,
       ename,
       job,
       sal,
       LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal_prev,
       sal - LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal_diff
FROM   emp;
```

```
SQL> SELECT empno,
  2         ename,
  3         job,
  4         sal,
  5         LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal_prev,
  6         sal - LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal_diff
  7  FROM   emp;

     EMPNO ENAME      JOB              SAL   SAL_PREV   SAL_DIFF
---------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            800          0        800
      7900 JAMES      CLERK            950        800        150
      7876 ADAMS      CLERK           1100        950        150
      7521 WARD       SALESMAN        1250       1100        150
      7654 MARTIN     SALESMAN        1250       1250          0
      7934 MILLER     CLERK           1300       1250         50
      7844 TURNER     SALESMAN        1500       1300        200
      7499 ALLEN      SALESMAN        1600       1500        100
      7782 CLARK      MANAGER         2450       1600        850
      7698 BLAKE      MANAGER         2850       2450        400
      7566 JONES      MANAGER         2975       2850        125

     EMPNO ENAME      JOB              SAL   SAL_PREV   SAL_DIFF
---------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         3000       2975         25
      7902 FORD       ANALYST         3000       3000          0
      7839 KING       PRESIDENT       5000       3000       2000

14 rows selected.
```

If the LAG would span a partition boundary, the default value is returned. In the following example we partition by department, so the SAL_PREV column has a default value of "0" for the first row in each department.

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

```
SQL> SELECT deptno,
  2         empno,
  3         ename,
  4         job,
  5         sal,
  6         LAG(sal, 1, 0) OVER (PARTITION BY deptno ORDER BY sal) AS sal_prev
  7  FROM   emp;

    DEPTNO      EMPNO ENAME      JOB              SAL   SAL_PREV
---------- ---------- ---------- --------- ---------- ----------
        10       7934 MILLER     CLERK            1300          0
        10       7782 CLARK      MANAGER          2450       1300
        10       7839 KING       PRESIDENT        5000       2450
        20       7369 SMITH      CLERK             800          0
        20       7876 ADAMS      CLERK            1100        800
        20       7566 JONES      MANAGER          2975       1100
        20       7788 SCOTT      ANALYST          3000       2975
        20       7902 FORD       ANALYST          3000       3000
        30       7900 JAMES      CLERK             950          0
        30       7654 MARTIN     SALESMAN         1250        950
        30       7521 WARD       SALESMAN         1250       1250

    DEPTNO      EMPNO ENAME      JOB              SAL   SAL_PREV
---------- ---------- ---------- --------- ---------- ----------
        30       7844 TURNER     SALESMAN         1500       1250
        30       7499 ALLEN      SALESMAN         1600       1500
        30       7698 BLAKE      MANAGER          2850       1600

14 rows selected.
```

LEAD

The LEAD function is used to return data from rows further down the result set. The following query returns the salary from the next row to calculate the difference between the salary of the current row and the following row.

```
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;
```

```
SQL> SELECT empno,
  2         ename,
  3         job,
  4         sal,
  5         LEAD(sal, 1, 0) OVER (ORDER BY sal) AS sal_next,
  6         LEAD(sal, 1, 0) OVER (ORDER BY sal) - sal AS sal_diff
  7  FROM   emp;

     EMPNO ENAME      JOB              SAL   SAL_NEXT   SAL_DIFF
---------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            800        950        150
      7900 JAMES      CLERK            950       1100        150
      7876 ADAMS      CLERK           1100       1250        150
      7521 WARD       SALESMAN        1250       1250          0
      7654 MARTIN     SALESMAN        1250       1300         50
      7934 MILLER     CLERK           1300       1500        200
      7844 TURNER     SALESMAN        1500       1600        100
      7499 ALLEN      SALESMAN        1600       2450        850
      7782 CLARK      MANAGER         2450       2850        400
      7698 BLAKE      MANAGER         2850       2975        125
      7566 JONES      MANAGER         2975       3000         25

     EMPNO ENAME      JOB              SAL   SAL_NEXT   SAL_DIFF
---------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         3000       3000          0
      7902 FORD       ANALYST         3000       5000       2000
      7839 KING       PRESIDENT       5000          0      -5000

14 rows selected.
```

If the LEAD would span a partition boundary, the default value is returned. In the following example we partition by department, so the SAL_NEXT column has a default value of "0" for the last row in each department.

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

```
SQL> SELECT deptno,
  2         empno,
  3         ename,
  4         job,
  5         sal,
  6         LEAD(sal, 1, 0) OVER (PARTITION BY deptno ORDER BY sal) AS sal_next
  7  FROM   emp;

    DEPTNO      EMPNO ENAME      JOB              SAL   SAL_NEXT
---------- ---------- ---------- --------- ---------- ----------
        10       7934 MILLER     CLERK            1300       2450
        10       7782 CLARK      MANAGER          2450       5000
        10       7839 KING       PRESIDENT        5000          0
        20       7369 SMITH      CLERK             800       1100
        20       7876 ADAMS      CLERK            1100       2975
        20       7566 JONES      MANAGER          2975       3000
        20       7788 SCOTT      ANALYST          3000       3000
        20       7902 FORD       ANALYST          3000          0
        30       7900 JAMES      CLERK             950       1250
        30       7654 MARTIN     SALESMAN         1250       1250
        30       7521 WARD       SALESMAN         1250       1500

    DEPTNO      EMPNO ENAME      JOB              SAL   SAL_NEXT
---------- ---------- ---------- --------- ---------- ----------
        30       7844 TURNER     SALESMAN         1500       1600
        30       7499 ALLEN      SALESMAN         1600       2850
        30       7698 BLAKE      MANAGER          2850          0

14 rows selected.
```

```
mysql> CREATE TABLE Stores (
    ->       cust_name VARCHAR(40),
    ->       bill_no VARCHAR(20) NOT NULL,
    ->       store_id INT PRIMARY KEY NOT NULL,
    ->       bill_date DATE NOT NULL,
    ->       amount DECIMAL(8,2) NOT NULL
    ->  )
    ->   PARTITION BY LIST(store_id) (
    ->   PARTITION pEast VALUES IN (101, 103, 105),
    ->   PARTITION pWest VALUES IN (102, 104, 106),
    ->   PARTITION pNorth VALUES IN (107, 109, 111),
    ->   PARTITION pSouth VALUES IN (108, 110, 112));
Query OK, 0 rows affected (1.32 sec)
```

❖❖❖❖

Module III

ORDBMS

Aim: Implementation of Abstract Data Type & Reference

Creating Tables Under the Relational Model

The relational approach normalizes everything into tables. The table names are Customer_reltab, PurchaseOrder_reltab, and Stock_reltab.

Each part of an address becomes a column in the Customer_reltab table. Structuring telephone numbers as columns sets an arbitrary limit on the number of telephone numbers a customer can have.

The relational approach separates line items from their purchase orders and puts each into its own table, named Purchase Order_reltab and LineItems_reltab.

The relational approach results in the tables describe in the following sections.

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));
```

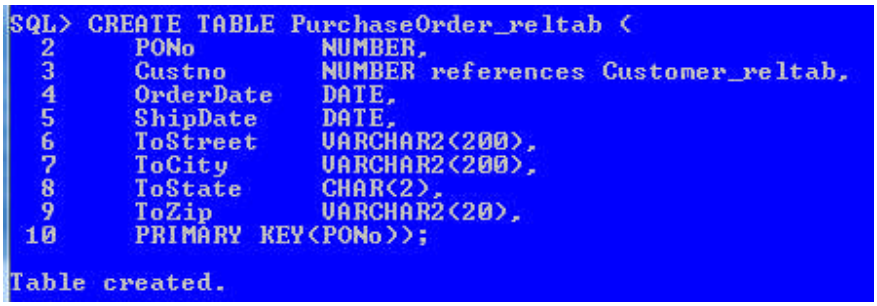
```
SQL> desc customer_reltab;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTNO                                    NOT NULL NUMBER
 CUSTNAME                                  NOT NULL VARCHAR2(200)
 STREET                                    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)
```

This table, Customer_reltab, stores all the information about customers, which means that it fully contains information that is intrinsic to the customer (defined with the NOTNULL constraint) and information that is not as essential. According to this definition of the table, the application requires that every customer have a shipping address.

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));
```



Purchase Order_reltab manages the relationship between the customer and the purchase order by means of the foreign key (FK) column Cust No, which references the CustNo key of the Customer_reltab. The Purchase Order_reltab table contains no information about related line items. The line items table (next section) uses the purchase order number to relate a line item to its parent purchase order.

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.
```

The table name is in the plural form LineItems_reltab to emphasize to someone reading the code that the table holds a collection of line items.
PONo, which references the PONo column in Purchase Order_reltab
StockNo, which references the StockNo column in Stock_reltab

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;
```

Module IV

EXPERIMENT NO.1

Aim: To study ETL process.

Objective: To understand ETL process in the Data Warehouse.

Theory:

❖ **What is ETL?**

ETL is a process that extracts the data from different source systems, then transforms the data (like applying calculations, concatenations, etc.) and finally loads the data into the data Warehouse system.

Before loading into the warehouse, the data is transformed from a raw state into the format required by the enterprise data warehouse.

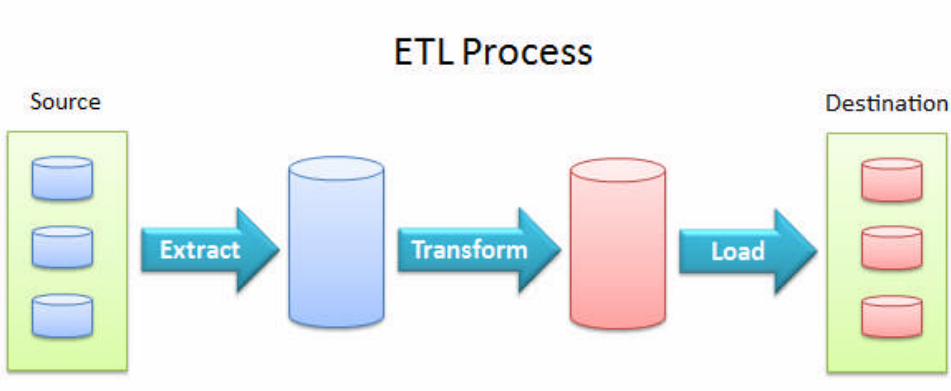


Figure 1: ETL process

Extract:

It is the process of fetching (reading) the information from the database. At this stage, data is collected from different types of sources.

Transform:

It is the process of converting the extracted data from its previous form into the required form. Data can be placed into another database. Transformation can occur by using rules or lookup tables or by combining the data with other data.

Load:

It is the process of writing the data into the target database.

In the ETL process, data is extracted from the multiple source system and converted into a format that can be examined and stored into a **data warehouse** or any other system. It is often used to build a **data warehouse**.

Example:

Let’s take an example of retail store which has different departments like sales, marketing, logistics, etc. Each department is handling the customer's information independently, and the way each department stores the data is quite different.

Sales department stores it by the **customer's name and** marketing department store it by **customer id.**

Now, if we want to check the history of the customer and want to know what different types of products, he/she bought owing to various campaigns; it would be very tedious.

The solution for this is to use a data warehouse to store information from different sources in a uniform structure using ETL. ETL tools extract the data from all these data sources and transform the data (like applying calculations, joining fields, removing incorrect data fields, etc.) and loads into a data warehouse.

ETL can transform unique data sets into a unified structure. After that, we will use BI tools to find out the meaningful reports, dashboards, visualization from this data.

✓ **Need of ETL**

There are many reasons the need for ETL is arising:

- ETL helps the companies to analyze their business data for making critical business decisions.
- ETL provides a method of moving data from various sources into a data warehouse. As the data sources change, the data warehouse will automatically update.
- Well-designed and documented ETL system is essential for the success of the data warehouse project.
- For business purpose, ETL offers deep historical context.
- It helps to improve productivity because it is codified and can be reused without a need for technical skills.

The popular ETL tools available in the market are:

- IBM- Websphere DataStage

- Informatica- Power Center
- SAP- Business objects data service BODS
- SAS - Data Integration Studio
- Oracle- Warehouse Builder
- Open source Clover ETL.

References

Figure 1: ETL process- https://blog.appliedinformaticsinc.com/etl-extract-transform-and-load-process-concept/, accessed on 14th July 2021.

Questions:

1. ETL stands for _____?
a) Extract, Transfer and Load
b) Extract, Transform and Load
c) Extract, Time and Load
d) Extract, Transform and Loss

2. Extract is the process of _____.
a)adding new data to a database.
b)reading and collecting data from a database, the data is often collected from multiple sources.
c)analyzing collected information
d)none of the above

3. Load is the process of _____.
a)publishing the data to decision makers.
b)reviewing data in the database.
c)writing data into a RPA tool.
d)writing the data into the target database.

Experiment no.2

Aim: Installation of Pentaho Software.

Objective:
1. Download the Pentaho Data Integration software.
2. Install JRE and JDK.
3. Set up JRE and JDK environment variables for Pentaho Data Integration.

Theory:

Pentaho Data Integration - Kettle ETL tool

Kettle (K.E.T.T.L.E - Kettle ETTL Environment) has been recently acquired by the Pentaho group and renamed to **Pentaho Data Integration**. Kettle is a leading open source ETL application on the market. It is classified as an ETL tool, however the concept of classic ETL process (extract, transform, load) has been slightly modified in Kettle as it is composed of four elements, **ETTL**, which stands for:

- ✓ Data **extraction** from source databases
- ✓ **Transport** of the data
- ✓ Data **transformation**
- ✓ **Loading** of data into a data warehouse

Kettle is a set of tools and applications which allows data manipulations across multiple sources.

The main components of Pentaho Data Integration are:

- **Spoon** –

  It is a graphical tool which make the design of an ETTL process transformations easy to create. It performs the typical data flow functions like reading, validating, refining, transforming, writing data to a variety of different data sources and destinations. Transformations designed in Spoon can be run with Kettle Pan and Kitchen.

- **Pan** –

  It is an application dedicated to run data transformations designed in Spoon.

- **Chef –**

  It is a tool to create jobs which automate the database update process in a complex way.

- **Kitchen –**

  It is an application which helps execute the jobs in a batch mode, usually using a schedule which makes it easy to start and control the ETL processing.

- **Carte –**

  It is a web server which allows remote monitoring of the running Pentaho Data Integration ETL processes through a web browser.

❖ **Installation steps for Pentaho Data Integration Software**

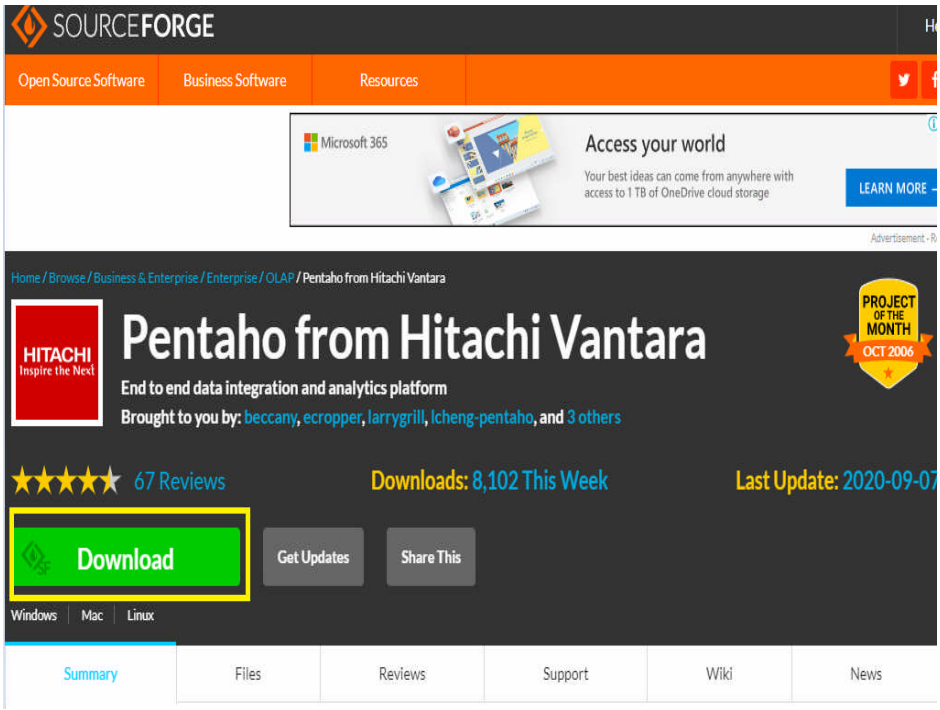
Step 1: Download Pentaho Data Integration Software.

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/.

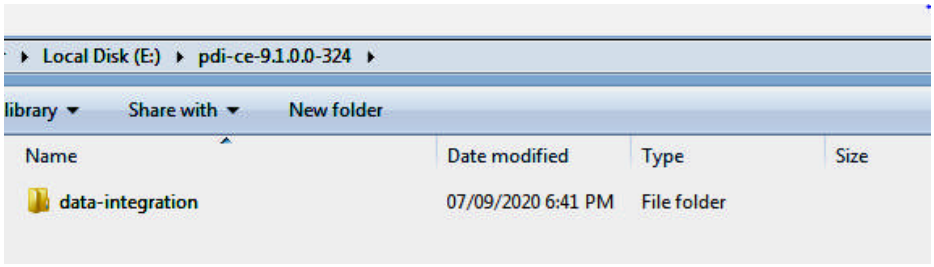
The page will look something like this:



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.

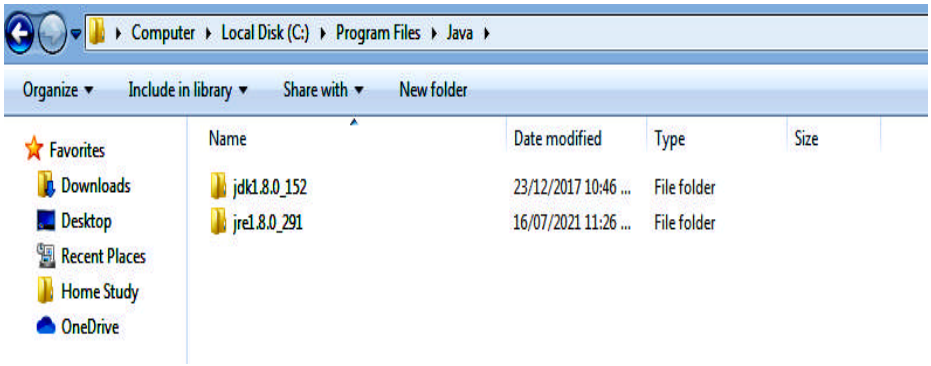
To run Pentaho Data Integration, Java Runtime Environment and Java Development Kit are required.

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:

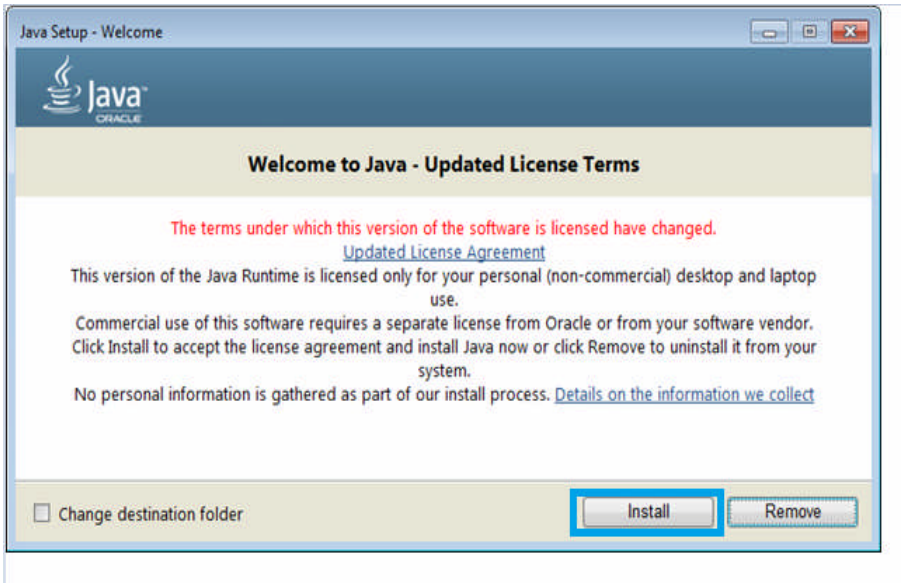


Then you have the required files.

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:



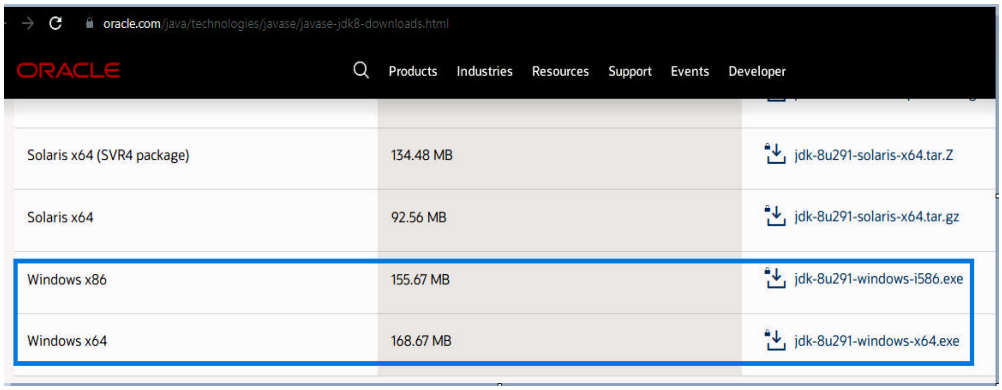
Follow the instructions until finished.

Next, download the JDK from this link https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html.

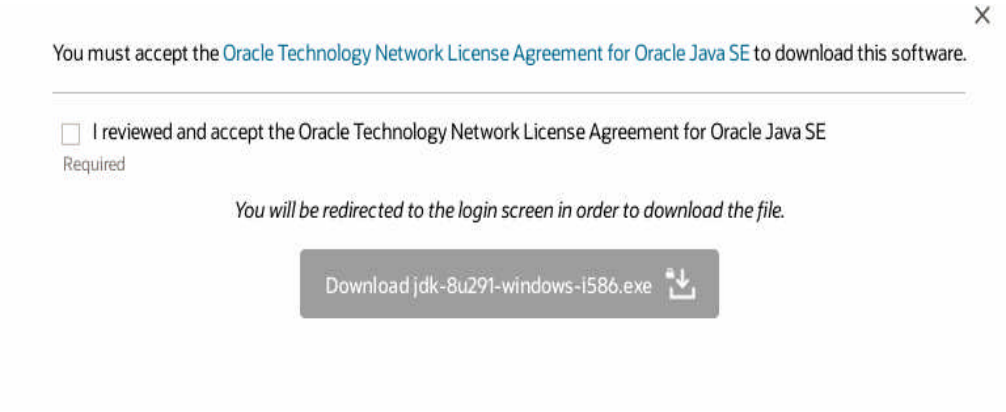
Please note that there have been substantial changes to the Oracle JDK licensing agreement. Details are available at Oracle Technology Network License Agreement for Oracle Java SE.

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

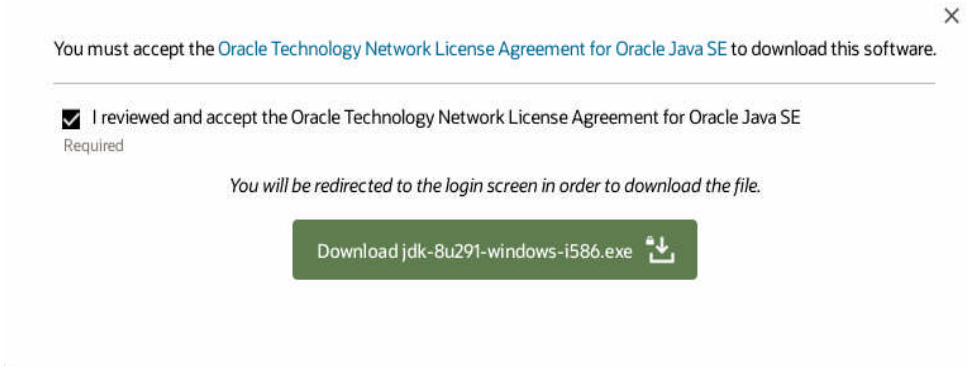
If you’re unsure about which version (x64 or x86) your Windows is, select x86.



It will open following window.

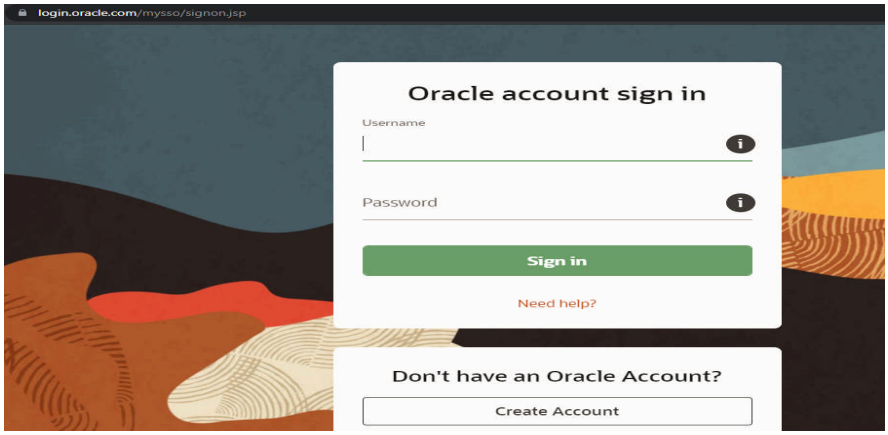


Press “Download”.

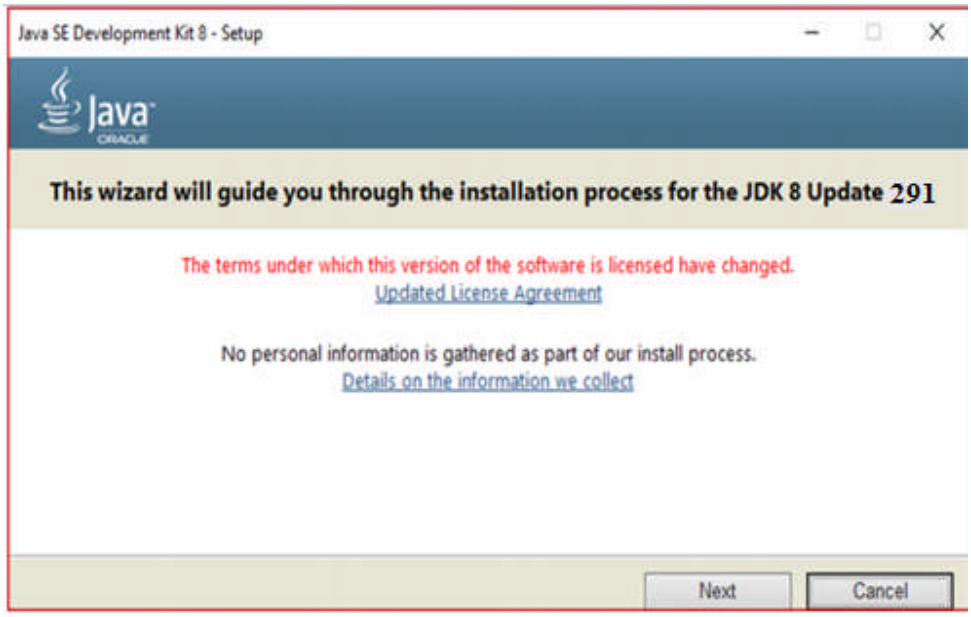


If you’re not logged in to Oracle, then you will be prompted to log in.

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:



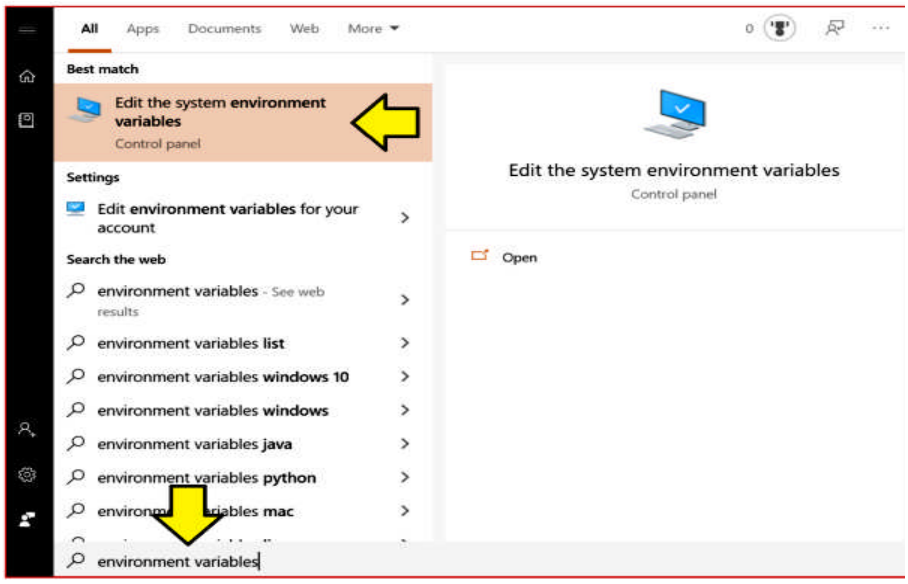
Graphics:

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.

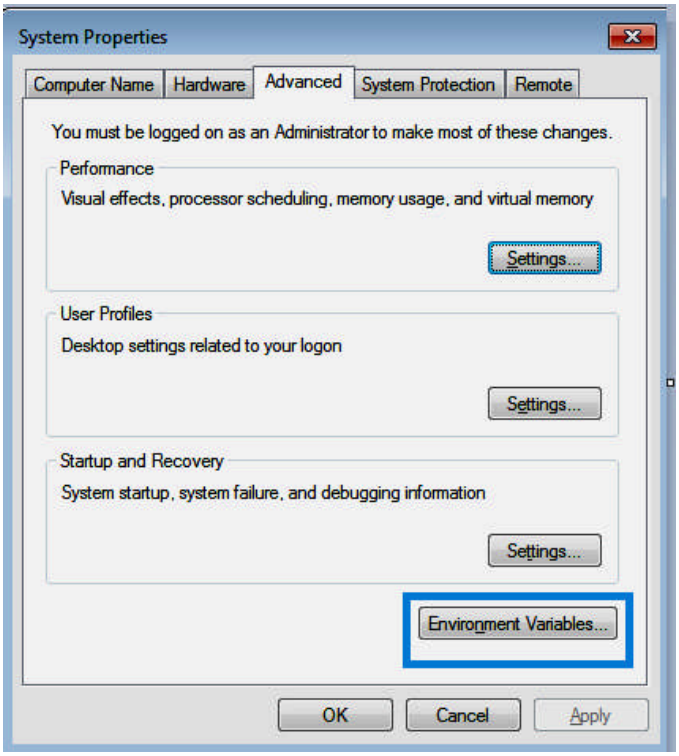
To open the environment variables menu type in “environment variables” in the Windows search bar like this:



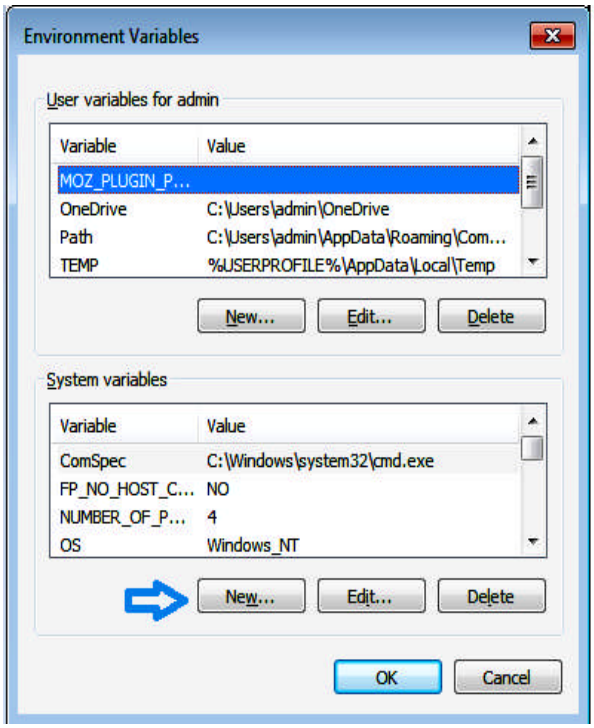
Click the “Edit the system environment variables” option.

That will open the “System Properties” window.

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

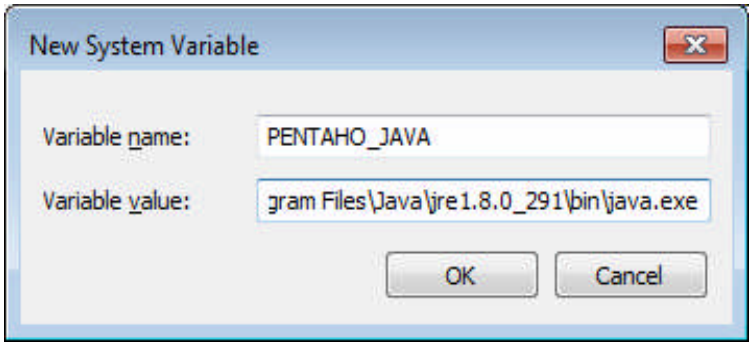


That will open a window that looks like this:

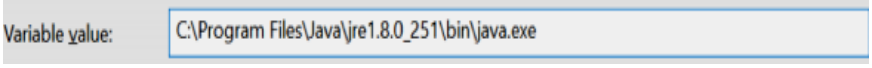


We need to add three new System variables.

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

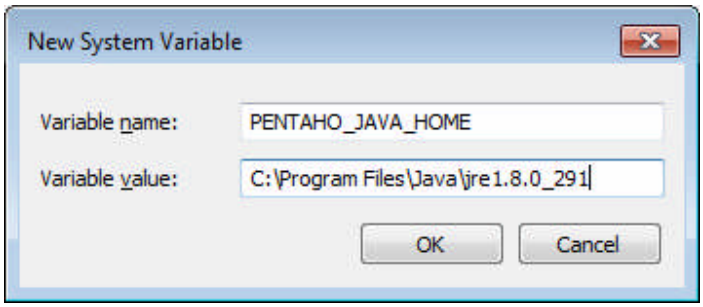


Note:

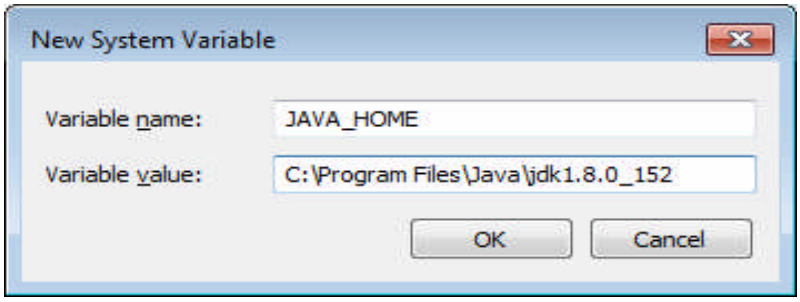


Make sure your variable value file path is the same one on your computer.

Press "OK" and then enter two more.



Press ‘’OK’’.

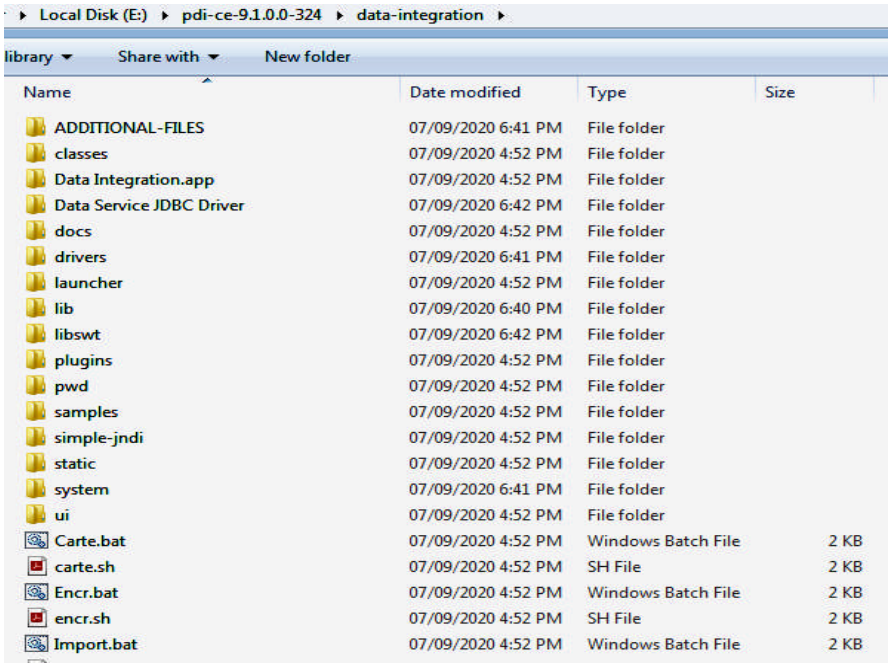


Press ‘’OK’’ and close all the previous windows by pressing “OK.”

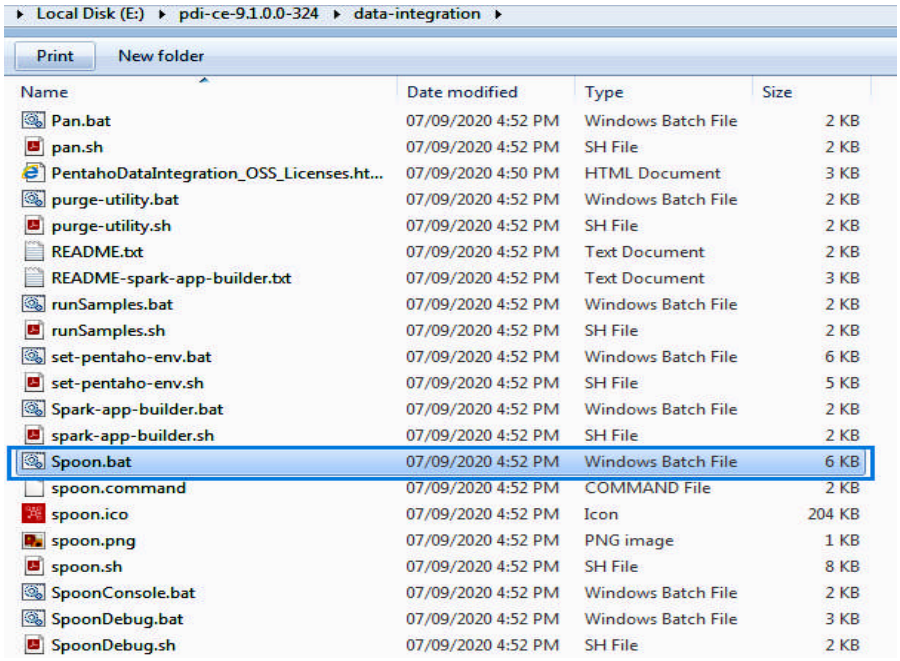
Step 4: Open the Pentaho Data Integration App

Now that Java is installed successfully and the environment variables are also set, we can start running the Pentaho Data Integration app.

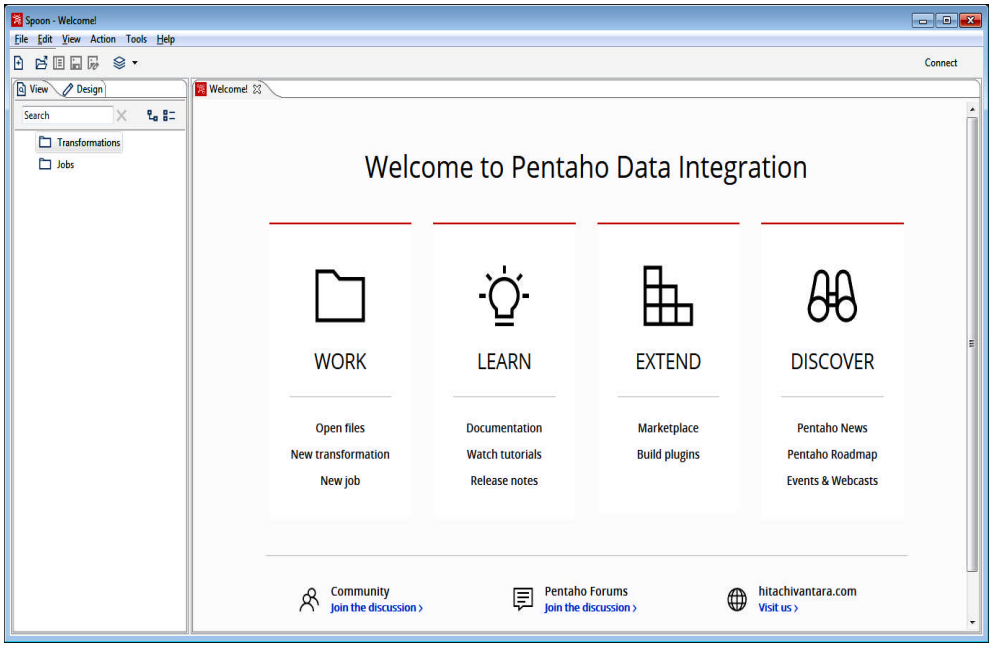
The data integration folder that you downloaded earlier will looklike this:



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.”

Questions:

1. GUI Provided by Pentaho to design Transformation and Jobs that can be run with Kettle Tools is _____.

   a) Pan b) Kitchen c) Spoon d) No GUI provided

2. Tool which performs reading, manipulating and writing various data sources is called _____.

   a) Spoon b) Kitchen c) Kettle d) Pan

3. The program that execute jobs that are designed in Pentaho Data Integration Tools is

   a) Kettle b) Kitchen c) ETTL d) Pan

Experiment no.3

Aim: Extract and load data using Pentaho Data integration tool.

Objective: Demonstration of how to build a data integration transformation and a job using the features and tools provided by Pentaho Data Integration (PDI).

Theory:

The Data Integration perspective of PDI (also called Spoon) allows us to create two basic file types: **transformations and jobs.**

Transformations describe the data flows for ETL such as reading from a source, transforming data and loading it into a target location.

Jobs coordinate ETL activities such as defining the flow and dependencies for what order transformations should be run, or prepare for execution by checking conditions such as, "Is my source file available?" or "Does a table exist in my database?"

We will learn basic concepts and processes involved in building a transformation with PDI in a typical business scenario. In this scenario, you are loading a flat file (.CSV) of **sales data** into a database so that mailing lists can be generated.

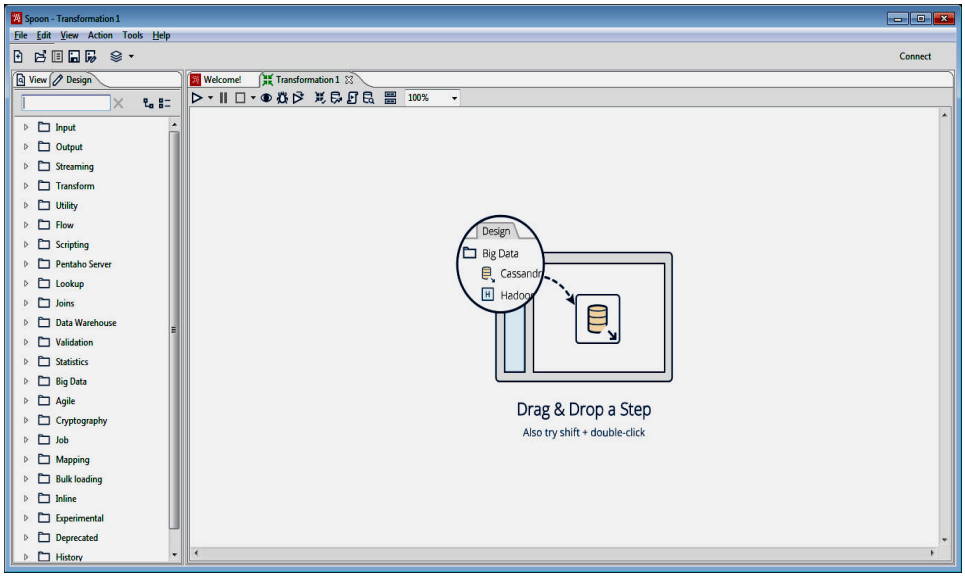
Extract and load data

We will retrieve data from a **.CSV flat file** and use the Text File Input step to: connect to a repository, view the file schema, and retrieve the data contents.

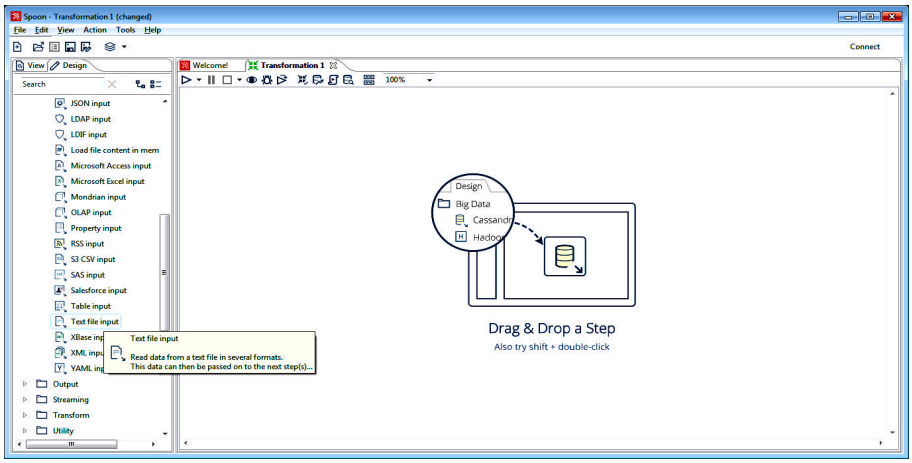
❖ Create a new transformation

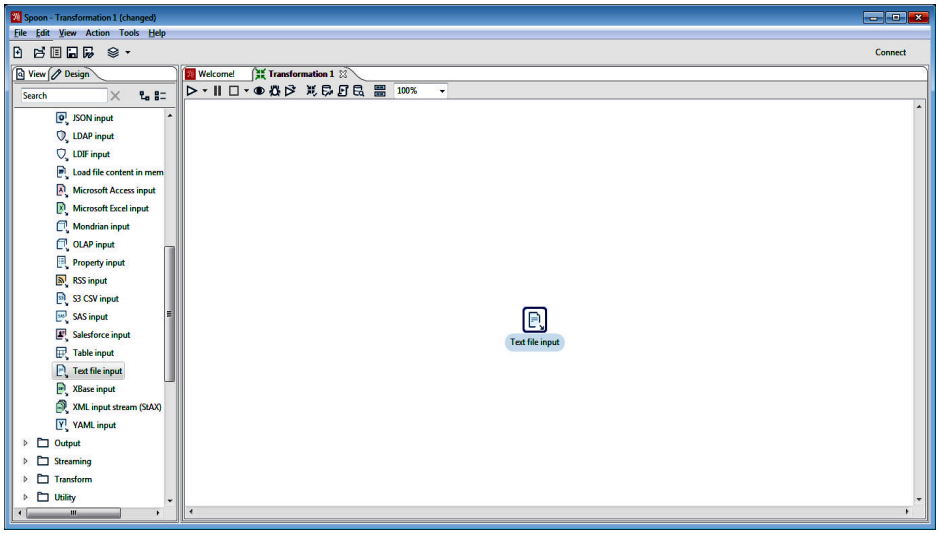
Follow these steps to create a new transformation.

1. Select File **New Transformation** in the upper left corner of the PDI window.

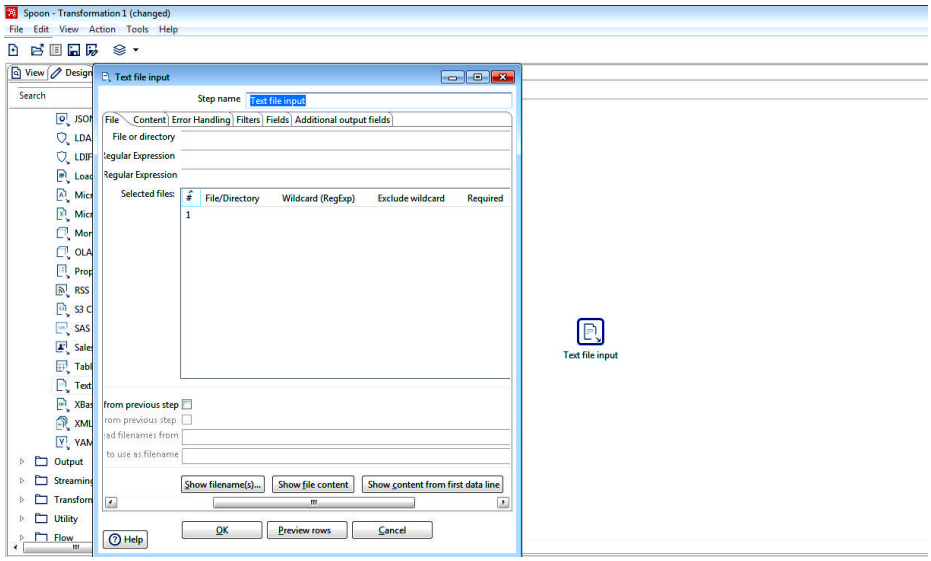


2. Under the **Design tab**, expand the **Input node**, then select and drag a **Text File Input** step onto the canvas.

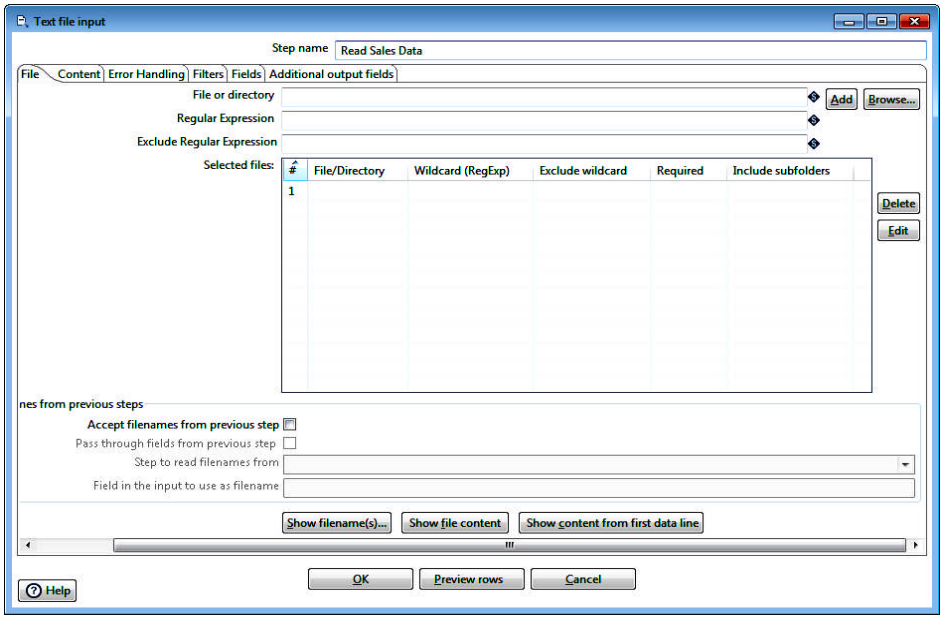




3. Double-click the **Text File input** step. It will open following window. you can set the step's various properties.



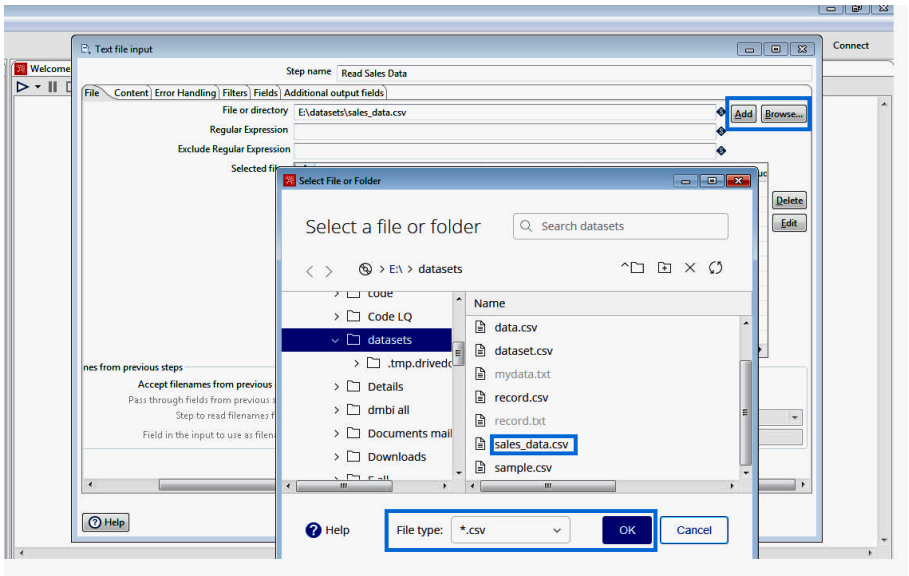
4. Now, In **Step Name field**, type **Read Sales Data.** The **Step name: Text file input** step is now renamed to **Read Sales Data.**



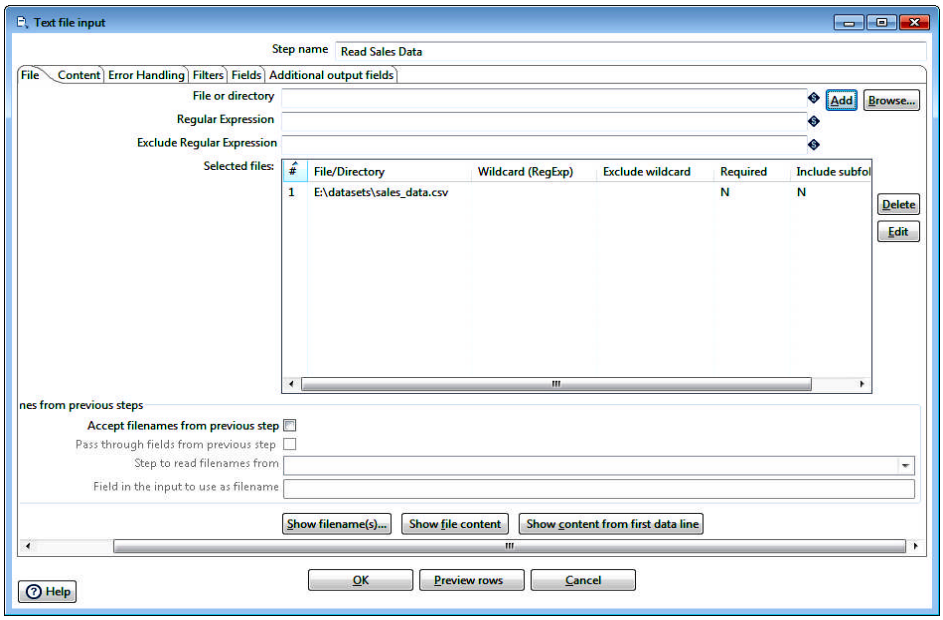
5. Click Browse to locate the source file, **sales_data.csv**, in the E:\datasets\sales_data.csv. The Browse button appears in the top right side of the window near the File or Directory field.

6. Change File type to *.csv. Select sales_data.csv, then click OK.

   The path to the source file appears in the File or directory field.



7. Click **Add.** The path to the file appears under Selected Files.



❖ View the content in the sample file

Follow these steps to look at the contents of the sample file.

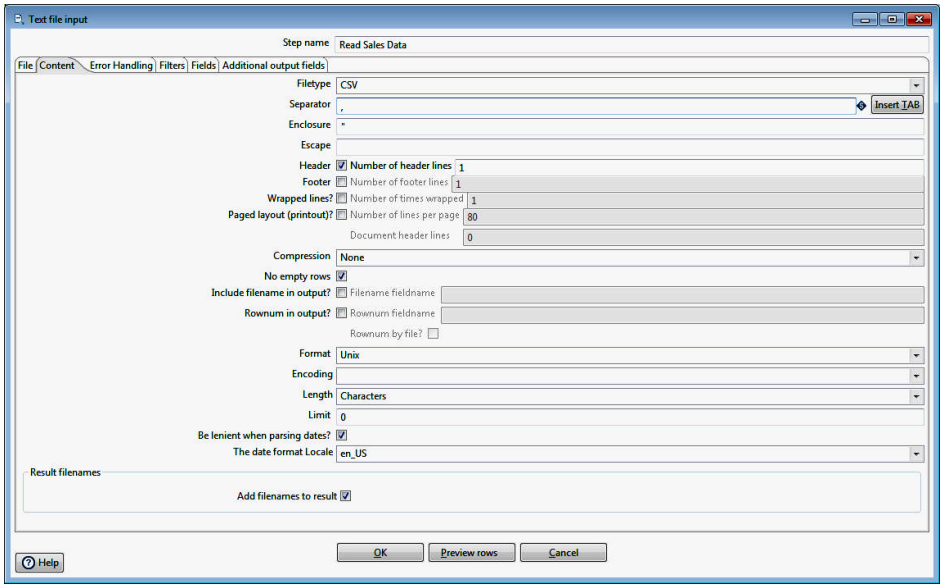
1. Click the **Content tab**, then set the **Format** field to **Unix**.
2. Click the **File** tab again and click the **Show file content** near the bottom of the window.
3. The **Number of lines (0-all lines)** window appears. Click the **OK** button to accept the default.
4. The **Content of first file** window displays the file. Examine the file to see how that input file is delimited, what enclosure character is used, and whether or not a header row is present.

   In the sample, the input file is comma delimited, the enclosure character being a quotation mark (") and it contains a single header row containing field names.
5. Click the **Close** button to close the window.

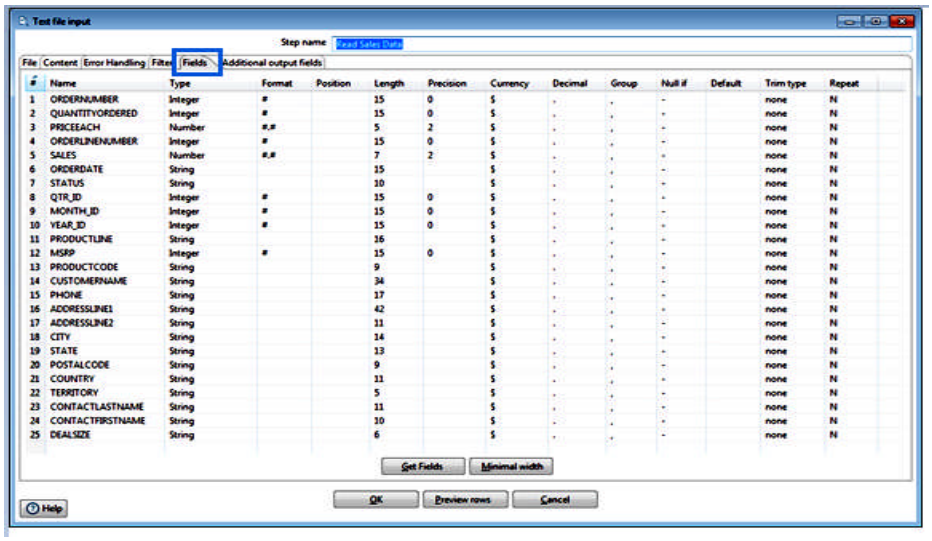
Edit and save the transformation

Follow these steps to provide information about the data's content.

1. Click the **Content** tab. The fields under the **Content** tab allow you to define how your data is formatted.
2. Verify that the **Separator** is set to **comma** (,) and the **Enclosure** is set to **quotation mark** ("). **Enable Header** because there is one line of header rows in the file and set the **Format field to Unix**.



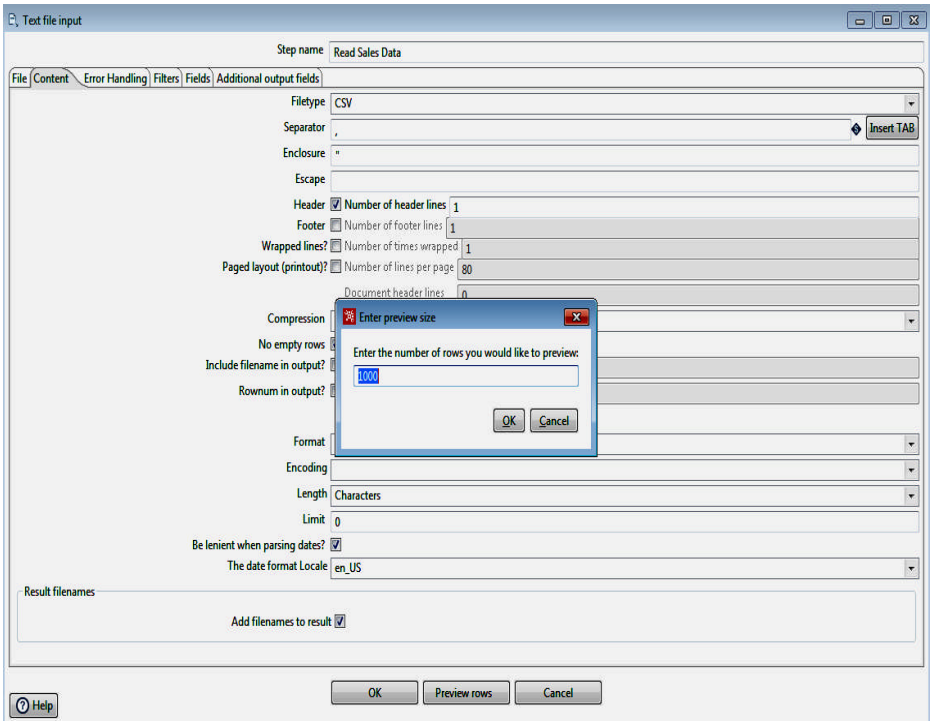
3. Click the **Fields** tab and click **Get Fields** to retrieve the input fields from your source file. When the **Number of lines** to sample window appears, enter **0** in the field then click **OK**.



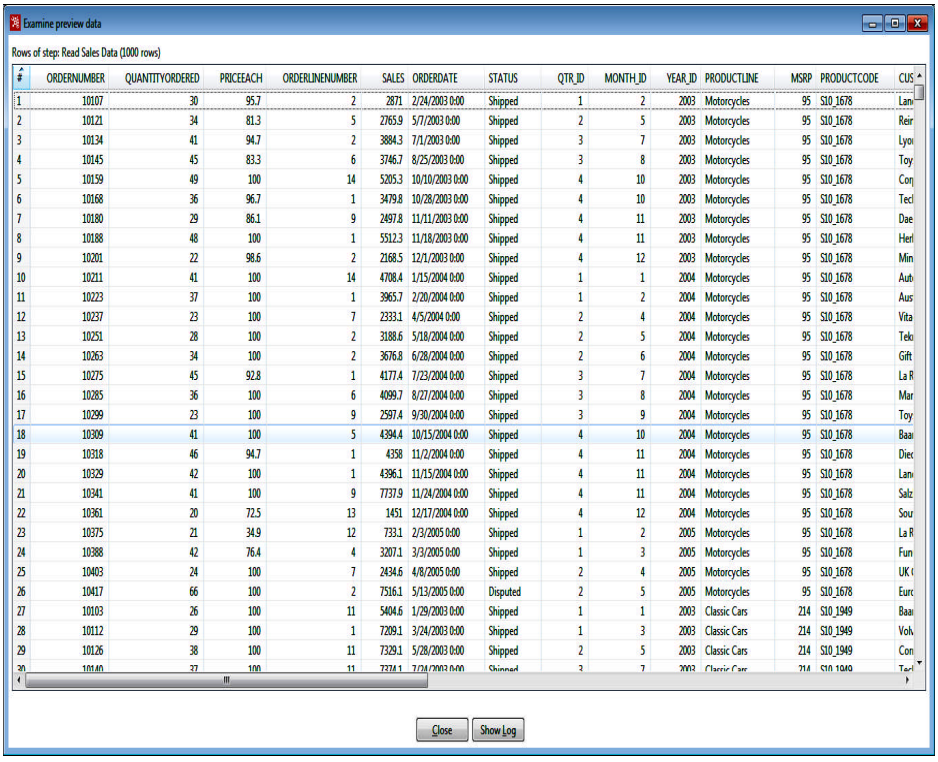
4. If the **Scan Result** window displays, click **Close** to close the window.

5. To verify that the data is being read correctly, click the **Content tab**, then click **Preview rows.**

6. In the Enter the number of rows you would like to preview window, click OK to accept the default.



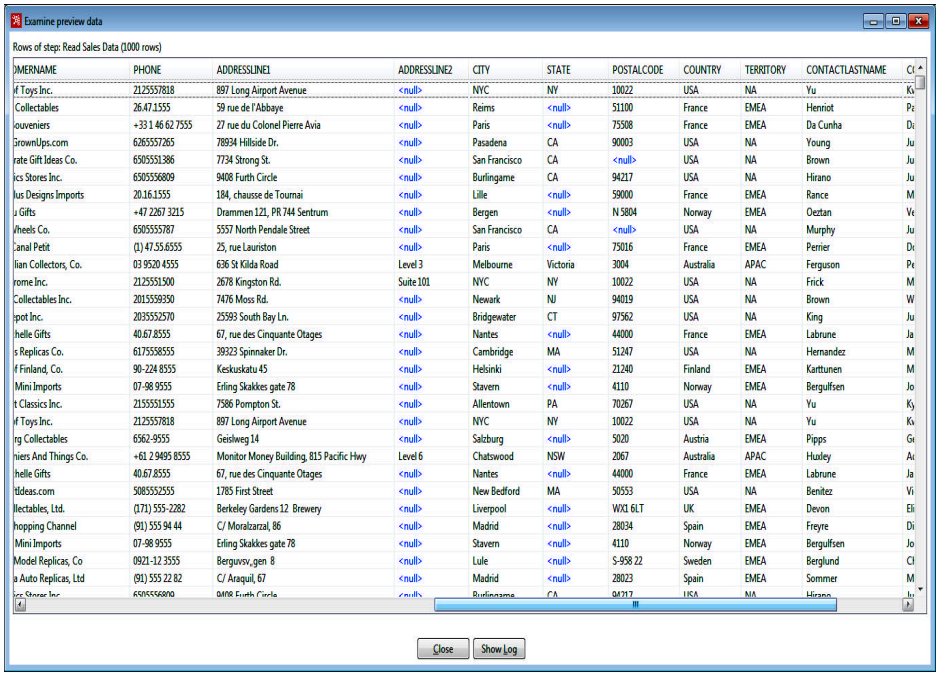
The Examine preview data window appears.



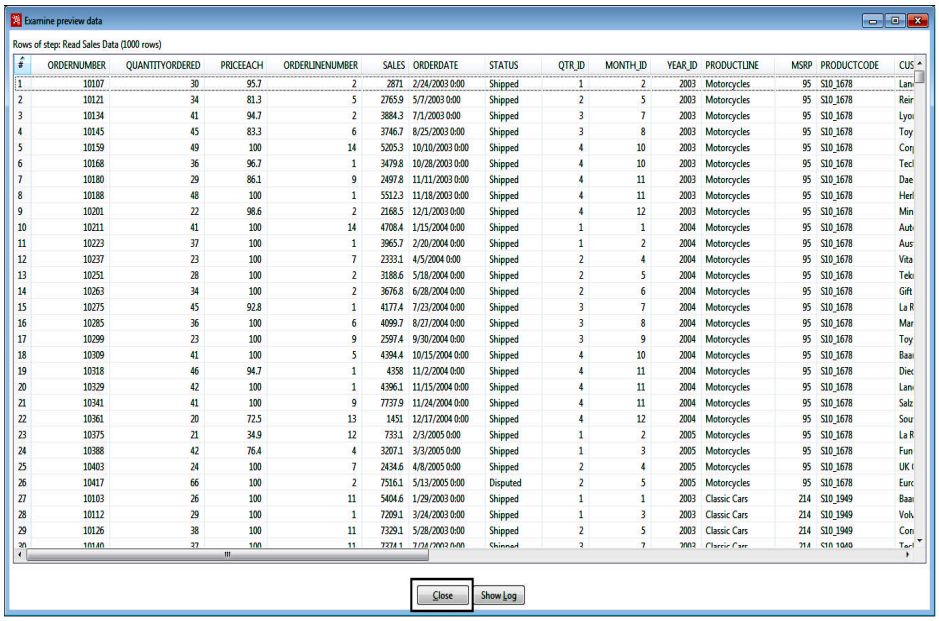
#	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE	CUS
1	10107	30	95.7	2	2871	2/24/2003 0:00	Shipped	1	2	2003	Motorcycles	95	S10_1678	Lan
2	10121	34	81.3	5	2765.9	5/7/2003 0:00	Shipped	2	5	2003	Motorcycles	95	S10_1678	Rei
3	10134	41	94.7	2	3884.3	7/1/2003 0:00	Shipped	3	7	2003	Motorcycles	95	S10_1678	Lyo
4	10145	45	83.3	6	3746.7	8/25/2003 0:00	Shipped	3	8	2003	Motorcycles	95	S10_1678	Toy
5	10159	49	100	14	5205.3	10/10/2003 0:00	Shipped	4	10	2003	Motorcycles	95	S10_1678	Cor
6	10168	36	96.7	1	3479.8	10/28/2003 0:00	Shipped	4	10	2003	Motorcycles	95	S10_1678	Tec
7	10180	29	86.1	9	2497.8	11/11/2003 0:00	Shipped	4	11	2003	Motorcycles	95	S10_1678	Dae
8	10188	48	100	1	5512.3	11/18/2003 0:00	Shipped	4	11	2003	Motorcycles	95	S10_1678	Her
9	10201	22	98.6	2	2168.5	12/1/2003 0:00	Shipped	4	12	2003	Motorcycles	95	S10_1678	Min
10	10211	41	100	14	4708.4	1/15/2004 0:00	Shipped	1	1	2004	Motorcycles	95	S10_1678	Aut
11	10223	37	100	1	3965.7	2/20/2004 0:00	Shipped	1	2	2004	Motorcycles	95	S10_1678	Aus
12	10237	23	100	7	2333.1	4/5/2004 0:00	Shipped	2	4	2004	Motorcycles	95	S10_1678	Vita
13	10251	28	100	2	3188.6	5/18/2004 0:00	Shipped	2	5	2004	Motorcycles	95	S10_1678	Tek
14	10263	34	100	2	3676.8	6/28/2004 0:00	Shipped	2	6	2004	Motorcycles	95	S10_1678	Gift
15	10275	45	92.8	1	4177.4	7/23/2004 0:00	Shipped	3	7	2004	Motorcycles	95	S10_1678	La R
16	10285	36	100	6	4099.7	8/27/2004 0:00	Shipped	3	8	2004	Motorcycles	95	S10_1678	Mar
17	10299	23	100	9	2597.4	9/30/2004 0:00	Shipped	3	9	2004	Motorcycles	95	S10_1678	Toy
18	10309	41	100	5	4394.4	10/15/2004 0:00	Shipped	4	10	2004	Motorcycles	95	S10_1678	Baa
19	10318	46	94.7	1	4358	11/2/2004 0:00	Shipped	4	11	2004	Motorcycles	95	S10_1678	Die
20	10329	42	100	1	4396.1	11/15/2004 0:00	Shipped	4	11	2004	Motorcycles	95	S10_1678	Lan
21	10341	41	100	9	7737.9	11/24/2004 0:00	Shipped	4	11	2004	Motorcycles	95	S10_1678	Salz
22	10361	20	72.5	13	1451	12/17/2004 0:00	Shipped	4	12	2004	Motorcycles	95	S10_1678	Sou
23	10375	21	34.9	12	733.1	2/3/2005 0:00	Shipped	1	2	2005	Motorcycles	95	S10_1678	La R
24	10388	42	76.4	4	3207.1	3/3/2005 0:00	Shipped	1	3	2005	Motorcycles	95	S10_1678	Fun
25	10403	24	100	7	2434.6	4/8/2005 0:00	Shipped	2	4	2005	Motorcycles	95	S10_1678	UK (
26	10417	66	100	2	7516.1	5/13/2005 0:00	Disputed	2	5	2005	Motorcycles	95	S10_1678	Eur
27	10103	26	100	11	5404.6	1/29/2003 0:00	Shipped	1	1	2003	Classic Cars	214	S10_1949	Baa
28	10112	29	100	1	7209.1	3/24/2003 0:00	Shipped	1	3	2003	Classic Cars	214	S10_1949	Vol
29	10126	38	100	11	7329.1	5/28/2003 0:00	Shipped	2	5	2003	Classic Cars	214	S10_1949	Con
30	10140	37	100	11	7374.1	7/24/2003 0:00	Shipped	3	7	2003	Classic Cars	214	S10_1949	Tel

7. Review the data. Do you notice any missing, incomplete, or variations of the data?

   ADDRESSLINE2, STATE & POSTALCODE contains <null> value.



8. Click **Close** to Examine preview data window. Click **OK** to save the information that you entered in the step i.e **Text file input** window.



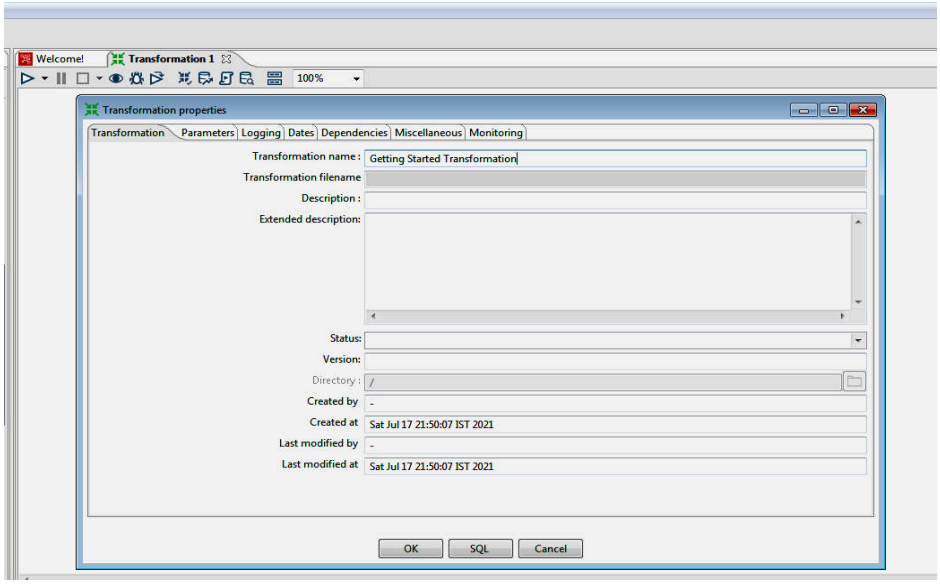
9. Give the transformation a name and provide additional properties using the Transformation Properties window. There are multiple ways to open the Transformation Properties window.

   a. Right-click on any empty space on the canvas and select properties.

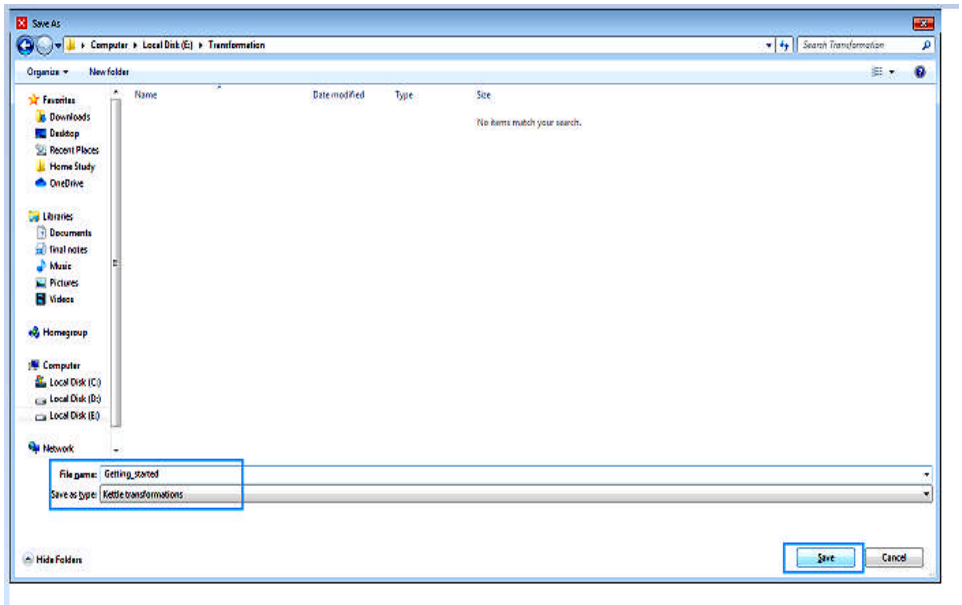
   b. Double-click on any empty space on the canvas to select properties.

10. In the **Transformation Name** field, type: **Getting Started Transformation**. Below the name you will see that the filename is empty.

11. Click **OK** to close the Transformation Properties window.



12. To save the transformation, select File Save. This is the first time you are saving transformation so you will be prompted for a file location and name of your choice. You will also see that.ktr is the usual file extension for transformations.



Questions

1. What is the extension for saving the transformations?

a.ktr b. kjb c. kmr d. tr

2. Which command is used to run a job in windows platform?

a. sh b.bat c.cmd d.notepad

3. What are the Steps of ETL Process?

a. define the source b. define the target c. create the mapping.
d. All the above

Module V

EXPERIMENT NO.1

Aim : Introduction to R.

Objective: To learn basics of R Programming. How to download and install R.

Theory:

❖ What is R Programming

R is an **interpreted computer programming language** developed by Ross Ihaka and Robert Gentleman in 1993. It is a software environment used to analyze statistical **information, graphical representation** and **reporting**.

In the current era, R is one of the most important tools used by researchers, data analyst, statisticians, and marketers for retrieving, cleaning, analyzing, visualizing, and presenting data. R allows integration with the procedures written in the C, C++, .Net, Python, and FORTRAN languages to improve efficiency.

❖ Installation of R

R programming is a very popular language and to work on it we must install two things, i.e., R and R Studio.

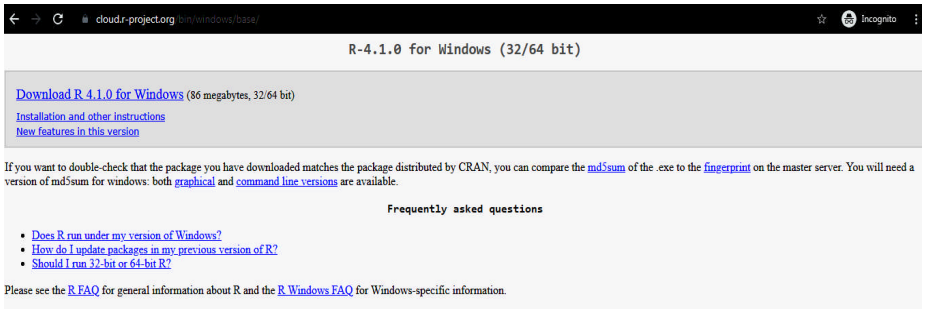
R and R Studio works together to create a project on R. Installing R to the local computer is easy. First, we must know which operating system we are using so that we can download the setup accordingly. The official site https://cloud.r-project.org provides binary files for major operating systems including Windows, Linux, and Mac OS. In some Linux distributions, R is installed by default, which we can verify from the console by entering R.

❖ R Installation in Windows

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

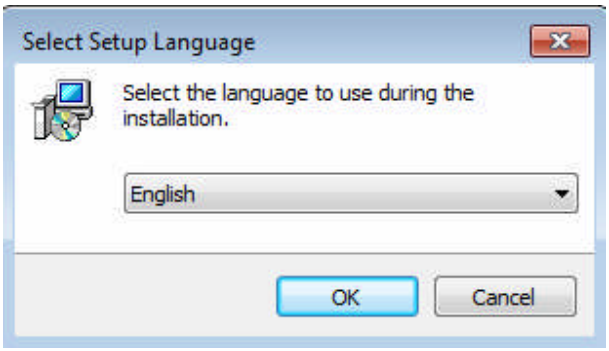
Step 1:

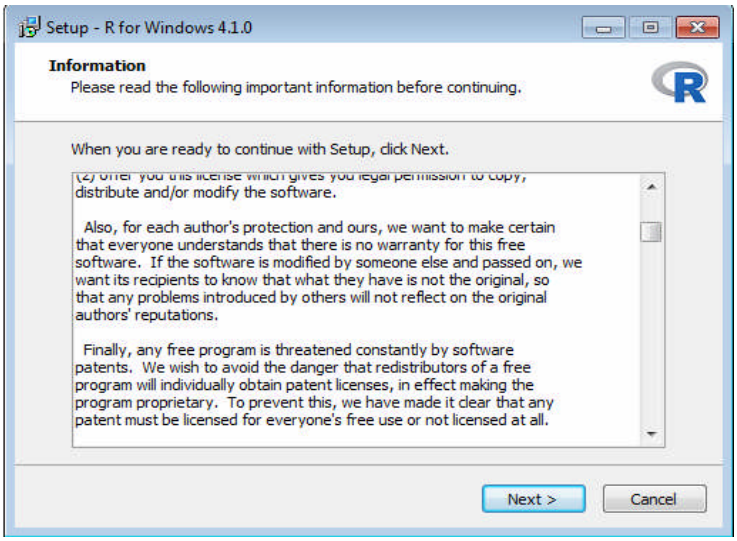
First, we have to download the R setup from https://cloud.r-project.org/bin/windows/base/.



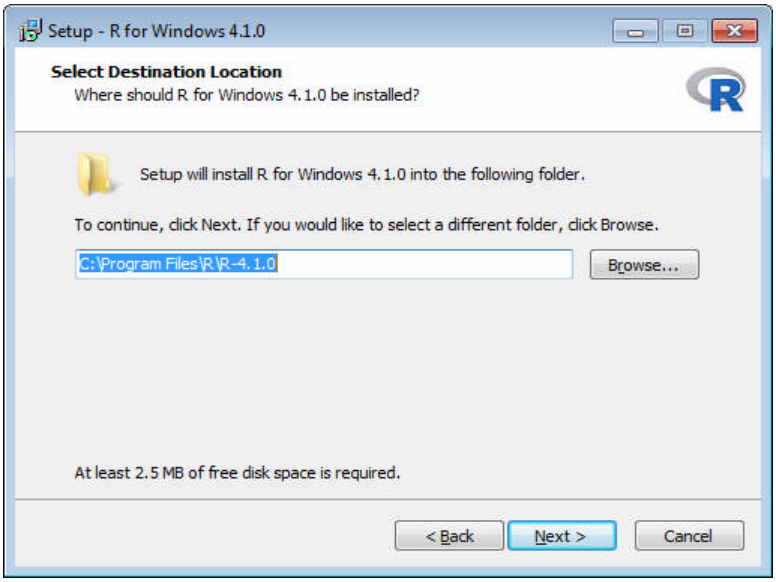
Step 2:

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:

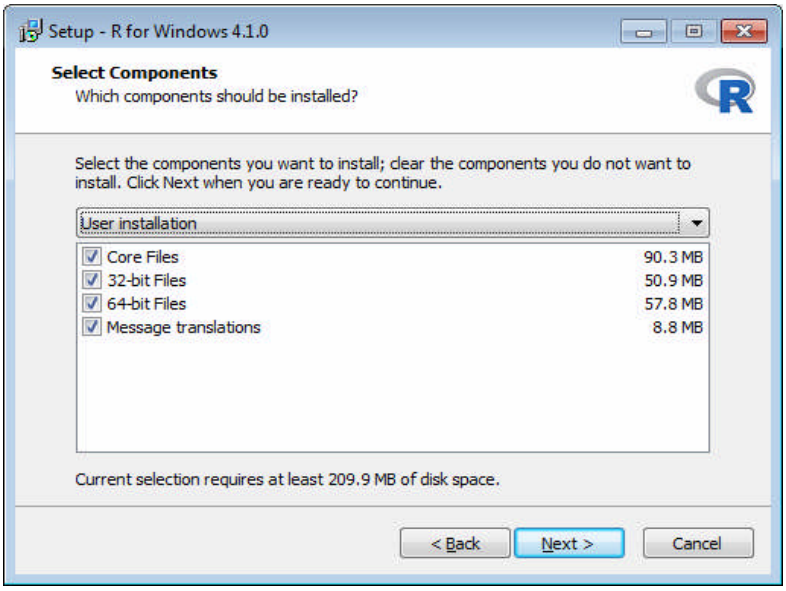




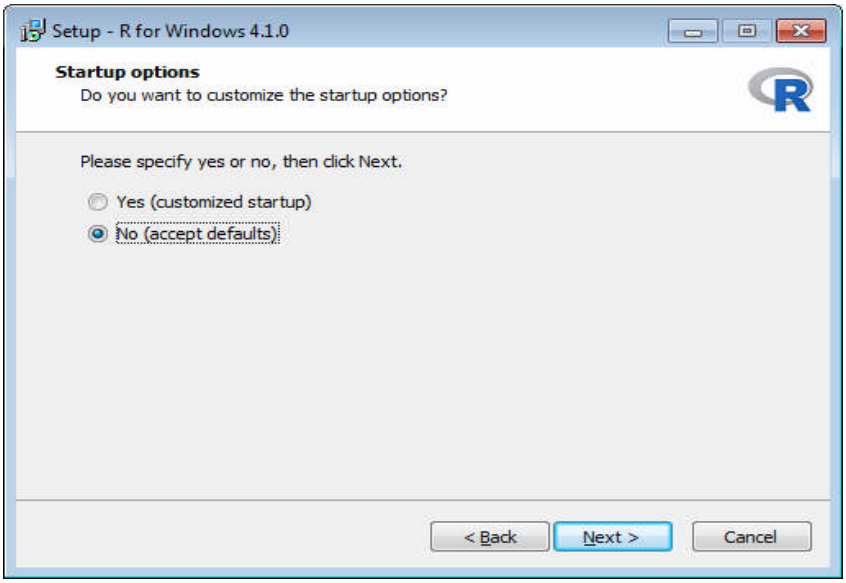
1) Select the path where we want to download the R and click **Next.**



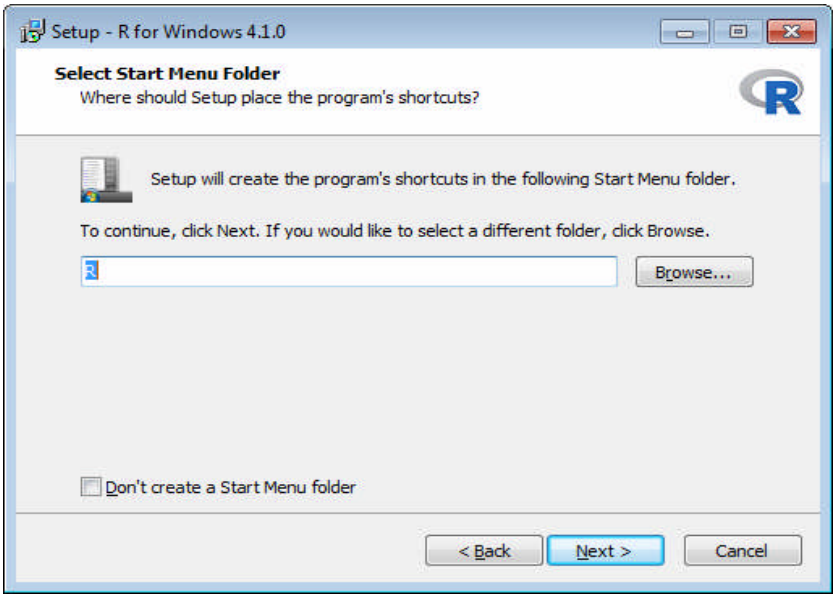
2) Select all components which we want to install, and then click **Next.**



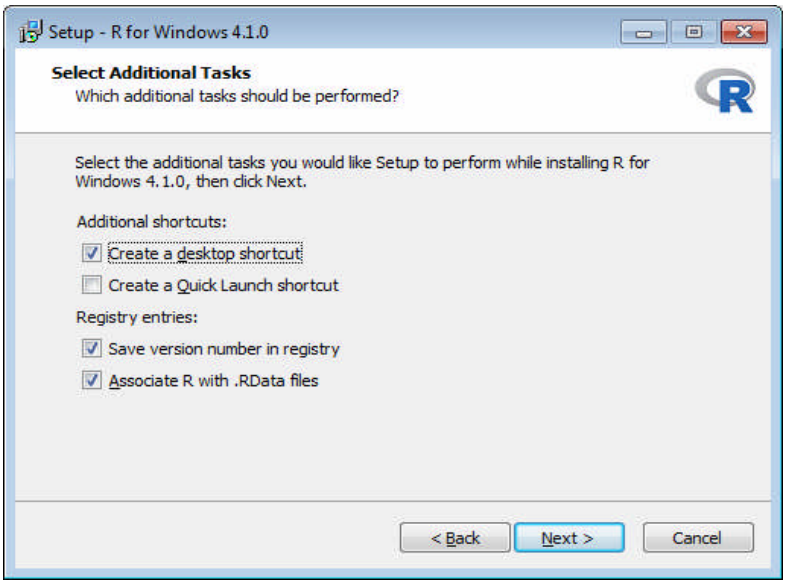
3) Now, we have to select either (customized startup) or (accept the default), and then click **Next**.



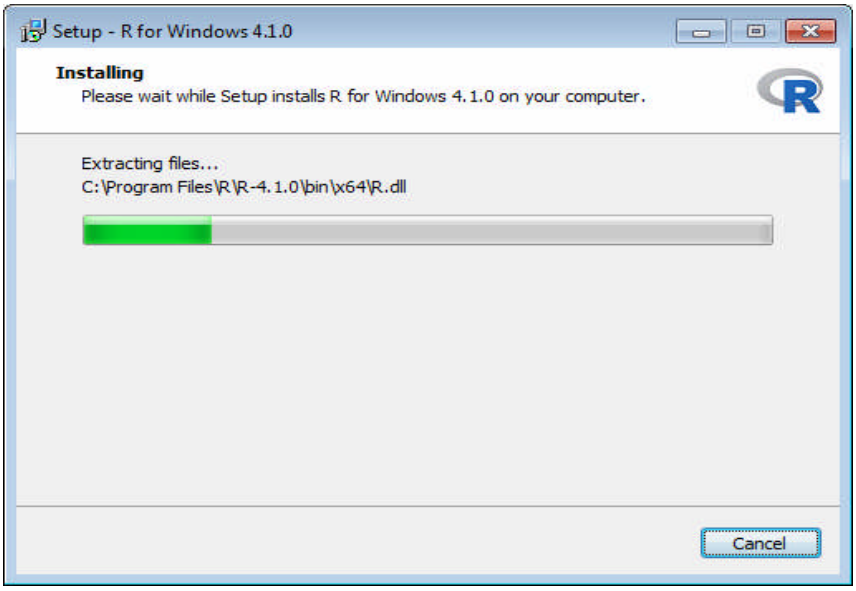
4) Now **Select Start Menu Folder** window will appear, click **Next**

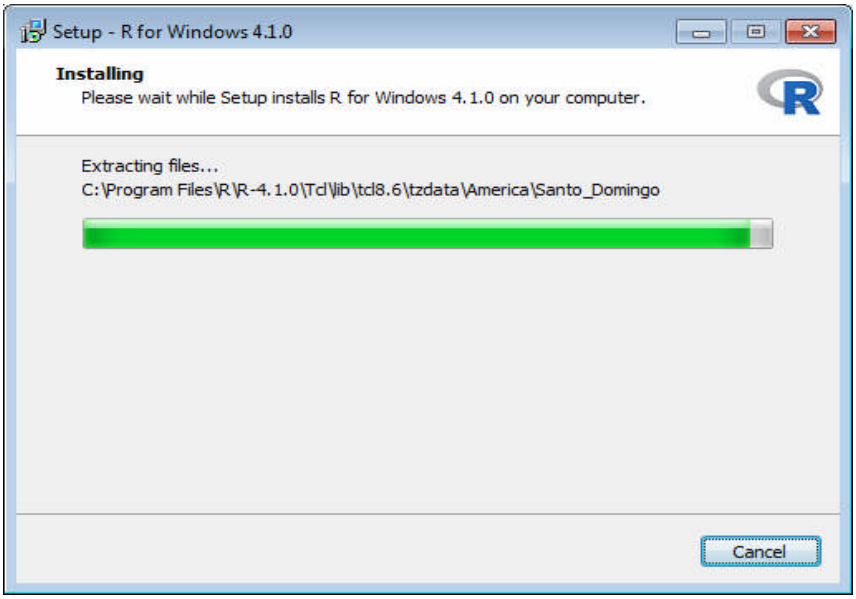


Click **Next**

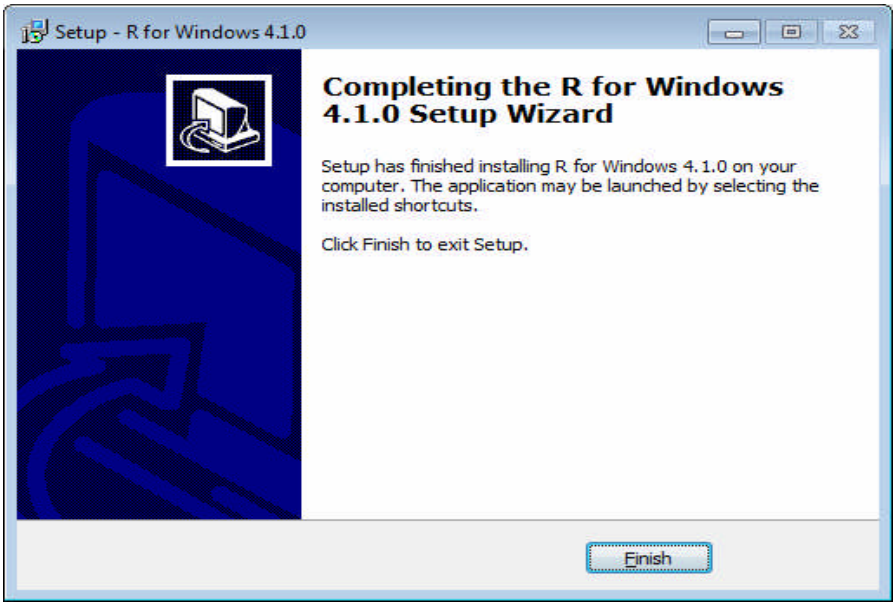


5) When we proceed to **Next**, installation of R will get started:





6) Finally, we will click on **Finish**.



R has been successfully installed.

❖ **R Studio IDE**

R Studio is an integrated development environment which allows us to interact with R more readily. R Studio considered more user-friendly. This IDE has various drop-down menus, Windows with multiple tabs, and so many customization processes.

First time when we open R Studio, we will see three Windows. The fourth Window will be hidden by default. We can open this hidden Window by clicking the **File** drop-down menu, then **New File** and then **R Script**.

RStudio Windows/Tabs	Location	Description
Console Window	Lower-left	The location where commands are entered and output is printed.
Source Tabs	Upper-left	Built-in test editor
Environment Tab	Upper-left	An interactive list of loaded R objects.
History Tab	Upper-left	List of keystrokes entered into the console.
Files Tab	Lower-right	File explorer to navigate C drive folders.
Plots Tab	Lower-right	Output location for plots.
Packages Tab	Lower-right	List of installed packages.
Help Tab	Lower-right	Output location for help commands and help search Window.
Viewer Tab	Lower-right	Advanced tab for local web content.

❖ **Installation of R Studio**

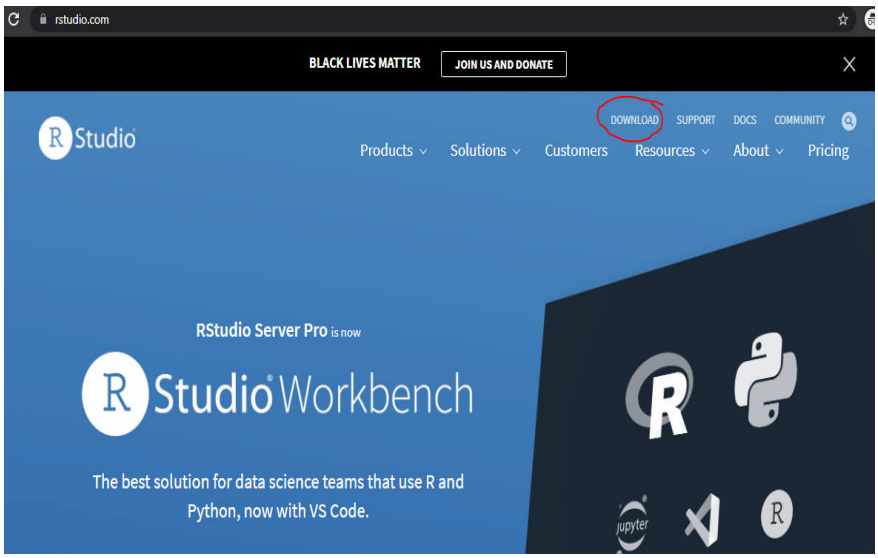
R Studio Desktop version is available for both Windows and Linux. The open-source R Studio Desktop installation is very simple to install on both operating systems.

Installation on Windows/Linux

The process of installing RStudio in both the OS Windows/Linux is same. Steps to install RStudio in Windows/Linux are as follows:

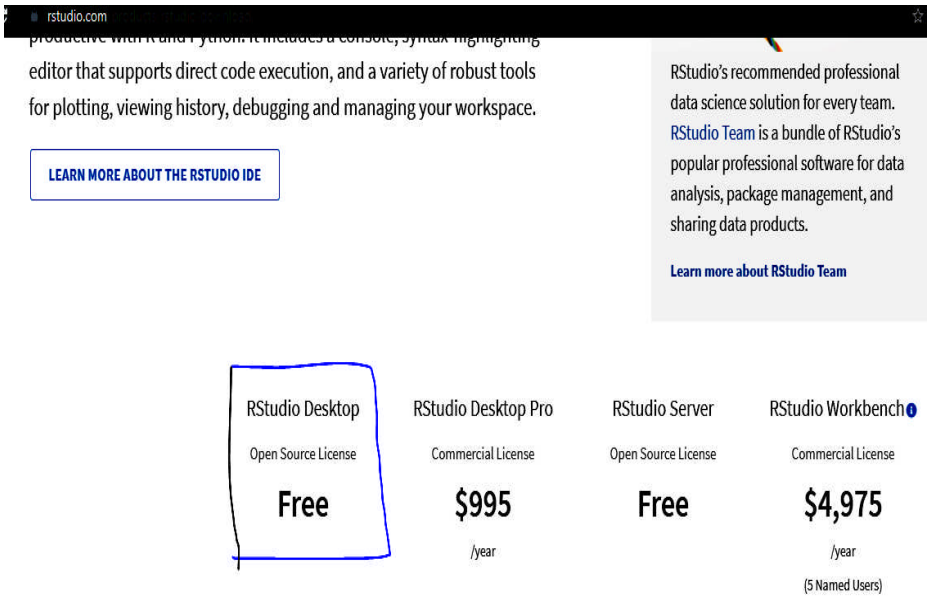
Step 1:

Visit the RStudio's official website and click on **Download.**



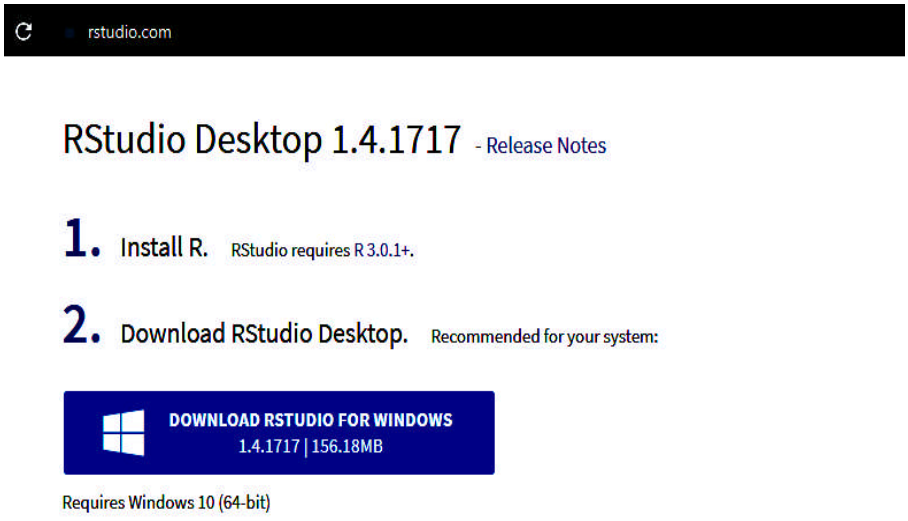
Step 2:

In the next step, we will select the RStudio desktop for open-source license.



Step 3:

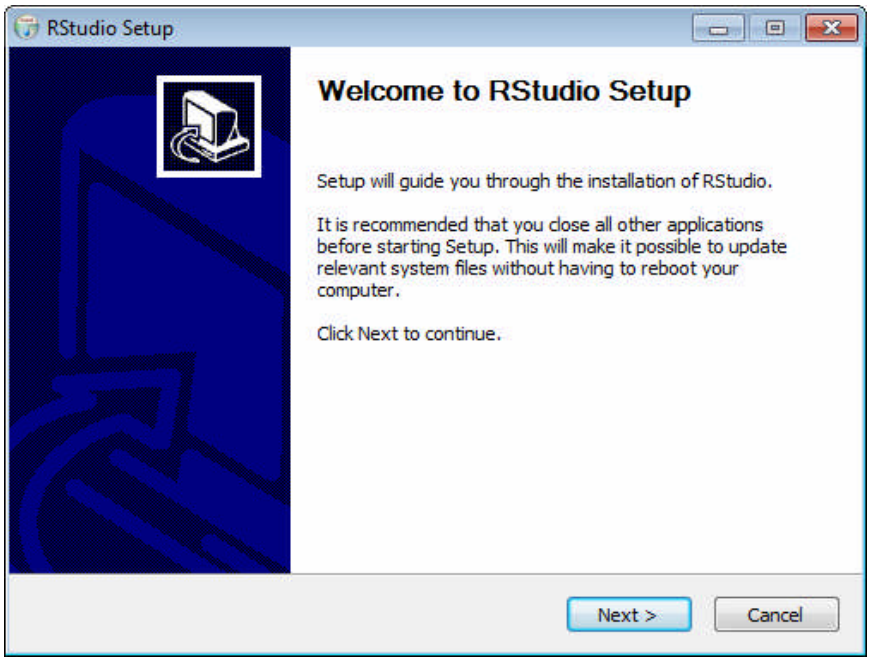
Now, Click on **Download RStudio for Windows**.



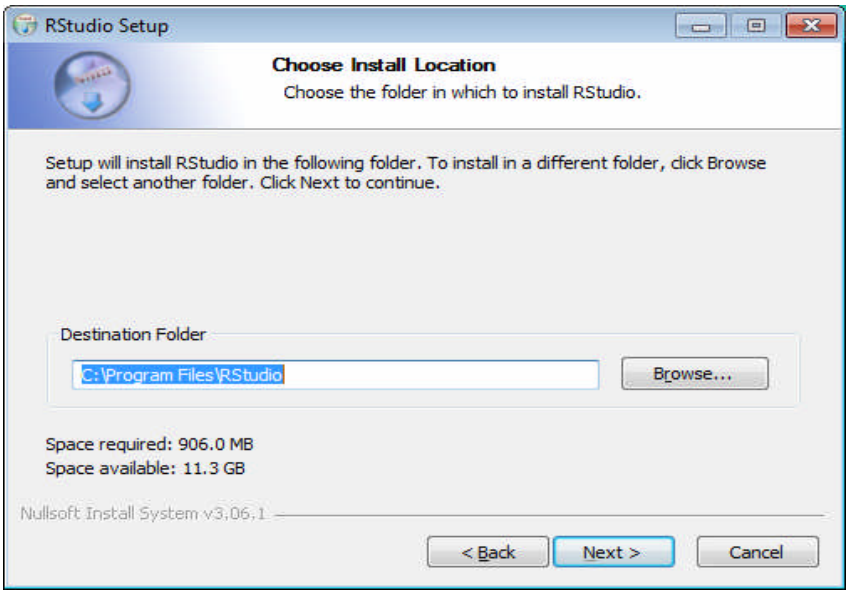
Downloading of RStudio setup will start.

Step 4:

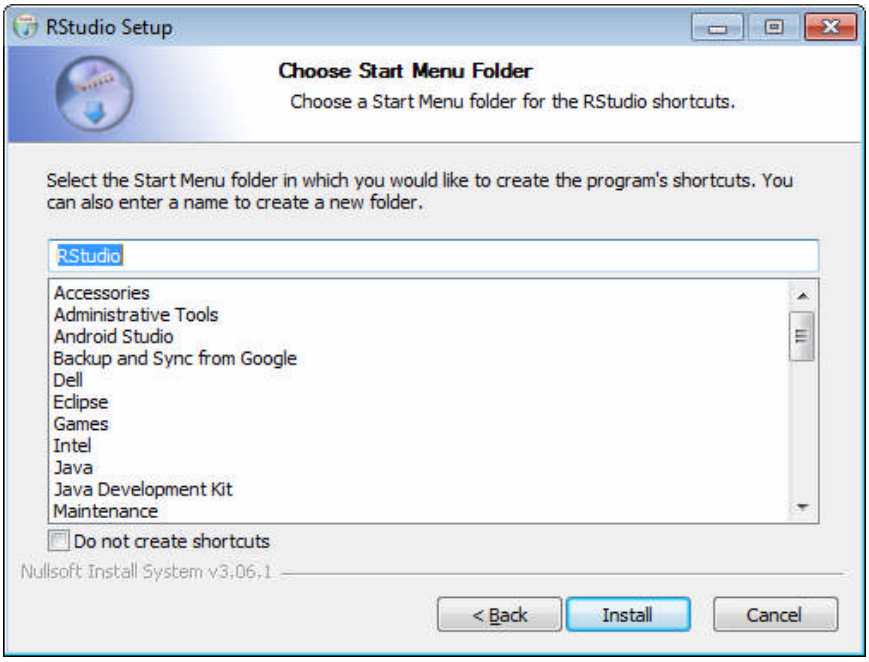
Double click on downloaded (RStudio setup) file, it will open **Welcome to RStudio Setup window**. Click **Next.**



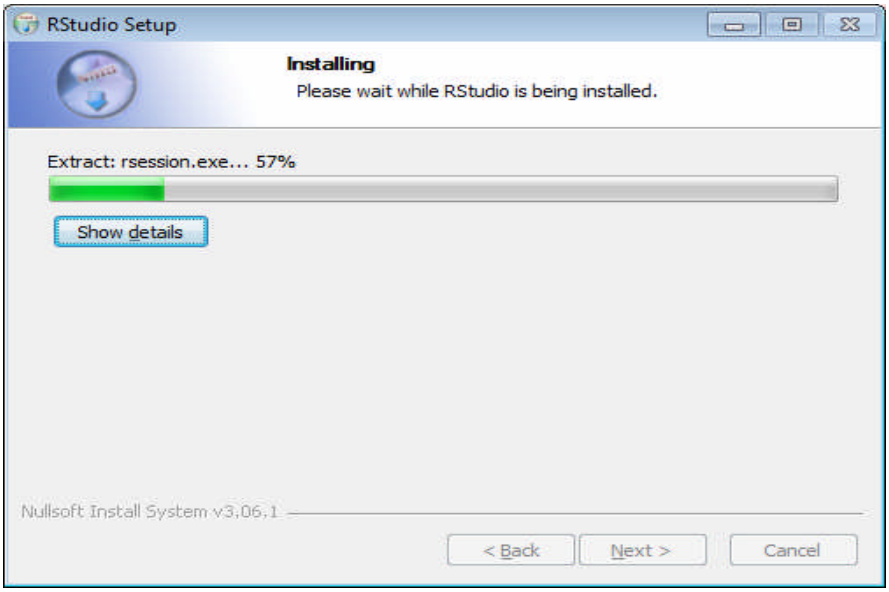
1) Click **Next**.



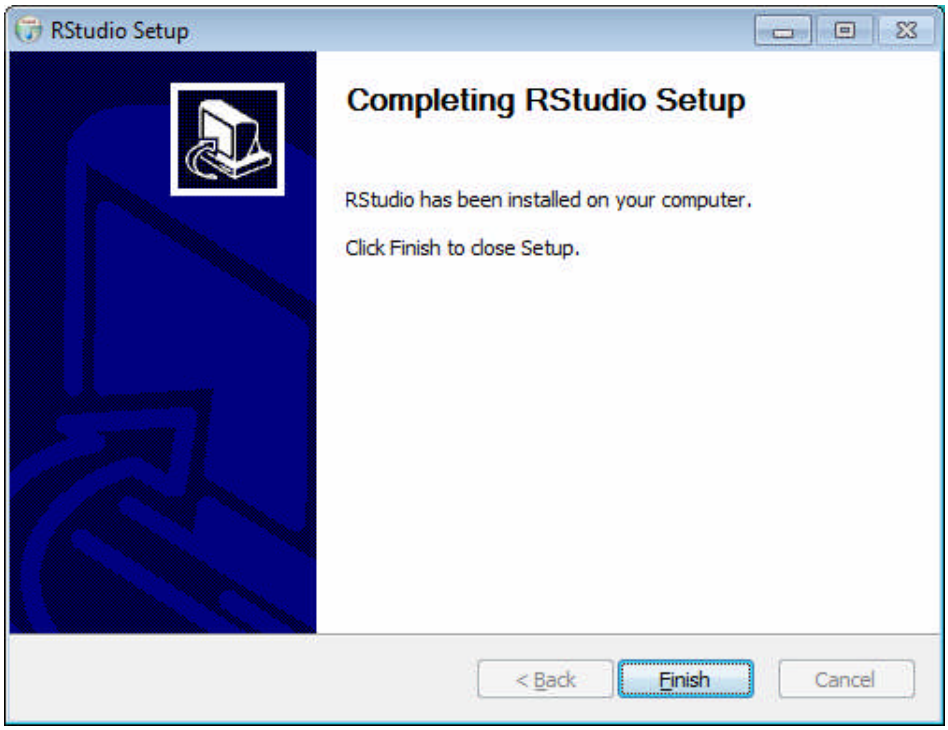
2) Click on **Install**.



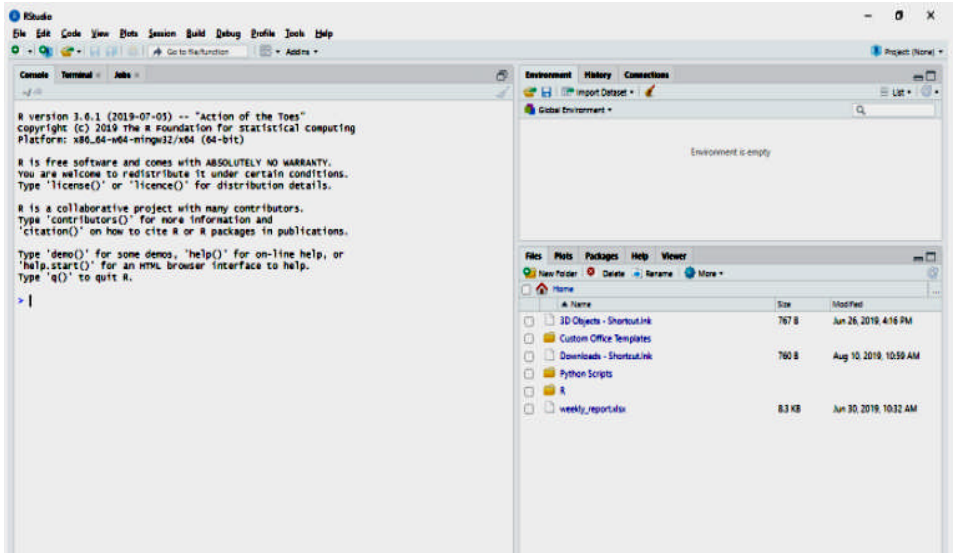
3) Now, Installation will start.



4) Click on **Finish**.



5) RStudio is ready to work.



Questions

1. Who developed R?
   a). Ross Ihaka b). Robert Gentleman
   c). Dennis Ritchie d). Both A and B

2. R allows integration with the procedures written in the?
   a). C b). Ruby c). Java d). All of the above

3. Which of the following is used for executing R programs?
   a). Google Chrome b). Microsoft word
   c) Command prompt d) R Studio

4. _____ is a software environment used to analyze statistical information**,** graphical representation **and** reporting**.**
   a) Notepad b) C c) Firefox d) R

Experiment no.2

Aim: Data types in R Programming.

Objective: To understand different data types used in R programming. We will also learn how to use R console to execute programs.

Theory:

In programming languages, we need to use various variables to store information. Variables are reserved memory locations to store values. It means that when you create a variable you reserve some space in memory.

Each variable in R has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it.

The following table shows the data type and the values that each data type can take.

Basic Data Types	Values
Numeric	Set of all real numbers
Integer	Set of all integers, Z
Logical	TRUE and FALSE
Complex	Set of complex numbers
Character	"a", "b", "c", ..., "@", "#", "$", ..., "1", "2", ...etc

- **Numeric Datatype**

Decimal values are called numeric in R. It is the default data type for numbers in R. If you assign a decimal value to a variable x as follows, x will be of numeric type.

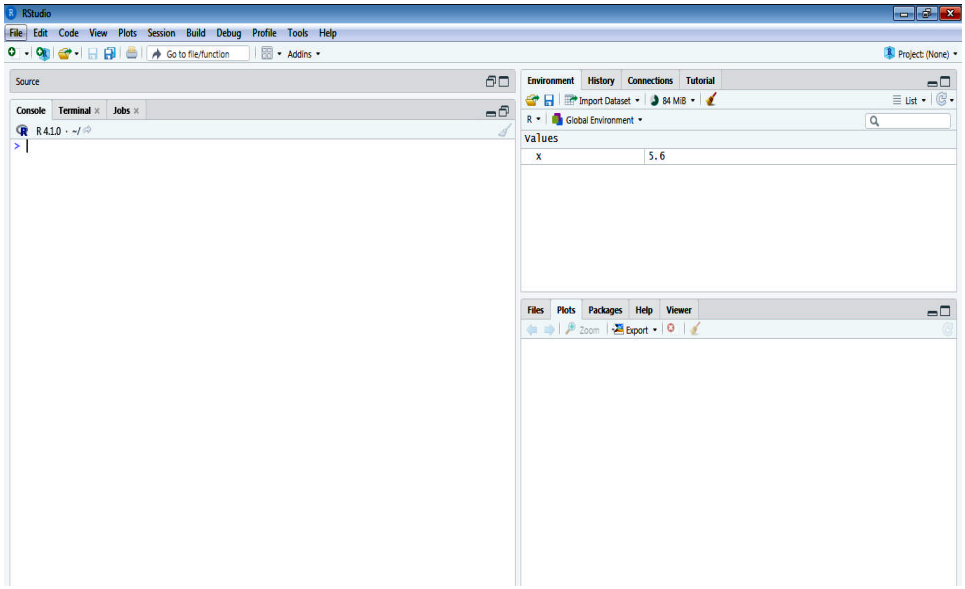
```
# R Program to illustrate Numeric data
type
# Assign a decimal value to variable x
   x = 5.6
# print the class name of variable x
    print(class(x))

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

Output:

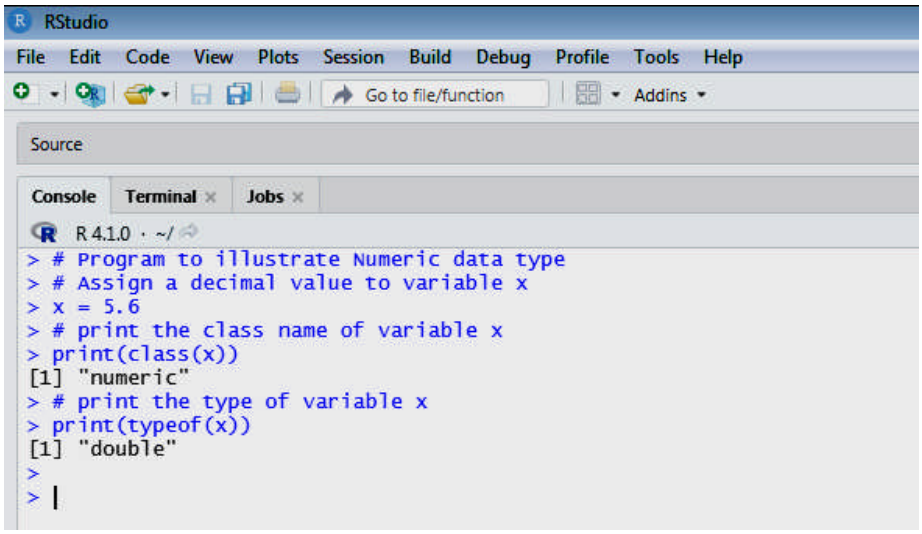
```
[1] "numeric"

[1] "double"
```

Steps to execute above program in RStudio:

1. Open RStudio.



2. Type your program in console and press enter to see the output.



Even if an integer is assigned to a variable y, it is still being saved as a numeric value.

```
# 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))
```

Output:

```
[1] "numeric"

[1] "double"
```

When R stores a number in a variable, it converts the number into a “double” value or a decimal type with at least two decimal places. This means that a value “5” here, is stored as 5.00 with a type of double and a class of numeric.

And also variable y is not an integer here. It can be confirmed by using **is.integer()** function.

```
# 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))
```

Output:

[1] FALSE

- **Integer Datatype**

R supports integer data types which are the set of all integers. You can create as well as convert a value into an integer type by using the **as.integer()** function. You can also use the capital letter 'L' as a suffix to denote that a particular value is of the integer data type.

 # 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(type of(y))
```

Output:

[1] "integer"

[1] "integer"

[1] "integer"

[1] "integer"

- ## Logical Datatype

R has logical data types which take only two values either a true or false.

 # 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(type of(z))
```

Output:

```
[1] TRUE

[1] "logical"

[1] "logical"
```

- **Complex Datatype**

R supports complex data types which are set of all the complex numbers. The complex data type is used to store numbers with an imaginary part.

 # 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(type of(x))
```

Output:

```
[1] "complex"

[1] "complex"
```

- **Character Datatype**

R language supports character data types where you have all the alphabets and special characters. It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols. The easiest

way to show that a value is of character type in R is to enclose the value inside single or double inverted commas.

 # 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(type of(char))

Output:

[1] "character"

[1] "character"

Questions:

1. What will be output for the following code?

   >a <- TRUE

   >print(class(a))

   a) logical b) Numeric c) Integer d) Complex

2. If you explicitly want an integer, you need to specify the _____ suffix.

a) D b) R c) L d) K

3. What will be the output of the following R code?

   > x <- 6

   > class(x)

   a) “integer” b) “numeric” c) “real” d) “imaginary”

4. What will be output for the following code?

   >v <- "Mumbai University"

   >print(class(v))

   a). logical b) Numeric c) Integer d) Character

5. What will be output for the following code?

> sqrt(-17)

a) -4.02 b) 4.02 c) 3.67 d) NAN

6. Decimal values are referred as _____ data types in R.

a) Numeric b) Character c). Integer d) Lists

Experiment no.3

Aim: Reading and Writing data to and from R.

Objective: To learn how to read data in R and write (export) data to files in R.

Theory:

Functions for Reading Data into R:

1. **read.table()** and **read.csv()** : functions used for reading tabular data into R.
2. **readLines()** : for reading lines from a text file.
3. **source()** : function for reading in R code files from a another R program.
4. **dget():** function for reading in R code files.
5. **load():** function is used for reading in saved workspaces.

Functions for Writing Data to Files:

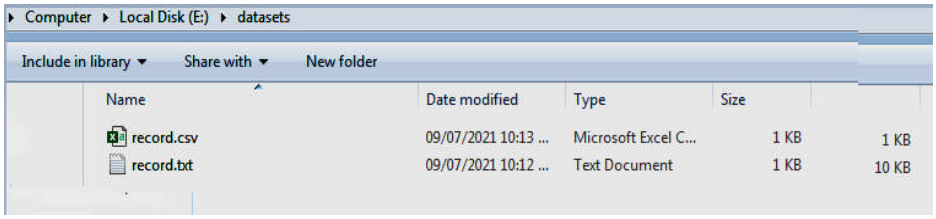
1. **write.table()** : for writing tabular data to text files (i.e. CSV).
2. **writeLines() :** function is useful for writing character data line-by-line to a file or connection.
3. **dump()** : function for dumping a textual representation of multiple R objects.
4. **dput()** : function is used for outputting a textual representation of an R object.
5. **save()** : for saving an arbitrary number of R objects in binary format to a file.

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.

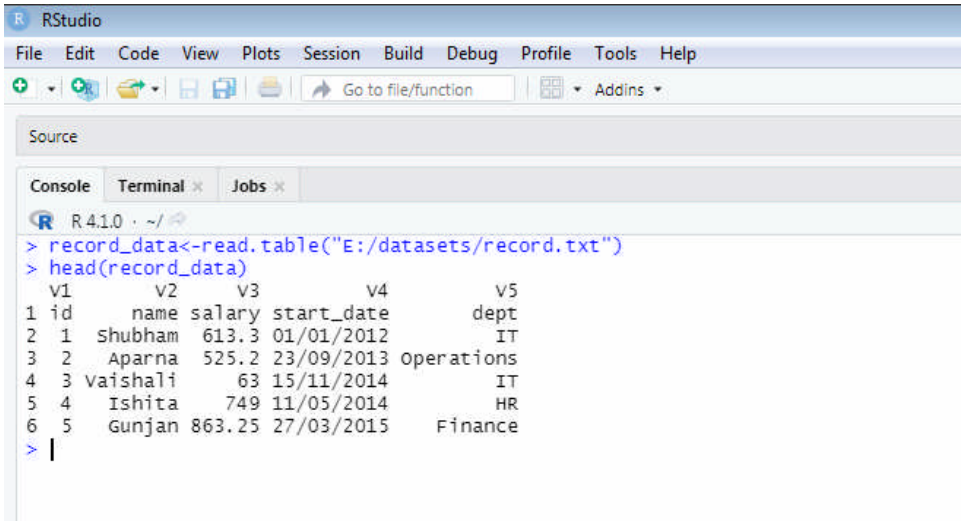
The function **read.table()** can be used to read the data frame.

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



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

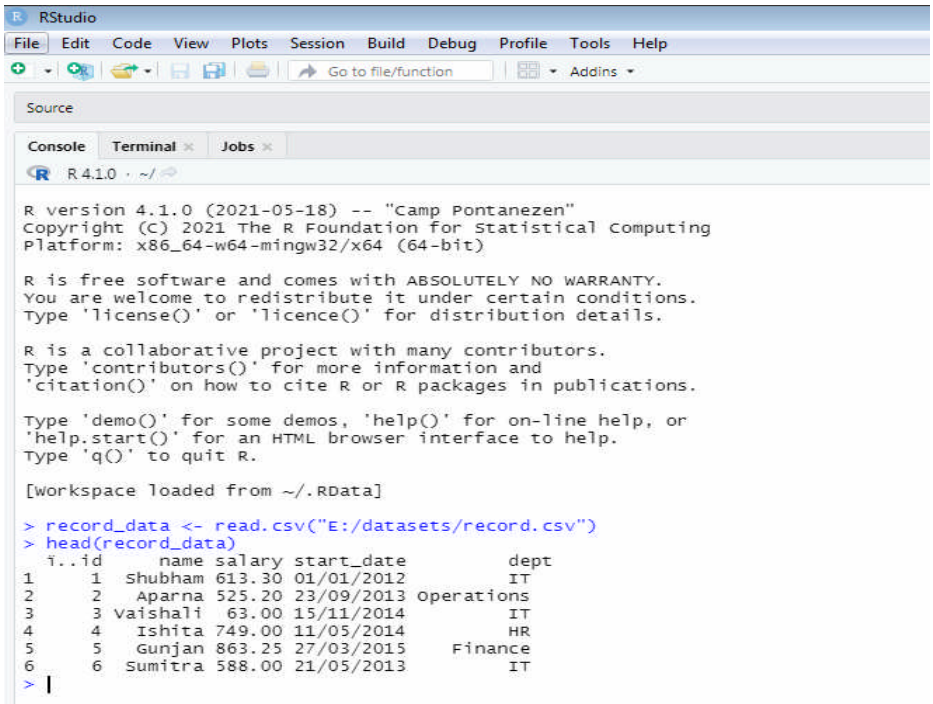
>head(record_data) #returns first n rows of the data



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

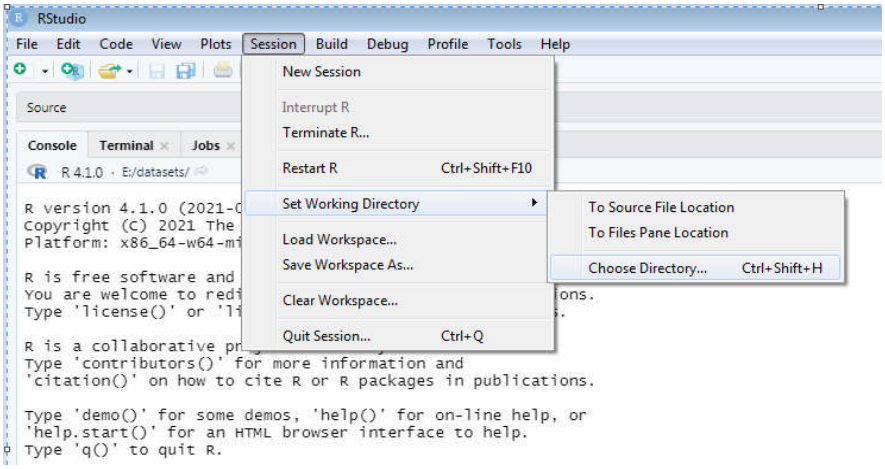


❖ Writing Data to a File

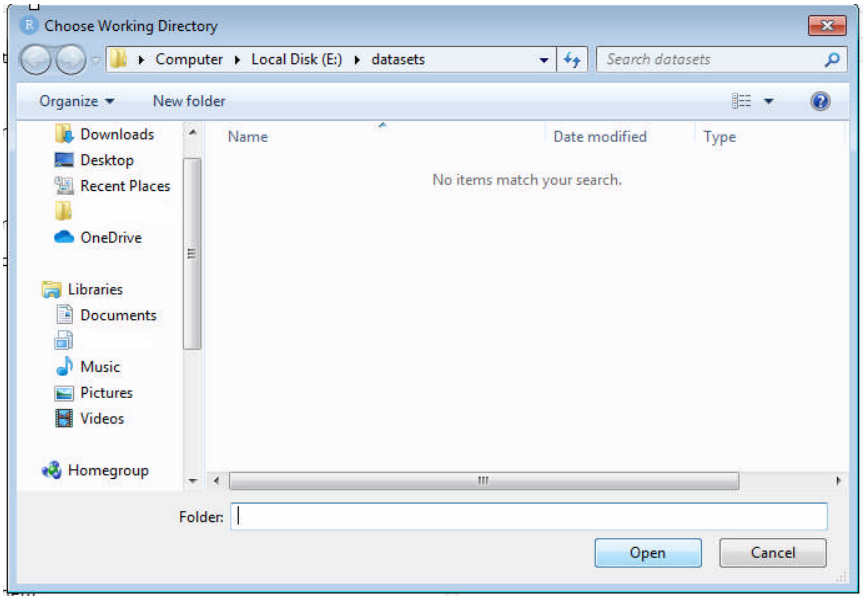
After working with a dataset, we might like to save it for future use. Before we do this, let's first set up a working directory so we know where we can find all our data sets and files later.

Setting up a Directory

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

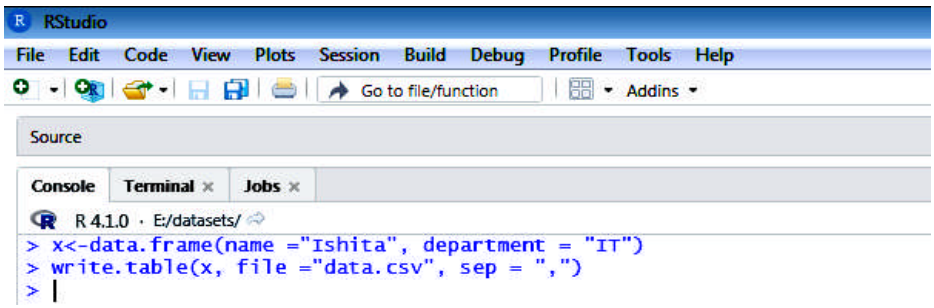


Click Open.



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

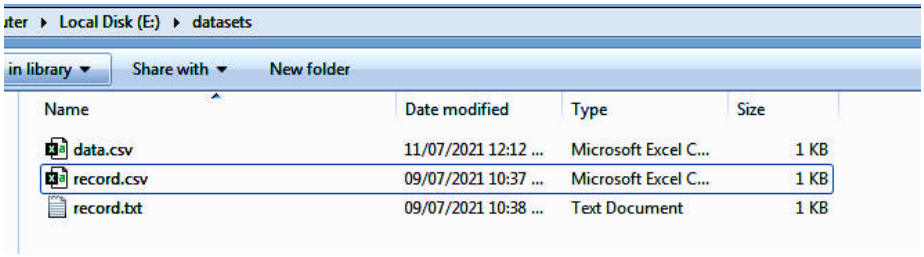
> setwd("E:/datasets")
To check your current working directory, type
> getwd()
In R, we can write data easily to a file, using the write.table() command.
x<-data.frame(name ="Ishita", department = "IT")
write.table(x, file ="data.csv", sep = ",")



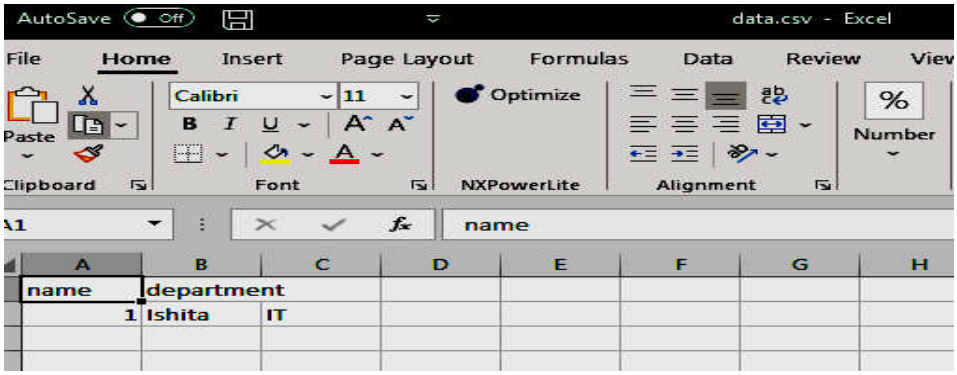
Following are few important arguments used in write.table() function.

- **x**, the object to be written, typically a data frame.
- **file**, the name of the file which the data are to be written to.
- **sep**, the field separator string.

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



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

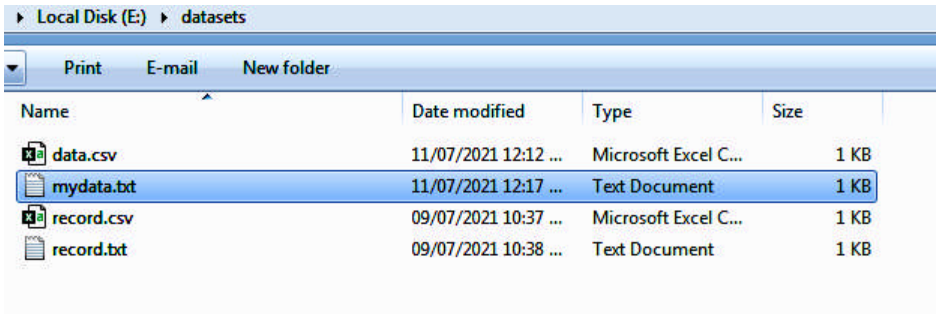


y<-data.frame(name ="Ankit", department = "HR")

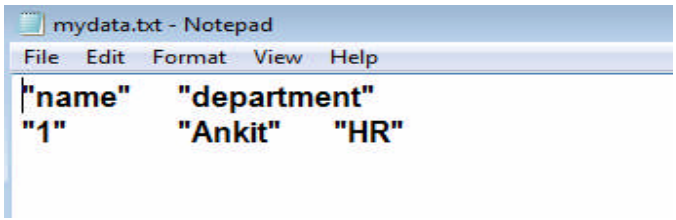
write.table(y,"E:/datasets/mydata.txt", sep = "\t")

```
> 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.

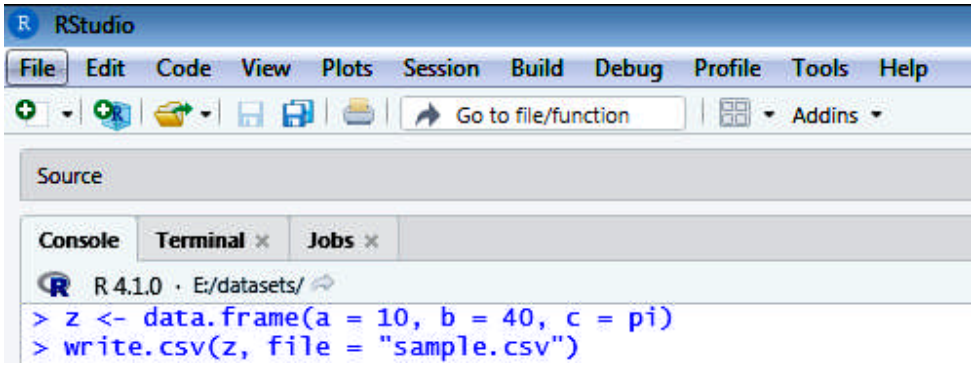


By going to this location E:/datasets, you should see a mydata.txt file.

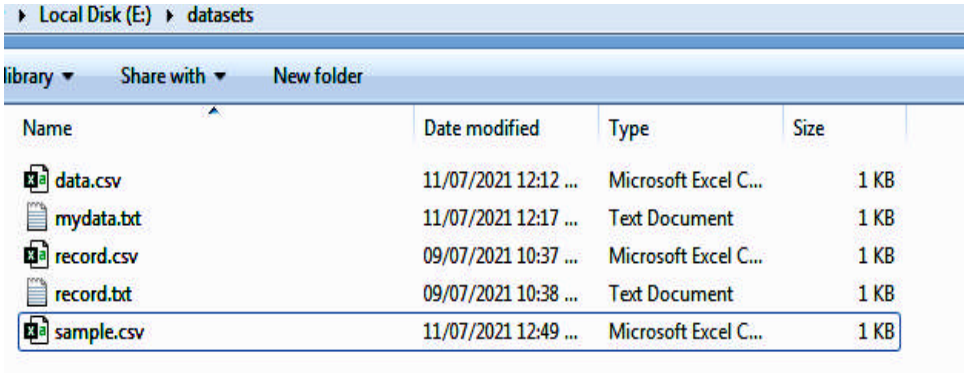


z <- data.frame(a = 10, b = 40, c = pi)

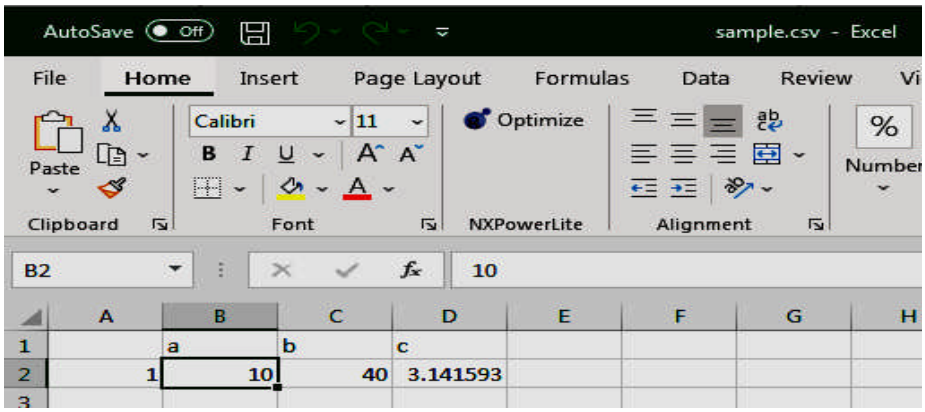
write.csv(z, file = "sample.csv")



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.



Questions:

1. Which of the following is used for reading tabular data?
   a) read.csv b) dget c) read Lines d) writeline

2. Which of the following function is identical to read .table?
   a) read.csv b) read.data c) read.tab d) read.del

3. _____ is used for outputting a textual representation of an R object.
   a) dput b) dump c) dget d) dset

Experiment no.4

Aim: Packages in R programming.

Objective: To learn R packages**.** How to install a new Package in R?

Theory:

R Packages

R packages are the collection of R functions, sample data and compile codes. In the R environment, these packages are stored under a directory called "**library**." By default, during installation R installs a set of packages. We can add packages later also when they are needed for some specific purpose.

When we start the R console, only the default packages are available by default. Other packages which are already installed will be loaded explicitly to be used by the R program.

All the packages available in R language are listed at

https://cran.r-project.org/web/packages/available_packages_by_name.html

List of commands that can be used to check, verify, and use the R packages are as follows.

- **Check Available R Packages**

✓ Get library locations containing R packages.

```
.libPaths()
```

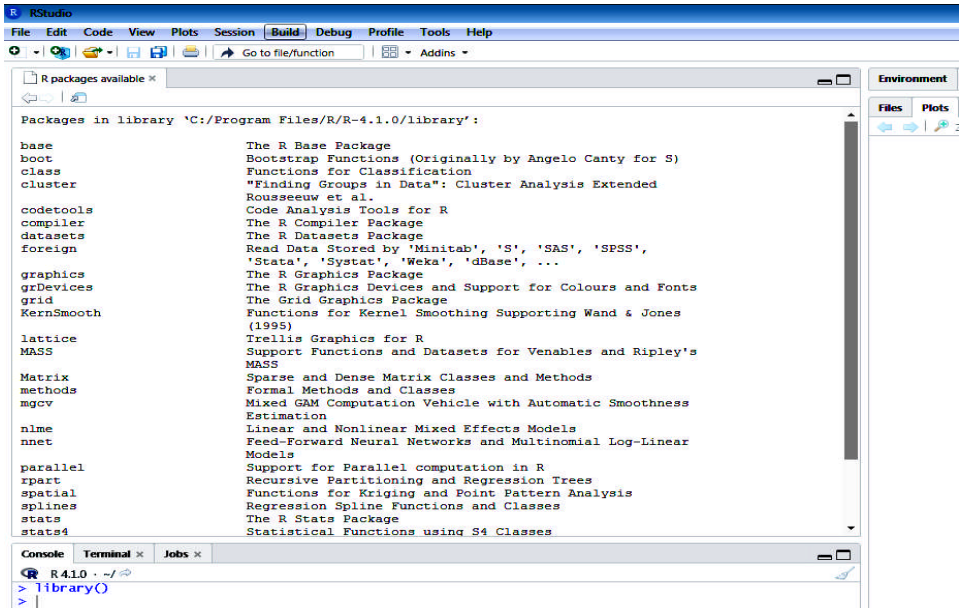
When we execute the above code, it will produce the following result.

```
> .libPaths()
[1] "C:/Program Files/R/R-4.1.0/library"
>
```

- **Get the list of all the packages installed**

```
library()
```

When we execute the above code, it will produce the following result.



R provides **search() function** to get all packages currently loaded in the R environment.

search()

When we execute the above code, it will produce the following result, which may vary depending on the local settings of our PCs and laptops:

```
> search()
 [1] ".GlobalEnv"        "tools:rstudio"     "package:stats"     "package:graphics"
 [5] "package:grDevices" "package:utils"     "package:datasets"  "package:methods"
 [9] "Autoloads"         "package:base"
>
```

❖ Install a New Package

In R, there are two ways to add new packages. One is install it directly from the CRAN directory and another is download the package to your local system and install it manually.

✓ Install directly from CRAN

The following command gets the packages directly from CRAN webpage and installs the package in the R environment.

install.packages("Package Name")

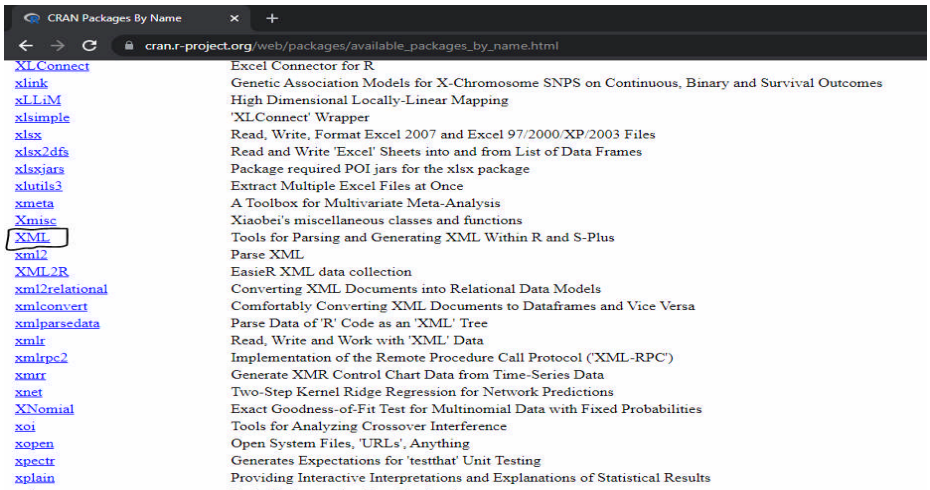
install.packages("XML") # Install the package named "XML".

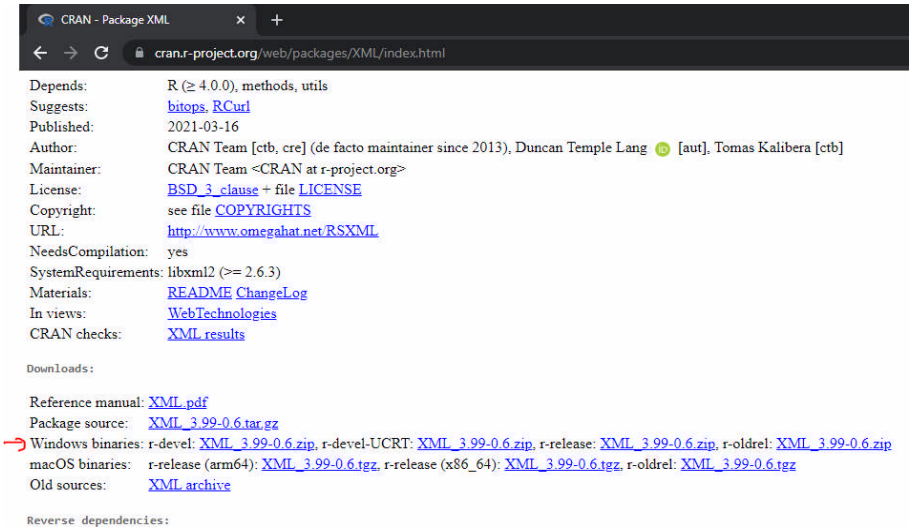
outpu

```
> install.packages("XML")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and i
nstall the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/XML_3.99-0.6.zip'
Content type 'application/zip' length 4261469 bytes (4.1 MB)
downloaded 4.1 MB

package 'XML' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpILcnsc\downloaded_packages
> |
```

Install package manually

To install a package manually, we first have to download it from https://cran.r-project.org/web/packages/available_packages_by_name.html. Save the package as a .zip file in a suitable location in the local system.



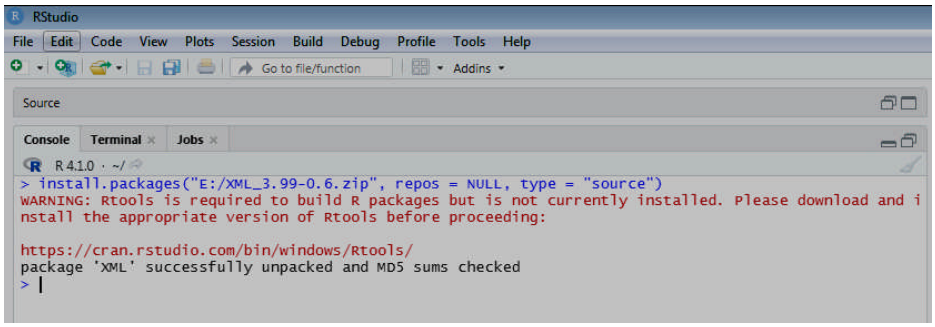


Once the downloading has finished, we will use the following command:

install.packages(file_name_with_path, repos = NULL, type = "source")

 # Install the package named "XML"

install.packages("E:/XML_3.99-0.6.zip", repos = NULL, type = "source")



❖ Load Package to Library

We cannot use the package in our code until its not loaded into the current R environment. We also need to load a package which is already installed but not available in the current environment.

A package is loaded using the following command –

library("package Name", lib.loc = "path to library")

 # Load the package named "XML"

install.packages("E:/XML_3.99-0.6.zip", repos = NULL, type = "source")

❖ **List of R packages**

R is the language of data science which includes a vast repository of packages. CRAN has 10,000 packages, making it an ocean of superlative statistical work. Most popular packages which are used in R as follows:

1) tidyr

The word tidyr comes from the word tidy, which means **clear**. **tidyr** package is used to make the data' tidy'.

2) ggplot2

R provides the **ggplot** package for creating graphics declaratively. This package is famous for its elegant and quality graphs which sets it apart from other visualization packages.

3) ggraph

R provides an extension of ggplot known as **ggraph**. The limitation of **ggplot** is the dependency on tabular data is taken away in ggraph.

4) dplyr

R provides the **dplyr** library for performing data wrangling and data analysis.This library facilitates several functions for the data frame in R.

5) tidyquant

The tidyquant is a financial package which is used for carrying out quantitative financial analysis. This package adds to the **tidyverse** universe as a financial package which is used for importing, analyzing and visualizing the data.

6) dygraphs

The dygraphs package provides an interface to the main JavaScript library which we can use for charting. This package is essentially used for plotting time-series data in R.

7) leaflet

For creating interactive visualization, R provides the **leaflet** package. This package is an open-source JavaScript library. The world's popular websites like the New York Times, Github and Flicker, etc. are using leaflet. The leaflet package makes it easier to interact with these sites.

8) ggmap

This is a mapping package that is used for delineating spatial visualizations. It also consists of various tools for geolocating and routing.

9) glue

R provides the **glue** package to perform the operations of data wrangling. This package is used for evaluating R expressions which are present within the string.

10) shiny

R allows us to develop interactive and aesthetically pleasing web apps by providing a **shiny** package. This package provides various extensions with HTML widgets, CSS, and JavaScript.

11) plotly

The plotly package provides online interactive and quality graphs. This package extends upon the JavaScript library **-plotly.js**.

12) dichromat

The R dichromat package is used to remove Red-Green or Blue-Green contrasts from the colors.

13) digest

The digest package is used for the creation of cryptographic hash objects of R functions.

14) caret

R allows us to perform classification and regression tasks by providing the caret package. **CaretEnsemble** is a feature of caret which is used for the combination of different models.

15) e1071

The **e1071** library provides useful functionsessential for data analysis like Naive Bayes, Fourier Transforms, SVMs, Clustering, and other miscellaneous functions.

16) sentimentr

The sentiment package provides functions for carrying out sentiment analysis. It is used to calculate text polarity at the sentence level and to perform aggregation by rows or grouping variable.

Questions:

1. Which of the following syntax is used to install forecast package in R?
    a) install.pack("forecast")
    b) installing.packages("cast")
    c) install.packages("forecast")
    d) install.pack["forecast"]

2. _____ is used to view all packages installed in R.
    a) library()
    b) search()
    c) .libPaths()
    d) stringr()

3. _____ function is used to get library location in R.
    a) library()
    b) search()
    c) .libPaths()
    d) stringr()

4. _____ function is used to view packages currently loaded.
    a) library()
    b) search()
    c) .libPaths()
    d) stringr()

Module VI

6

EXPERIMENT NO.1

Aim : Data preprocessing in R.

Objective: What is data preprocessing? Different steps involved in data preprocessing.

Theory:

❖ **Data Preprocessing**

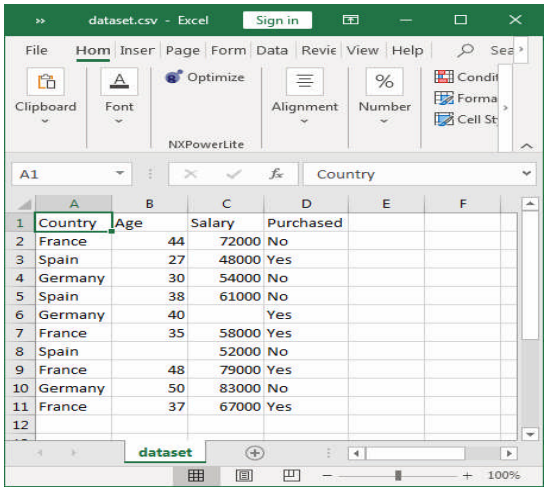
Data preprocessing is a process of preparing the raw data to make it suitable for a machine learning model. It is the first and crucial step while making any machine learning model.

When creating a machine learning model, it is not a case that we come across the clean and formatted data always. It is mandatory to clean the data and put it in a formatted way before using it for any model. So, for this we use data preprocessing.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is used for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

❖ **Data Preprocessing in R**

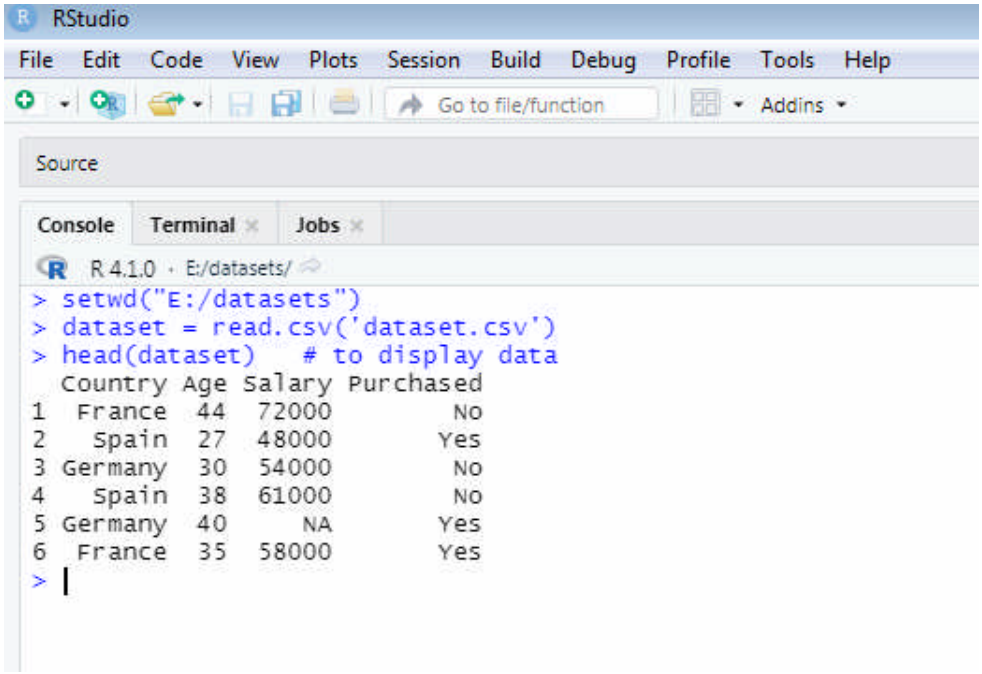


	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes
12				

dataset.csv file

❖ **Importing the Dataset**

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



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
>
```

This dataset consists of four features. The dependent factor is the **'Purchased'** column.

If the above dataset is to be used for machine learning model, the idea will be to predict if an item got purchased or not depending on the Country, Age and Salary of a person. The highlighted cells with value 'NA' denote missing values in the dataset.

❖ 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 and Salary).

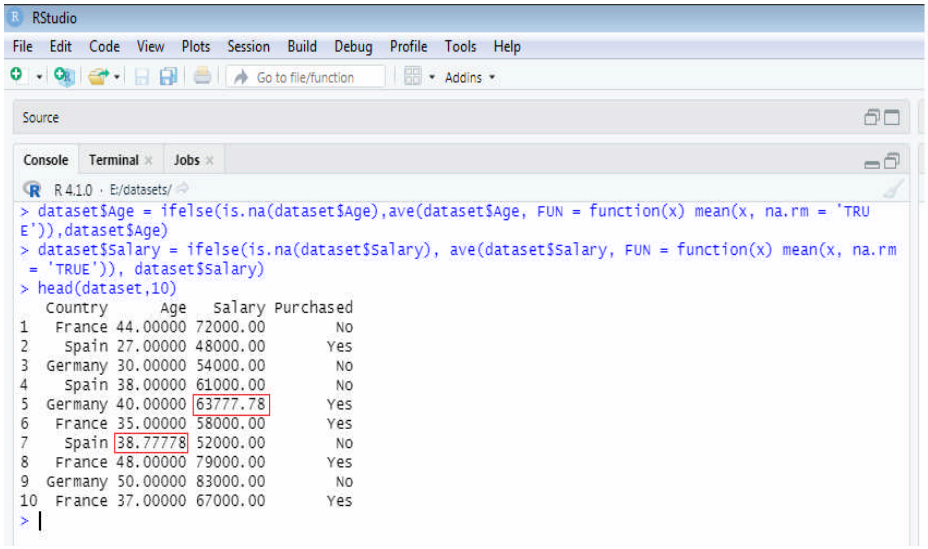
- **is.na(dataset$column_header):**

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

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

  This method calculates the average of the column passed as argument.

Output:



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, 0)))
```

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 72000.00        No
2    Spain  27 48000.00       Yes
3  Germany  30 54000.00        No
4    Spain  38 61000.00        No
5  Germany  40 63777.78       Yes
6   France  35 58000.00       Yes
7    Spain  39 52000.00        No
8   France  48 79000.00       Yes
9  Germany  50 83000.00        No
10  France  37 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$olumn_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
```

Questions:

1. Incorrect or invalid data is known as _____.
   a.Missing data b.Outlier c.Changing data d.Noisy data

2. What will be the output of the following R code?
   > x <- c(2, 6, NaN, NA, 4)
   > is.na(x)
   a) FALSE FALSE TRUE TRUE FALSE
   b) FALSE TRUE TRUE TRUE FALSE
   c) TRUE FALSE TRUE TRUE FALSE
   d) TRUE FALSE TRUE FALSE FALSE

3. _____is used for cleaning the data and making it suitable for a machine learning model.
   a. Data preprocessing
   b. Saving the data
   c. Data Repairing
   d. Data removing

❖❖❖❖

Module VII

EXPERIMENT NO. 1

Aim: To implement and analyse linear regression

Objective:- To understand linear regression which is a statistical model to study the relationship that could exist between two variable quantities : one of the variables is called independent variable(x) and the other is known as dependent variable(y).

Theory:

LINEAR REGRESSION

The independent and dependent variables are assumed to have a linear relationship in linear regression. This implies that a line can be drawn between the two.

The relation between two quantitative variables is estimated using simple linear regression. When you need to know the following, you can apply simple linear regression:

1)What is the extent of the association between the two variables?

2) The value of the dependent variable at a given value of the independent variable.

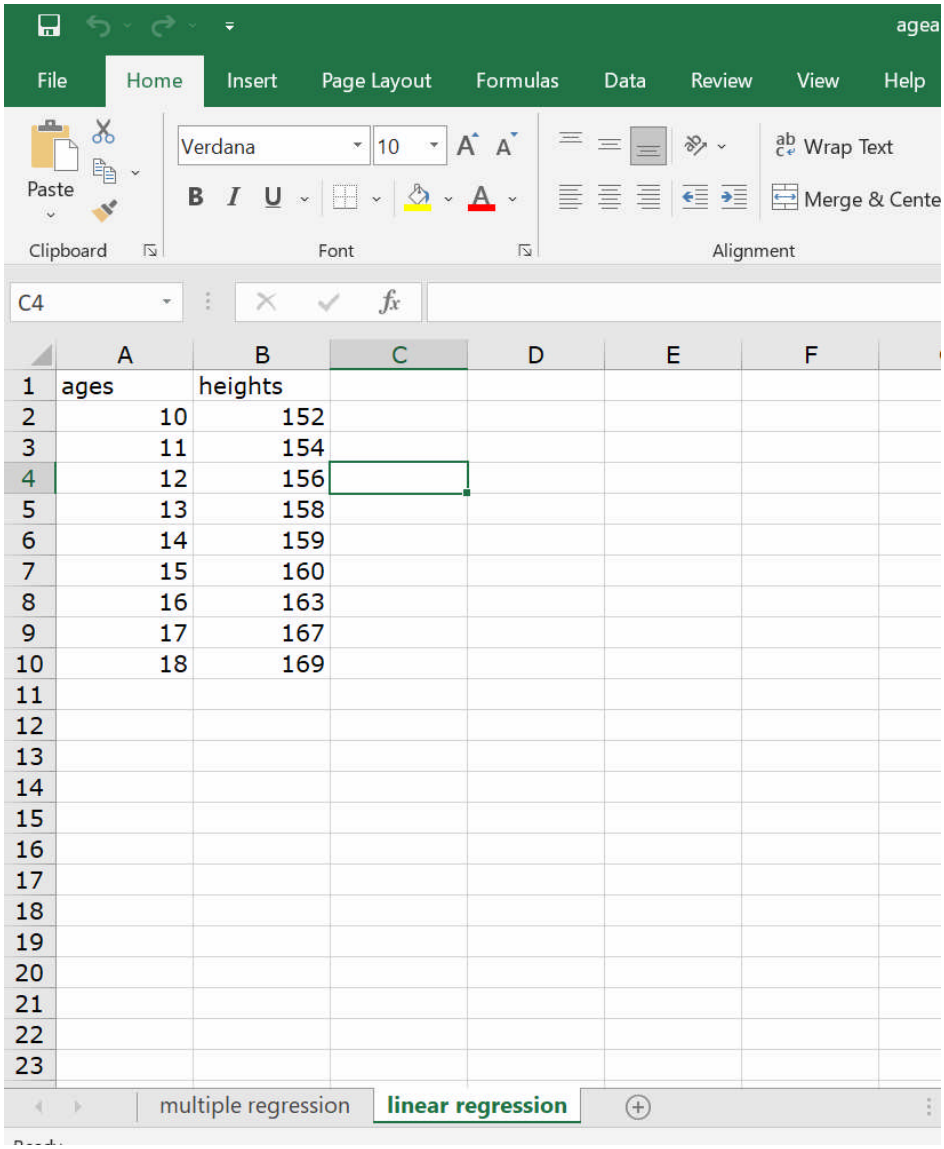
Program:

In this program, ages and heights of people are recorded in an excel file “ageandheight.xls” and the relationship between ages (independent variable) and heights(dependent variable) is studied.

This relationship between heights and ages can be expressed as a linear equation:

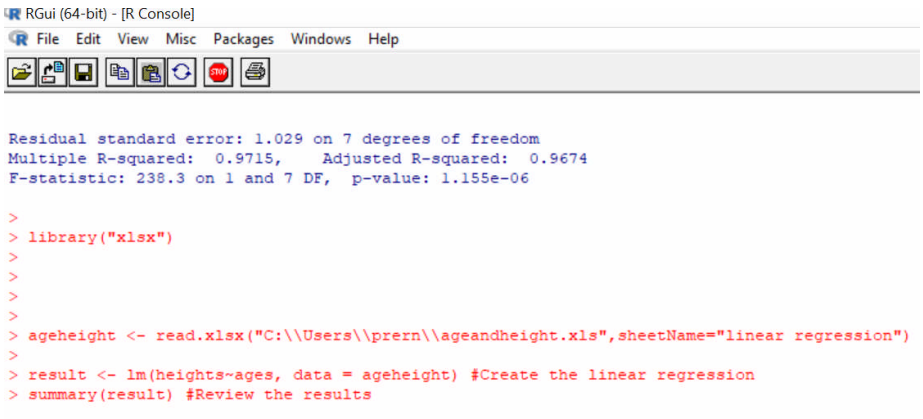
Heights = m*ages + c.

M is the slope of the line and c is the intercept.



	A	B	C	D	E	F
1	ages	heights				
2	10	152				
3	11	154				
4	12	156				
5	13	158				
6	14	159				
7	15	160				
8	16	163				
9	17	167				
10	18	169				
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

(ageandheight.xls file)



```
Residual standard error: 1.029 on 7 degrees of freedom
Multiple R-squared:  0.9715,    Adjusted R-squared:  0.9674
F-statistic: 238.3 on 1 and 7 DF,  p-value: 1.155e-06

>
> library("xlsx")
>
>
>
>
> ageheight <- read.xlsx("C:\\Users\\prern\\ageandheight.xls",sheetName="linear regression")
>
> result <- lm(heights~ages, data = ageheight) #Create the linear regression
> summary(result) #Review the results
```

Output :

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

Residuals:
    Min      1Q  Median      3Q     Max
-1.8278 -0.7778  0.3222  0.4222  1.0722

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 131.0778     1.8904   69.34 3.41e-11 ***
ages          2.0500     0.1328   15.44 1.15e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.029 on 7 degrees of freedom
Multiple R-squared:  0.9715,    Adjusted R-squared:  0.9674
F-statistic: 238.3 on 1 and 7 DF,  p-value: 1.155e-06
```

Residuals: The intention is for the sum of the residuals to be close to zero or as low as possible. Most cases will not follow a completely straight line in real life, hence residuals are always to be expected.

Coefficients: The values of the intercept (“c” value) and the slope (“m” value) for the age can be seen These “c” and “m” values are used to draw a line between all the points of the data.

So in this case, if there is a person whose age is 18, then the height can be calculated as (18*2.0500 + 131.0778)

The p-value for the age is 0.000001155. The smaller the value the better is ‘ages’ a good determinant of ‘heights’.

R^2 value is almost 1 for models that fit well and in case of models that poorly fit the data have R^2 value near about 0. In this output R squared value is 0.9715 which explains almost 97% of the variability.

Questionnaire

1. The number of variables used in linear regression is
   a) 2 b) 3 c) 4 d) 0

2. Linear regression is a _____ model.
   a) statistical b) non-statistical
   b) c) cannot say d) both

3. The smaller the p value for independent variable the better it is a predictor for dependent variable’s value
   a) True b) False c) Cannot Say d) All of the above

4. Generally, a _____ R squared value suggests a better fitting model.
   a) greater b) smaller c) both d)None

5. In linear regression, two variables form _____ relationship
   a) linear b) non-linear c) quadratic d) polynomial

6. If the sum of the residuals is zero, then it is the ideal scenario.
   a) True b) False c) Cannot Say

Experiment No. 02

Aim: To implement and analyse multiple linear regression

Objective:- To understand multiple linear regression which is a statistical model to study the relationship that could exist between variable quantities : here there are multiple independent variables(x1, x2, x3…..) to predict a dependent variable(y).

Theory:

MULTIPLE LINEAR REGRESSION

The independent and dependent variables are assumed to have a linear relationship in linear regression. This implies that a line can be drawn using them.

The relation between three or more quantitative variables is estimated using multiple linear regression. Analysts can use multiple linear regression to determine the model's variance and the relative contribution of each independent variable.

Multiple linear regression is used when one wants to know:

1. How strong the relationship is between two or more independent variables and one dependent variable (eg price of the houses and factors determining them like location,area etc)

2. The value of the dependent variable at a certain value of the independent variables (e.g. the expected price of a house at a particular value of location ,area etc).

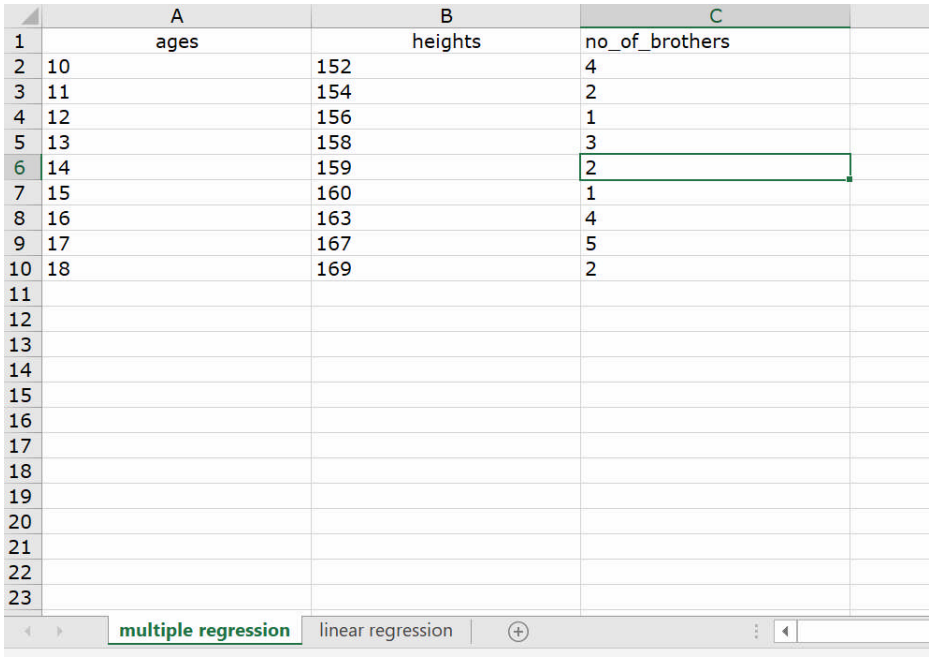
Program:

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.

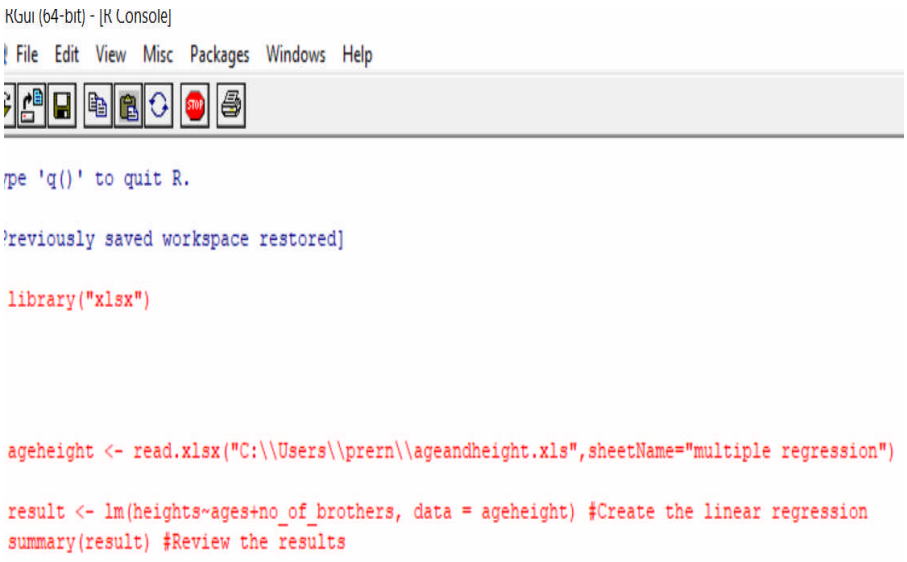


	A	B	C
1	ages	heights	no_of_brothers
2	10	152	4
3	11	154	2
4	12	156	1
5	13	158	3
6	14	159	2
7	15	160	1
8	16	163	4
9	17	167	5
10	18	169	2
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

multiple regression | linear regression

(ageandheight.xls file)

Program:



```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

ype 'q()' to quit R.

previously saved workspace restored]

library("xlsx")


ageheight <- read.xlsx("C:\\Users\\prern\\ageandheight.xls",sheetName="multiple regression")

result <- lm(heights~ages+no_of_brothers, data = ageheight) #Create the linear regression
summary(result) #Review the results
```

Output :

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

Residuals:
    Min      1Q  Median      3Q     Max
-1.9060 -0.4425  0.3123  0.4786  0.9631

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    131.3778     2.0791   63.19 1.06e-09 ***
ages             2.0437     0.1408   14.51 6.71e-06 ***
no_of_brothers  -0.1268     0.2439   -0.52    0.622
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.087 on 6 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.9727,    Adjusted R-squared:  0.9636
F-statistic: 106.9 on 2 and 6 DF,  p-value: 2.035e-05
```

Residuals: The intention is for the sum of the residuals to be close to zero or as low as possible. Most cases will not follow a completely straight line in real life, hence residuals are always to be expected.

Coefficients: The values of the intercept ("c" value) and the "m1" value for the ages and "m2" value for the number of brothers can be seen. These "c" and "m1,m2" values are used to draw a line between all the points of the data.

So in this case, if there is a person whose age is 18 and no of brothers is 2, then the height can be calculated as (18*2.0437 + 2* -0.1268 + 131.3778)

The p-value for the age is 0.00000671. The smaller the value the better is 'ages' a good determinant of 'heights'.

The p-value for the no_of _brothers is 0.622. It means there is a 62% chance that number of brothers is not a good determinant of 'heights'

R^2 value is almost 1 for models that fit well and in case of models that poorly fit the data have R^2 value near about 0. In this output R squared value is 0.9727 which explains almost 97% of the variability.

Questionnaire

1. The number of independent variables used in multiple linear regression is
   a) >=2 b) 1 c) 2 d) 0

2. Multiple Linear regression is an extension of _____ linear regression.
   a) simple b) complex c) cannot say d) both

3. The smaller the p value for independent variable the better it is a predictor for dependent variable's value
   a) True b) False c) Cannot Say d) All of the above

4. Generally, a _____ R squared value suggests a not so good fitting model.
   a) greater b) smaller c) both d)None

5. In multiple linear regression, quantitative variables form _____ relationship
   a) linear b)non-linear c)quadratic d) polynomial

6. If the sum of the residuals is not zero, then it is the ideal scenario.
   a) True b) False c) Cannot Say

Experiment No. 03

Aim: To implement and analyse Classification model - logistic regression.

Objective:- To understand logistic regression which is a classification algorithm that predicts categorical value of the response variable given multiple predictor variables values.

Theory:

LOGISTIC REGRESSION

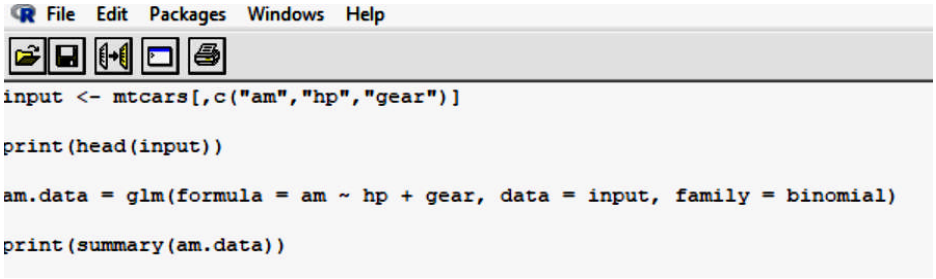
It is used to predict a binary result from a set of independent variables (Eg : 1 / 0, Yes / No, True / False,Male/Female). If the outcome variable is categorical, you may think of logistic regression as a particular instance of linear regression with the log of chances as the dependent variable. To put it another way, it forecasts the likelihood of something happening.

One may use logistic regression as an example of a classification approach to predict a qualitative response.

Program:

Mtcars is an inbuilt dataset in R. We have considered the attributes 'hp' i.e. gross horsepower of car engine and 'gear' i.e. number of forward gears in determining transmission mode i.e. 0 which means automatic and 1 which means manual. The variable "am" is categorical that is it can have only 2 values.

The glm() (generalised linear model) function creates the regression model and summary() function generates the summary for analysis.




```
input <- mtcars[,c("am","hp","gear")]

print(head(input))

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

print(summary(am.data))
```

Output:

```
> 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: 19.99

Number of Fisher Scoring iterations: 20
```

From the above observations, it is seen that both hp and gear values have p values more than 0.05. So it can be said that neither hp nor gear are significant in the logistic regression model.

Deviance is a metric of goodness of fit of a model. Larger numbers always mean it is a bad fit.

The number of Fisher Scoring iterations : It indicates the ideal number of iterations to fit the model. For example, beyond some number of iterations there is nothing to be practically gained.

Questionnaire

1. Logistic Regression is based on probability
   a) True b) False c)Both d)None

2. Logistic Regression is useful in case of qualitative data
   a) True b) False c) cannot say d) both

3. Logistic Regression measures the probability of ____ response
   a) Binary b) Tertiary c) Cannot Say d) All of the above

4. Logistic Regression algorithms are a part of class of_____ linear model.
   a) generalized b) particular c) both d)None

5. Predicting a qualitative response from an observation is _____ of observation.
   a) classification b) summarizing c) reducing the number d) None

6. Which one predicts probability?
   a) Linear Regression b) Logistic Regression c) Cannot Say

Experiment No. 04

Aim: To implement one classification algorithm in Weka.

Objective:- To implement classification algorithm K-Nearest Neighbour using WEKA

Theory:

KNN Algorithm

The K-Nearest Neighbour method is based on the Supervised Learning approach and is one of the most basic Machine Learning algorithms.

The K-NN method assumes that the new case/data and existing cases are comparable and places the new case in the category that is most similar to the existing categories.

The K-NN method saves all available data and classifies a new data point based on its similarity to the existing data.

It is called as lazy method because when you give the training data, it conducts no training at all. At training time, it does nothing except store the whole data set and does not do any calculations.

Algorithm:

Step-1: Select K number of the neighbours

Step-2: Compute the Euclidean distance of those K number of neighbours

Step-3: Take the K nearest neighbours according to the computed Euclidean distance.

Step-4: Between these k neighbours, count the number of the data points in each category.

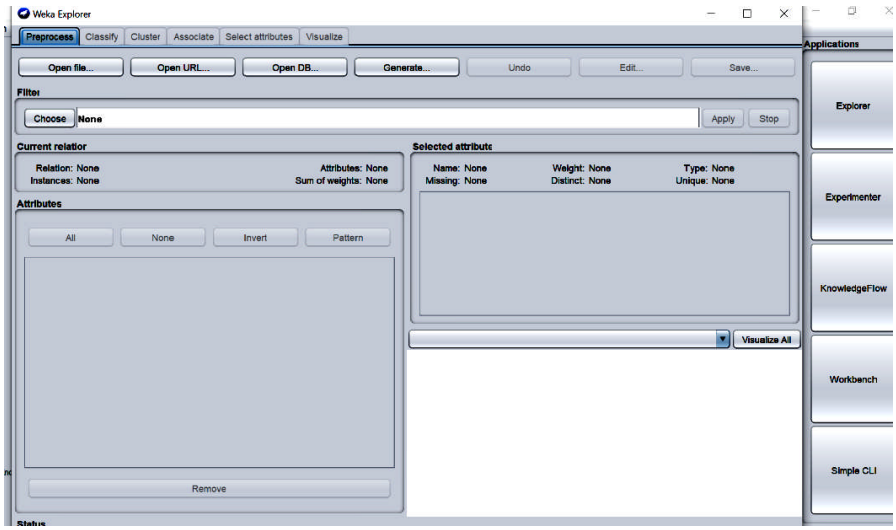
Step-5: Assign the new data to that category for which the number of the neighbours is the maximum value.

Program:

Steps :

1) Choose Explorer option

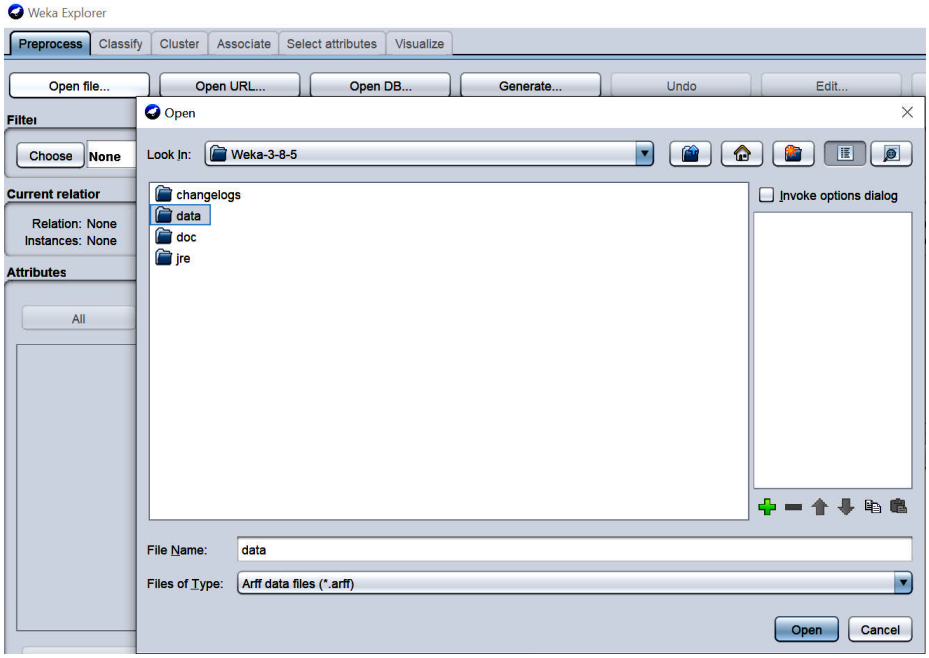
   The following Window will pop up



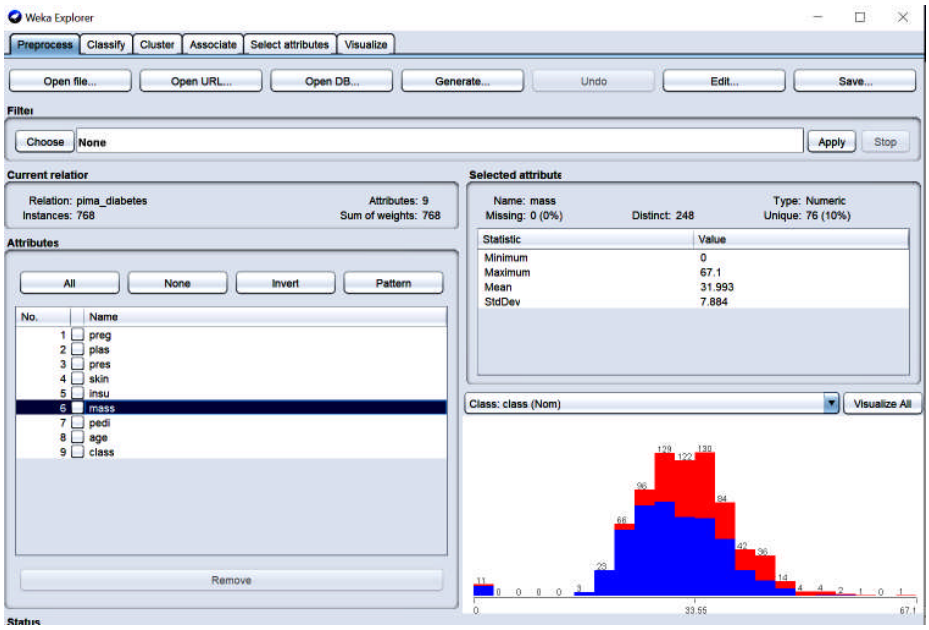
2) Select Open File

3) Select Drive name and choose Weka Folder

4) Select Data Folder

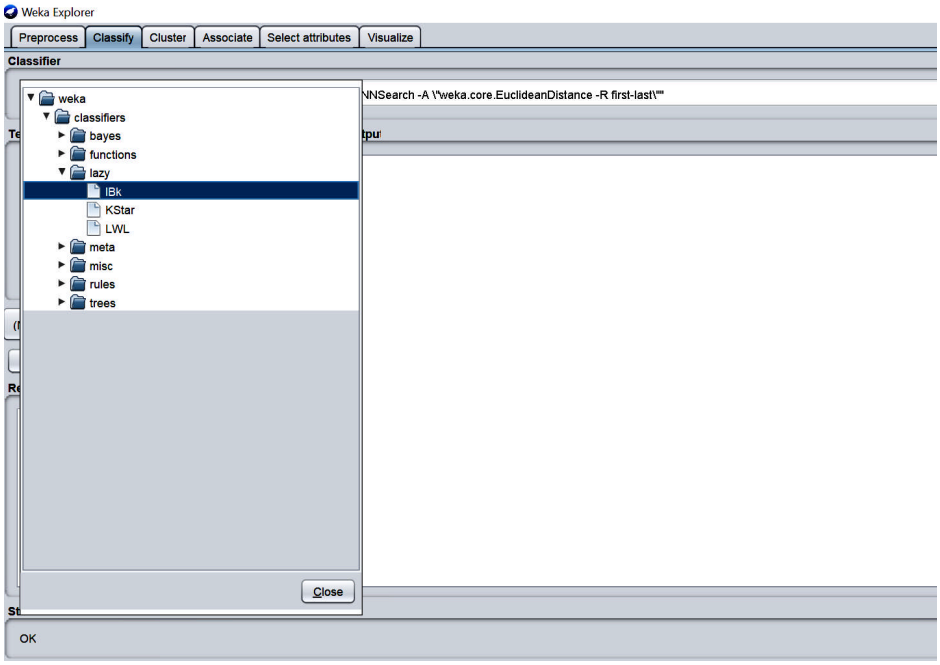


5) Select diabetes.arff file. The following window will be seen :

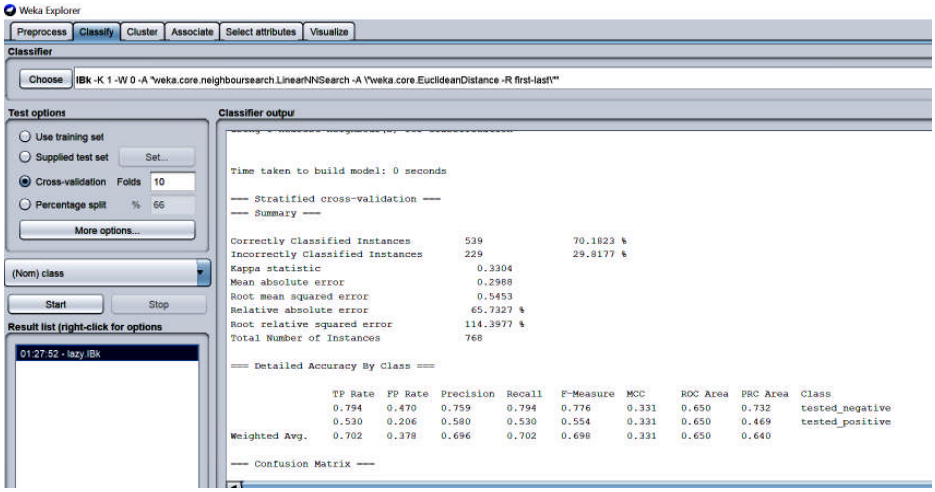


The relation or table name is pima_diabetes. It has 9 attributes or columns and 768 instances or rows. Every selected attribute has a statistic which is shown in the right part. Here, the selected attribute is mass --- which has values: Minimum – 0, Maximum 67.1 , Mean 31.993 ,Std. Deviation 7.884. All other attributes have their own statistics.

6) Choose Classify tab and then select Lazy option, then select IBk (Instance Based Learner).The IBk generates a prediction for all the rows in test dataset. Here it uses Euclidean Distance as a measure.



7) Click on Start . The following screen can be seen :



Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. Folds are the number of subsets that can be made. Here the number of folds is 10.

8) The following is the full output screen :

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanI
Relation:     pima_diabetes
Instances:    768
Attributes:   9
              preg
              plas
              pres
              skin
              insu
              mass
              pedi
              age
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds
```

```
=== Summary ===

Correctly Classified Instances         539               70.1823 %
Incorrectly Classified Instances       229               29.8177 %
Kappa statistic                          0.3304
Mean absolute error                      0.2988
Root mean squared error                  0.5453
Relative absolute error                 65.7327 %
Root relative squared error            114.3977 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.794    0.470    0.759      0.794    0.776      0.331    0.650     0.732     tested_negative
               0.530    0.206    0.580      0.530    0.554      0.331    0.650     0.469     tested_positive
Weighted Avg.  0.702    0.378    0.696      0.702    0.698      0.331    0.650     0.640

=== Confusion Matrix ===

   a   b   <-- classified as
 397 103 |   a = tested_negative
 126 142 |   b = tested_positive
```

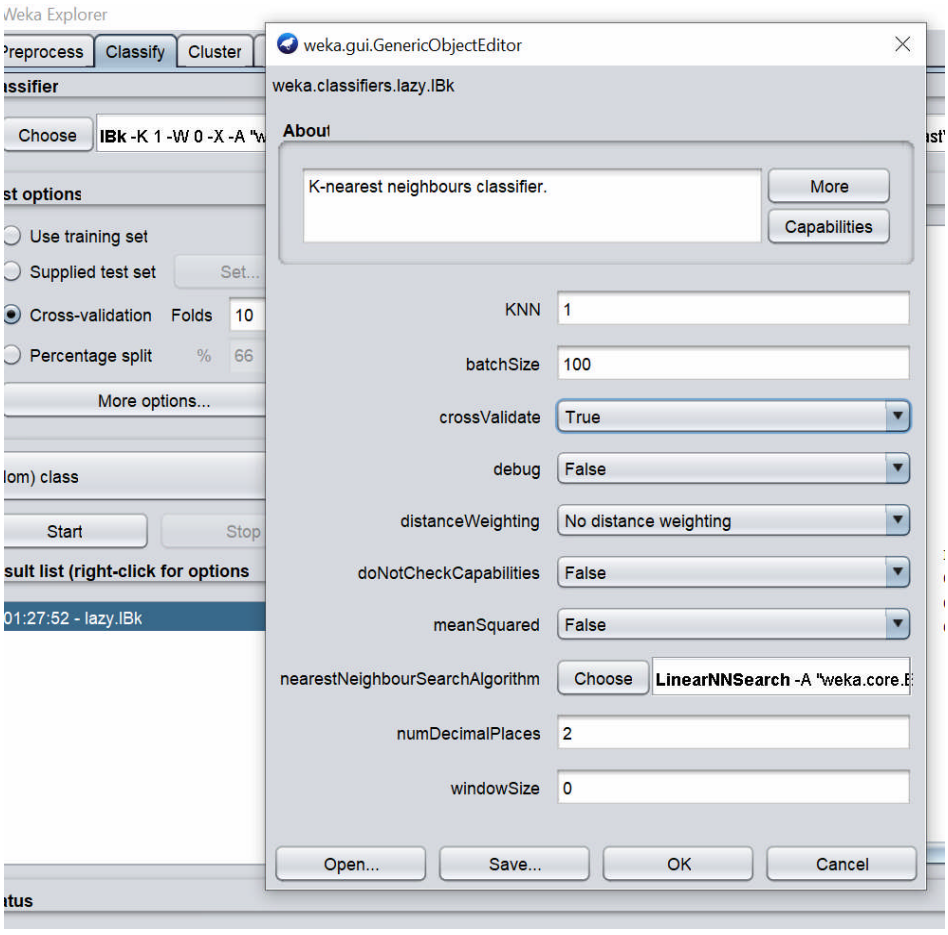
The correctly classified instances are 539 i.e. around 70% and the incorrectly classified instances are 229 i.e. 29.8177%.

Confusion matrix :

```
=== Confusion Matrix ===

   a   b   <-- classified as
 397 103 |   a = tested_negative
 126 142 |   b = tested_positive
```

The values 103 and 126 are False Negative and False Positive respectively.

9) By left clicking on IBk option near Choose tab, we get the following window:



Here the values for different options can be set like eg: KNN i.e. the number of neighbours can be set.

Questionnaire

1. In KNN algorithm K stands for number of _____
   a) neighbours b) errors c)Both d)None

2. Euclidean distance is the only measure of distance in KNN algorithm.
   a) True b) False c) cannot say d) both

3. WEKA stands for Waikato Environment for _____ Analysis
   a) Knowledge b) Kappa c) Cannot Say d) All of the above

4. KNN is _____ algorithm.
   a) lazy b) active c) both d)None

5. _____ can have many number of folds.
   a) Cross validation b) Standard deviation c) Absolute Error d) None

6. KNN is _____ machine learning algorithm
   a) Supervised b) Unsupervised c) Cannot Say d) Both

Module VIII

EXPERIMENT NO. 1

Aim: Implementation of market basket analysis.

Objective:- To implement market basket analysis in R

Theory:

Market Basket Analysis is a form of frequent itemset mining that examines consumer purchasing patterns by identifying relationships between the many goods in their "shopping baskets." By getting insight into which goods are commonly purchased together by customers, businesses may build marketing strategies based on the finding of these relationships.

Market Basket Analysis is a method of determining the value of a market basket.

MBA is most often used to help in cross-selling and up-selling. If you know that customers who buy trousers also buy belts, for example, you may advertise the belts on the same page or offer them as part of a bundle to try to boost sales. You may also advertise one product while seeing an increase in the other.

Customers' purchase patterns are depicted using "Association Rules" in Market Basket Analysis. A rule's interestingness is determined by two metrics: support and confidence.

Example:

Tea_powder => sugar [support = 4%, confidence = 70%]

a. A support of 2% for the above rule states that 2% of all the transaction under analysis show that tea powder and sugar are purchased together.

   support(B => C) = P(B U C)

b. A confidence of 70% means that 70% of the customers who purchased tea powder also bought the sugar.

c. Lift is a metric that helps us figure out if combining two products increases our chances of making a sale.

Packages/functions used:

1) arules

It is used for displaying, manipulating, and analysing transaction data and patterns (frequent item sets and association rules)

2) arulesViz

Extends package 'arules' with various visualization techniques for association rules and item sets.

3) inspect()

It summarizes all relevant options, plots and statistics that should be usually considered

4) is.redundant()

It finds redundant rules

5) apriori()

From a given collection of transaction data, apriori() creates the most relevant set of rules. It also demonstrates the rules' support, confidence, and lifting. The relative strength of the rules may be determined using these three criteria.

6)plot()

It is used to visualize association rules and item sets. It has in it implemented several popular visualization methods like scatter plots.

Algorithm:

The Apriori algorithm seeks out "often recurring item sets." An itemset is a collection of related items (such as products in a basket) whose frequency of co-occurrence is determined by a user-defined "support" level.

1) Read through the entire transaction.

2) Calculate the value of support for each item.

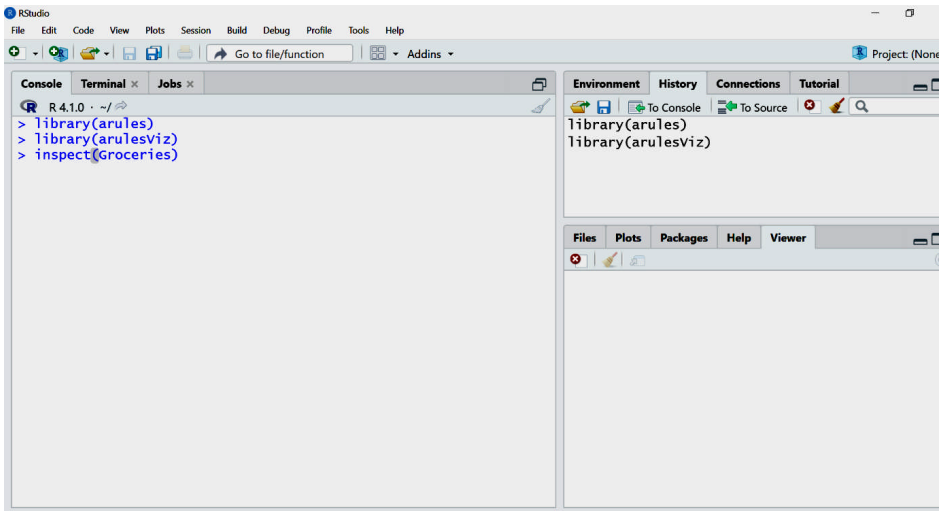
3) If the item's support is less than the minimum, it should be discarded. Otherwise, add it to the frequently used itemset.

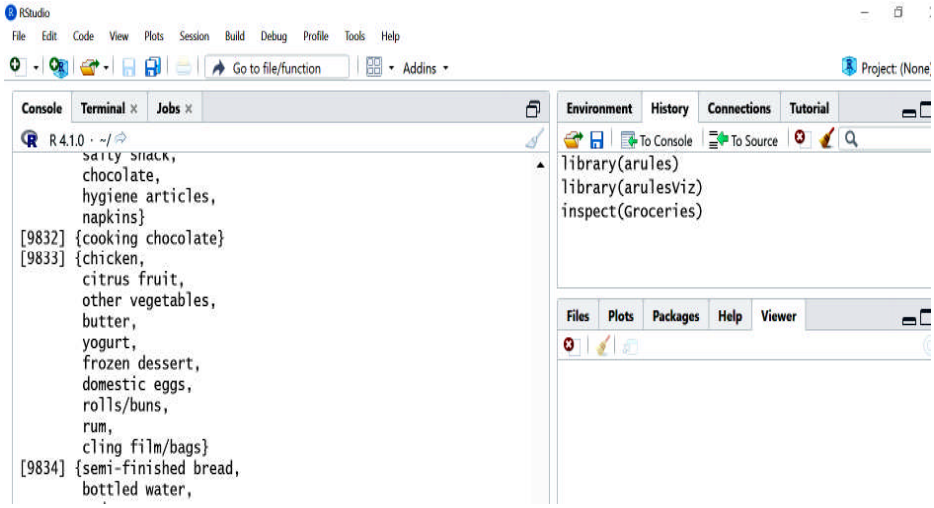
4) Determine the level of confidence for each non-empty subset.

If the confidence is less than the minimum, the subgroup should be discarded.

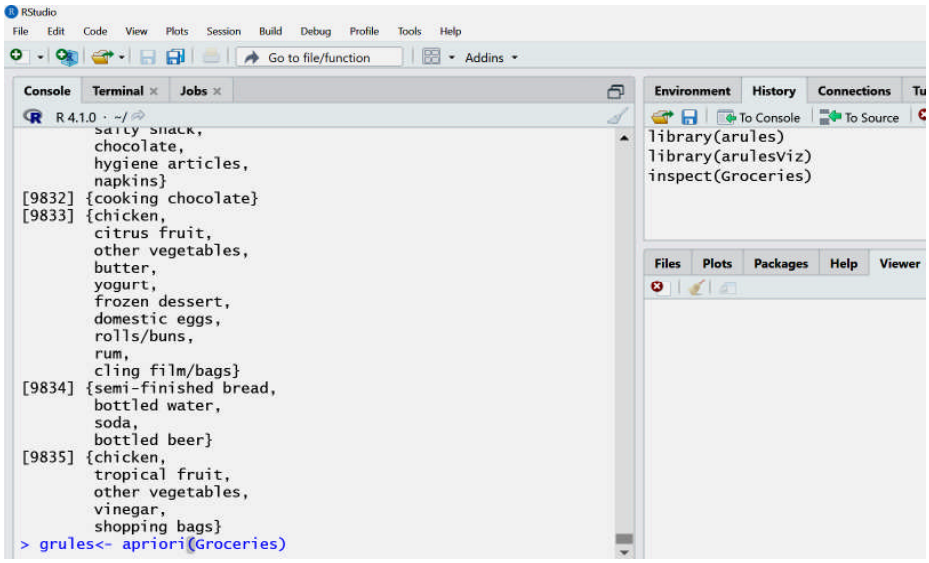
Program:

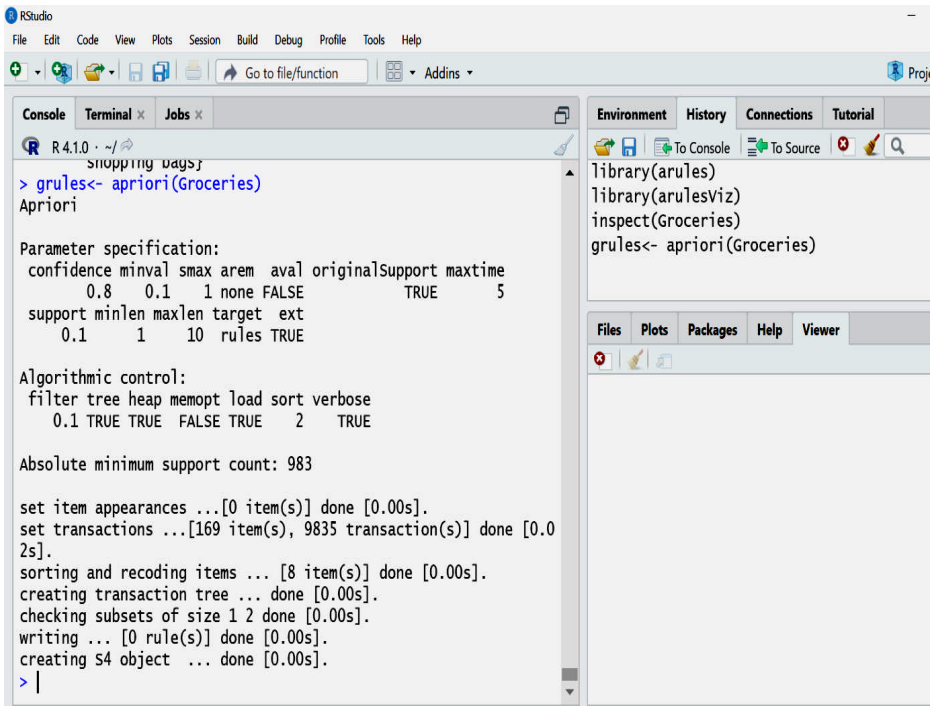
```
1) library(arules)

2) library(arulesViz)

3) inspect(Groceries)
```



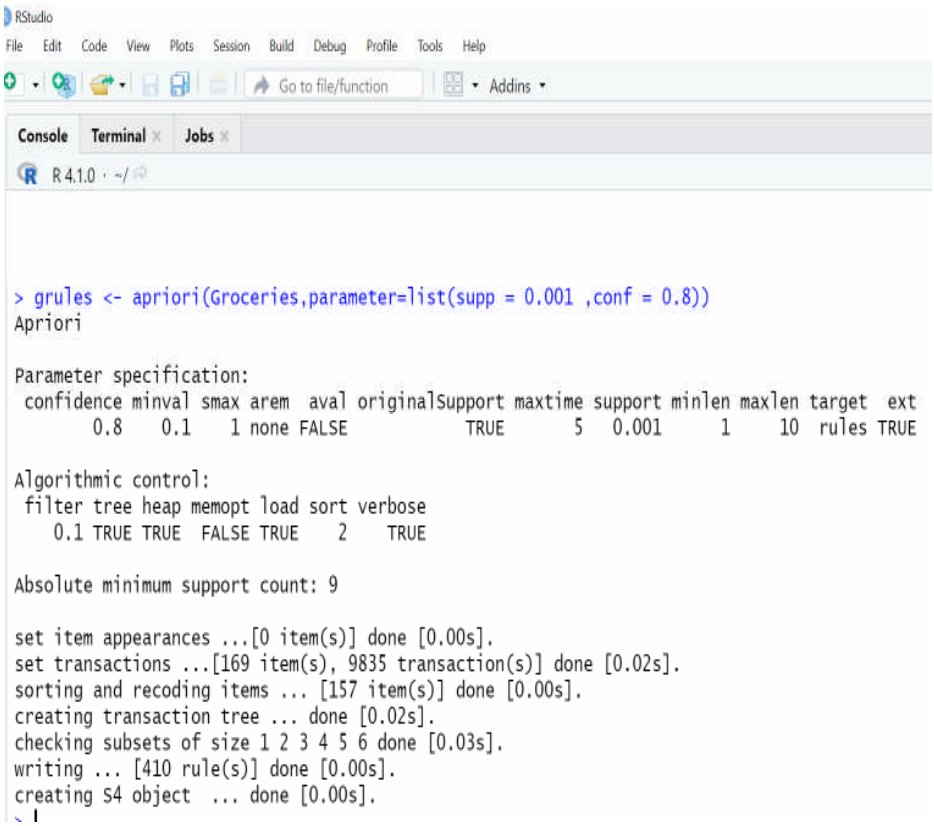


4) grules<-apriori(Groceries)

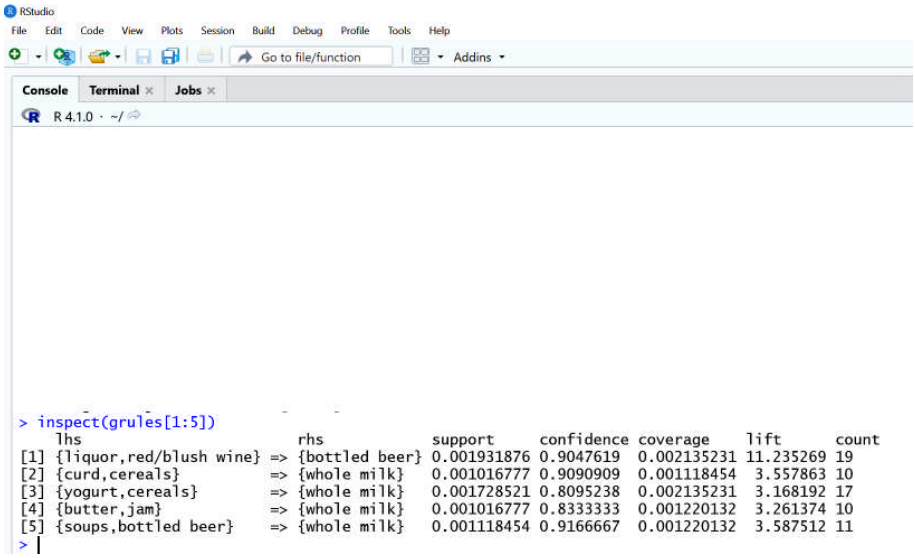




5) grules <-apriori(Groceries, parameter=list(supp = 0.001 , conf = 0.8))



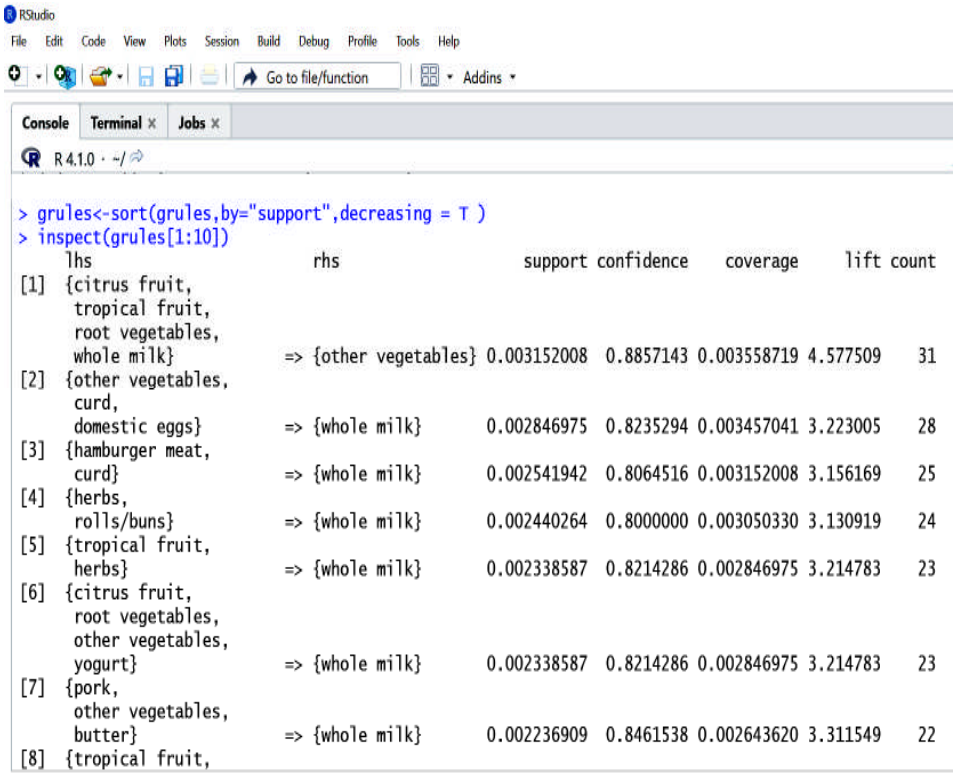
6) inspect(grules[1:5])



```
> inspect(grules[1:5])
    lhs                          rhs            support     confidence coverage    lift      count
[1] {liquor,red/blush wine} => {bottled beer} 0.001931876 0.9047619  0.002135231 11.235269 19
[2] {curd,cereals}          => {whole milk}   0.001016777 0.9090909  0.001118454  3.557863 10
[3] {yogurt,cereals}        => {whole milk}   0.001728521 0.8095238  0.002135231  3.168192 17
[4] {butter,jam}            => {whole milk}   0.001016777 0.8333333  0.001220132  3.261374 10
[5] {soups,bottled beer}    => {whole milk}   0.001118454 0.9166667  0.001220132  3.587512 11
>
```

7) grules <- sort (grules, by = ” support ”,decreasing = T)

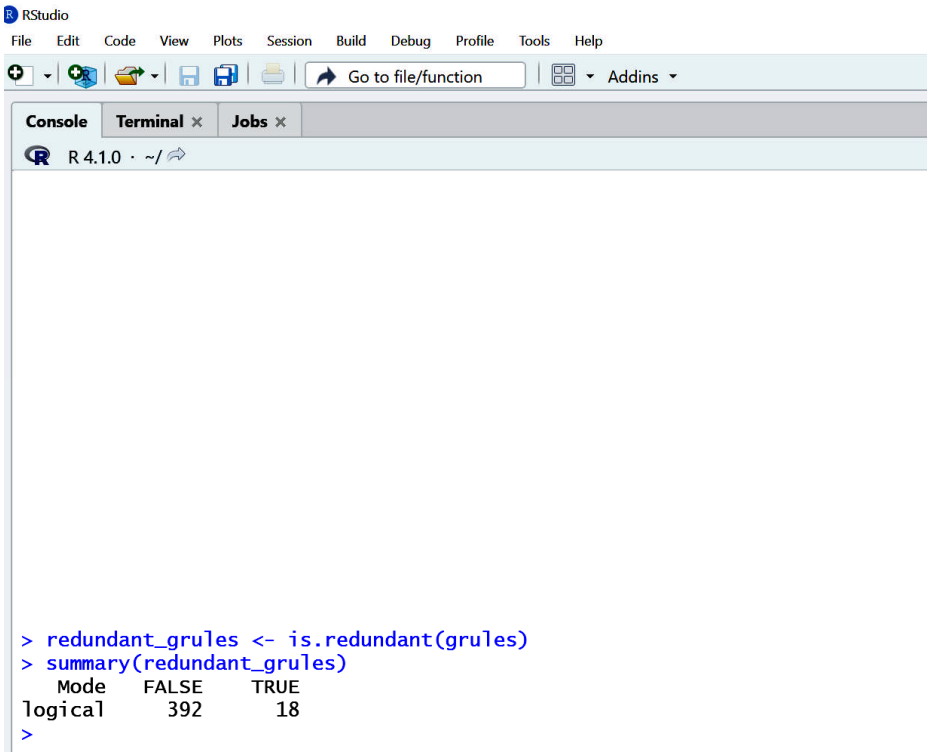
inspect(grules[1:10])



```
> grules<-sort(grules,by="support",decreasing = T )
> inspect(grules[1:10])
     lhs                        rhs                  support confidence    coverage     lift count
[1]  {citrus fruit,
      tropical fruit,
      root vegetables,
      whole milk}           => {other vegetables} 0.003152008  0.8857143 0.003558719 4.577509    31
[2]  {other vegetables,
      curd,
      domestic eggs}        => {whole milk}       0.002846975  0.8235294 0.003457041 3.223005    28
[3]  {hamburger meat,
      curd}                 => {whole milk}       0.002541942  0.8064516 0.003152008 3.156169    25
[4]  {herbs,
      rolls/buns}           => {whole milk}       0.002440264  0.8000000 0.003050330 3.130919    24
[5]  {tropical fruit,
      herbs}                => {whole milk}       0.002338587  0.8214286 0.002846975 3.214783    23
[6]  {citrus fruit,
      root vegetables,
      other vegetables,
      yogurt}               => {whole milk}       0.002338587  0.8214286 0.002846975 3.214783    23
[7]  {pork,
      other vegetables,
      butter}               => {whole milk}       0.002236909  0.8461538 0.002643620 3.311549    22
[8]  {tropical fruit,
```

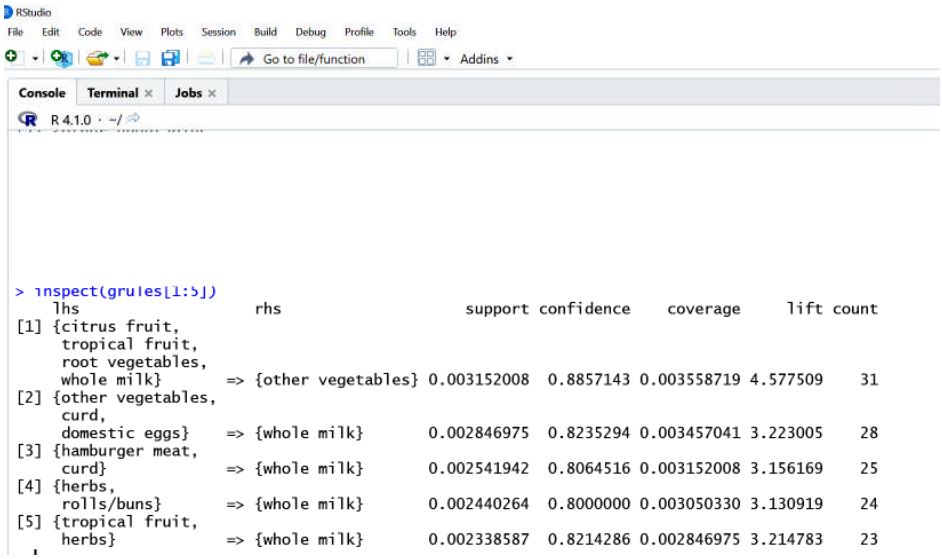
8) redundant_grules <- is.redundant(grules)

9) summary(redundant_grules)

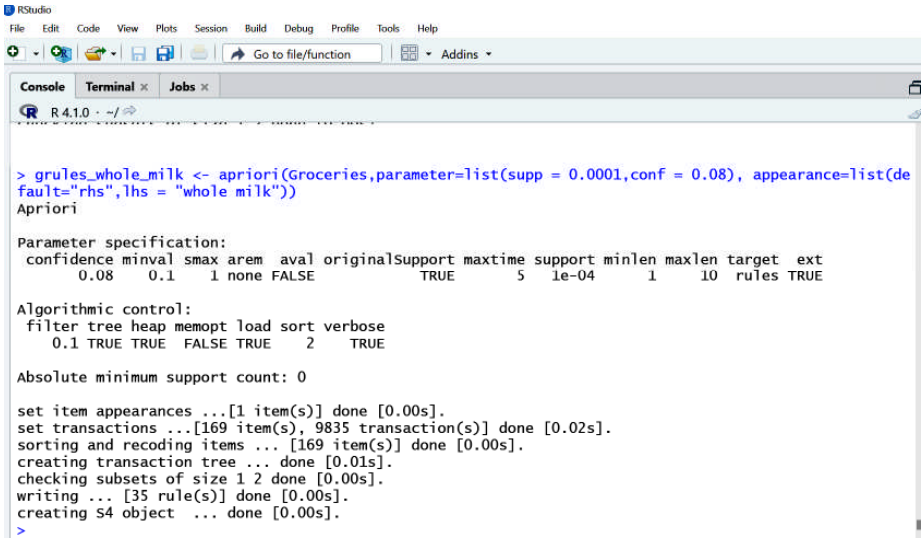


10) grules<-grules[!redundant_grules]

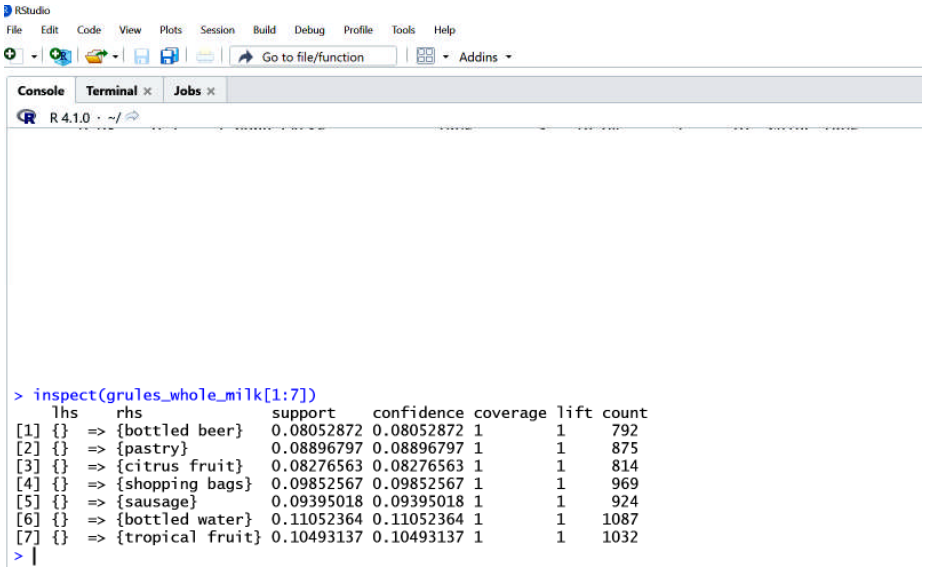
11) inspect(grules)



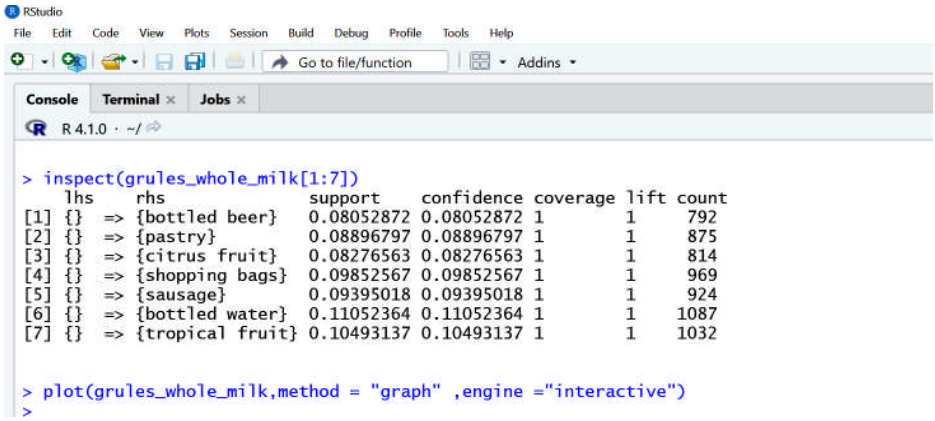
12) grules_whole_milk <- apriori(Groceries, parameter=list(supp = 0.0001,conf = 0.08), appearance=list(default="rhs", lhs = "whole milk"))



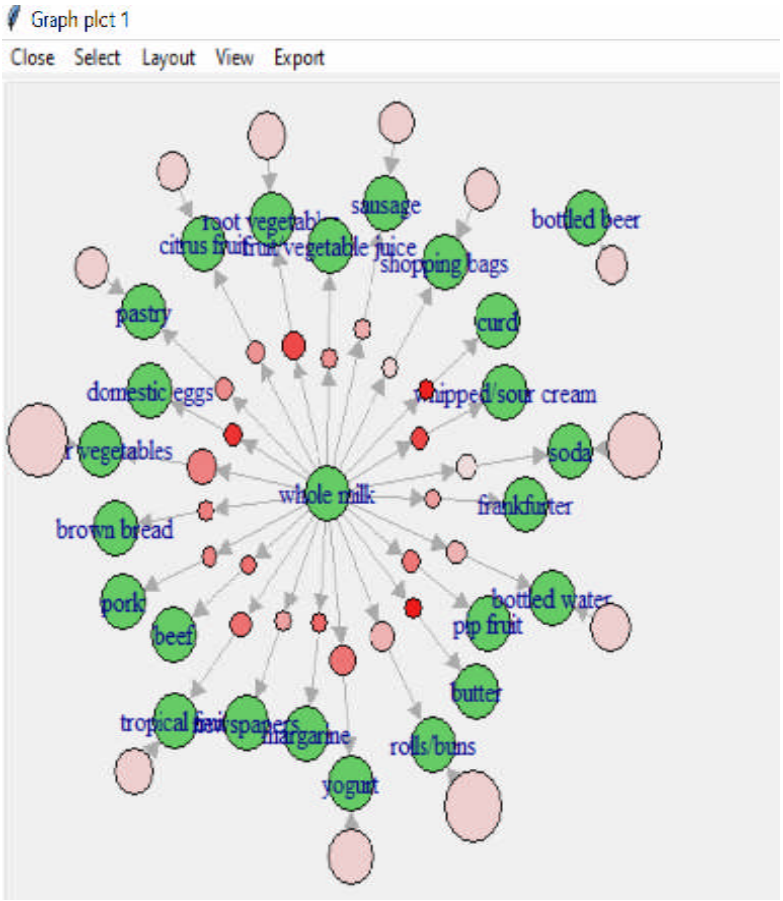
13) inspect(grules_whole_milk[1:7])



14) plot(grules_whole_milk, method = "graph" ,engine ="interactive")



Graph:



Questionnaire

1. Lift value _____ than 1 is desirable
   a)greater b) lesser c)Both d)None

2. Support means number of occurrences of a transaction.
   a) True b) False c) cannot say d) both

3. arulesViz package is used for _____ of association rules.
   a) visualization b) addition c) summarization d) All of the above

4. Apriori uses _____ knowledge of frequent itemset properties.
   a) prior b) output c) transaction d)None

5. Support and confidence measure interestingness of a rule.
   a)True b)False c)Both d) None

6. _____ is used when you want to find an association between different objects in a set.
   a) Association Rule Mining b) Clustering
   c) Classification d) None

Experiment No. 02

Aim: To implement K means Clustering in R.

Objective:- To understand how Kmeans clustering is performed in R

Theory:

K means for Clustering is an unsupervised learning method that attempts to group data by similarity. There is no outcome to anticipate with unsupervised learning, thus the algorithm just tries to discover patterns in the data. We must indicate the number of clusters we want the data to be classified into in k, which stands for clustering.

The algorithm allocates each observation to a cluster at random and finds the cluster centroid. After that, the algorithm repeats two steps:

Reassign data points to the cluster with the closest centroid.

Calculate each cluster's new centroid.

Packages/functions used:

1) cluster

   It is used in finding groups in data

2) set. seed()

   sets the starting number used to generate random numbers

3) kmeans()

   Perform k-means clustering on a data matrix.

4) plot()

   It is used for grouping items in terms of a smaller number of observed clusters.

5) table()

   Carries out categorical data tabulation with the variable and its frequency.

Algorithm:

a. Select the number of clusters K.

b. The centroids are chosen at random from a set of K locations (Not necessarily from the given data).

c. Assign each data point to the centroid that is closest to it, resulting in K clusters.

d. Calculate and position each centroid's new centroid.

e. Each data point should be assigned to a new cluster.

Program:

```
library(ClusterR)
library(cluster)
# Removing previous label of Species from the original Iris dataset
iris_new <- iris[, -5]
# Initializing seed value
set.seed(200)
kmeans.re <- kmeans(iris_new, centers = 3, nstart = 10)
kmeans.re
kmeans.re$cluster
# Building confusion matrix table
 confusion_matrix <- table(iris$Species, kmeans.re$cluster)
confusion_matrix
#Plotting the clusters
plot(iris_new[c("Sepal.Length", "Sepal.Width")],col =
kmeans.re$cluster,main = "Showing 3 different Clusters")
```

Output :

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

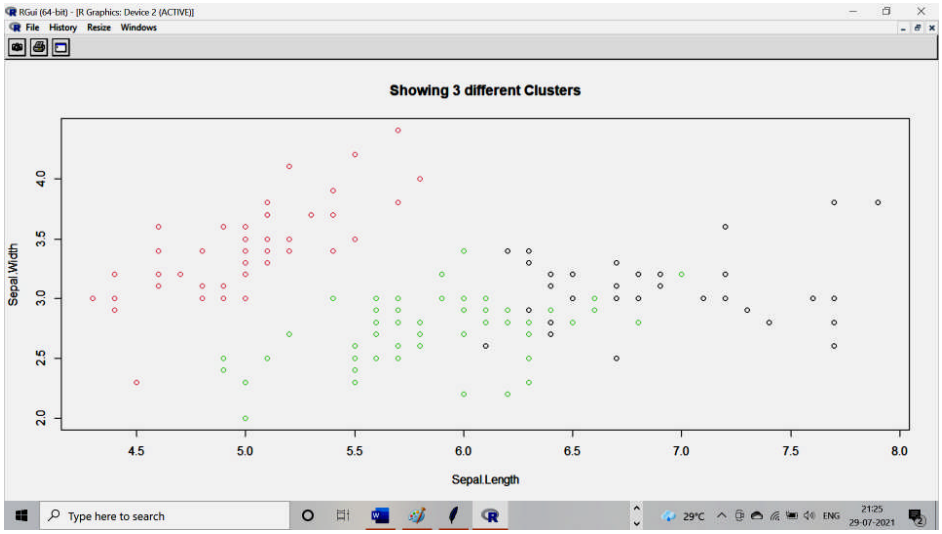
```
> library(ClusterR)
> library(cluster)
>
> # Removing previous label of Species from the original Iris dataset
>
> iris_new <- iris[, -5]
>
>  # Initializing seed value
> set.seed(200)
>
> kmeans.re <- kmeans(iris_new, centers = 3, nstart = 10)
> kmeans.re
K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
[112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1
[149] 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
 [75] 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
[112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1
[149] 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
>
> kmeans.re$cluster
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
[112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1
[149] 1 3
>
> # Building confusion matrix table
>
> confusion_matrix <- table(iris$Species, kmeans.re$cluster)
> confusion_matrix

              1  2  3
  setosa      0 50  0
  versicolor  2  0 48
  virginica  36  0 14
>
>
> #Plotting the clusters
>
> plot(iris_new[c("Sepal.Length", "Sepal.Width")],col = kmeans.re$cluster,main = "Showing 3 different Clusters")
>
```

Output:



Questionnaire

1. K Means Clustering is an _____ learning algorithm
   a)unsupervised b) supervised c)Both d)None

2. K is the number of clusters in a dataset.
   a) True b) False c) cannot say d) both

3. _____ learning means no outcome can be predicted, and the algorithm just tries to find patterns in the data.
   a) unsupervised b) supervised c)Both d)None

4. Iris dataset is inbuilt dataset.
   a) True b) False c) cannot say d) both

5. The more variation we have within clusters, the more similar the data points are within the same cluster.
   a)True b)False c)Both d) None

6. The groups of customers can be formed based on their behaviour ---- we can use k-means clustering for this task.
   a) True b)False c)Both d) None

❖❖❖❖