# Practical 1

## Aim: Implementation of Data partition using Range partitioning

```
SQL> create table sales_range2
  (
    salesman_id NUMBER(5),
   sales_name VARCHAR2(30),
  sales_amount NUMBER(10),
  sales_date DATE
  )
  PARTITION BY RANGE(sales_date)
  (
    PARTITION sales_jan2000 VALUES LESS THAN (TO_DATE('31/01/2000','DD/mm/yyyy')),
  PARTITION sales_feb2000 VALUES LESS THAN (TO_DATE('28/02/2000','DD/mm/yyyy')),
  PARTITION sales_mar2000 VALUES LESS THAN (TO_DATE('31/03/2000','DD/mm/yyyy')),
  PARTITION sales_apr2000 VALUES LESS THAN (TO_DATE('30/04/2000','DD/mm/yyyy'))
  );
Table created.
```

- **Inserting data in table**

```
SQL> insert into sales_range2 values(1,'RAM',7000,to_date('15/01/2000','DD/MM/YYYY'));
1 row created.

SQL> insert into sales_range2 values(2,'SAM',5500,to_date('23/02/2000','DD/MM/YYYY'));
1 row created.

SQL> insert into sales_range2 values(3,'ROHIT',6500,to_date('13/03/2000','DD/MM/YYYY'));
1 row created.

SQL> insert into sales_range2 values(4,'VINAY',4500,to_date('22/04/2000','DD/MM/YYYY'));
1 row created.
```

- **Displaying data from the table**

```
SQL> SELECT * FROM sales_range2;

SALESMAN_ID SALES_NAME                 SALES_AMOUNT SALES_DAT
----------- ------------------------------ ------------ ---------
          1 RAM                            7000 15-JAN-00
          2 SAM                            5500 23-FEB-00
          3 ROHIT                          6500 13-MAR-00
          4 VINAY                          4500 22-APR-00


SQL> SELECT * FROM sales_range2 partition(sales_jan2000);

SALESMAN_ID SALES_NAME                 SALES_AMOUNT SALES_DAT
----------- ------------------------------ ------------ ---------
          1 RAM                            7000 15-JAN-00
```

# Practical 2

## Aim: Implementation of Data partition using List partitioning

```
create table sales_list
 2  (
 3    salesman_id NUMBER(5),
 4    salesman_name VARCHAR2(30),
 5  sales_state varchar2(20),
 6  sales_amount NUMBER(10),
 7  sales_date DATE
 8  )
 9  PARTITION BY LIST(sales_state)
10  (
11    PARTITION sales_west values('california','hawaii'),
12    PARTITION sales_east values('New york','virginia','Florida'),
13    PARTITION sales_central values('texas','Illinios'),
14    PARTITION sales_other values(DEFAULT)
15  );
```

## Output:

```
SQL> select * from sales_list
  2  ;

SALESMAN_ID SALESMAN_NAME                 SALES_STATE          SALES_AMOUNT
----------- ----------------------------- -------------------- ------------
SALES_DAT
---------
          1 Rukhsar                       California                   2000
01-DEC-23

          2 Priyanka                      Florida                      4000
02-DEC-23

          3 Pooja                         New York                     6000
03-DEC-23


SALESMAN_ID SALESMAN_NAME                 SALES_STATE          SALES_AMOUNT
----------- ----------------------------- -------------------- ------------
SALES_DAT
---------
          4 Shalini                       Texas                        8000
04-DEC-23

          5 Sejal                         Texas                        4000
12-NOV-23
```

# Practical 3

## Aim: Implementation of Analytical Queries.

- **Creating a table**

```
SQL> create table student
(
    rollnumber number,
    name varchar(10),
    subject varchar(7),
   marks number
 );
```

Table created.

- **Inserting the data in table student**

```
SQL> insert into student values(10,'rohan','ADTA',67);
```

1 row created.

```
SQL>insert into student values(11,'Mohan','ADTA',57);
```

1 row created.

```
SQL> insert into student values(12,'han','ADTA',47);
```

1 row created.

```
SQL> insert into student values(13,'jay','SMS',47);
```

1 row created.

```
SQL> insert into student values(14,'Ajay','SMS',87);
```

1 row created.

```
SQL> insert into student values(15,'jayan','SMS',97);
```

1 row created.

```
SQL> insert into student values(15,'ram','java',17);
```

1 row created.

```
SQL> insert into student values(19,'sumit','java',27);
```

1 row created.

SQL> insert into student values(20,'zam','java',87);

1 row created.

SQL> insert into student values(14,'shan','ADTA',47);

1 row created.

- **Displaying the data from table student**

SQL> select * from student;

```
SQL> select * from student;

ROLLNUMBER NAME       SUBJECT      MARKS
---------- ---------- ------- ----------
        10 rohan      ADTA            67
        11 Mohan      ADTA            57
        12 han        ADTA            47
        13 jay        SMS             47
        14 Ajay       SMS             87
        15 jayan      SMS             97
        15 ram        java            17
        19 sumit      java            27
        20 zam        java            87
        14 shan       ADTA            47

10 rows selected.
```

- **Calculating minimum marks using dense rank order by and partition by**

SQL> select rollnumber,name,subject,marks,min(marks)keep(dense_rank first order by marks) over (partition by subject) as "lowest" from student order by subject,marks;

```
ROLLNUMBER NAME       SUBJECT      MARKS     lowest
---------- ---------- ------- ---------- ----------
        12 han        ADTA            47         47
        11 Mohan      ADTA            57         47
        10 rohan      ADTA            67         47
        13 jay        SMS             47         47
        14 Ajay       SMS             87         47
        15 jayan      SMS             97         47
        15 ram        java            17         17
        19 sumit      java            27         17
        20 zam        java            87         17
```

- **Calculating maximum marks using dense rank order by and partition by**

SQL> select rollnumber,name,subject,marks,max(marks)keep(dense_rank last order by marks) over (partition by subject) as "highest" from student order by subject,marks;

| ROLLNUMBER | NAME | SUBJECT | MARKS | highest |
|---|---|---|---|---|
| 12 | han | ADTA | 47 | 67 |
| 11 | Mohan | ADTA | 57 | 67 |
| 10 | rohan | ADTA | 67 | 67 |
| 13 | jay | SMS | 47 | 97 |
| 14 | Ajay | SMS | 87 | 97 |
| 15 | jayan | SMS | 97 | 97 |
| 15 | ram | java | 17 | 87 |
| 19 | sumit | java | 27 | 87 |
| 20 | zam | java | 87 | 87 |

- **Calculating maximum marks & minimum marks using dense rank order by and partition by**

SQL> select rollnumber,name,subject,marks,max(marks)keep(dense_rank last order by marks) over (partition by subject) as "highest" , min(marks)keep(dense_rank first order by marks) over (partition by subject) as "lowest" from student order by subject,marks;

| ROLLNUMBER | NAME | SUBJECT | MARKS | highest | lowest |
|---|---|---|---|---|---|
| 12 | han | ADTA | 47 | 67 | 47 |
| 11 | Mohan | ADTA | 57 | 67 | 47 |
| 10 | rohan | ADTA | 67 | 67 | 47 |
| 13 | jay | SMS | 47 | 97 | 47 |
| 14 | Ajay | SMS | 87 | 97 | 47 |
| 15 | jayan | SMS | 97 | 97 | 47 |
| 15 | ram | java | 17 | 87 | 17 |
| 19 | sumit | java | 27 | 87 | 17 |
| 20 | zam | java | 87 | 87 | 17 |

9 rows selected.

- **Calculating the rank using rank() and partition by**

SQL> select rollnumber, name , subject marks, rank() over(partition by subject order by marks) as "RANK" from student;

| ROLLNUMBER | NAME | MARKS | RANK |
|---|---|---|---|
| 12 | han | ADTA | 1 |
| 11 | Mohan | ADTA | 2 |
| 10 | rohan | ADTA | 3 |
| 13 | jay | SMS | 1 |
| 14 | Ajay | SMS | 2 |
| 15 | jayan | SMS | 3 |
| 15 | ram | java | 1 |
| 19 | sumit | java | 2 |
| 20 | zam | java | 3 |

- **Calculating the rank using rank() and partition by when marks are same**

SQL> select rollnumber, name , subject marks, rank() over(partition by subject order by marks) as "RANK" from student;

| ROLLNUMBER | NAME | MARKS | RANK |
|---|---|---|---|
| 12 | han | ADTA | 1 |
| 14 | shan | ADTA | 1 |
| 11 | Mohan | ADTA | 3 |
| 10 | rohan | ADTA | 4 |
| 13 | jay | SMS | 1 |
| 14 | Ajay | SMS | 2 |
| 15 | jayan | SMS | 3 |
| 15 | ram | java | 1 |
| 19 | sumit | java | 2 |
| 20 | zam | java | 3 |

- **Calculating the rank using dense_rank() and partition by when marks are same**

SQL> select rollnumber, name , subject marks, dense_rank() over(partition by subject order by marks) as "RANK" from student;

| ROLLNUMBER | NAME | MARKS | RANK |
|---|---|---|---|
| 12 | han | ADTA | 1 |
| 14 | shan | ADTA | 1 |
| 11 | Mohan | ADTA | 2 |
| 10 | rohan | ADTA | 3 |
| 13 | jay | SMS | 1 |
| 14 | Ajay | SMS | 2 |
| 15 | jayan | SMS | 3 |
| 15 | ram | java | 1 |
| 19 | sumit | java | 2 |
| 20 | zam | java | 3 |

## AIM : Creating a table to use lead and lag operation in the employee table

```
SQL>  create  table employee
 2    (
 3      empid number,
 4      empname varchar2(22),
 5      dept varchar2(22),
 6      salary number
 7    );
```

```
SQL>  create  table employee
  2     (
  3        empid number,
  4       empname varchar2(22),
  5       dept varchar2(22),
  6       salary number
  7     );

Table created.
```

- **Inserting data in table**

SQL> insert into employee values(1,'ram','marketing',10000);

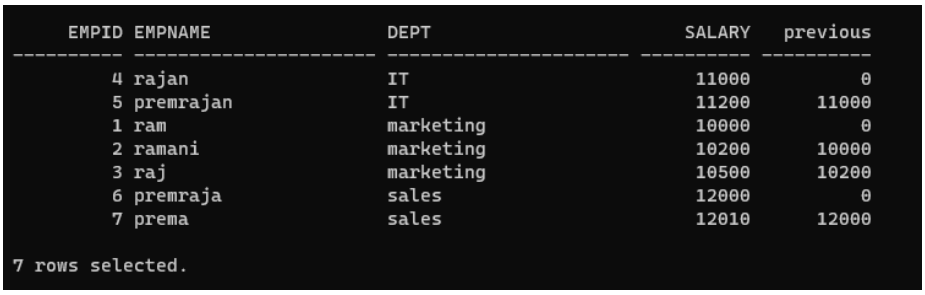
```
SQL> insert into employee values(1,'ram','marketing',10000);

1 row created.
```

- **Displaying data in table**

```
SQL> select * from employee;

     EMPID EMPNAME              DEPT                    SALARY
---------- -------------------- -------------------- ----------
         1 ram                  marketing                 10000
         2 ramani               marketing                 10200
         3 raj                  marketing                 10500
         4 rajan                IT                        11000
         5 premrajan            IT                        11200
         6 premraja             sales                     12000
         7 prema                sales                     12010
```

- **Performing LAG operation in the table employee**

SQL> select empid,empname,dept,salary,lag(salary,1,0) over (partition by dept order by salary) as "previous" from employee;

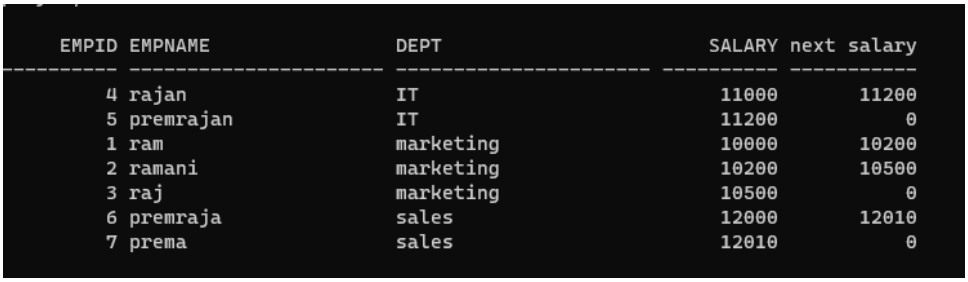


| EMPID | EMPNAME | DEPT | SALARY | previous |
|---|---|---|---|---|
| 4 | rajan | IT | 11000 | 0 |
| 5 | premrajan | IT | 11200 | 11000 |
| 1 | ram | marketing | 10000 | 0 |
| 2 | ramani | marketing | 10200 | 10000 |
| 3 | raj | marketing | 10500 | 10200 |
| 6 | premraja | sales | 12000 | 0 |
| 7 | prema | sales | 12010 | 12000 |

7 rows selected.

- **Performing LEAD operation in the table employee**

SQL> select empid,empname,dept,salary,lead(salary,1,0) over (partition by dept order by salary) as "next salary" from employee;



| EMPID | EMPNAME | DEPT | SALARY | next salary |
|---|---|---|---|---|
| 4 | rajan | IT | 11000 | 11200 |
| 5 | premrajan | IT | 11200 | 0 |
| 1 | ram | marketing | 10000 | 10200 |
| 2 | ramani | marketing | 10200 | 10500 |
| 3 | raj | marketing | 10500 | 0 |
| 6 | premraja | sales | 12000 | 12010 |
| 7 | prema | sales | 12010 | 0 |

# AIM: Operations on table student using Cube by function

- **Displaying the total marks of the same subject students using Cube by function**

SQL> select rollnumber,subject,SUM(marks)from student group by cube(rollnumber,subject)order by rollnumber,subject;

```
SQL> select rollnumber,subject,SUM(marks)from student group by cube(rollnumber,subject)order by rollnumber,subject;

ROLLNUMBER SUBJECT SUM(MARKS)
---------- ------- ----------
        10 ADTA            67
        10                 67
        11 ADTA            57
        11                 57
        12 ADTA            47
        12                 47
        13 SMS             47
        13                 47
        14 ADTA            47
        14 SMS             87
        14                134

ROLLNUMBER SUBJECT SUM(MARKS)
---------- ------- ----------
        15 SMS             97
        15 java            17
        15                114
        19 java            27
        19                 27
        20 java            87
        20                 87
           ADTA           218
           SMS            231
           java           131
                          580
```

- **Displaying the total marks of the same subject students using Roll up function**

SQL> select rollnumber,subject,SUM(marks)from student group by rollup(rollnumber,subject)order by rollnumber,subject;

```
ROLLNUMBER SUBJECT SUM(MARKS)
---------- ------- ----------
        10 ADTA            67
        10                 67
        11 ADTA            57
        11                 57
        12 ADTA            47
        12                 47
        13 SMS             47
        13                 47
        14 ADTA            47
        14 SMS             87
        14                134

ROLLNUMBER SUBJECT SUM(MARKS)
---------- ------- ----------
        15 SMS             97
        15 java            17
        15                114
        19 java            27
        19                 27
        20 java            87
        20                 87
                          580

19 rows selected.
```

# Practical 4

## AIM:- Implementation of ORDBMS (Abstract Data Types)

- **Creation of Type**

SQL> CREATE TYPE TypeName AS OBJECT

(

fname VARCHAR2(20),

mname VARCHAR2(20),

lname VARCHAR2(20)

);

/Type created.

```
SQL> CREATE TYPE TypeName AS OBJECT (
  2         fname VARCHAR2(20),
  3         mname VARCHAR2(20),
  4        lname VARCHAR2(20)
  5       );
  6       /

Type created.
```

create type type_address_1 as object

2 (street varchar(20),

3 city varchar(20),

4 pincode number(10)

5 );

6 /

Type created.

```
SQL> connect aniket/ani
Connected.
SQL> create type type_address_1 as object
  2  (street varchar(20),
  3   city varchar(20),
  4  pincode number(10)
  5  );
  6  /

Type created.
```

SQL> create table tb

2 (c_id number(5) primary key,

3 c_name typename,

4 c_add type_address_1,

5 c_phno number(10));

Table created.

```
SQL> create table tb
  2  (c_id number(5) primary ke
  3  c_name typename,
  4  c_add type_address_1,
  5  c_phno number(10));

Table created.
```

SQL> insert into tb

values(1,type_name_1('Arjun','V','Rajesh'),type_address_1('Vasai','Mumbai',412514),81280484

14);

1 row created.

```
SQL>  select * from tb ;

      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------
    C_PHNO
----------
         1
TYPENAME('Arjun', 'V', 'Rajesh')
TYPE_ADDRESS_1('Vasai', 'Mumbai', 412514)
8128048414
```

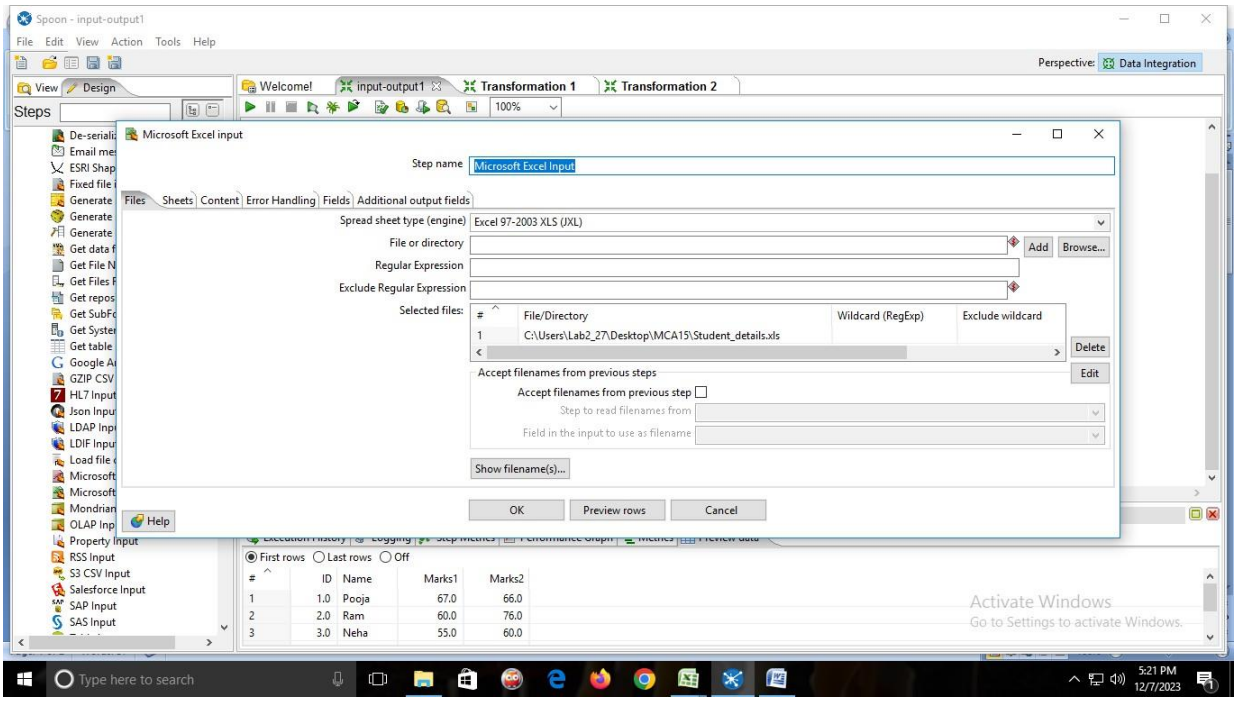
# PRACTCAL NO.5

## AIM: Implementation of extract transformation with Pentaho.

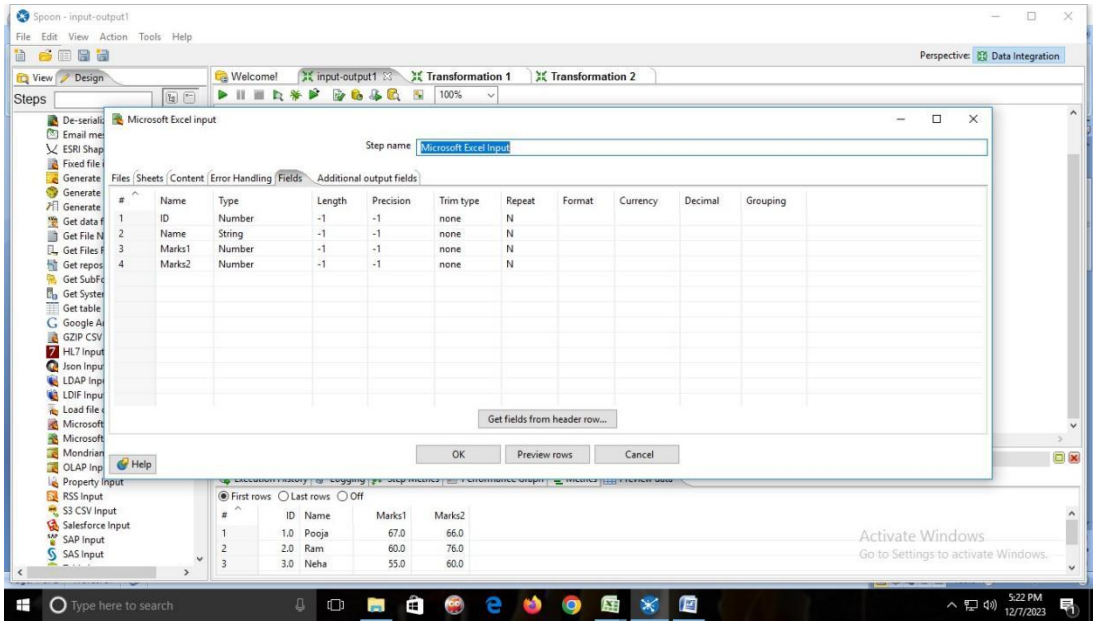
## A] calculator (table and Excel)

- **INPUT-OUTPUT**

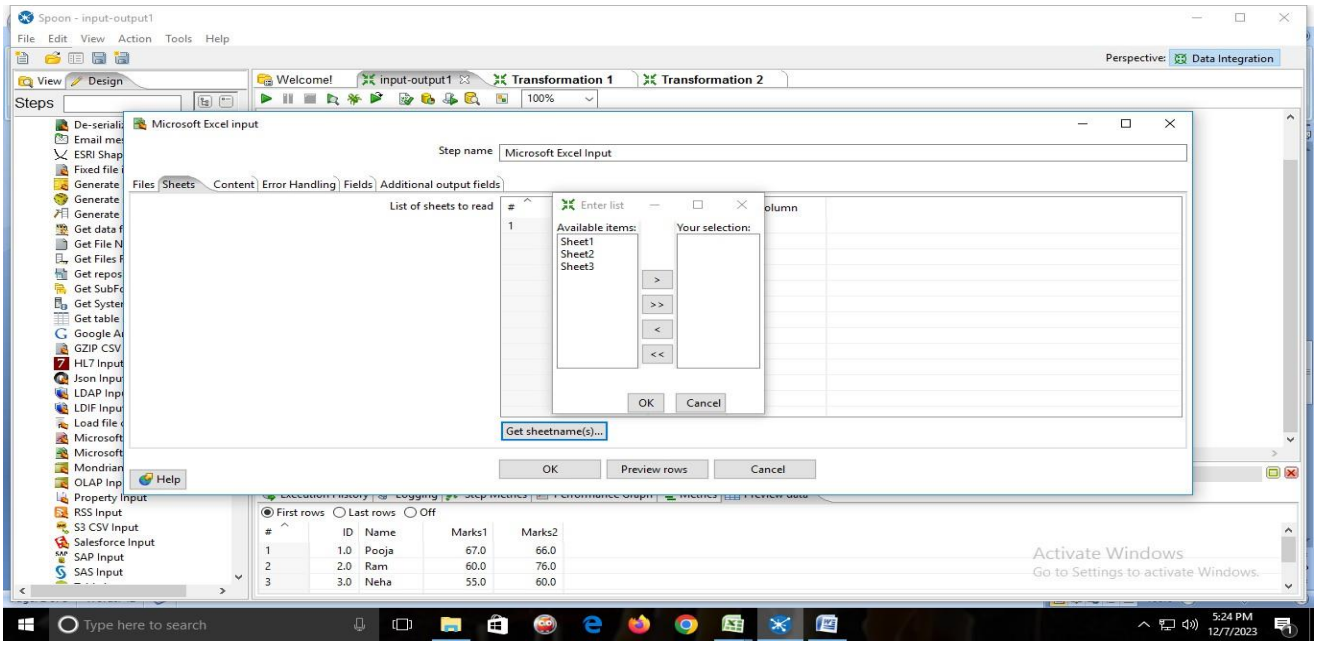
1) create an Excel file
2) In Pentaho click on transformation .
3) In design →input folder→ drag & drop → Microsoft Excel Input →double click on it→browse the excel file→add→fields→



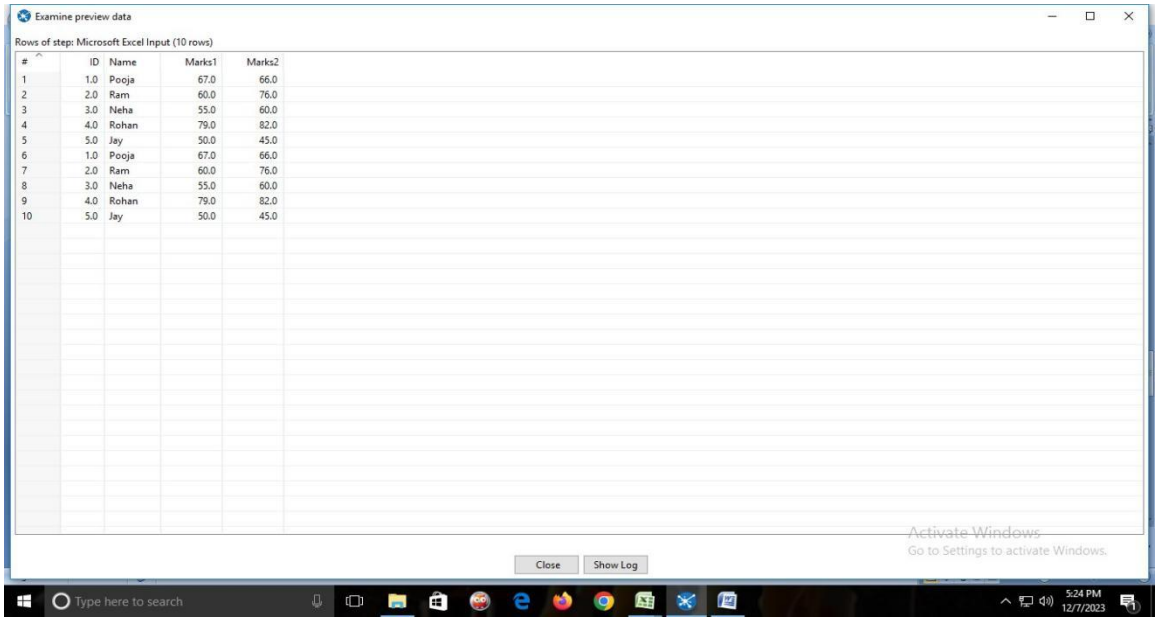
- click on get fields from header row



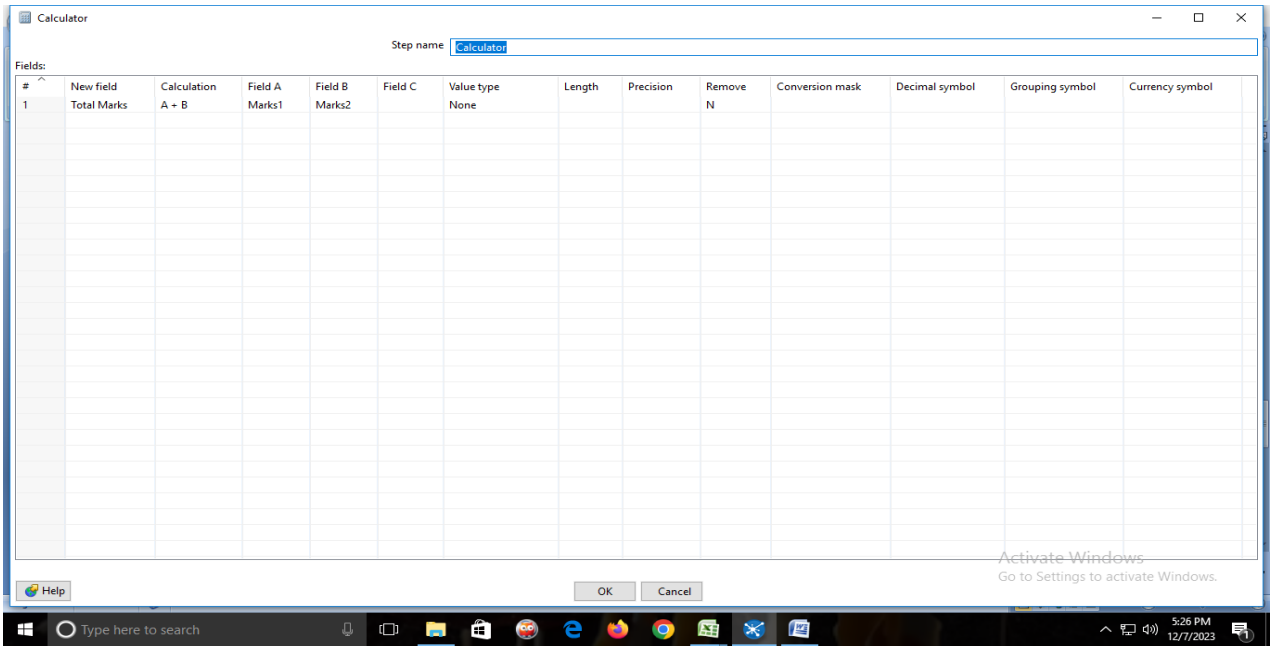

- then click on sheets→click on get sheetname→add sheet1→ok



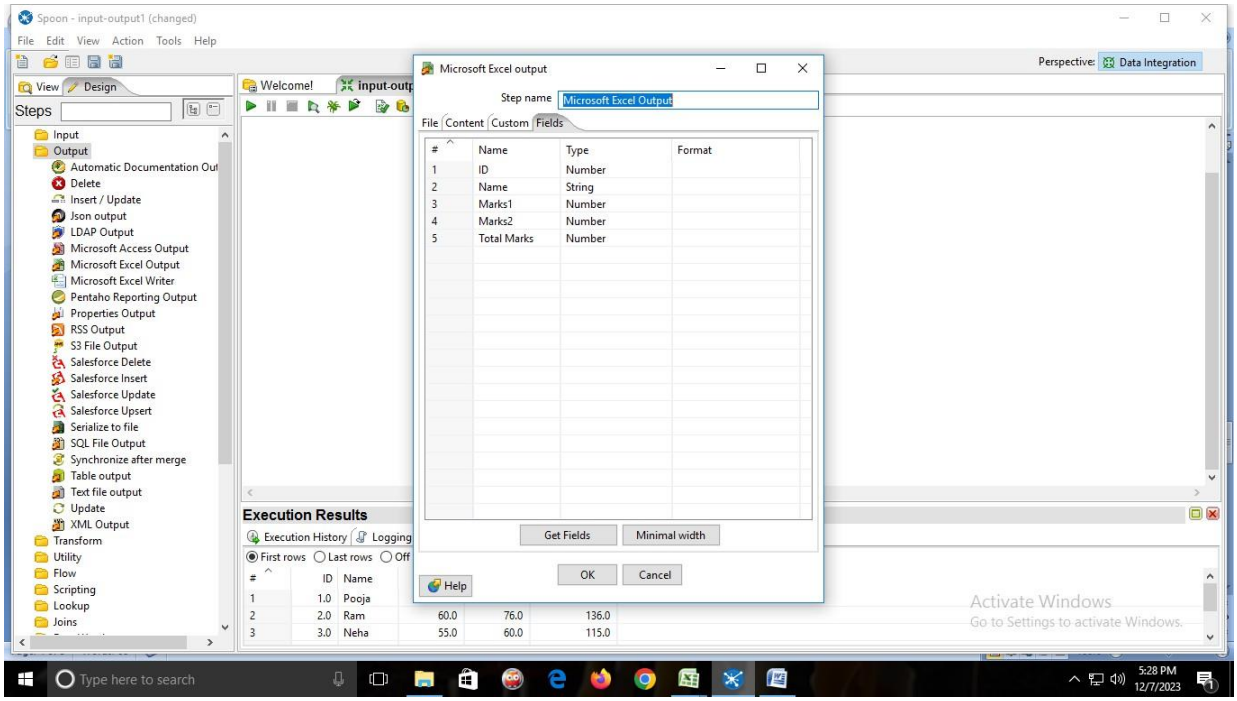
- click on Preview rows→Ok



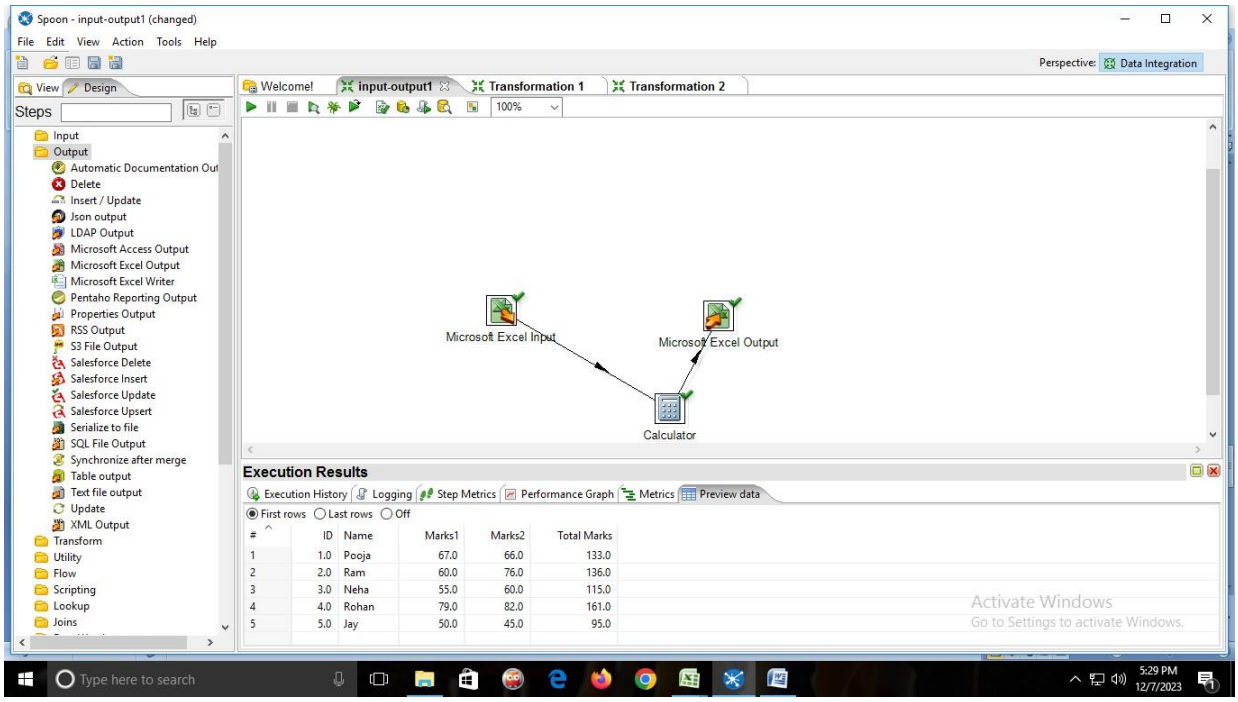

- In transform folder →drag & drop →calculator→double click and details as follow→click Ok



- In Output folder→drag & drop→Microsoft Excel Output→double click→fields→get fields→OK

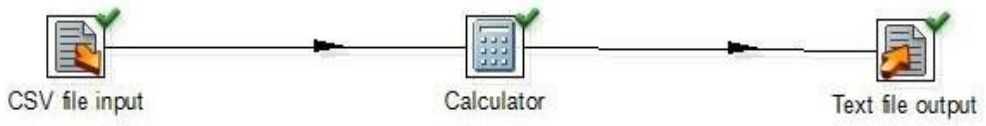


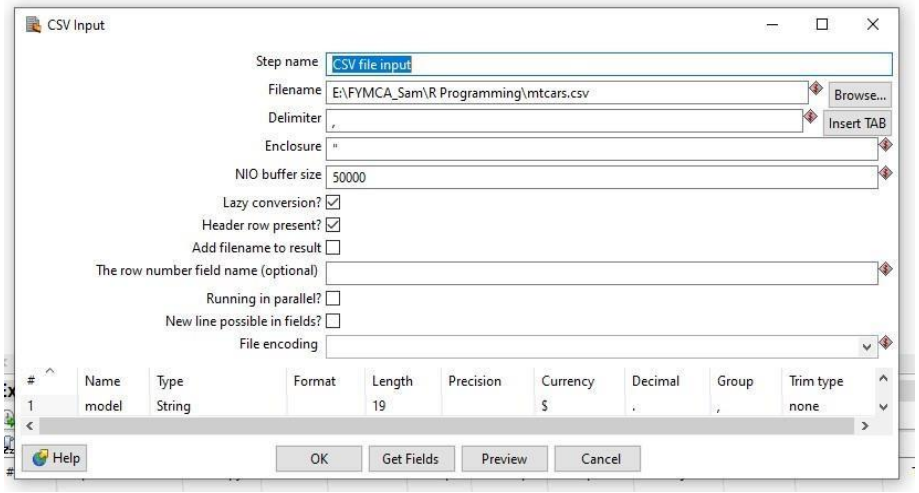

## OUTPUT:

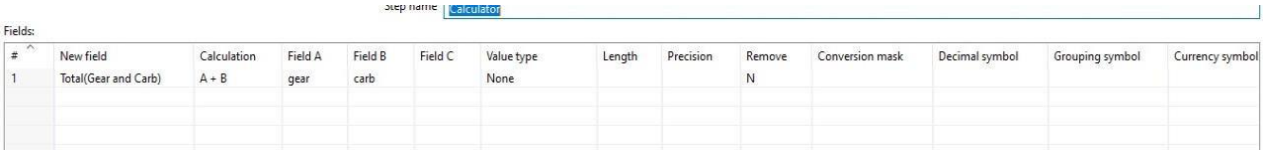


## Aim: Calculator using CSV input and text file.

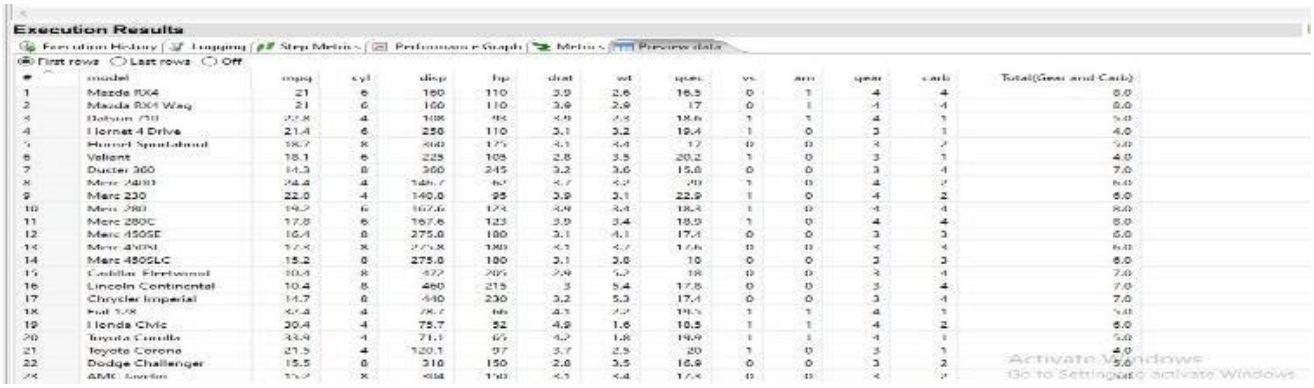
**STEP1:** Drag And Drop CSV file input, calculator and Text file output then give a connection then Double click on CSV file input browse existing csv file then click on get fields and then enter the field name, calculation=(A+B), enter the field name1 and field name 2.



**STEP2:** Launch the Transform then in the Execution Results click preview option to view the result the new column is added as shown in the output.




| # | model | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb | Total(Gear and Carb) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Mazda RX4 | 21 | 6 | 160 | 110 | 3.9 | 2.6 | 16.5 | 0 | 1 | 4 | 4 | 8.0 |
| 2 | Mazda RX4 Wag | 21 | 6 | 160 | 110 | 3.9 | 2.9 | 17 | 0 | 1 | 4 | 4 | 8.0 |
| 3 | Datsun 710 | 22.8 | 4 | 108 | 93 | 3.9 | 2.3 | 18.6 | 1 | 1 | 4 | 1 | 5.0 |
| 4 | Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.1 | 3.2 | 19.4 | 1 | 0 | 3 | 1 | 4.0 |
| 5 | Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.1 | 3.4 | 17 | 0 | 0 | 3 | 2 | 5.0 |
| 6 | Valiant | 18.1 | 6 | 225 | 105 | 2.8 | 3.5 | 20.2 | 1 | 0 | 3 | 1 | 4.0 |
| 7 | Duster 360 | 14.3 | 8 | 360 | 245 | 3.2 | 3.6 | 15.8 | 0 | 0 | 3 | 4 | 7.0 |
| 8 | Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.7 | 3.2 | 20 | 1 | 0 | 4 | 2 | 6.0 |
| 9 | Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.9 | 3.1 | 22.9 | 1 | 0 | 4 | 2 | 6.0 |
| 10 | Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.9 | 3.4 | 18.3 | 1 | 0 | 4 | 4 | 8.0 |
| 11 | Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.9 | 3.4 | 18.9 | 1 | 0 | 4 | 4 | 8.0 |
| 12 | Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.1 | 4.1 | 17.4 | 0 | 0 | 3 | 3 | 6.0 |
| 13 | Merc 450SL | 17.3 | 8 | 275.8 | 180 | 3.1 | 3.7 | 17.6 | 0 | 0 | 3 | 3 | 6.0 |
| 14 | Merc 450SLC | 15.2 | 8 | 275.8 | 180 | 3.1 | 3.8 | 18 | 0 | 0 | 3 | 3 | 6.0 |
| 15 | Cadillac Fleetwood | 10.4 | 8 | 472 | 205 | 2.9 | 5.2 | 18 | 0 | 0 | 3 | 4 | 7.0 |
| 16 | Lincoln Continental | 10.4 | 8 | 460 | 215 | 3 | 5.4 | 17.8 | 0 | 0 | 3 | 4 | 7.0 |
| 17 | Chrysler Imperial | 14.7 | 8 | 440 | 230 | 3.2 | 5.3 | 17.4 | 0 | 0 | 3 | 4 | 7.0 |
| 18 | Fiat 128 | 32.4 | 4 | 78.7 | 66 | 4.1 | 2.2 | 19.5 | 1 | 1 | 4 | 1 | 5.0 |
| 19 | Honda Civic | 30.4 | 4 | 75.7 | 52 | 4.9 | 1.6 | 18.5 | 1 | 1 | 4 | 2 | 6.0 |
| 20 | Toyota Corolla | 33.9 | 4 | 71.1 | 65 | 4.2 | 1.8 | 19.9 | 1 | 1 | 4 | 1 | 5.0 |
| 21 | Toyota Corona | 21.5 | 4 | 120.1 | 97 | 3.7 | 2.5 | 20 | 1 | 0 | 3 | 1 | 4.0 |
| 22 | Dodge Challenger | 15.5 | 8 | 318 | 150 | 2.8 | 3.5 | 16.9 | 0 | 0 | 3 | 2 | 5.0 |
| 23 | AMC Javelin | 15.2 | 8 | 304 | 150 | 3.1 | 3.4 | 17.3 | 0 | 0 | 3 | 2 | 5.0 |



CSVTransformation - Notepad

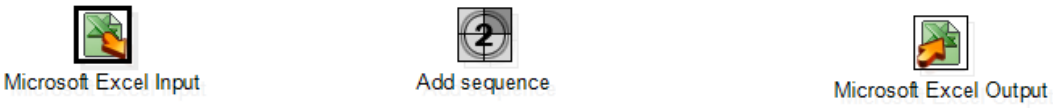
File Edit Format View Help

```
model;mpg;cyl;disp;hp;drat;wt;qsec;vs;am;gear;carb;Total(Gear and Carb)
Mazda RX4           ;21;6;160;110;3.9;2.6;16.5;0;1;4;4; 8.0
Mazda RX4 Wag       ;21;6;160;110;3.9;2.9;17;0;1;4;4; 8.0
Datsun 710          ;22.8;4;108;93;3.9;2.3;18.6;1;1;4;1; 5.0
Hornet 4 Drive      ;21.4;6;258;110;3.1;3.2;19.4;1;0;3;1; 4.0
Hornet Sportabout   ;18.7;8;360;175;3.1;3.4;17;0;0;3;2; 5.0
Valiant             ;18.1;6;225;105;2.8;3.5;20.2;1;0;3;1; 4.0
Duster 360          ;14.3;8;360;245;3.2;3.6;15.8;0;0;3;4; 7.0
Merc 240D           ;24.4;4;146.7;62;3.7;3.2;20;1;0;4;2; 6.0
Merc 230            ;22.8;4;140.8;95;3.9;3.1;22.9;1;0;4;2; 6.0
Merc 280            ;19.2;6;167.6;123;3.9;3.4;18.3;1;0;4;4; 8.0
Merc 280C           ;17.8;6;167.6;123;3.9;3.4;18.9;1;0;4;4; 8.0
Merc 450SE          ;16.4;8;275.8;180;3.1;4.1;17.4;0;0;3;3; 6.0
Merc 450SL          ;17.3;8;275.8;180;3.1;3.7;17.6;0;0;3;3; 6.0
Merc 450SLC         ;15.2;8;275.8;180;3.1;3.8;18;0;0;3;3; 6.0
Cadillac Fleetwood  ;10.4;8;472;205;2.9;5.2;18;0;0;3;4; 7.0
Lincoln Continental;10.4;8;460;215;3;5.4;17.8;0;0;3;4; 7.0
Chrysler Imperial   ;14.7;8;440;230;3.2;5.3;17.4;0;0;3;4; 7.0
Fiat 128            ;32.4;4;78.7;66;4.1;2.2;19.5;1;1;4;1; 5.0
Honda Civic         ;30.4;4;75.7;52;4.9;1.6;18.5;1;1;4;2; 6.0
Toyota Corolla      ;33.9;4;71.1;65;4.2;1.8;19.9;1;1;4;1; 5.0
Toyota Corona       ;21.5;4;120.1;97;3.7;2.5;20;1;0;3;1; 4.0
Dodge Challenger    ;15.5;8;318;150;2.8;3.5;16.9;0;0;3;2; 5.0
AMC Javelin         ;15.2;8;304;150;3.1;3.4;17.3;0;0;3;2; 5.0
Camaro Z28          ;13.3;8;350;245;3.7;3.8;15.4;0;0;3;4; 7.0
Pontiac Firebird    ;19.2;8;400;175;3.1;3.8;17.1;0;0;3;2; 5.0
Fiat X1-9           ;27.3;4;79;66;4.1;1.9;18.9;1;1;4;1; 5.0
Porsche 914-2       ;26;4;120.3;91;4.4;2.1;16.7;0;1;5;2; 7.0
Lotus Europa        ;30.4;4;95.1;113;3.8;1.5;16.9;1;1;5;2; 7.0
Ford Pantera L      ;15.8;8;351;264;4.2;3.2;14.5;0;1;5;4; 9.0
Ferrari Dino        ;19.7;6;145;175;3.6;2.8;15.5;0;1;5;6; 11.0
Maserati Bora       ;15;8;301;335;3.5;3.6;14.6;0;1;5;8; 13.0
Volvo 142E          ;21.4;4;121;109;4.1;2.8;18.6;1;1;4;2; 6.0
```

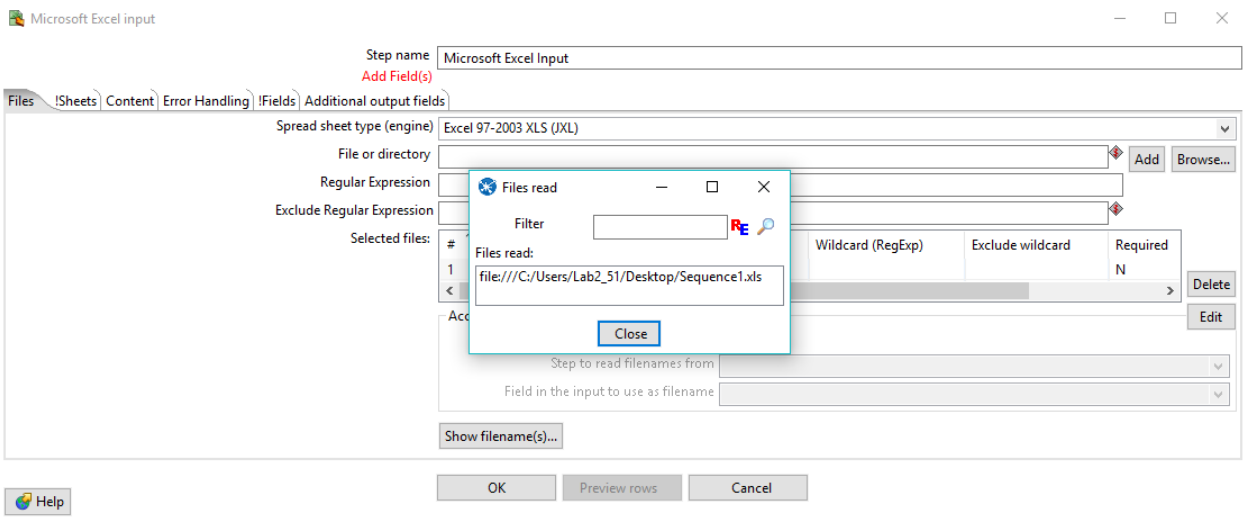
## B) SEQUENCE

### A] Aim: Using Excel Input

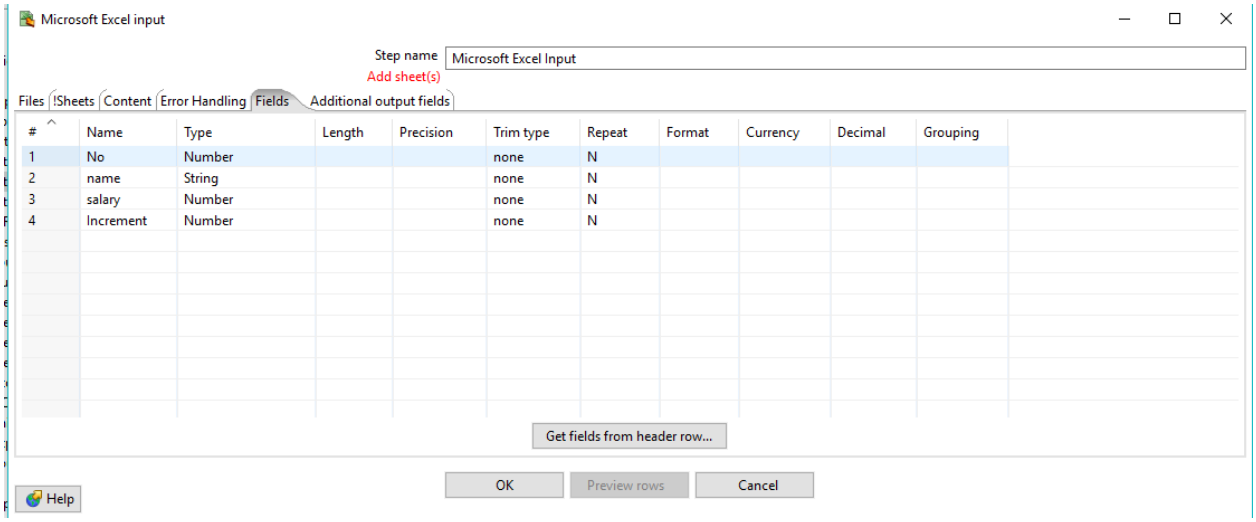
**Step1:** Drag and Drop Microsoft Excel Input, Add Sequence, Microsoft Excel output and give a connection.



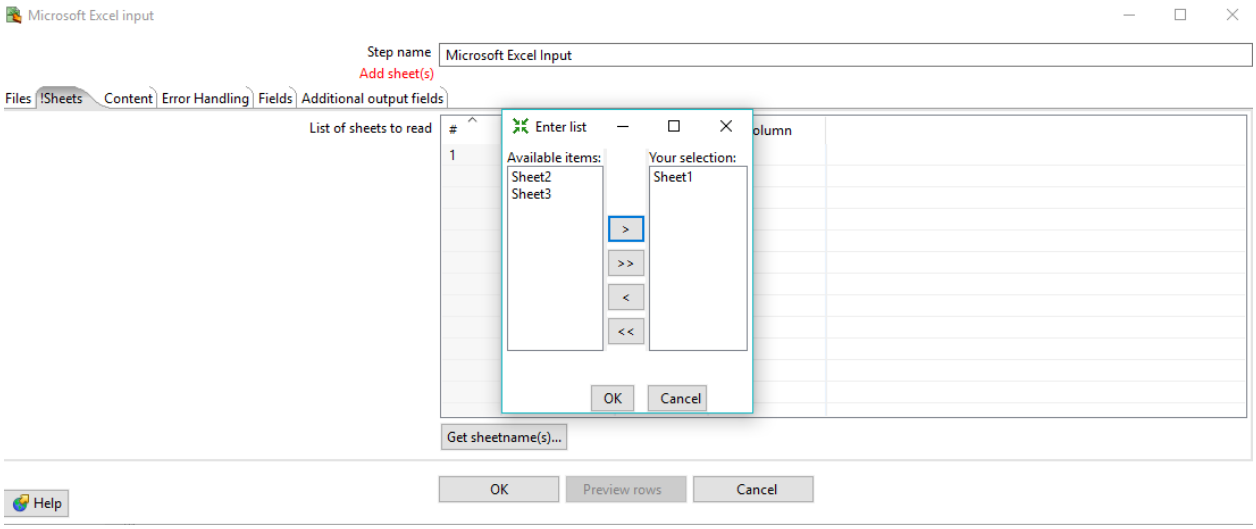

**Step2:** Double Click on Micosoft Excel Input, add the existing Excel file, click ok.



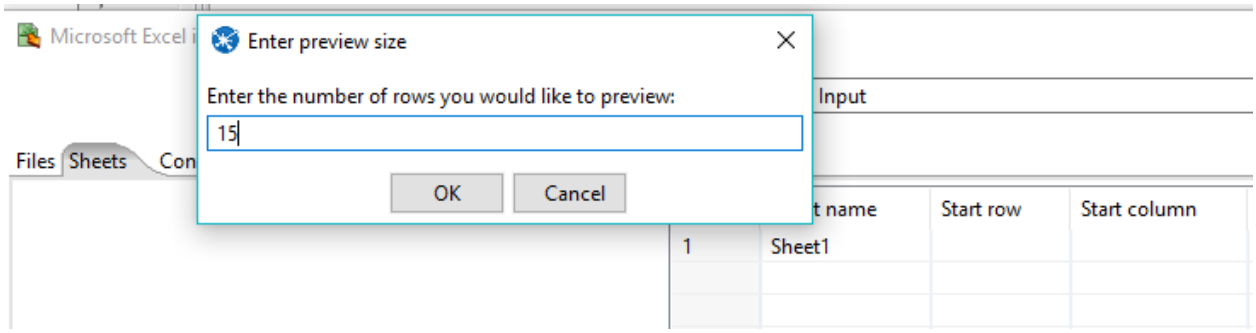
## Step3: Go to fields, click on get fields from header row.



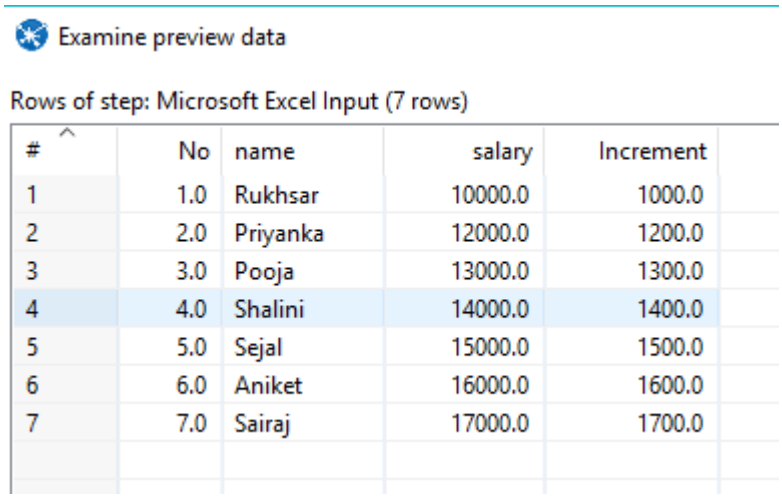

## Step4: Go to sheets, click on get sheetnames, click ok.



**Step5: enter the number of records, click ok.**

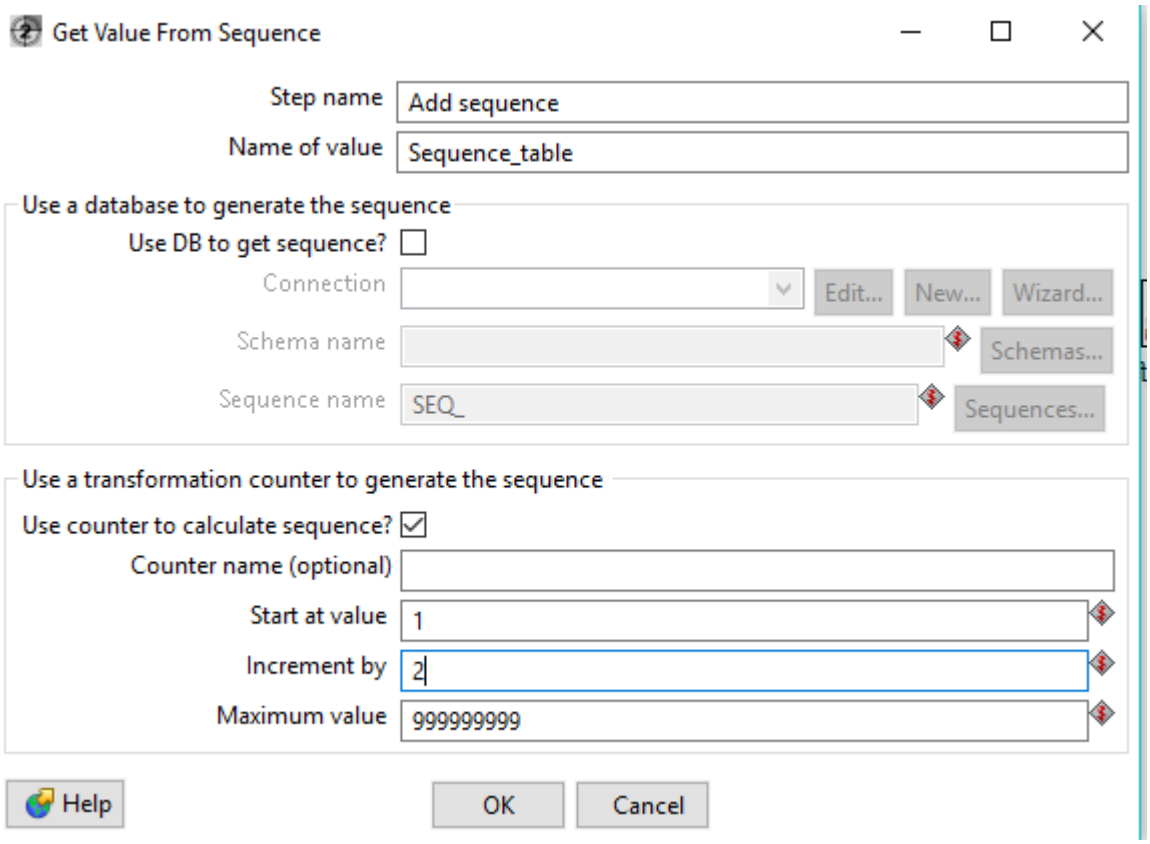



**Step6: click on preview rows.**

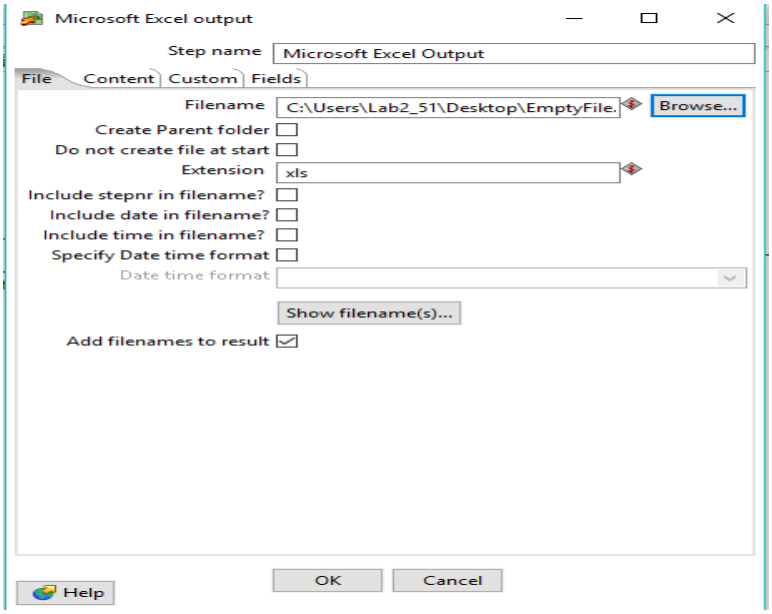



| # | No | name | salary | Increment |
|---|---|---|---|---|
| 1 | 1.0 | Rukhsar | 10000.0 | 1000.0 |
| 2 | 2.0 | Priyanka | 12000.0 | 1200.0 |
| 3 | 3.0 | Pooja | 13000.0 | 1300.0 |
| 4 | 4.0 | Shalini | 14000.0 | 1400.0 |
| 5 | 5.0 | Sejal | 15000.0 | 1500.0 |
| 6 | 6.0 | Aniket | 16000.0 | 1600.0 |
| 7 | 7.0 | Sairaj | 17000.0 | 1700.0 |

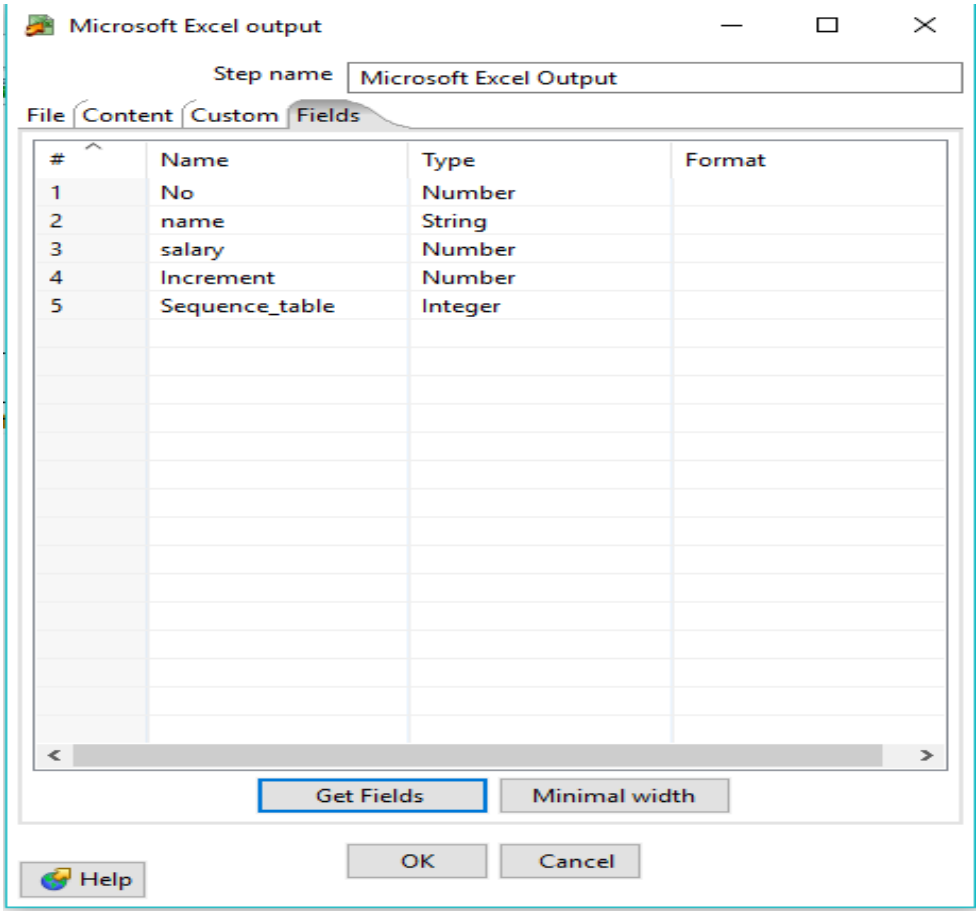
**Step7: Double click on Add Sequence, enter icrement by and start at value,click ok.**



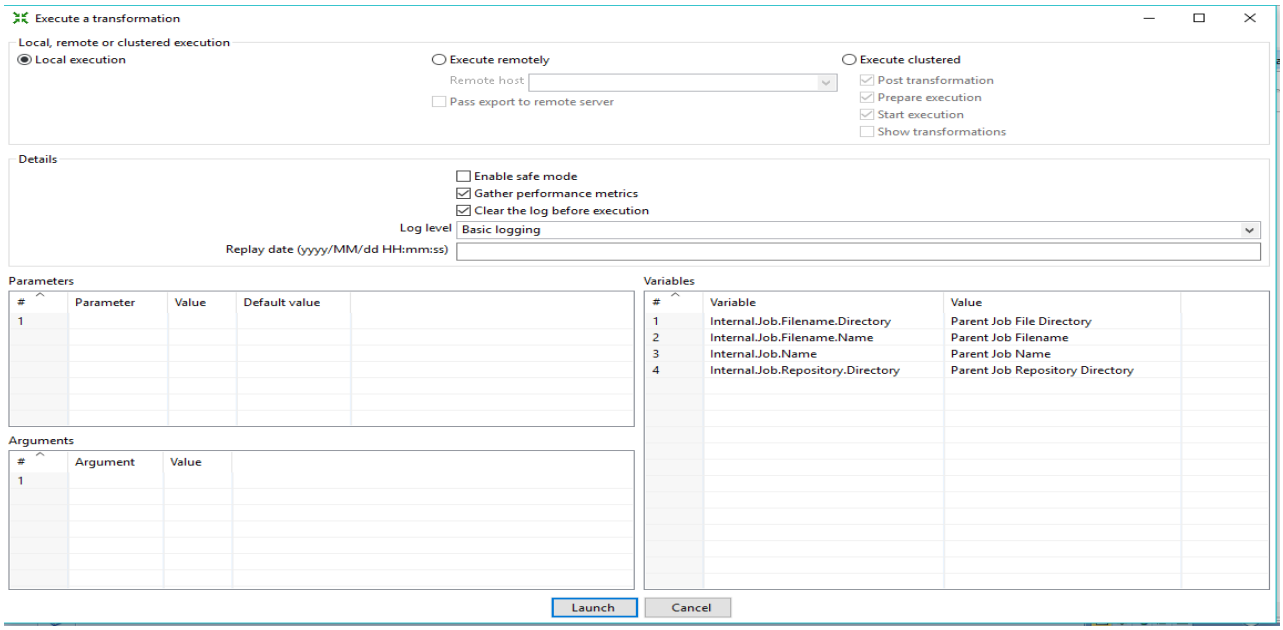

**Step8: Double click on Microsoft Excel Output, Browse The existing Empty file.**



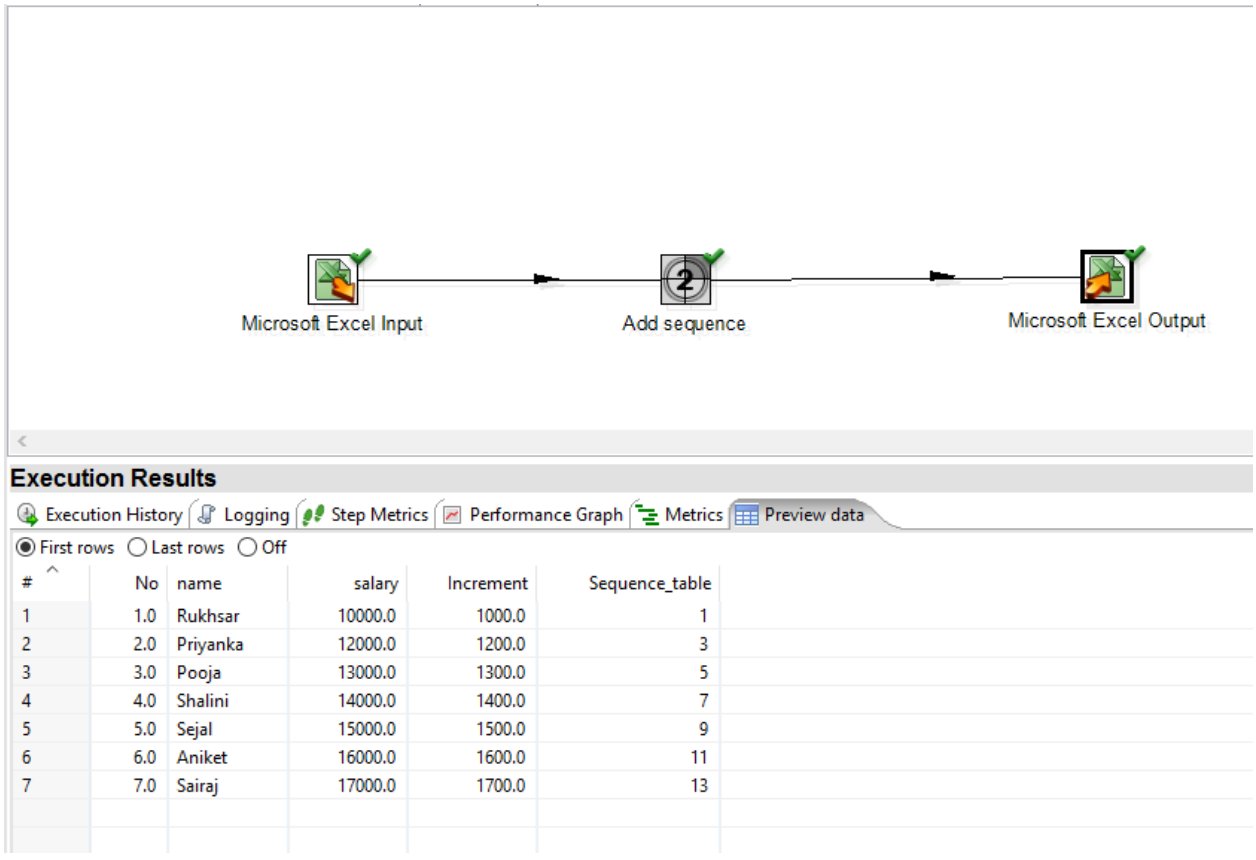

**Step9: Go to fields, click on get fields, click ok.**



## Step10: Run and Launch the transformation.




## Output:



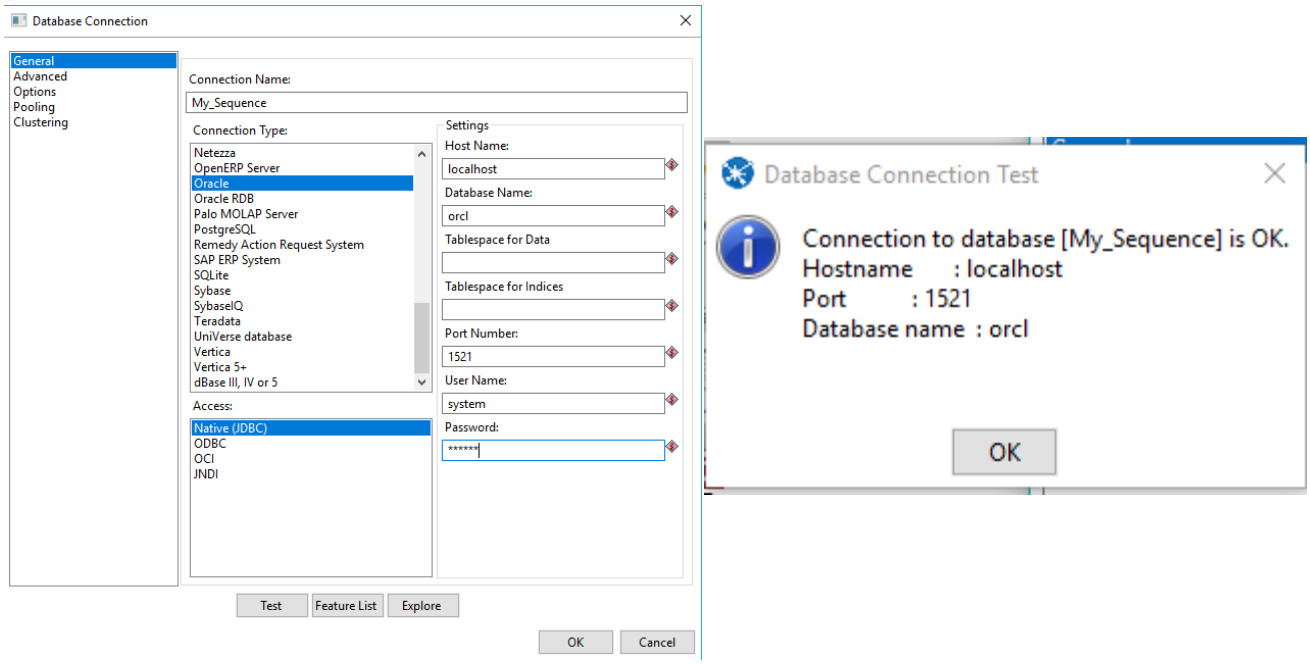

| # | No | name | salary | Increment | Sequence_table |
|---|---|---|---|---|---|
| 1 | 1.0 | Rukhsar | 10000.0 | 1000.0 | 1 |
| 2 | 2.0 | Priyanka | 12000.0 | 1200.0 | 3 |
| 3 | 3.0 | Pooja | 13000.0 | 1300.0 | 5 |
| 4 | 4.0 | Shalini | 14000.0 | 1400.0 | 7 |
| 5 | 5.0 | Sejal | 15000.0 | 1500.0 | 9 |
| 6 | 6.0 | Aniket | 16000.0 | 1600.0 | 11 |
| 7 | 7.0 | Sairaj | 17000.0 | 1700.0 | 13 |

## SEQUENCE(TABLE)

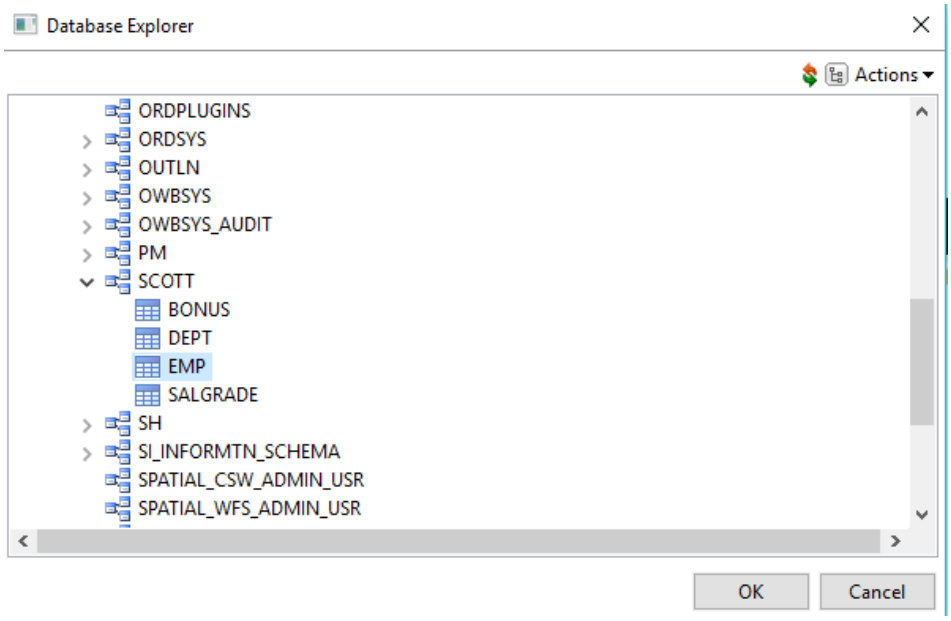
**A] Aim: Using Table**

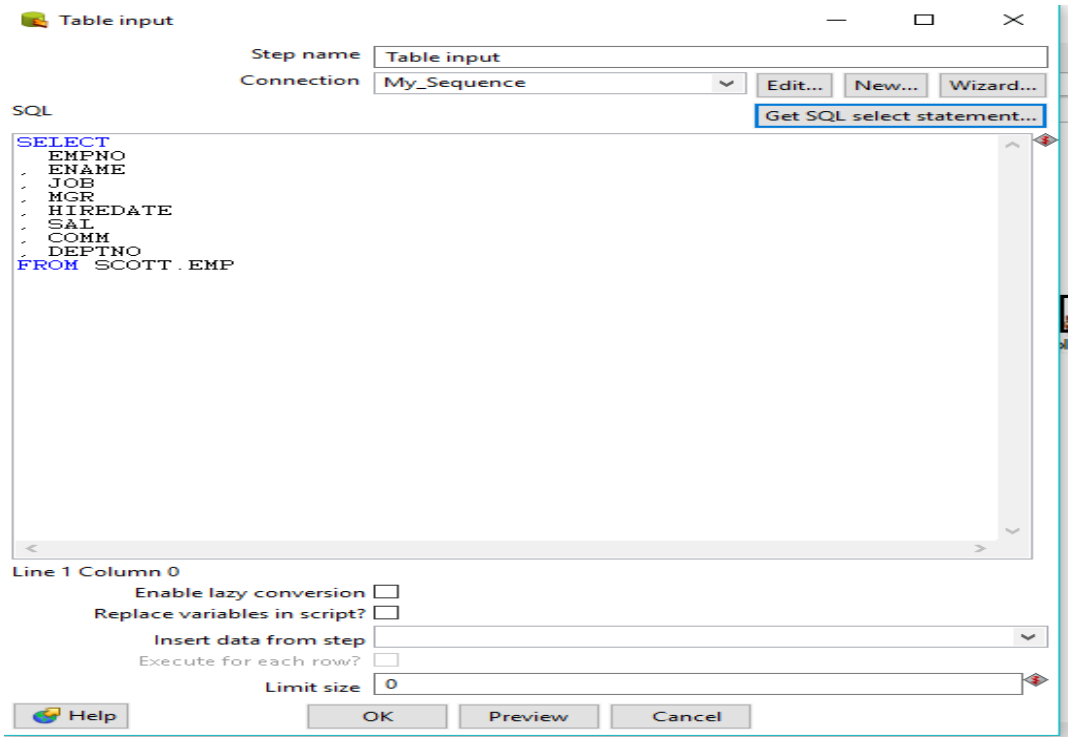
**Step1:** Drag and Drop TABLE Input, Add Sequence,TABLE output and give a connection.



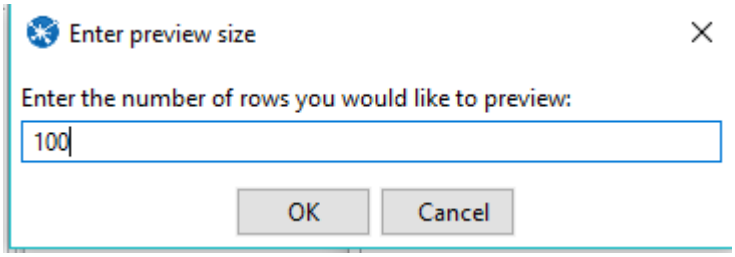



## Step2: Select the scott schema and table emp




- **Set previwe size=100**

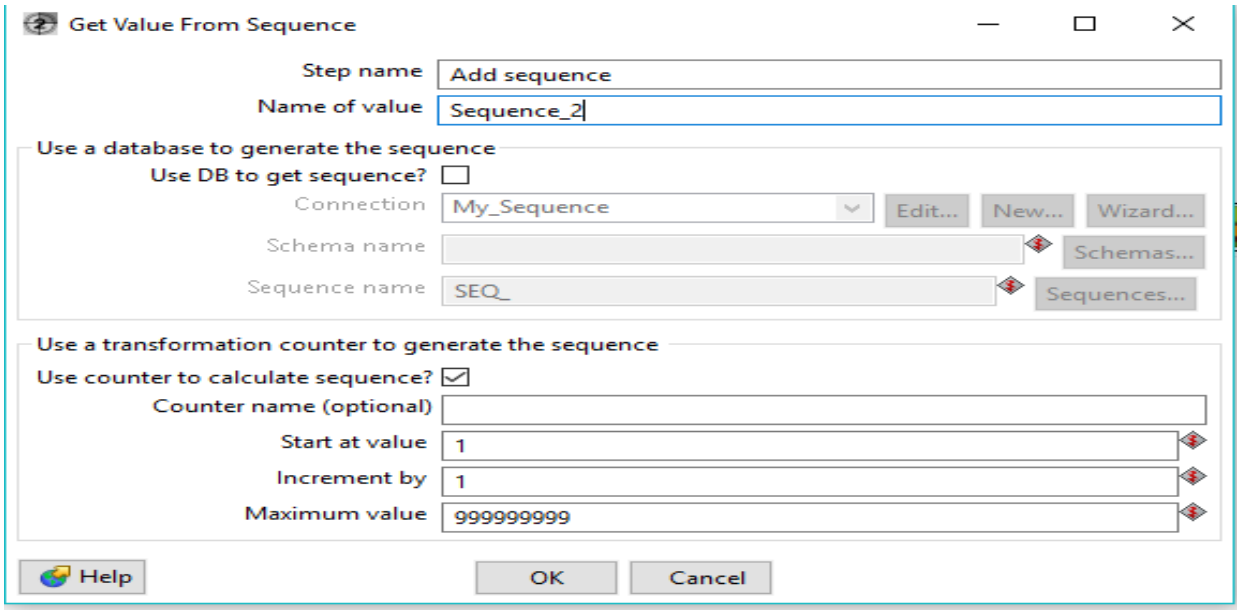



Examine preview data

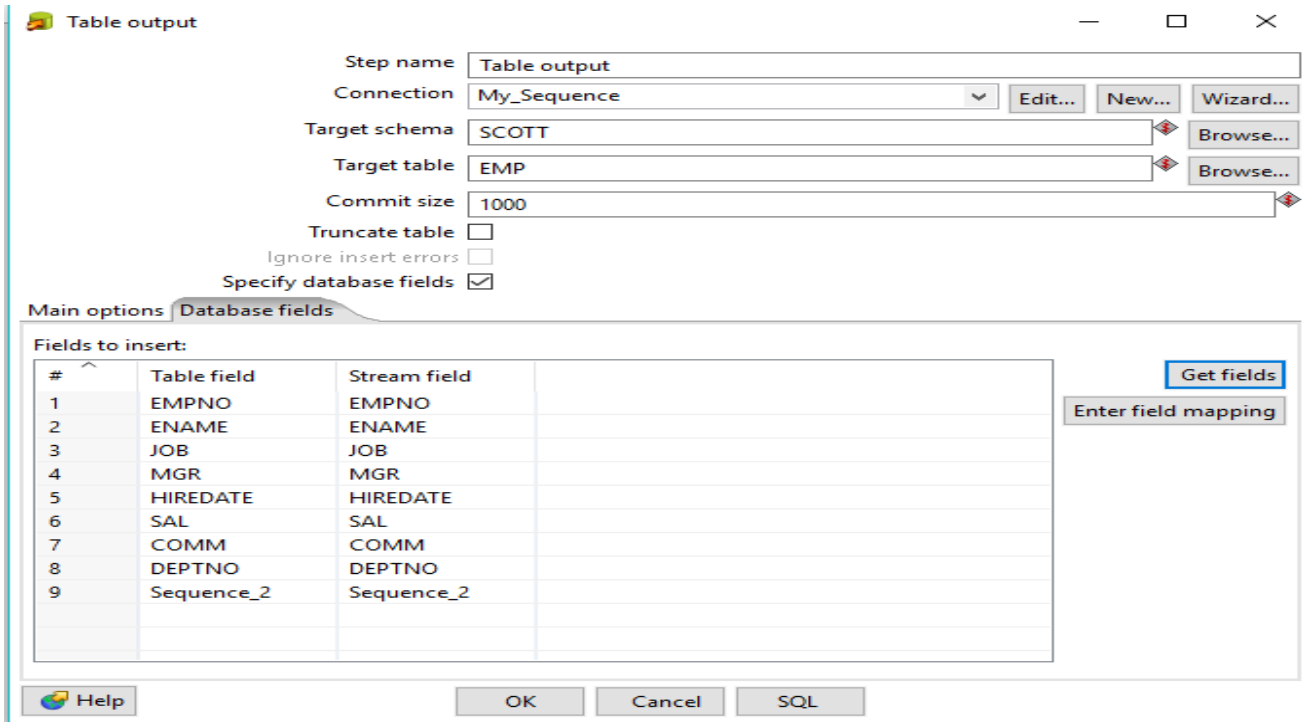
Rows of step: Table input (14 rows)

| # | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|---|
| 1 | 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00.000000000 | 800 | <null> | 20 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000000000 | 1600 | 300 | 30 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000000000 | 1250 | 500 | 30 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000000000 | 2975 | <null> | 20 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000000000 | 1250 | 1400 | 30 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000000000 | 2850 | <null> | 30 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000000000 | 2450 | <null> | 10 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00.000000000 | 3000 | <null> | 20 |
| 9 | 7839 | KING | PRESIDENT | <null> | 1981-11-17 00:00:00.000000000 | 5000 | <null> | 10 |
| 10 | 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00.000000000 | 1500 | 0 | 30 |
| 11 | 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00.000000000 | 1100 | <null> | 20 |
| 12 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000000000 | 950 | <null> | 30 |
| 13 | 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00.000000000 | 3000 | <null> | 20 |
| 14 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000000000 | 1300 | <null> | 10 |

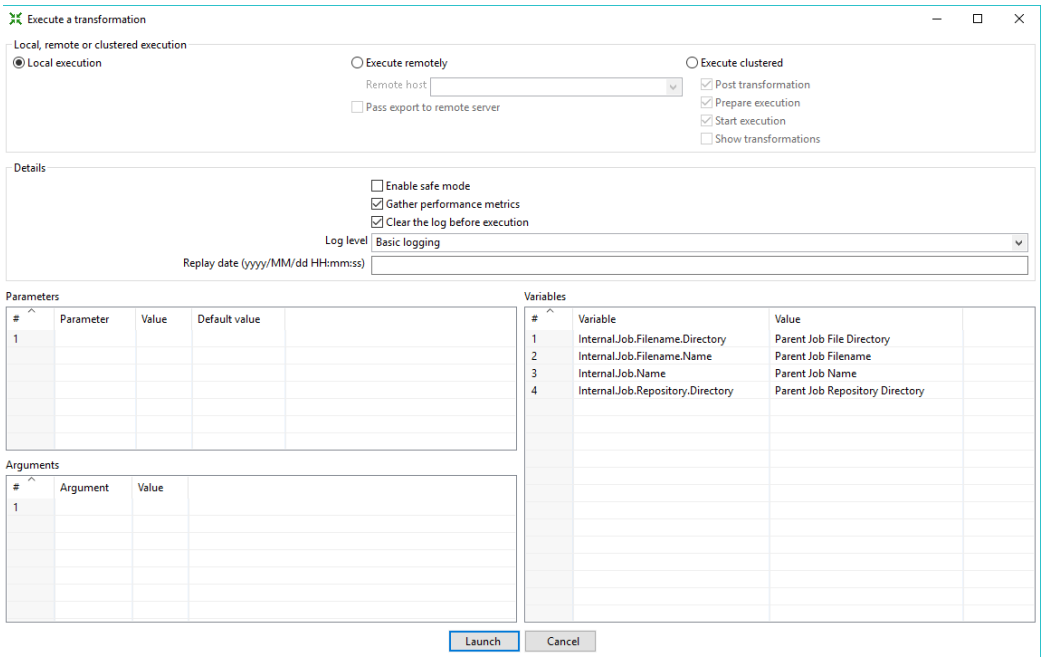
**Step3: Double click on Add Sequence, enter icrement by and start at value,click ok.**



**Step 4: Double click on table Output, Browse The existing Empty file.Go to fields, click on get fields, click ok.**




**Step 5: Run and Launch the transformation.**



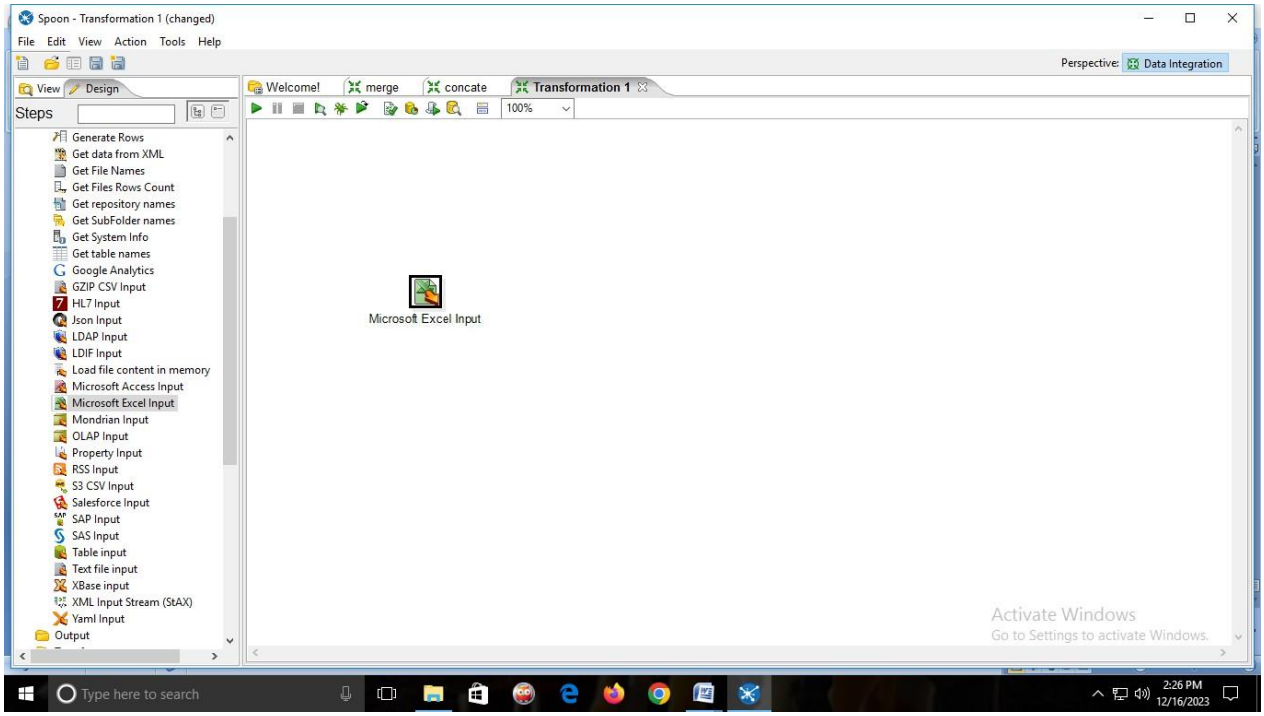
## Output:

First rows  Last rows  Off

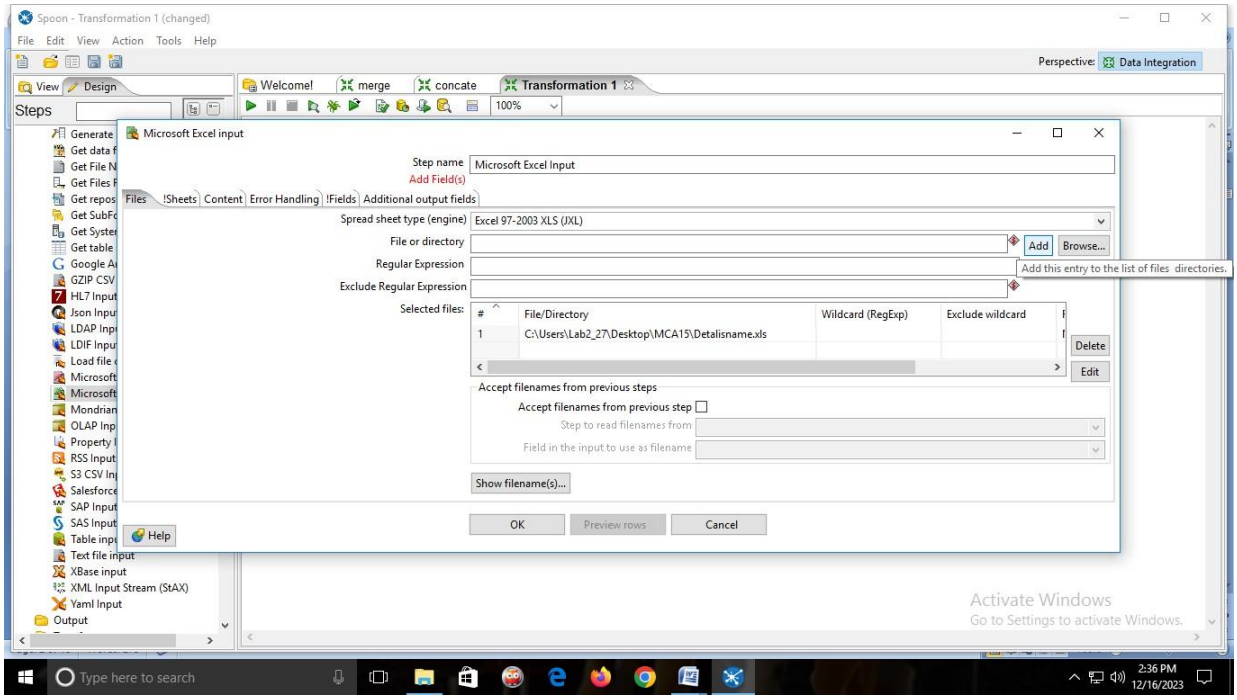
| # | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO | valuename |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00.000000000 | 800 | <null> | 20 | 1 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000000000 | 1600 | 300 | 30 | 4 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000000000 | 1250 | 500 | 30 | 7 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000000000 | 2975 | <null> | 20 | 10 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000000000 | 1250 | 1400 | 30 | 13 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000000000 | 2850 | <null> | 30 | 1 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000000000 | 2450 | <null> | 10 | 4 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00.000000000 | 3000 | <null> | 20 | 7 |
| 9 | 7839 | KING | PRESIDENT | <null> | 1981-11-17 00:00:00.000000000 | 5000 | <null> | 10 | 10 |
| 10 | 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00.000000000 | 1500 | 0 | 30 | 13 |
| 11 | 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00.000000000 | 1100 | <null> | 20 | 1 |
| 12 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000000000 | 950 | <null> | 30 | 4 |
| 13 | 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00.000000000 | 3000 | <null> | 20 | 7 |
| 14 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000000000 | 1300 | <null> | 10 | 10 |
| 15 | | | | | | | | | |

# C) AIM: Implementation of STRING OPERATIONS with Pentaho

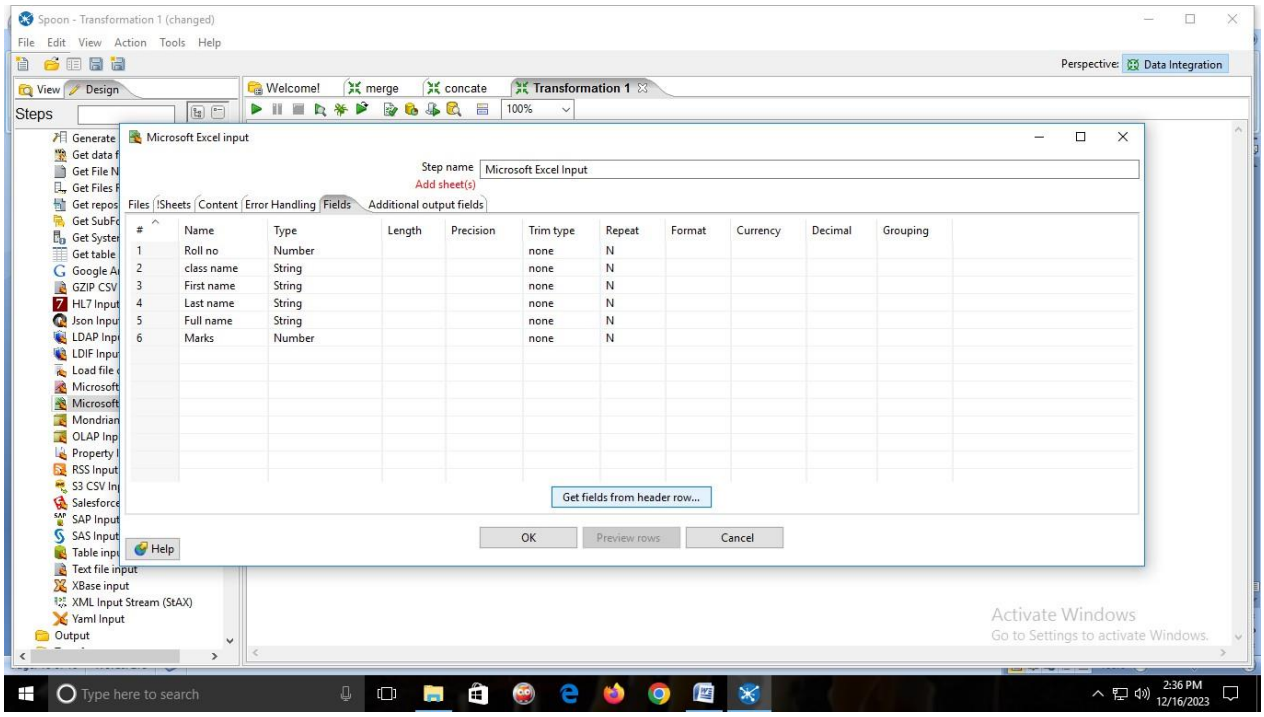
- create an Excel sheet with the fields firstname,lastname,fullname,rollno,marks
- From input folder→drag & drop→ MICROSOFT EXCEL INPUT



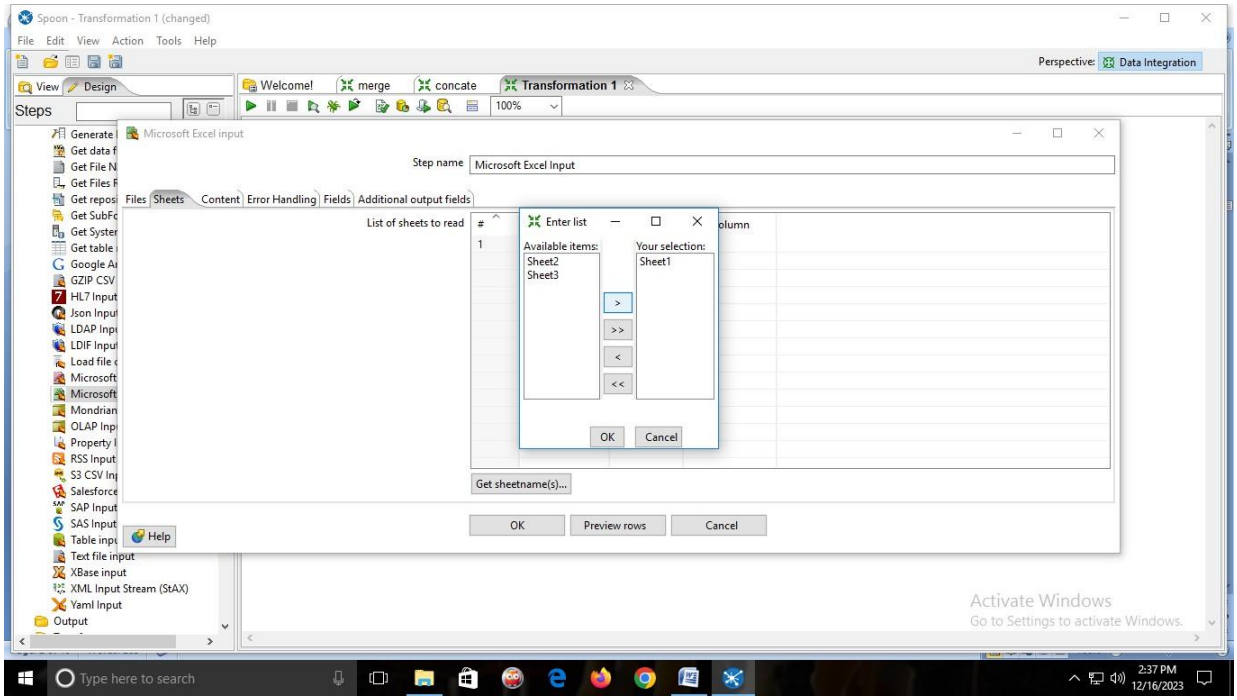

- →Double click on it and add the Excel sheet.



- →click on get fields from header row

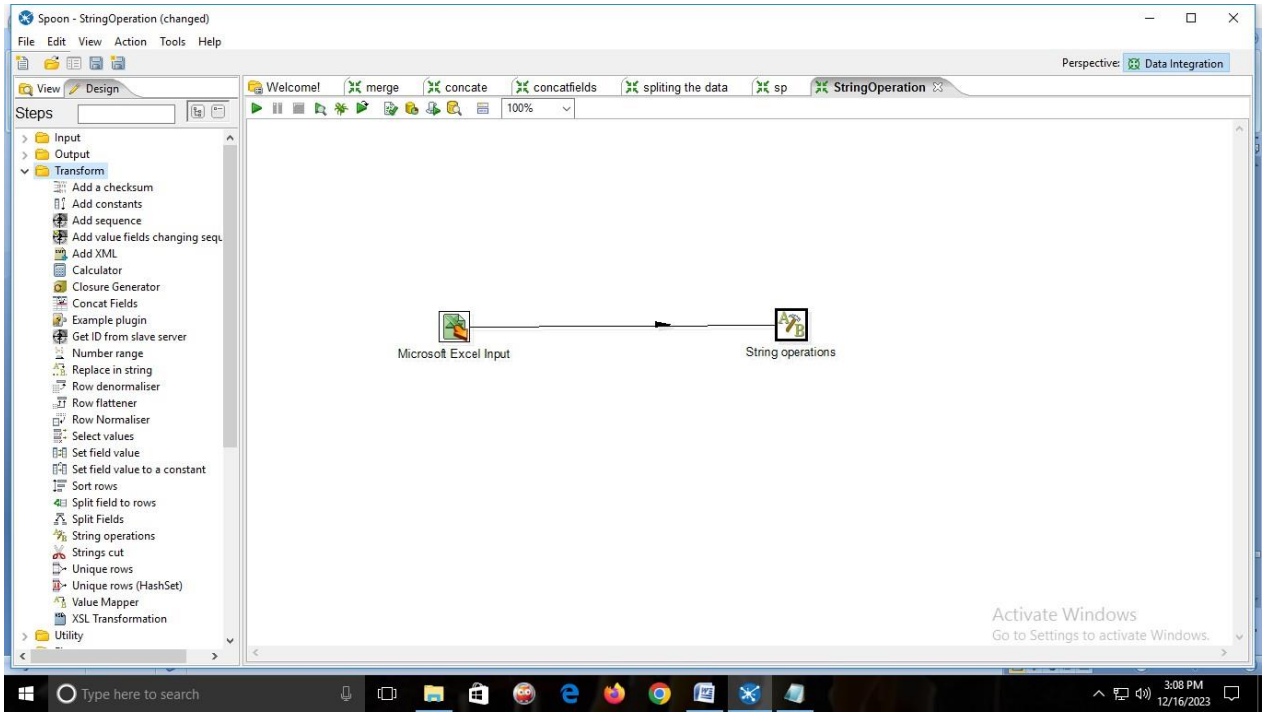



- →then click on sheets→click on get sheetname→add sheet1→ok

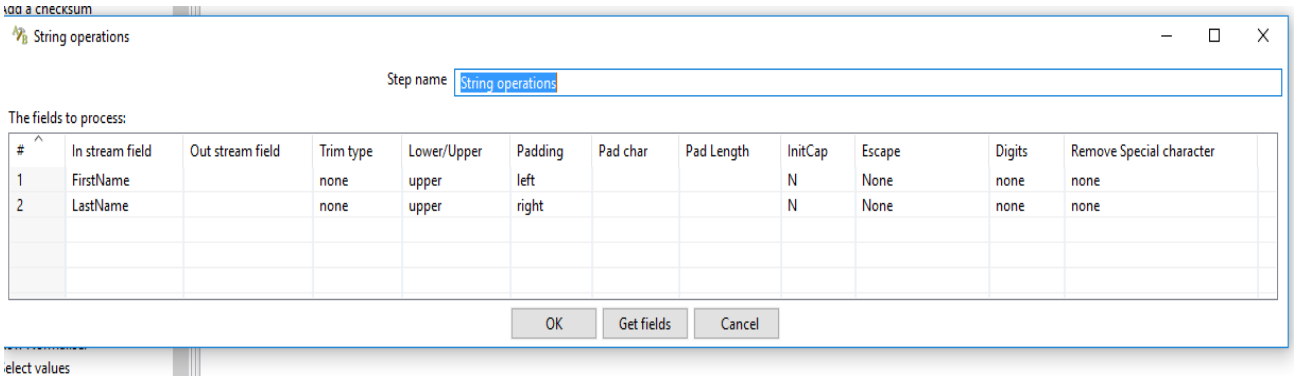


- →click on Preview rows→Ok

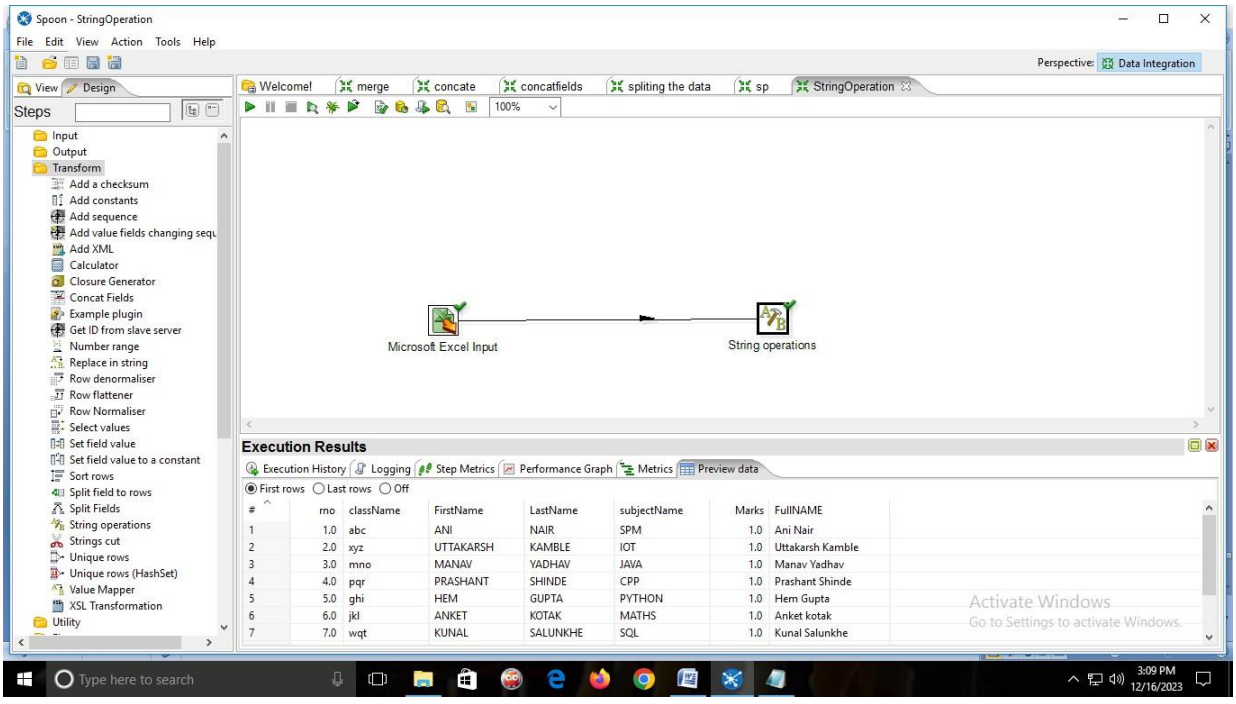

- From Transform Folder→drag & drop→STRING OPERATIONS→and connect MICROSOFT EXCEL INPUT to STRING OPERATIONS.



- double click on STRING OPERATIONS



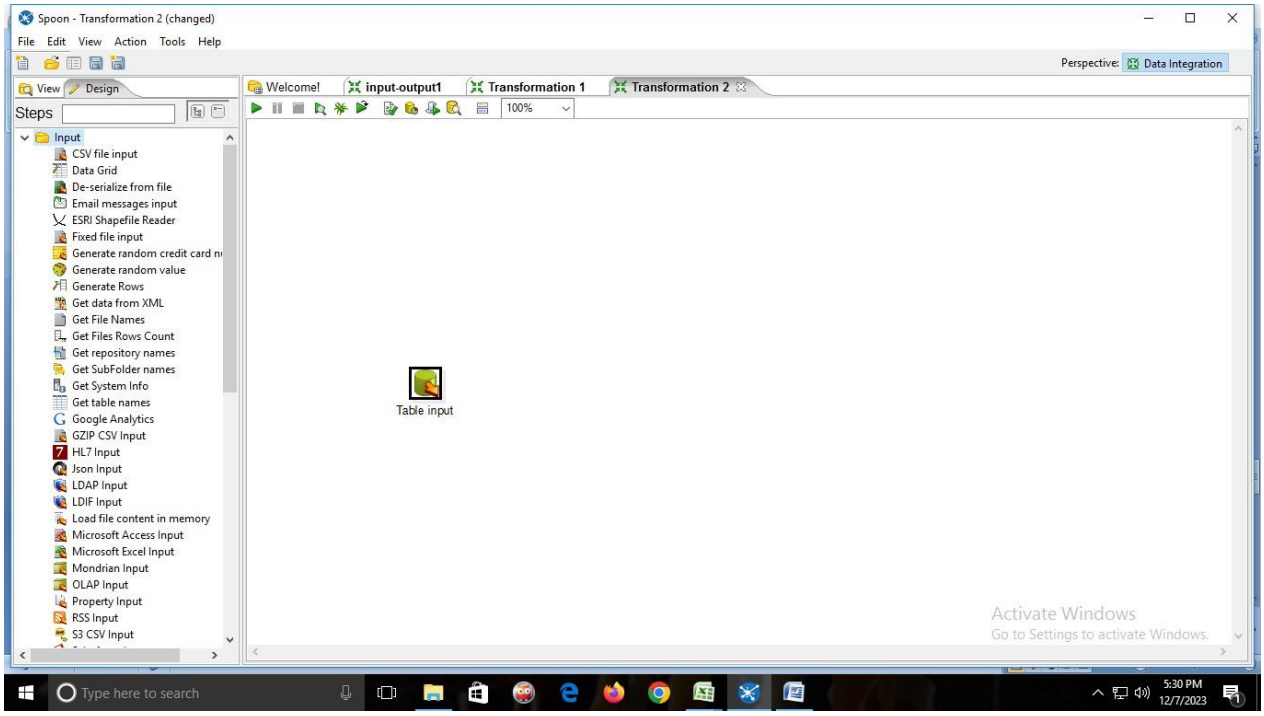

## OUTPUT:



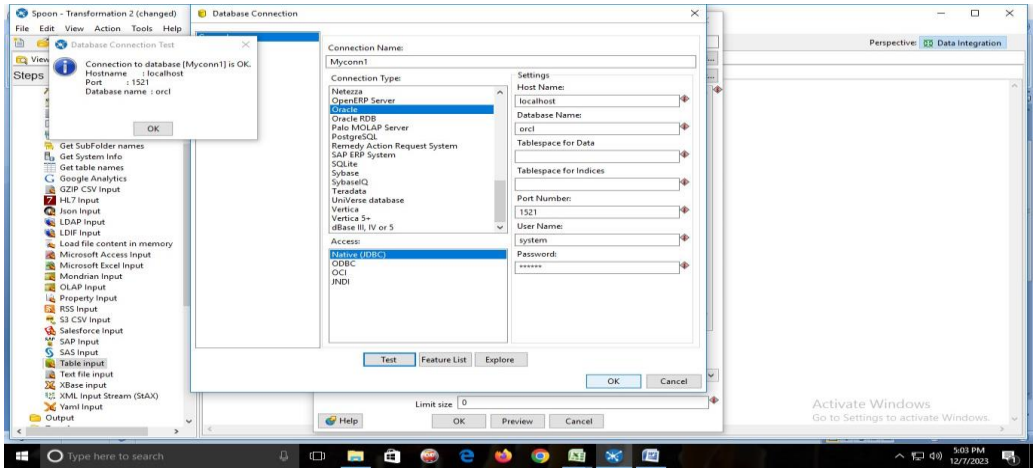
# D)AIM: Implementation of Sort operations with Pentaho.

## TABLE SORT

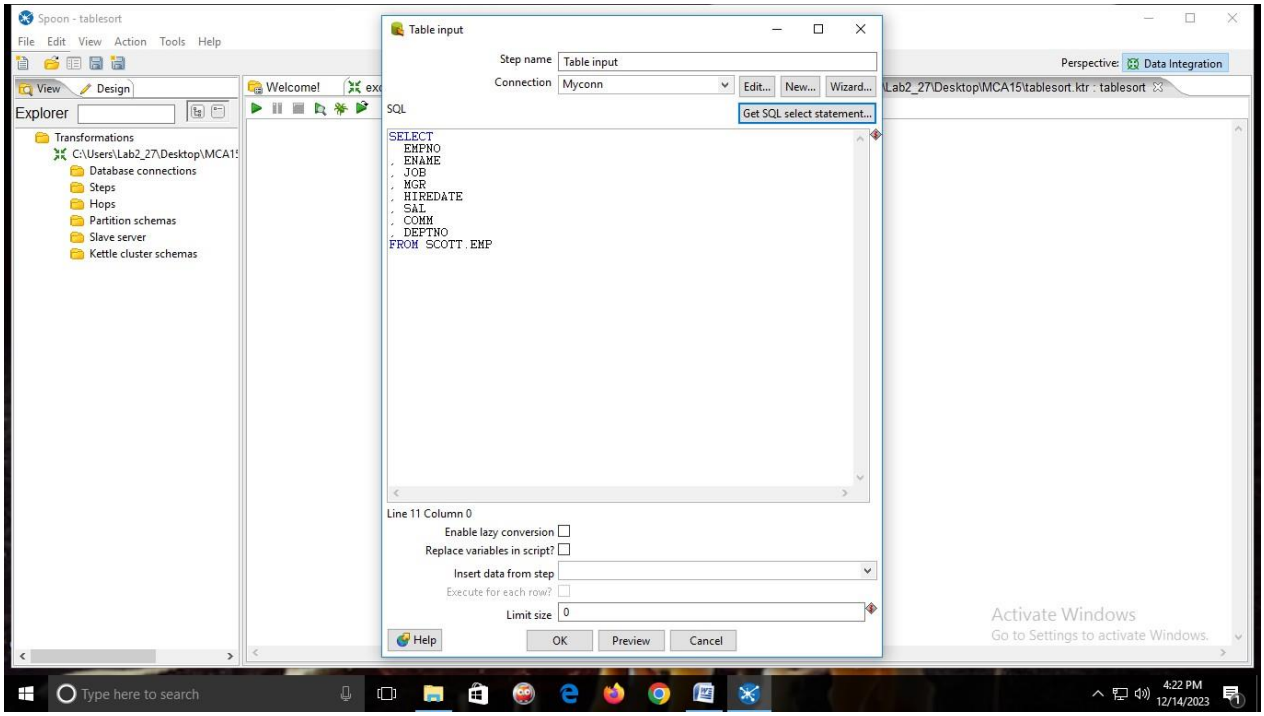
- input folder→drag & drop→table input



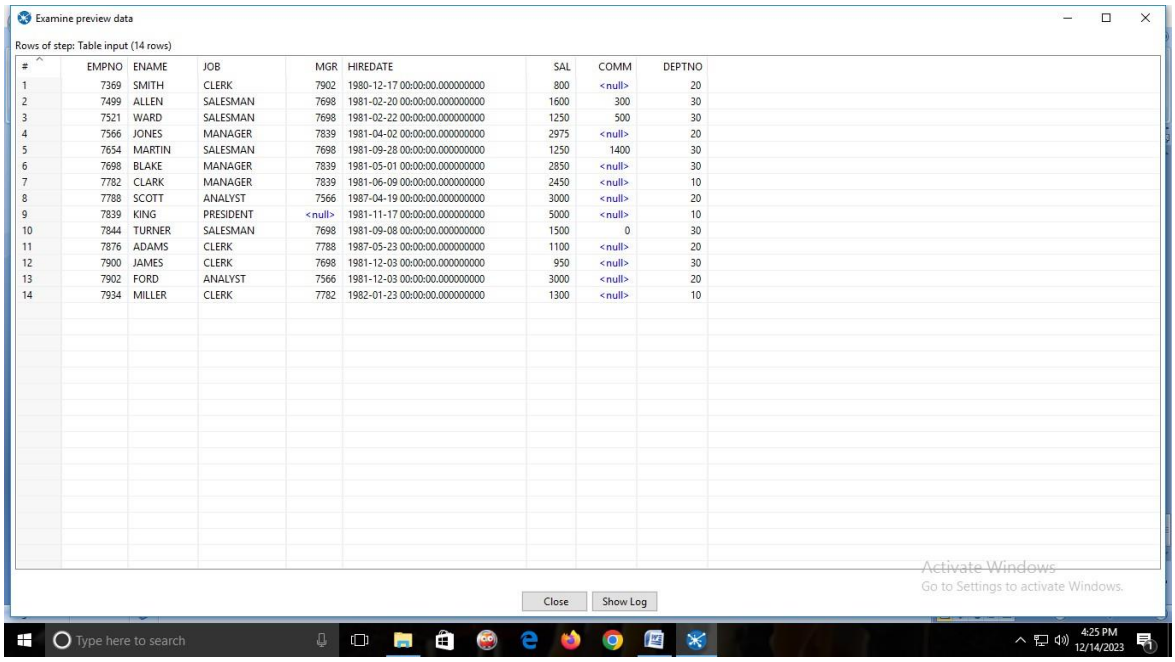

- New→connection name



- Click on→GET SQL SELECT STATEMENT→and select the table→then click on preview .

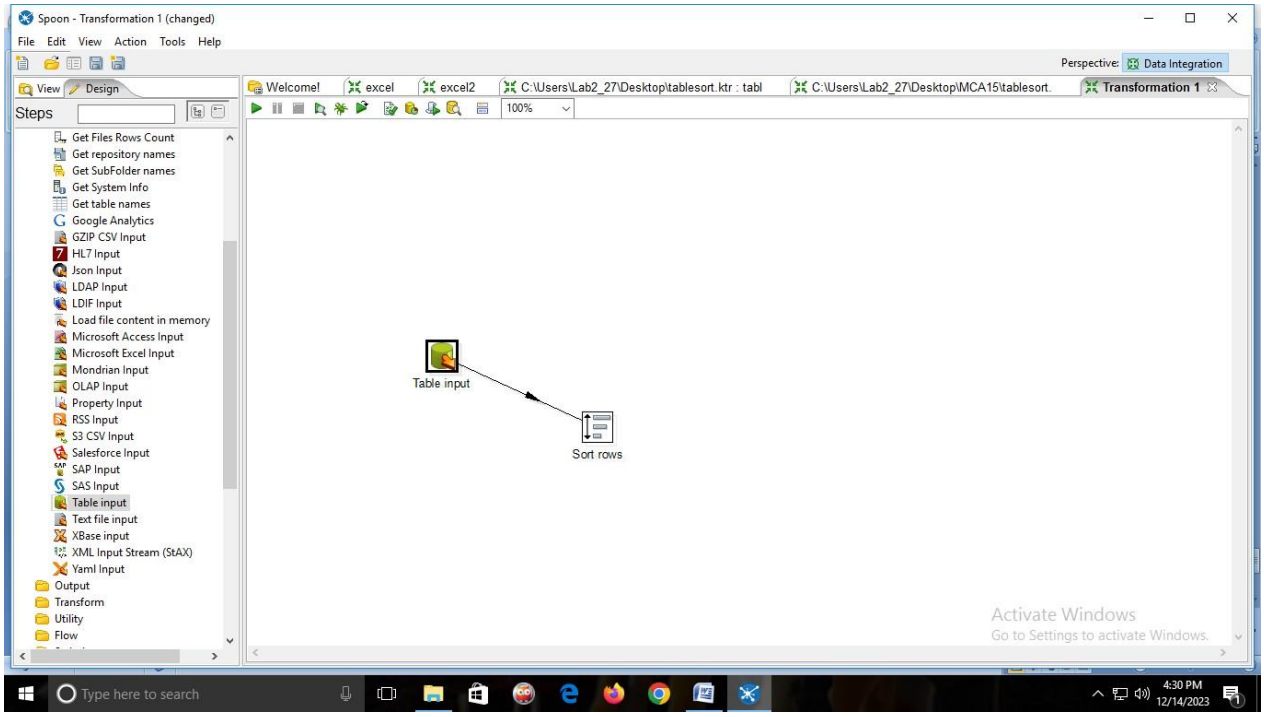



- Preview table

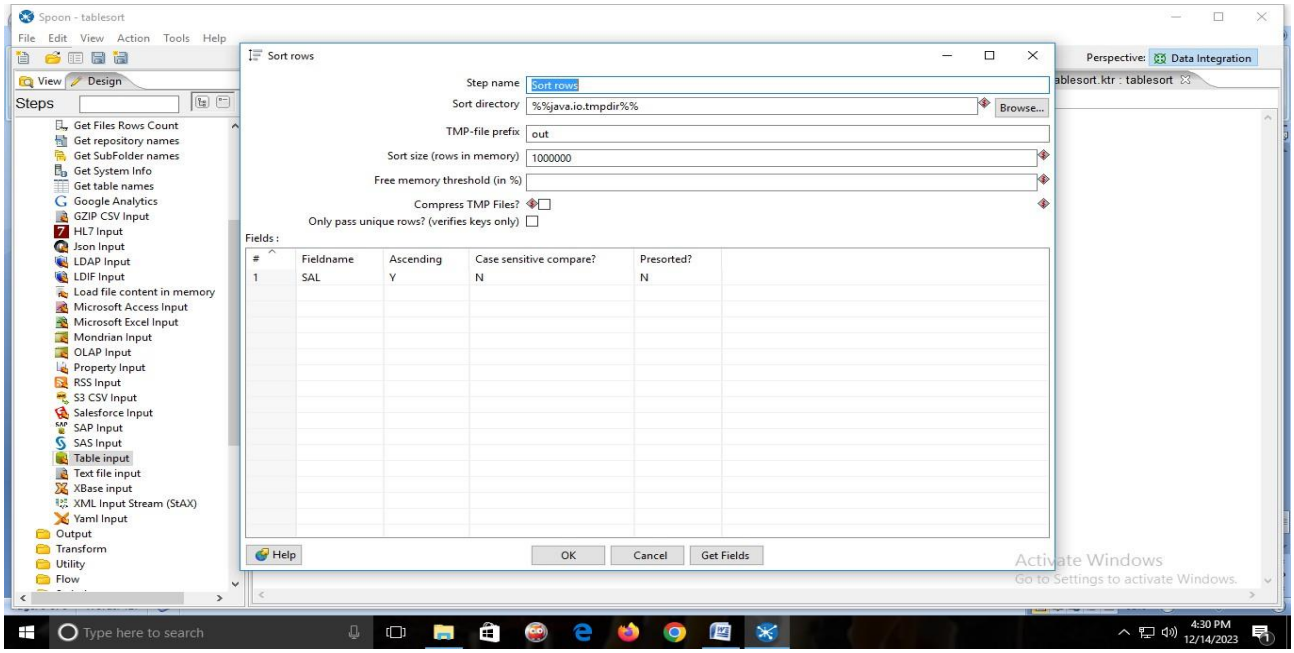



| # | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|---|
| 1 | 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00.000000000 | 800 | <null> | 20 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000000000 | 1600 | 300 | 30 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000000000 | 1250 | 500 | 30 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000000000 | 2975 | <null> | 20 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000000000 | 1250 | 1400 | 30 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000000000 | 2850 | <null> | 30 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000000000 | 2450 | <null> | 10 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00.000000000 | 3000 | <null> | 20 |
| 9 | 7839 | KING | PRESIDENT | <null> | 1981-11-17 00:00:00.000000000 | 5000 | <null> | 10 |
| 10 | 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00.000000000 | 1500 | 0 | 30 |
| 11 | 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00.000000000 | 1100 | <null> | 20 |
| 12 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000000000 | 950 | <null> | 30 |
| 13 | 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00.000000000 | 3000 | <null> | 20 |
| 14 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000000000 | 1300 | <null> | 10 |

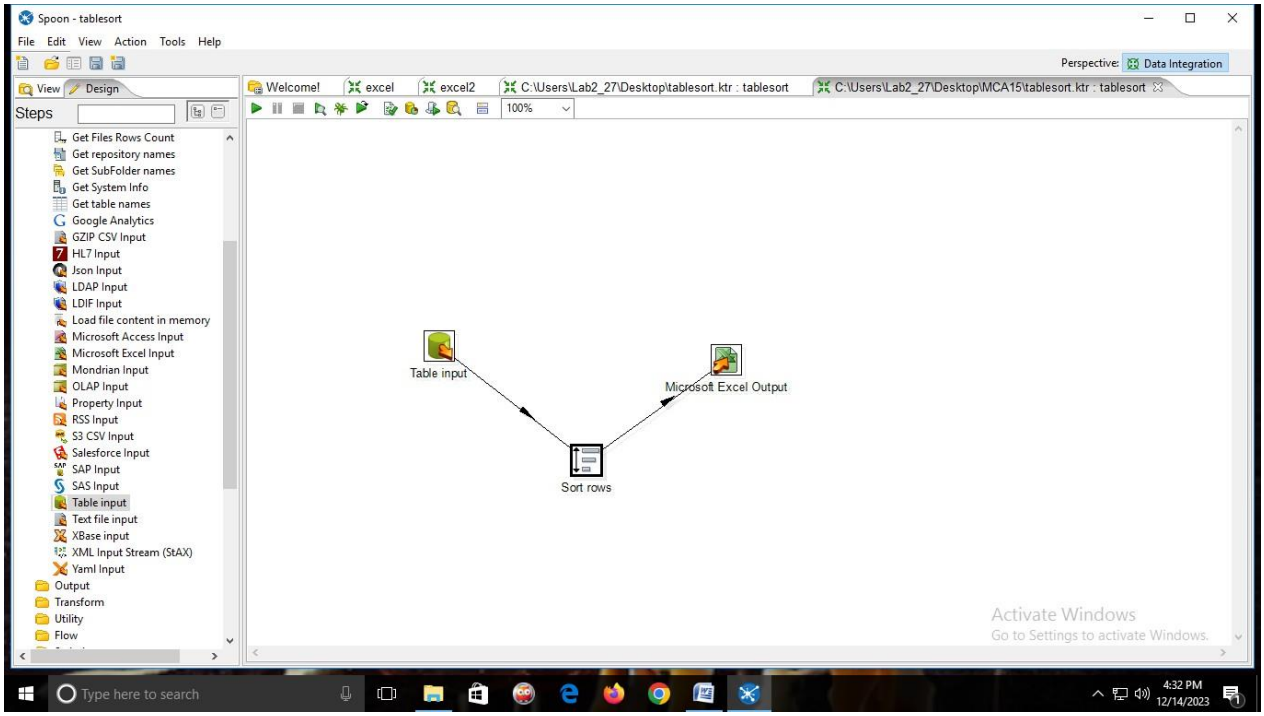
- From transform folder→drag & drop SORT ROWS and connect table input to sort rows



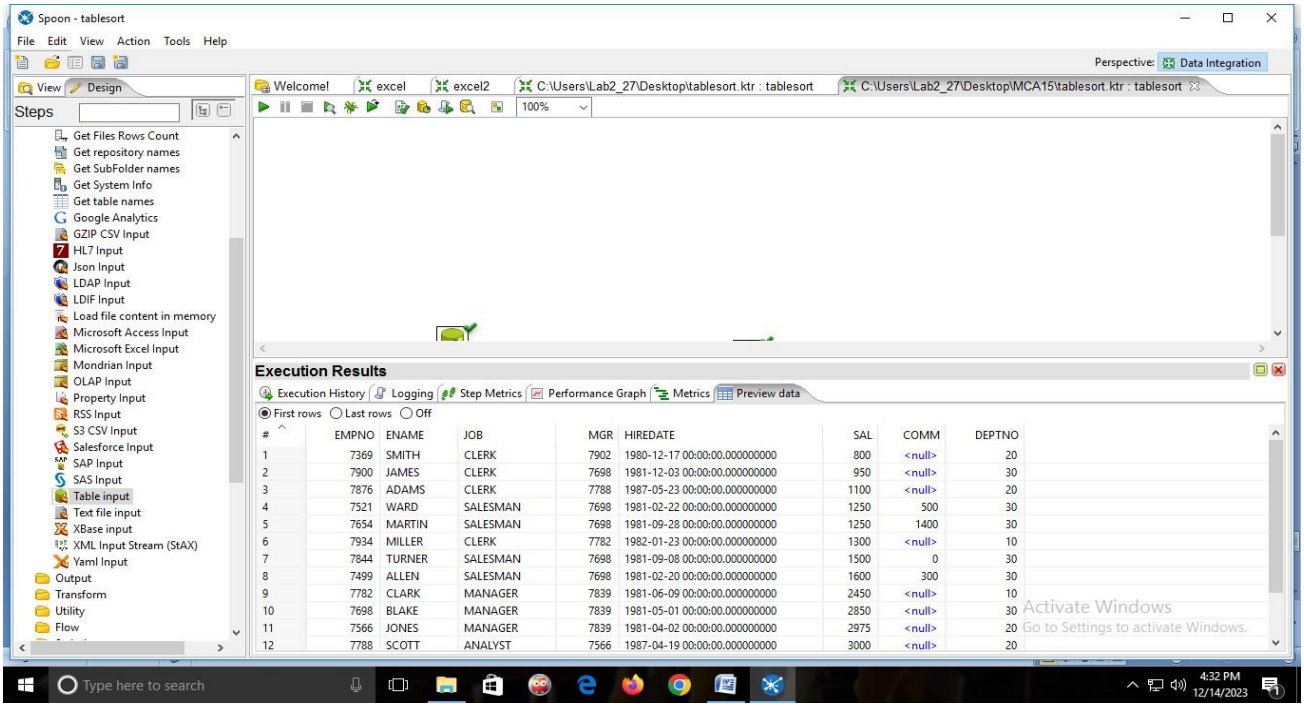

- Give the field name which we have to sort.



- From Output folder →drag & drop Microsoft excel output →connect sort row with Microsoft excel output.




## OUTPUT:

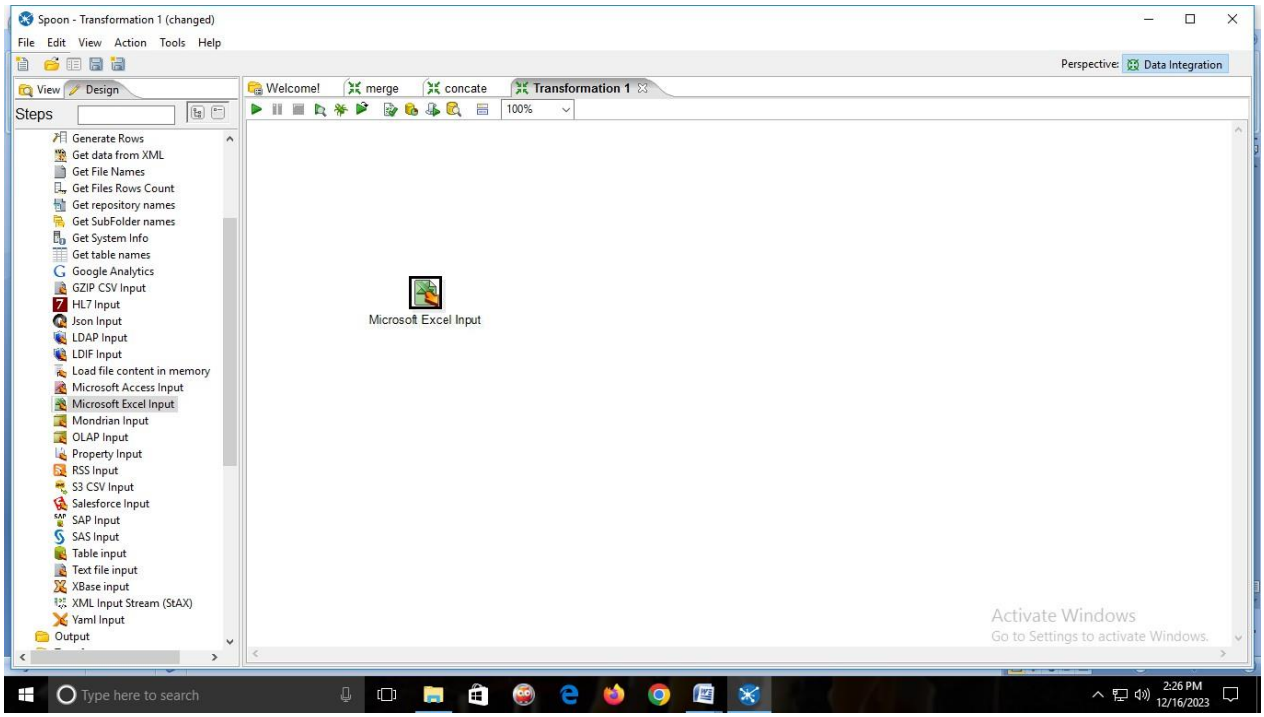



| # | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|---|
| 1 | 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00.000000000 | 800 | <null> | 20 |
| 2 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000000000 | 950 | <null> | 30 |
| 3 | 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00.000000000 | 1100 | <null> | 20 |
| 4 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000000000 | 1250 | 500 | 30 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000000000 | 1250 | 1400 | 30 |
| 6 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000000000 | 1300 | <null> | 10 |
| 7 | 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00.000000000 | 1500 | 0 | 30 |
| 8 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000000000 | 1600 | 300 | 30 |
| 9 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000000000 | 2450 | <null> | 10 |
| 10 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000000000 | 2850 | <null> | 30 |
| 11 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000000000 | 2975 | <null> | 20 |
| 12 | 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00.000000000 | 3000 | <null> | 20 |

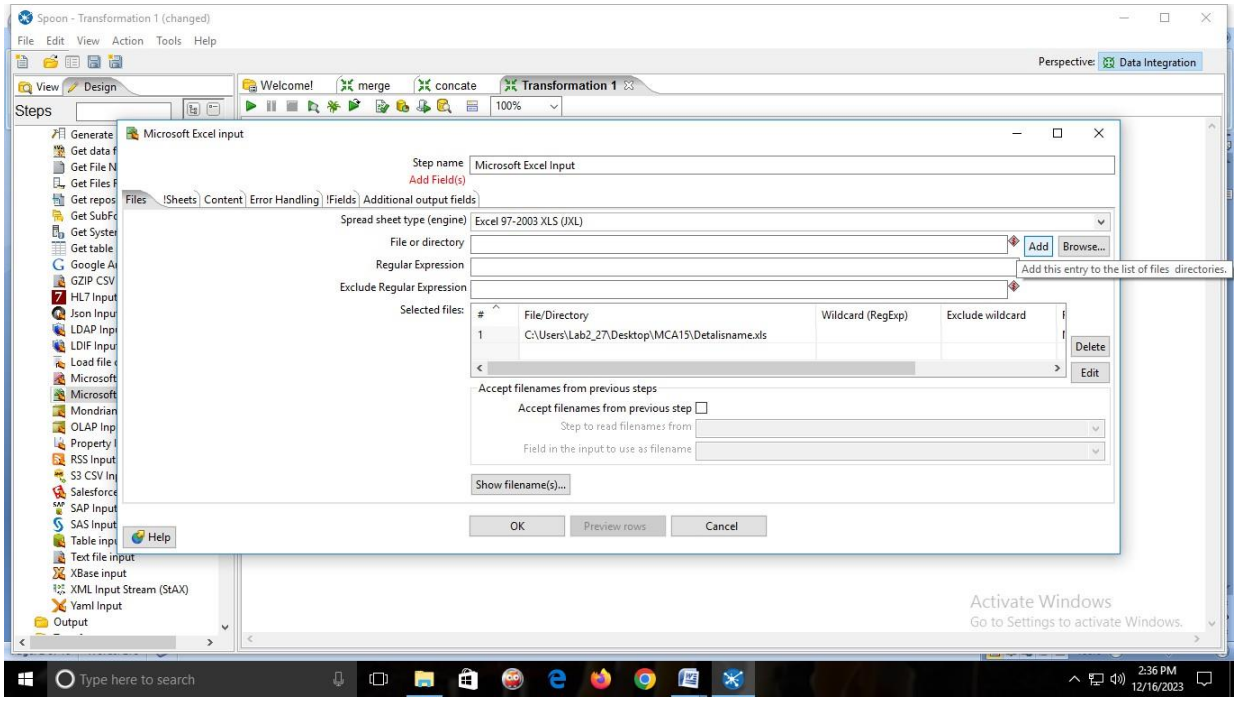
# E)AIM: Implementation of SPLIT FIELDS operations with Pentaho

## SPLIT FIELDS

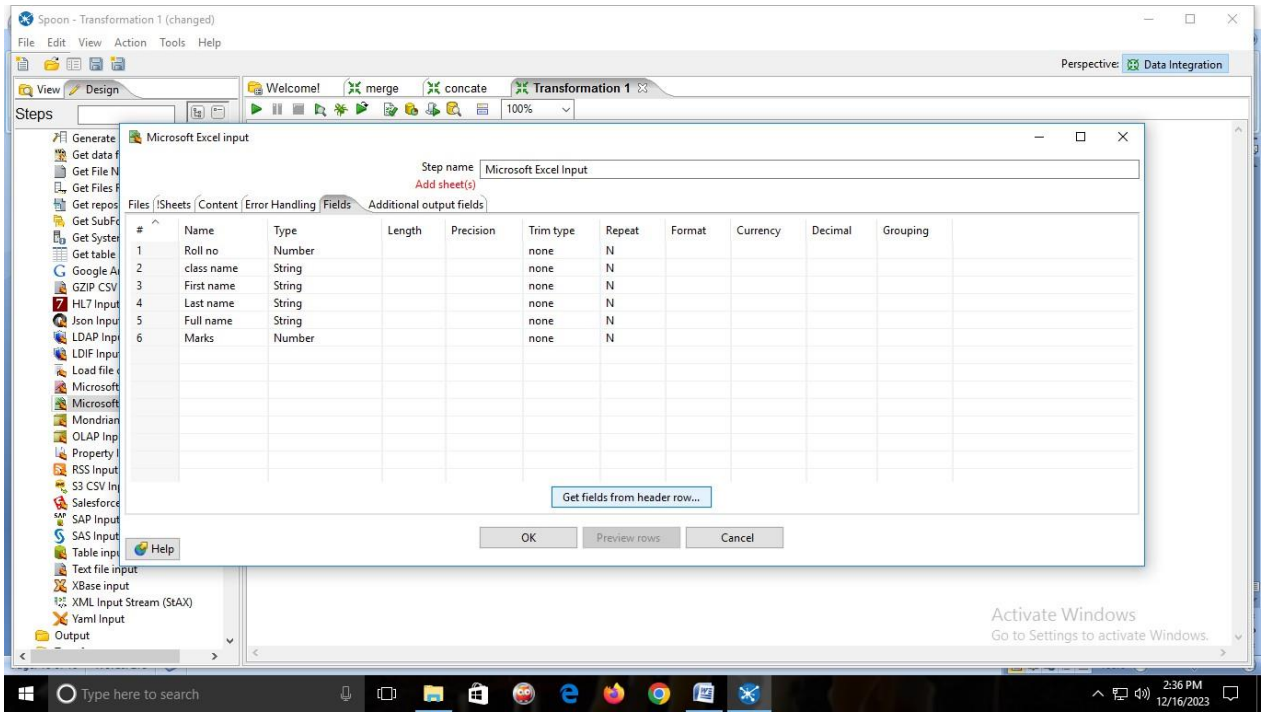
- create an Excel sheet with the fields firstname,lastname,fullname,rollno,marks
- From input folder→drag & drop→ MICROSOFT EXCEL INPUT



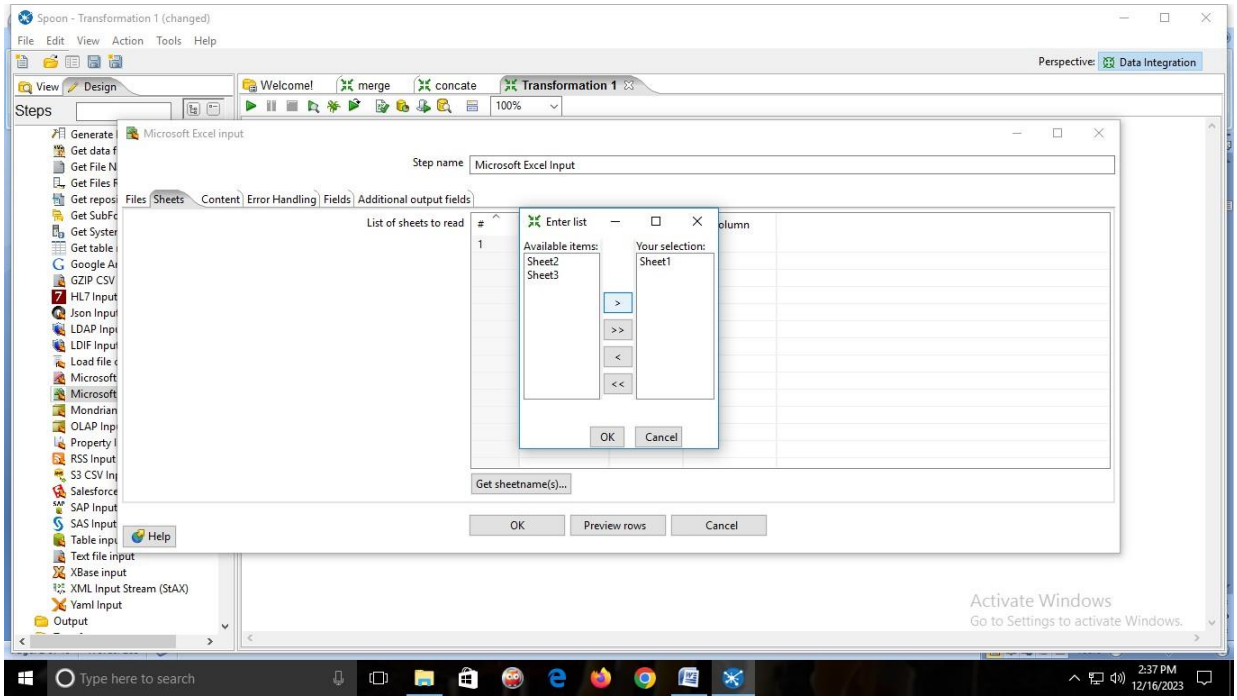

- →Double click on it and add the Excel sheet.



- →click on get fields from header row

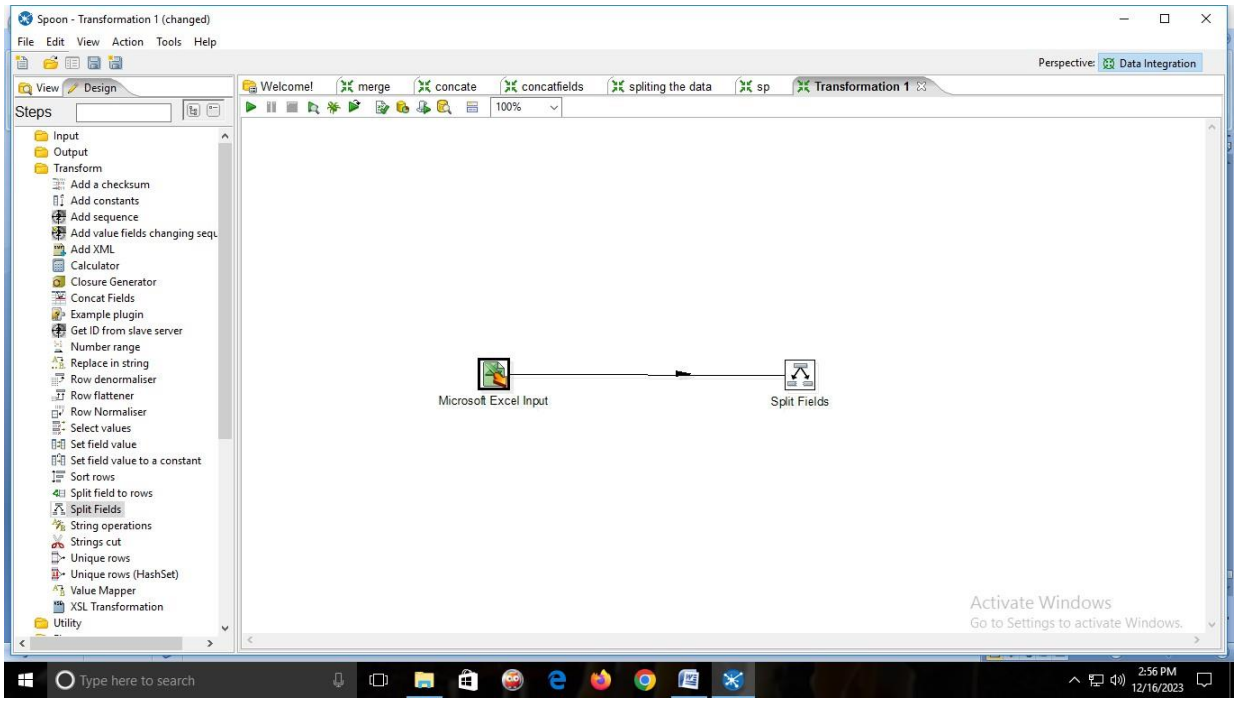



- →then click on sheets→click on get sheetname→add sheet1→ok

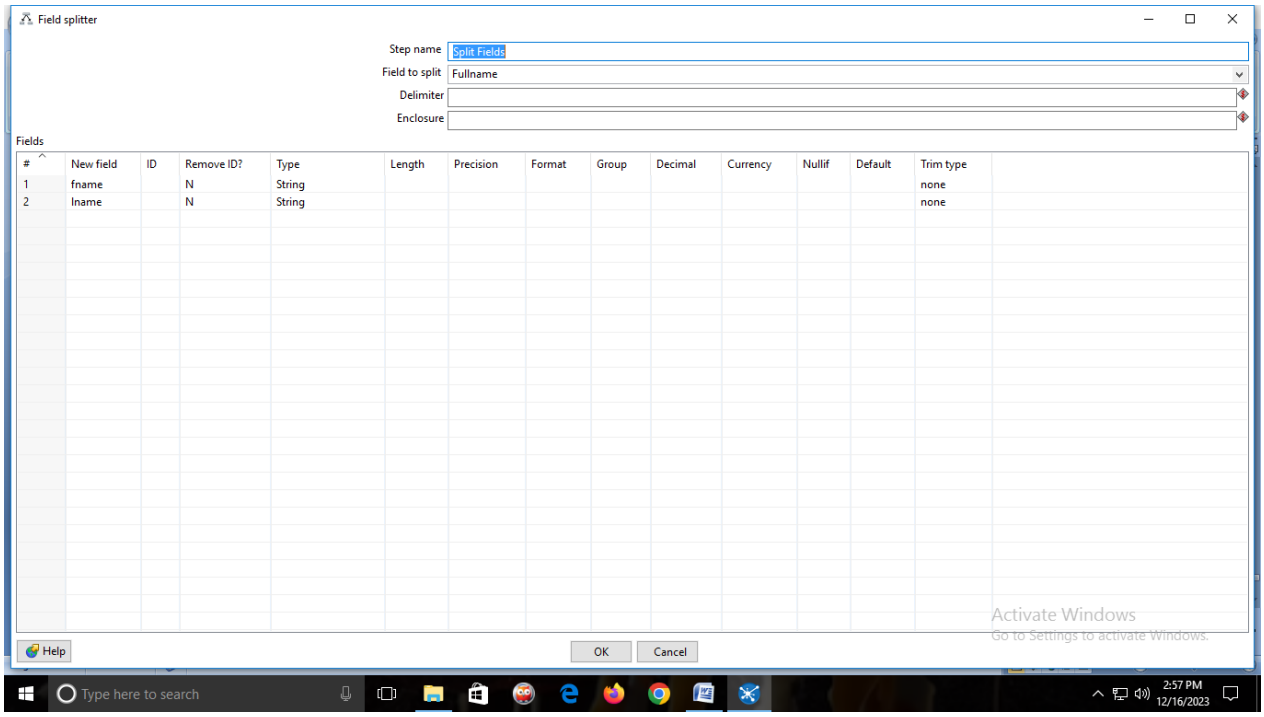


- →click on Preview rows→Ok

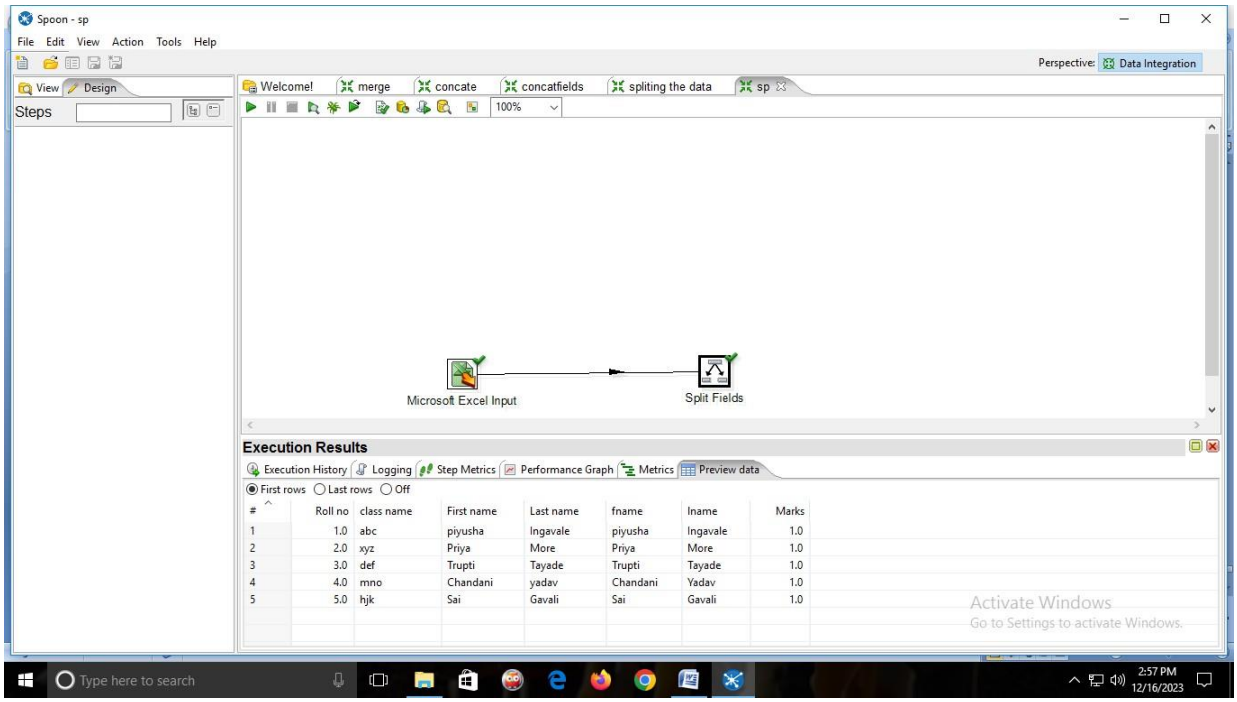

- From Transform folder→drag & drop→SPLIT FIELDS→and connect MICROSOFT EXCEL INPUT to SPLIT FIELDS.



- →Double click on SPLIT FIELD

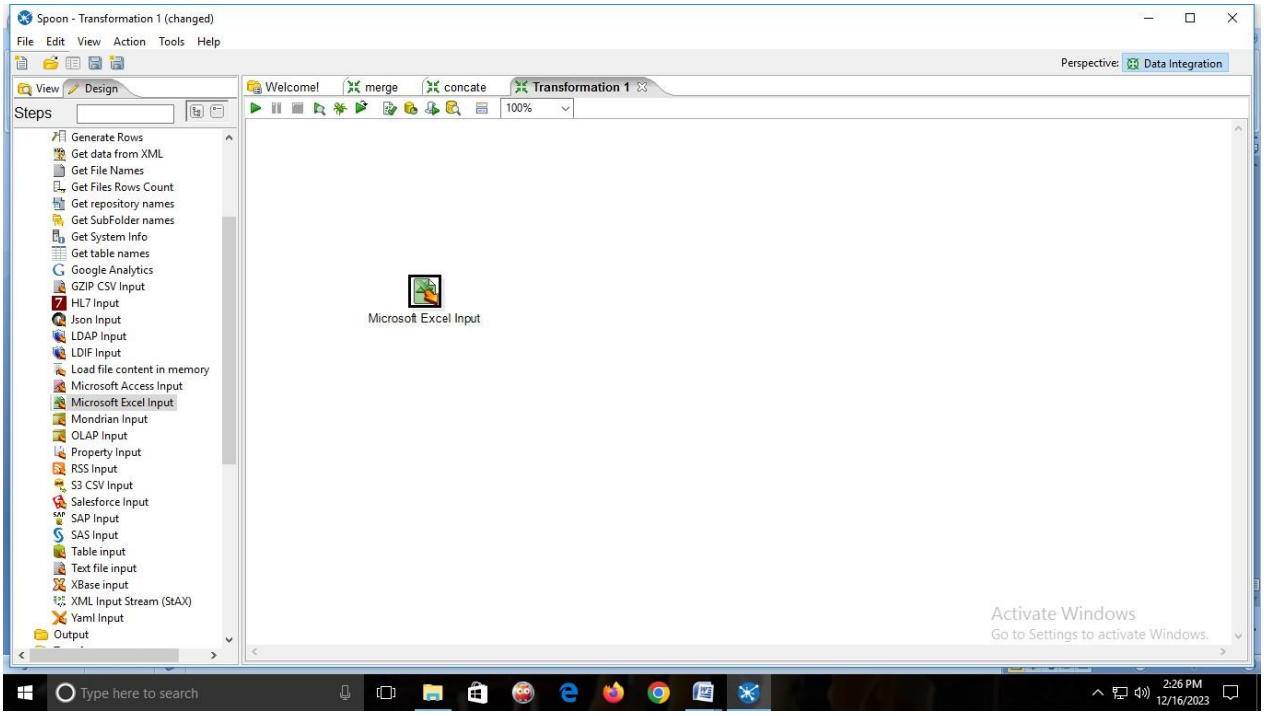



## OUTPUT:

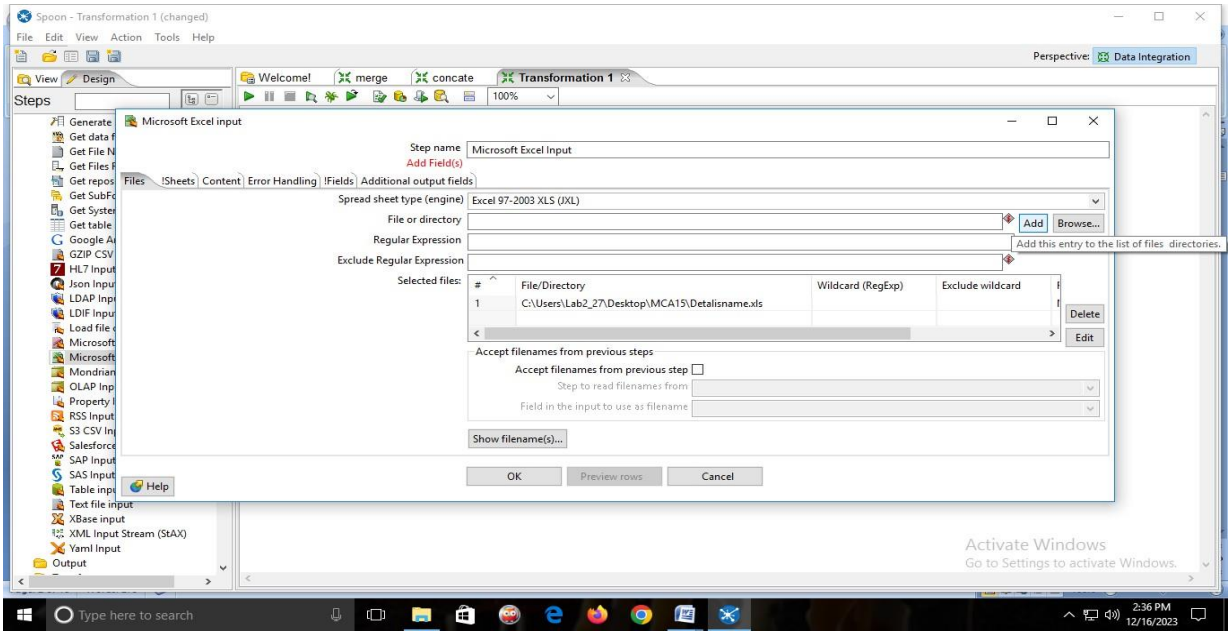


# F)AIM: Implementation of concatenation operations with Pentaho

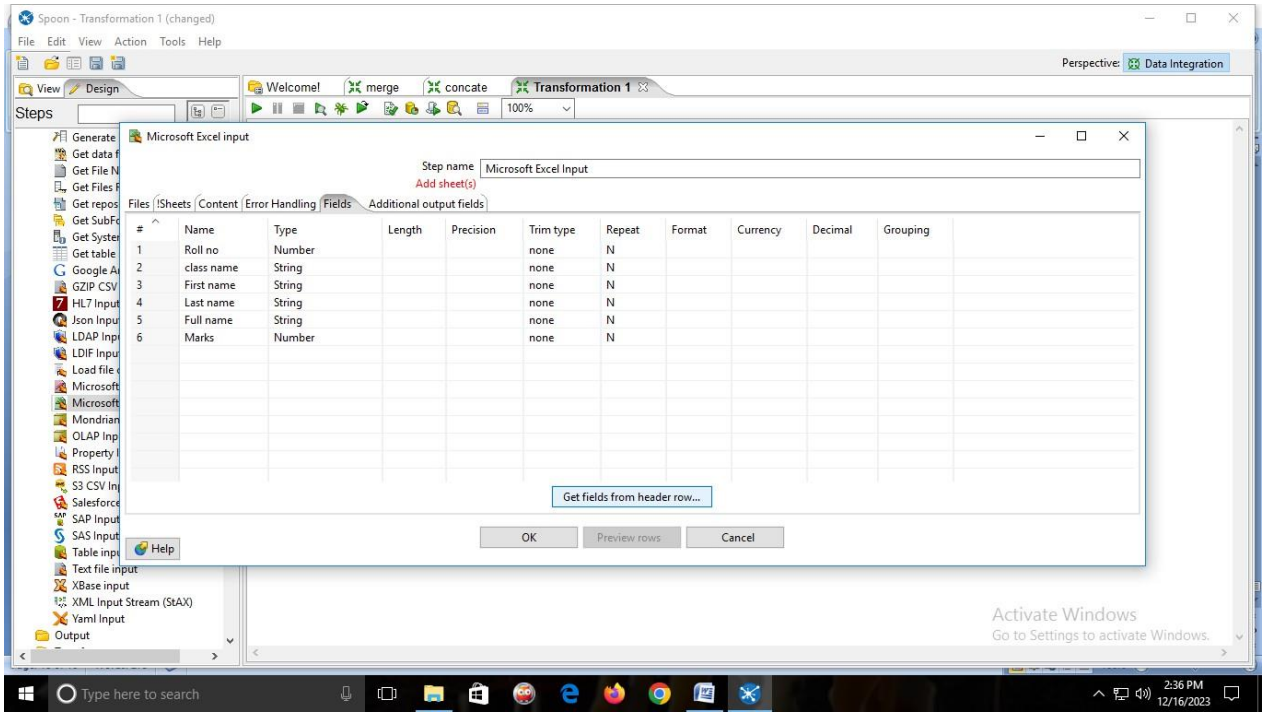
- create an Excel sheet with the fields firstname,lastname,fullname,rollno,marks
- From input folder→drag & drop→ MICROSOFT EXCEL INPUT



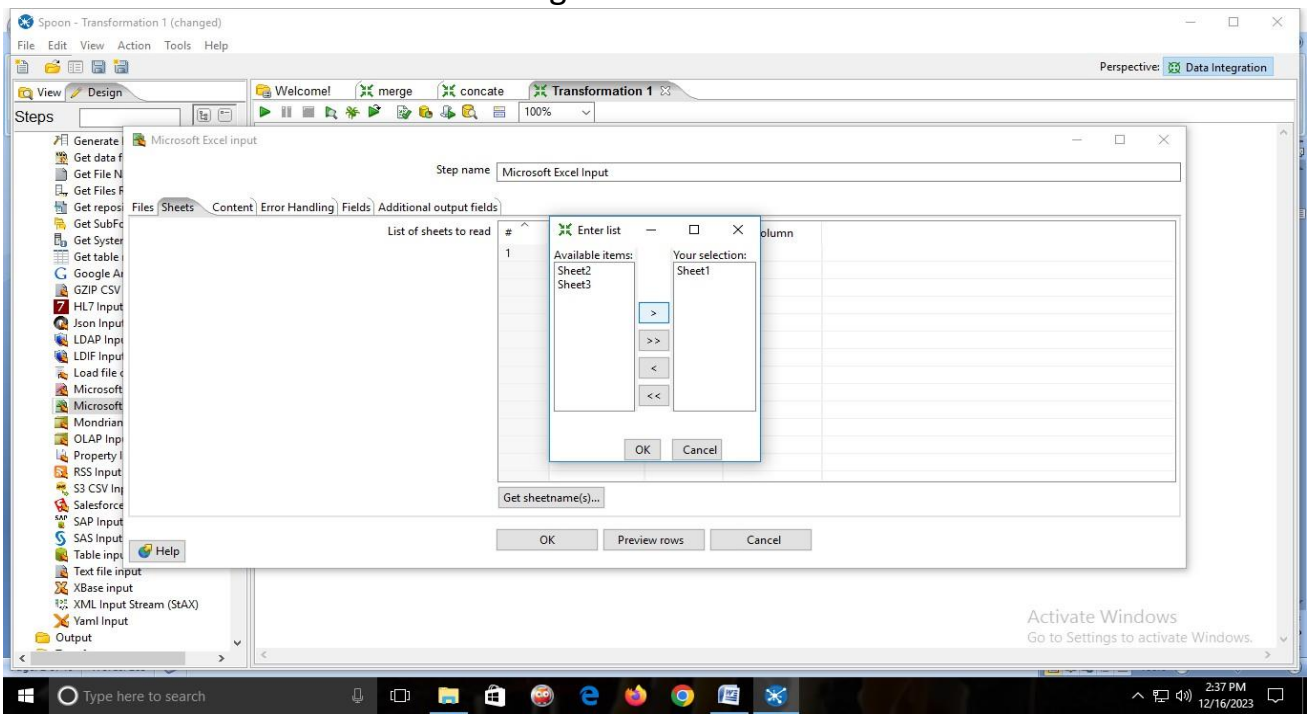

- Double click on it and add the Excel sheet.



- →click on get fields from header row

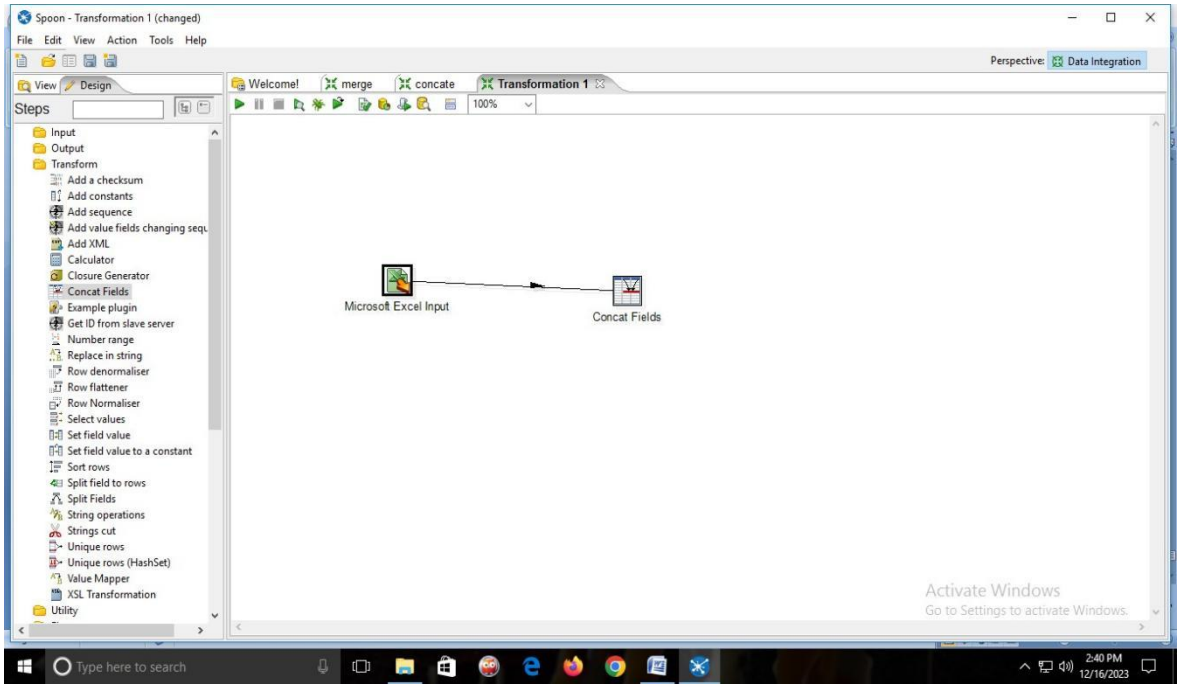



- →then click on sheets→click on get sheetname→add sheet1→ok

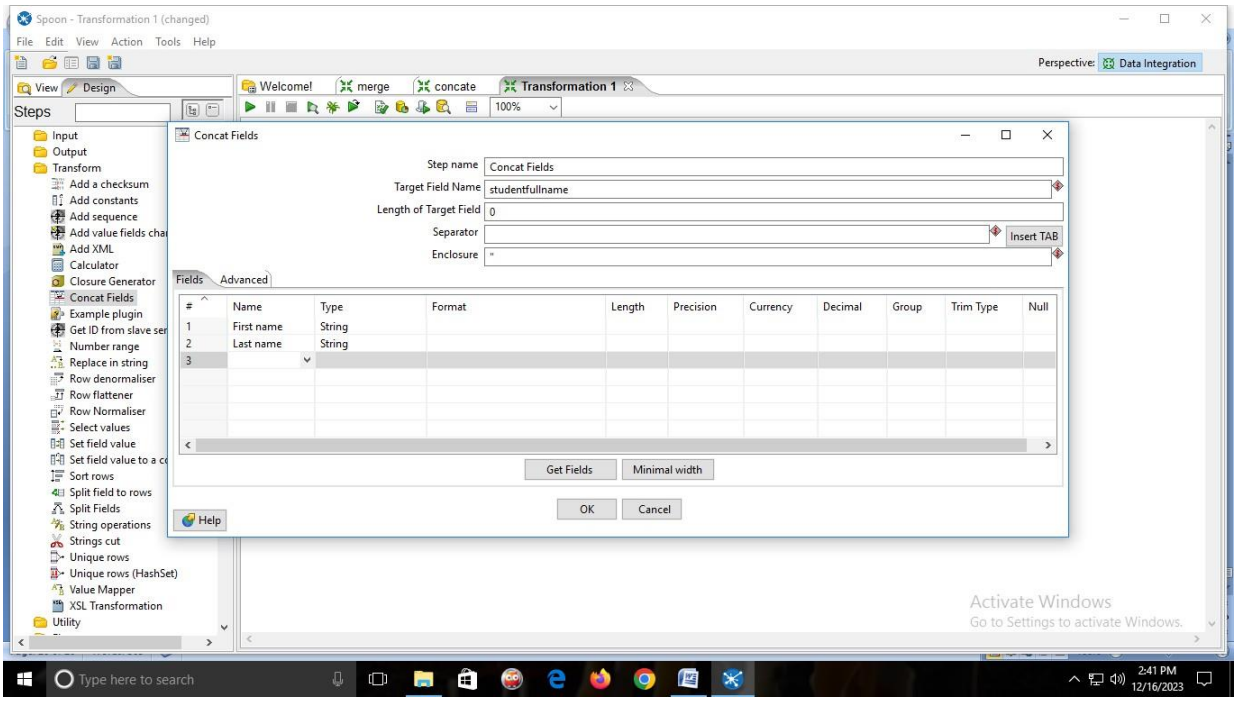


- →click on Preview rows→Ok

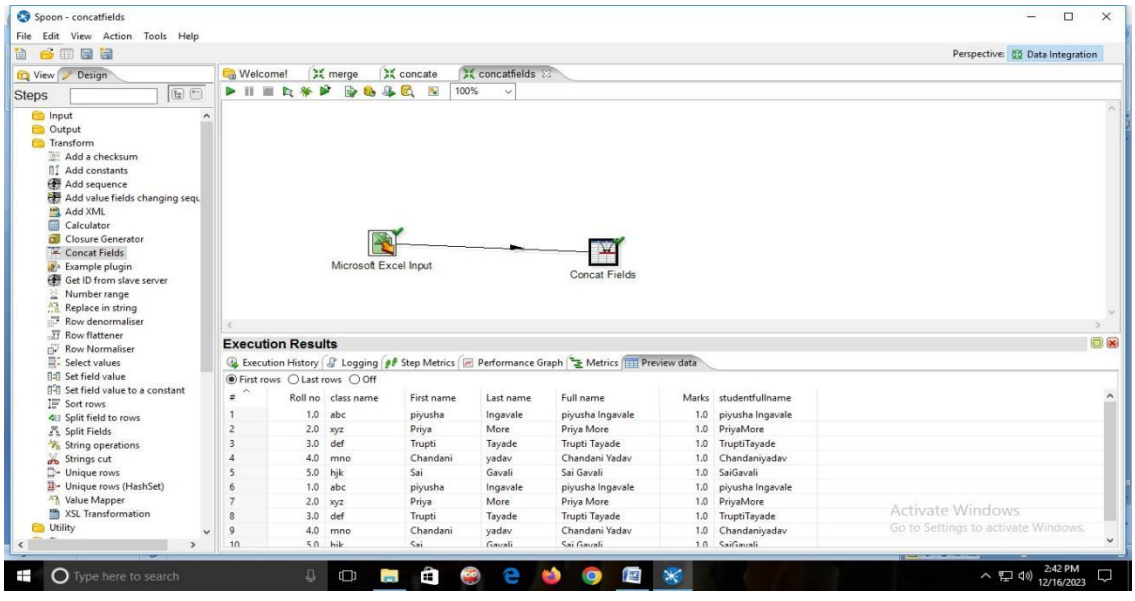

- from Transform folder→drag & drop→CONCAT FIELDS→and connect MICROSOFT EXCEL INPUT to CONCAT FIELDS.



- →double click on CONCAT FIELDS



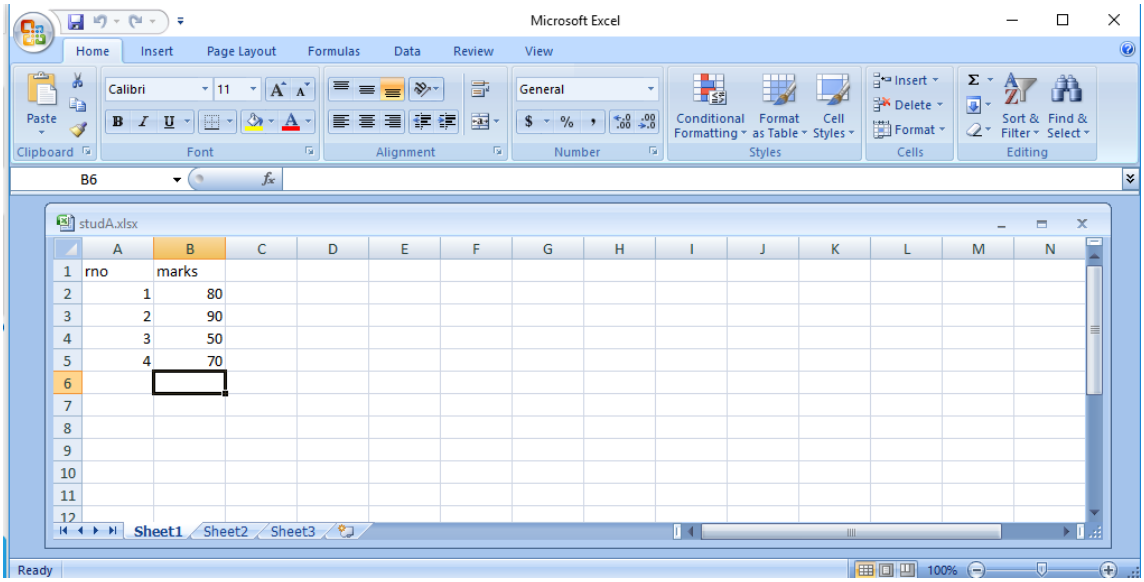

## OUTPUT:



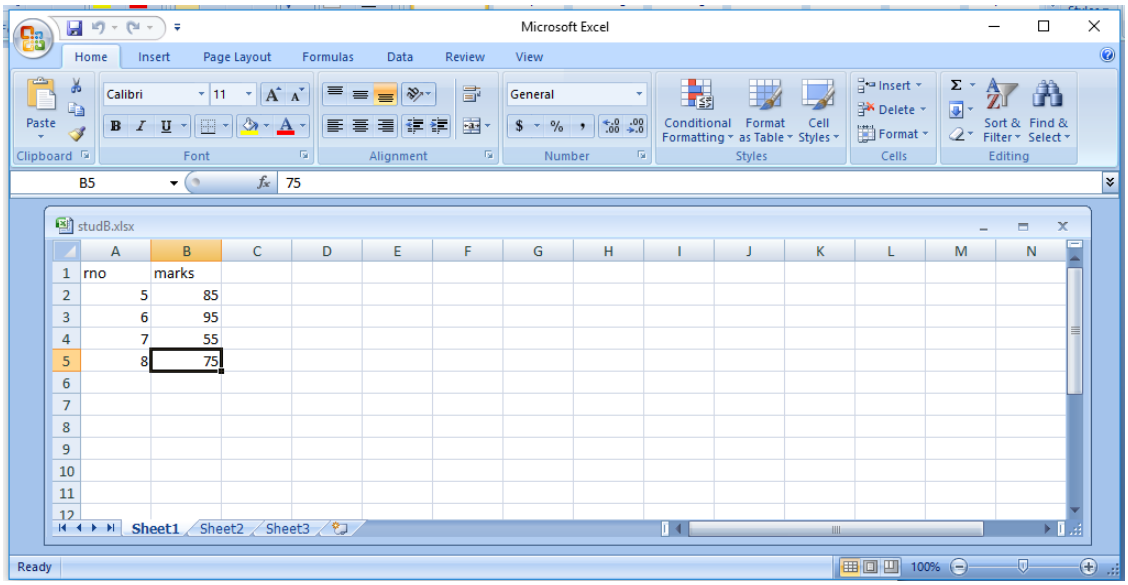
# G)AIM: Implementation of MERGE OPERATIONS with Pentaho

- Create Two Excel Sheets with the fields Roll no and Marks. →first Excel sheet will have roll no from 1-4 and Second Excel sheet will have roll no from 5-8.

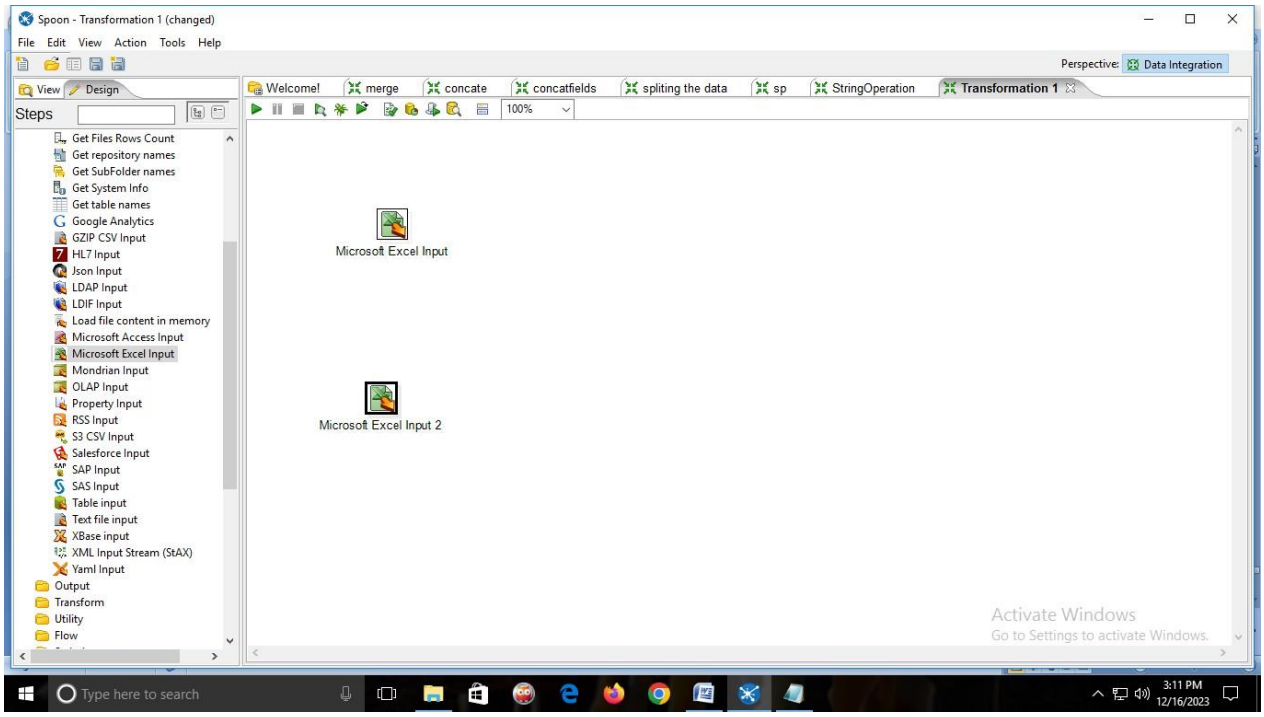
Excel 1




Excel 2

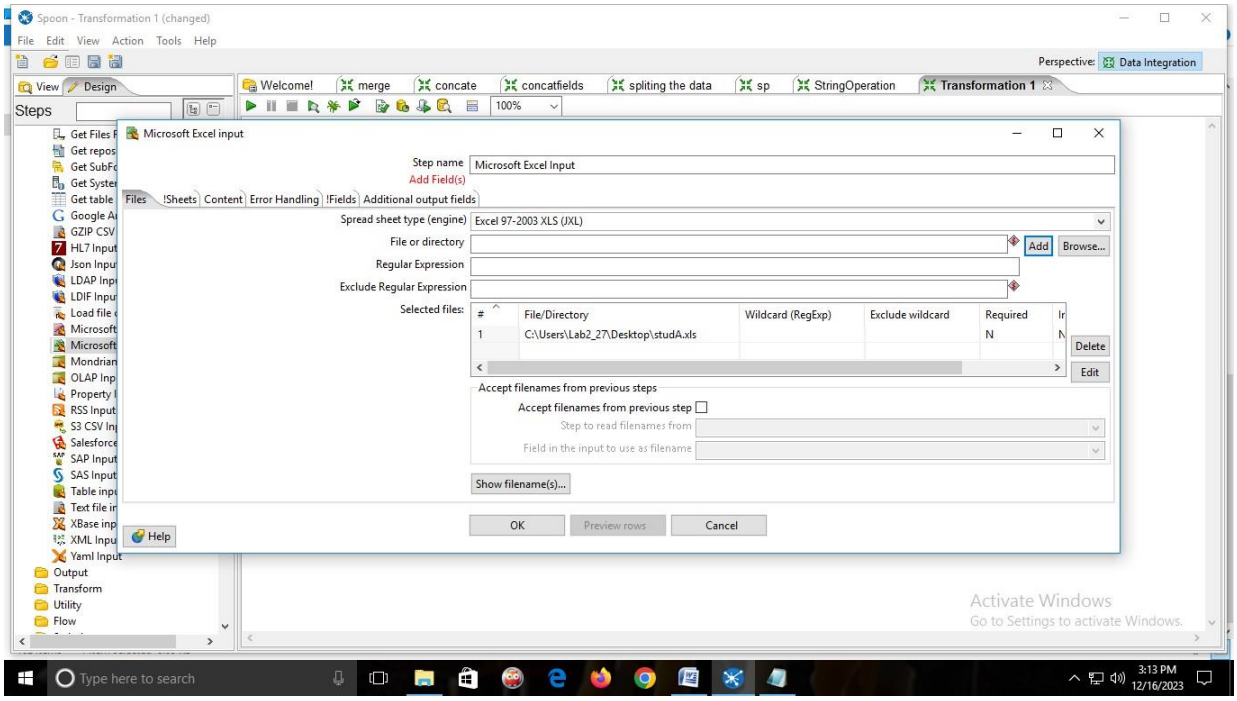


- From input folder→drag & drop Two→ MICROSOFT EXCEL INPUT

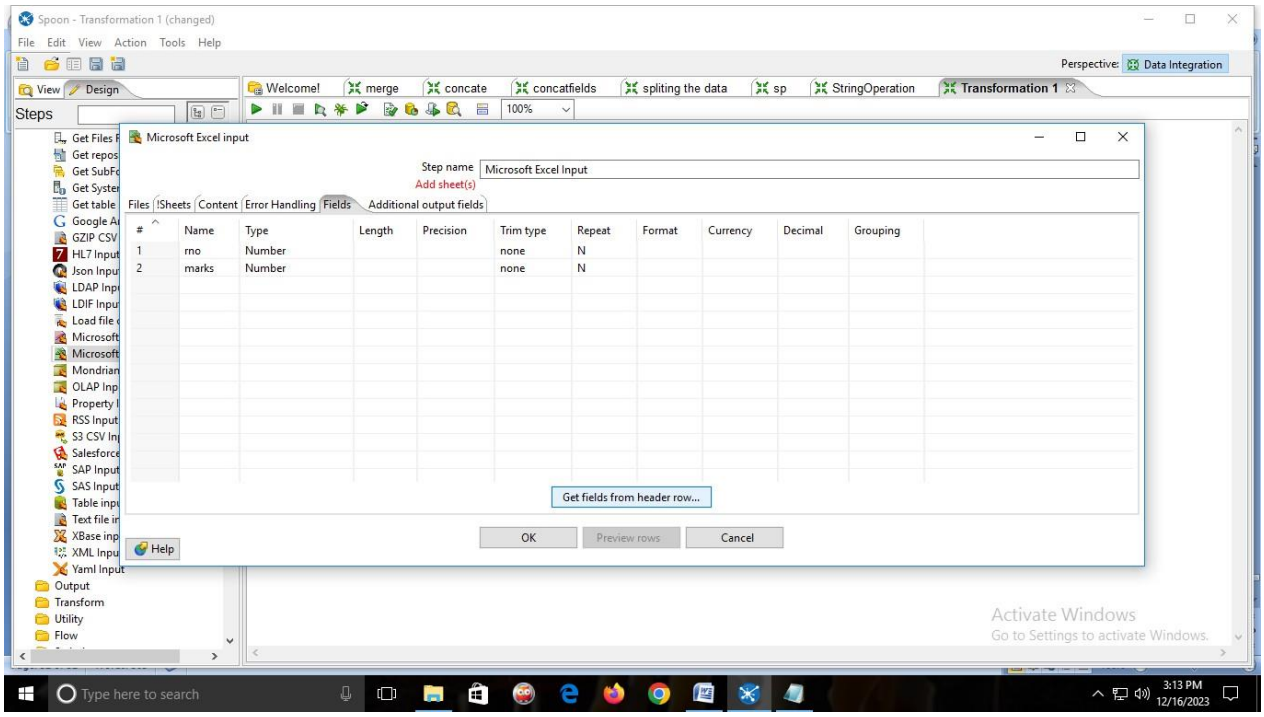



- →Add the EXCEL file in both the INPUT FIELD.

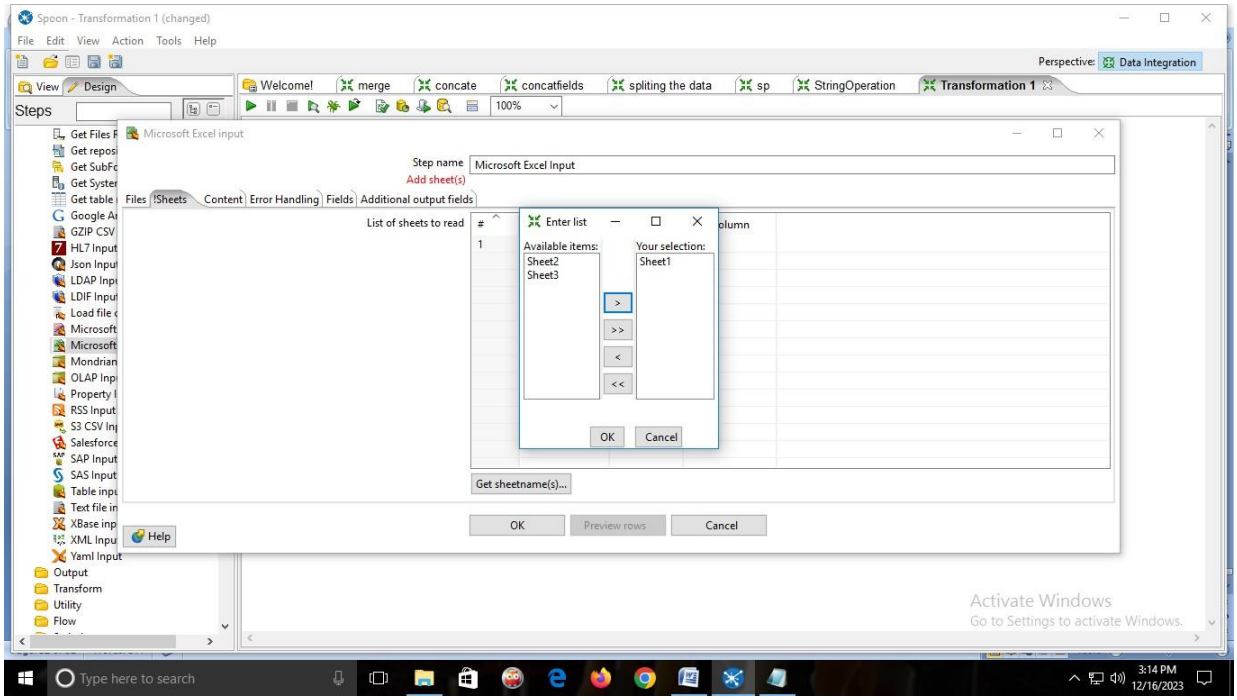
- →Double click on it and add the Excel sheet.



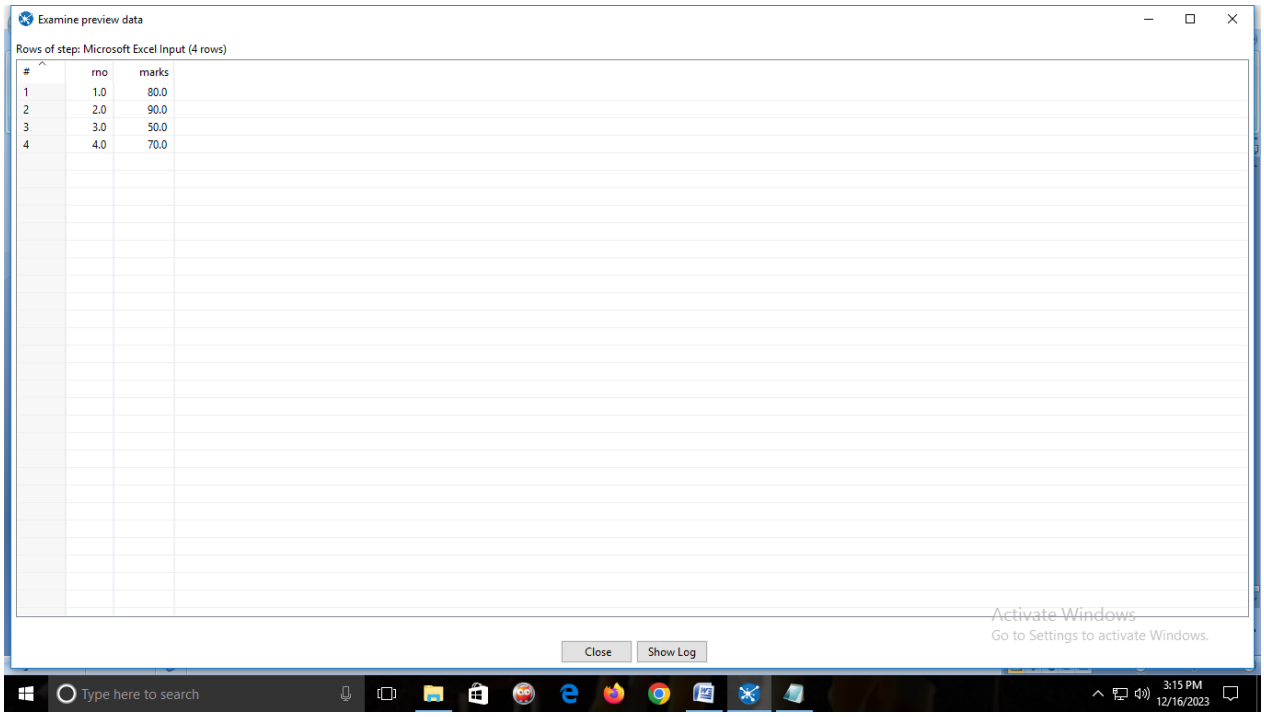
- →click on get fields from header row




- →then click on sheets→click on get sheetname→add sheet1→ok

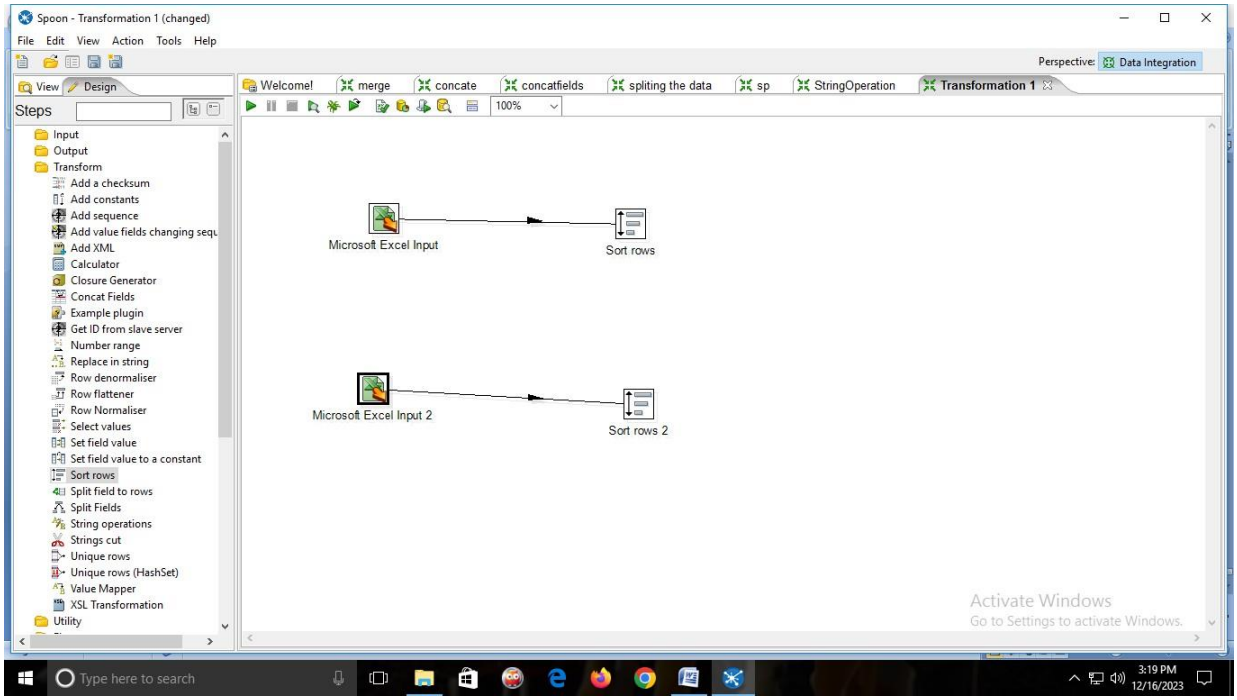


➢ →click on Preview rows→Ok

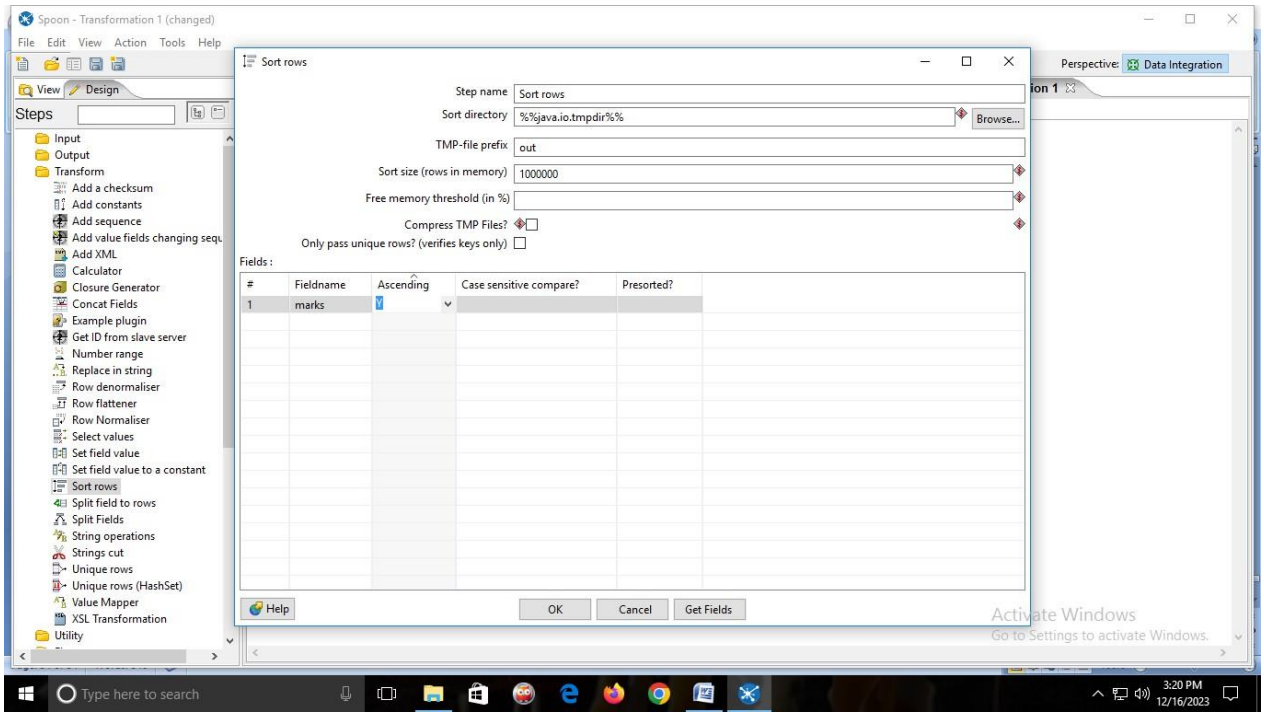



→Same process for the second Excel sheet.

➢ From Transform folder→drag & drop Two→SORT ROWS.And connect it with MICROSOFT EXCEL INPUT.

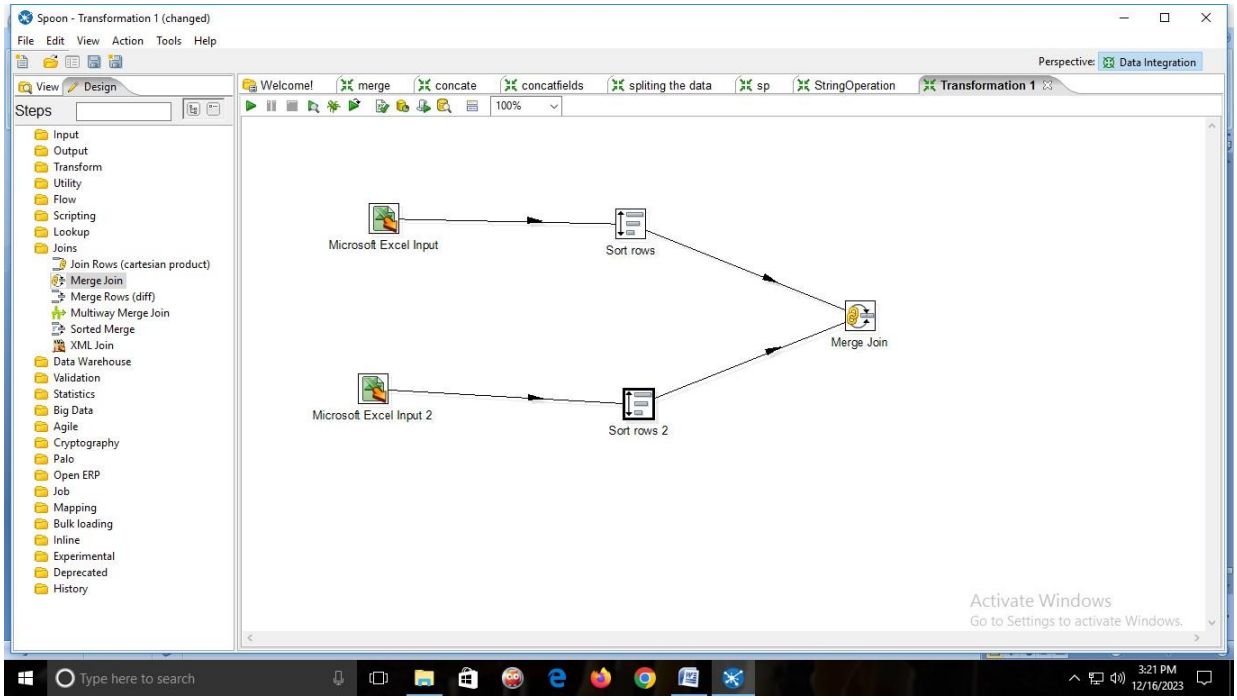


- →Double click on Sort rows

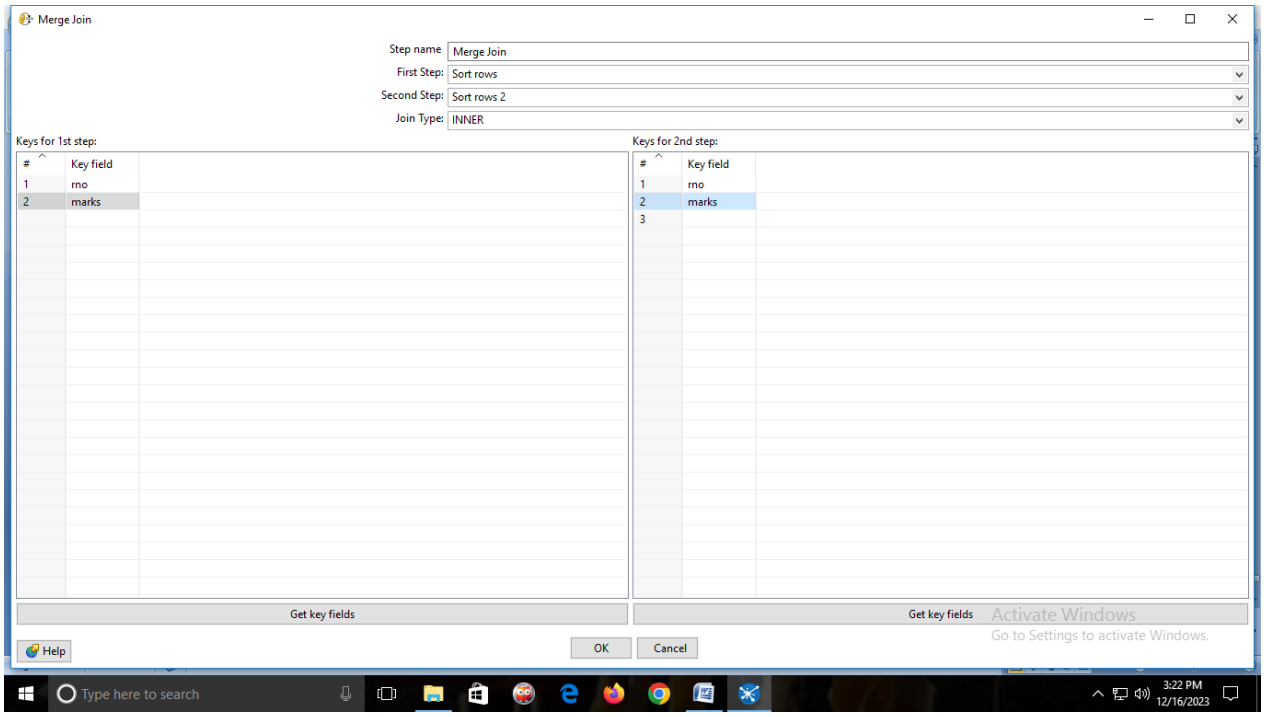



→Same for Second SORT ROWS.

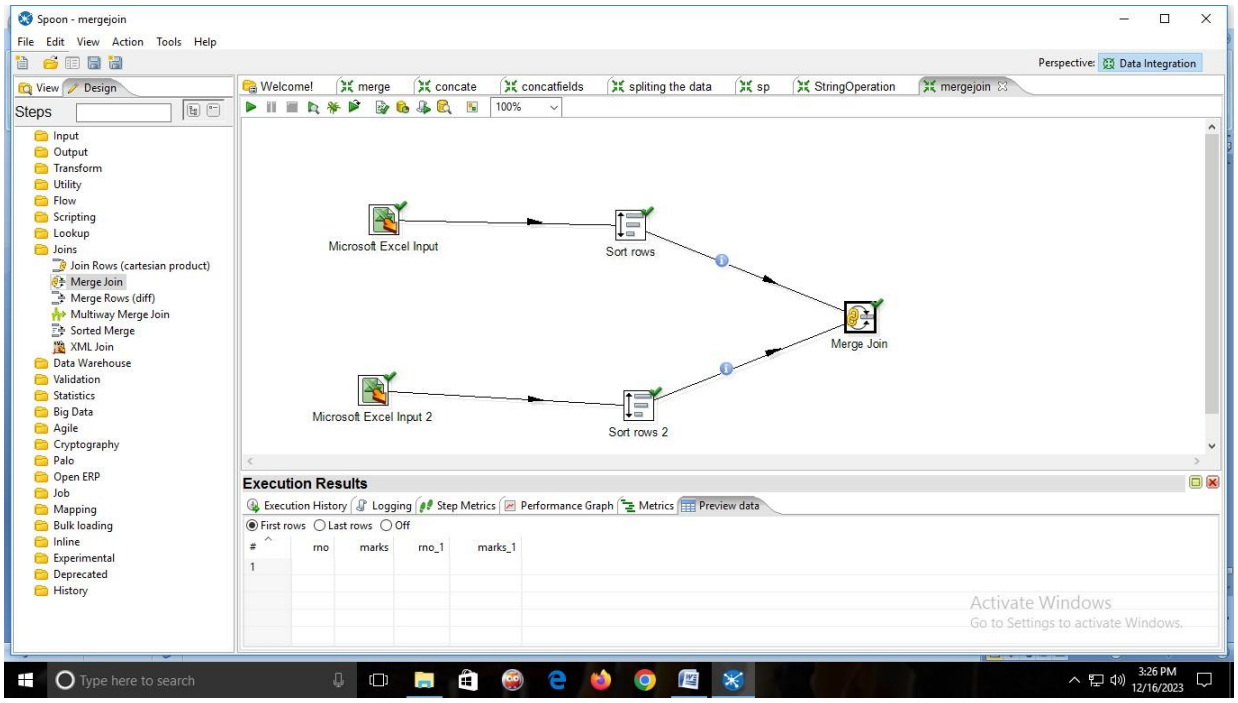
- from Join folder→drag & drop→MERGE JOIN. And connect it with both the SORT ROWS.



- →double click MERGE JOIN



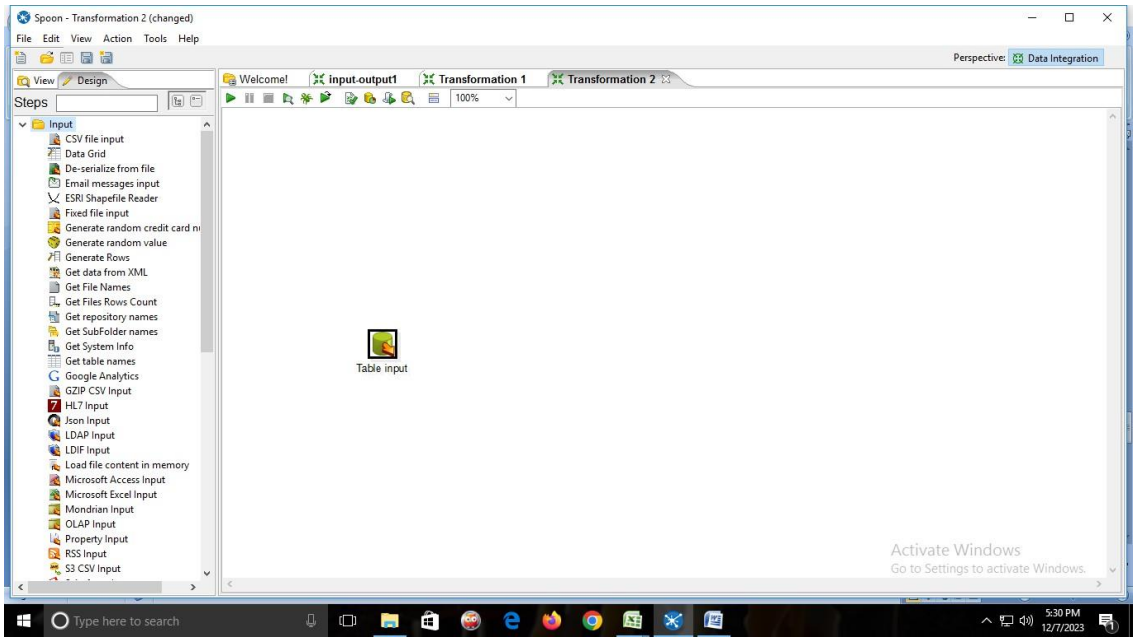

**OUTPUT:**



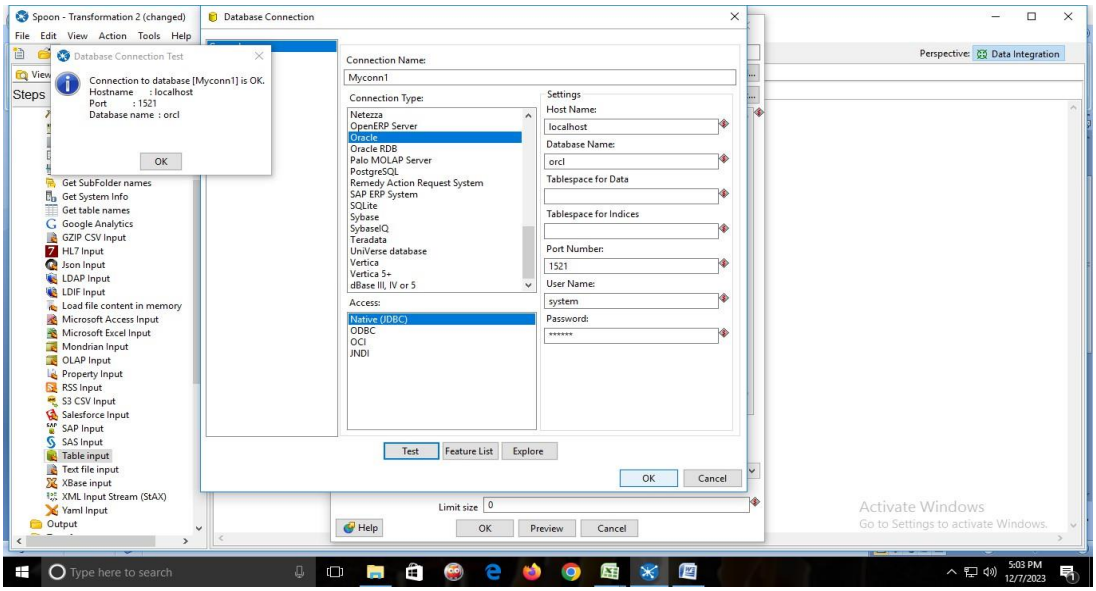
# H)AIM: Implementation of NULL IF operations with Pentaho.

## NULL IF

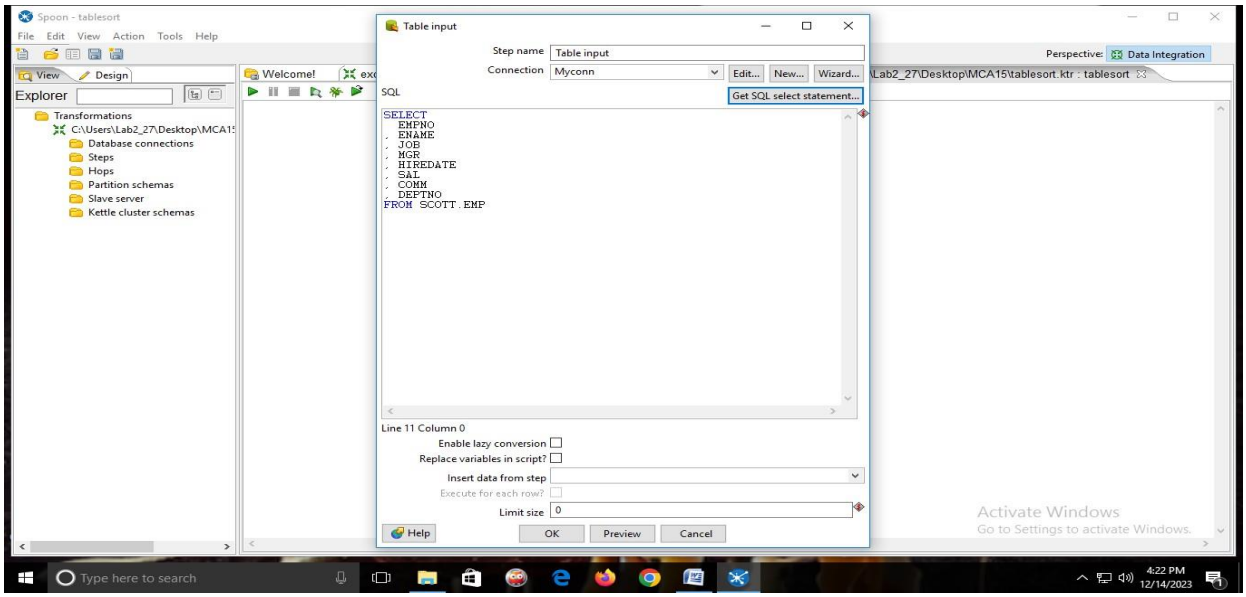
- from input folder→drag & drop table input.



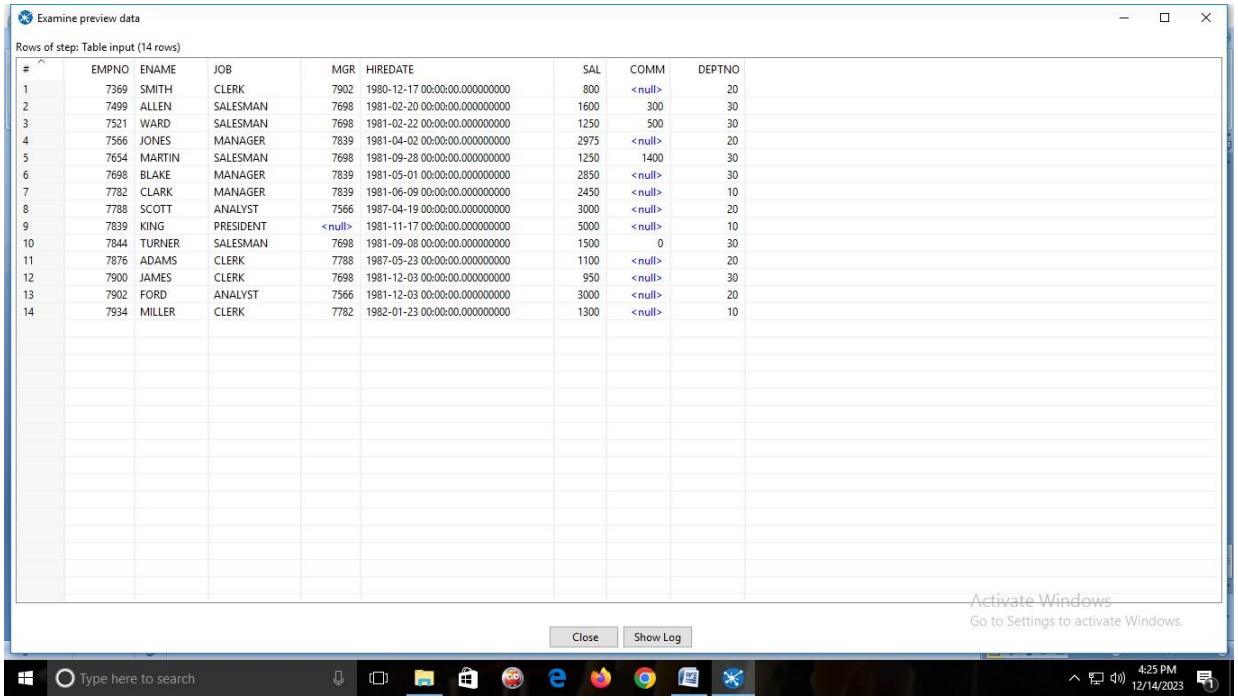

- New→connection name



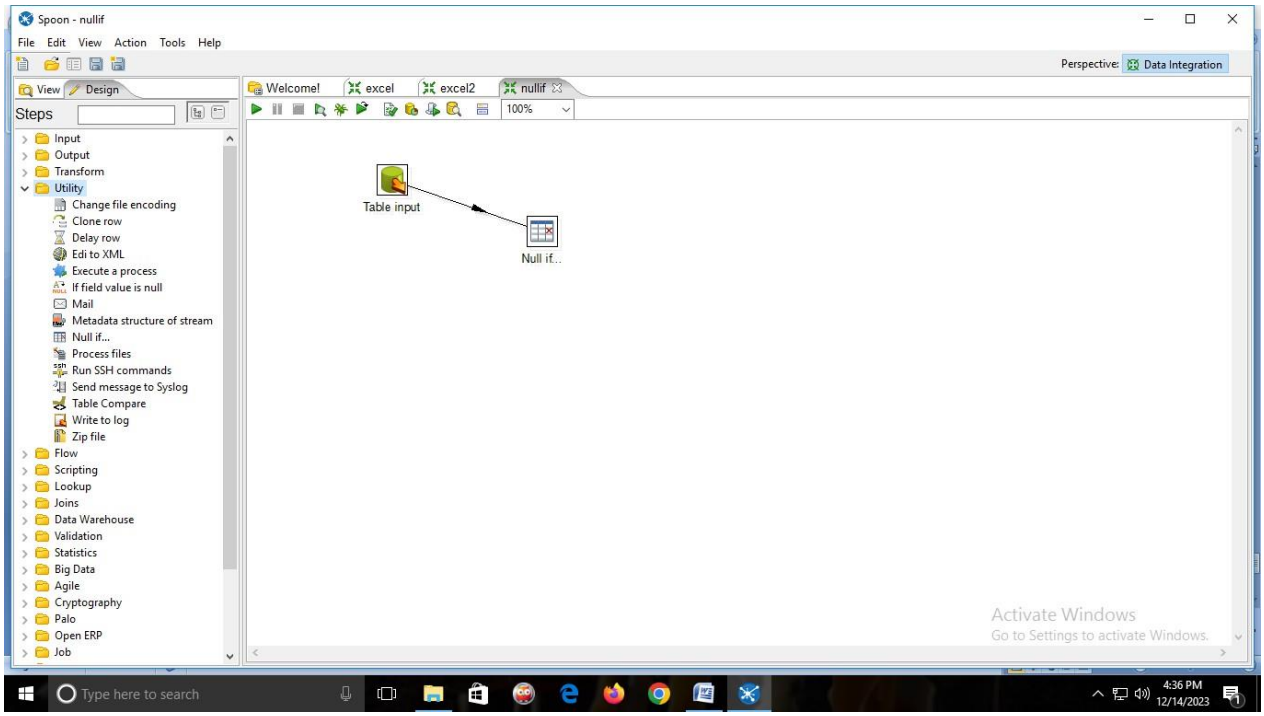
- Click on→GET SQL SELECT STATEMENT→and select the table→then click on preview .



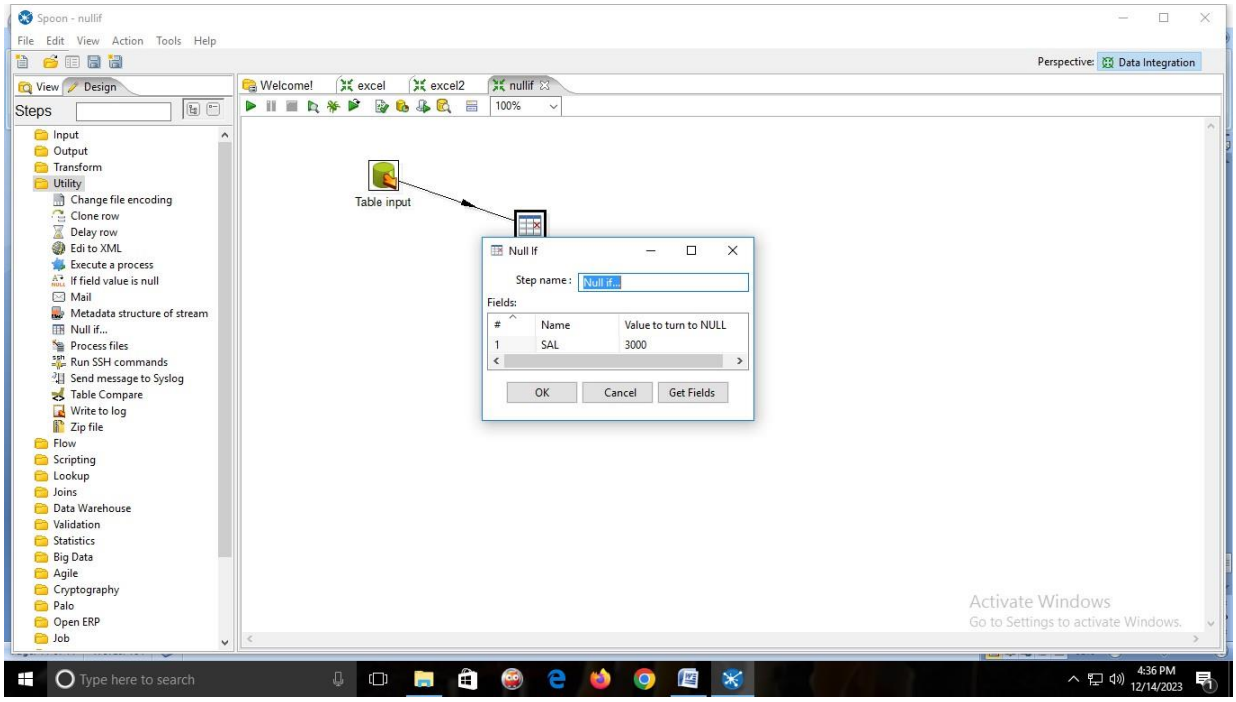

- Preview table



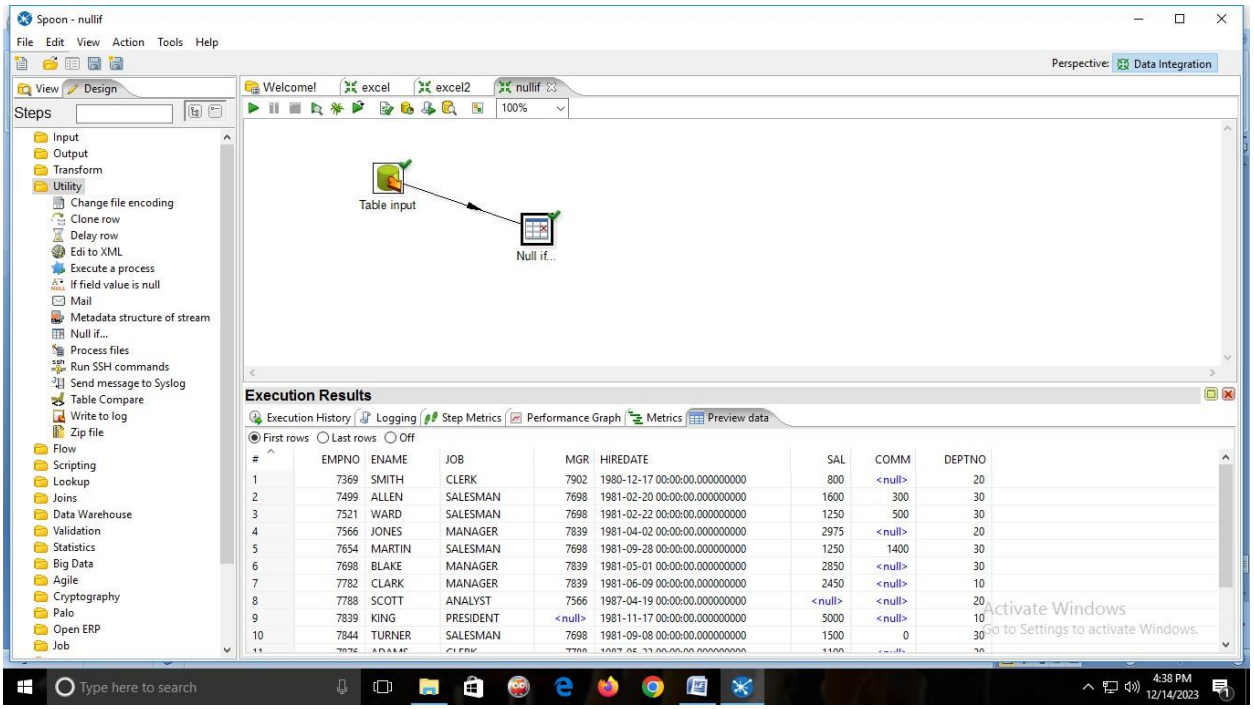
- From Utility folder→drag & drop NULLIF and connect table input to nullif




- Double click on null if and give name and value to turn to NULL.



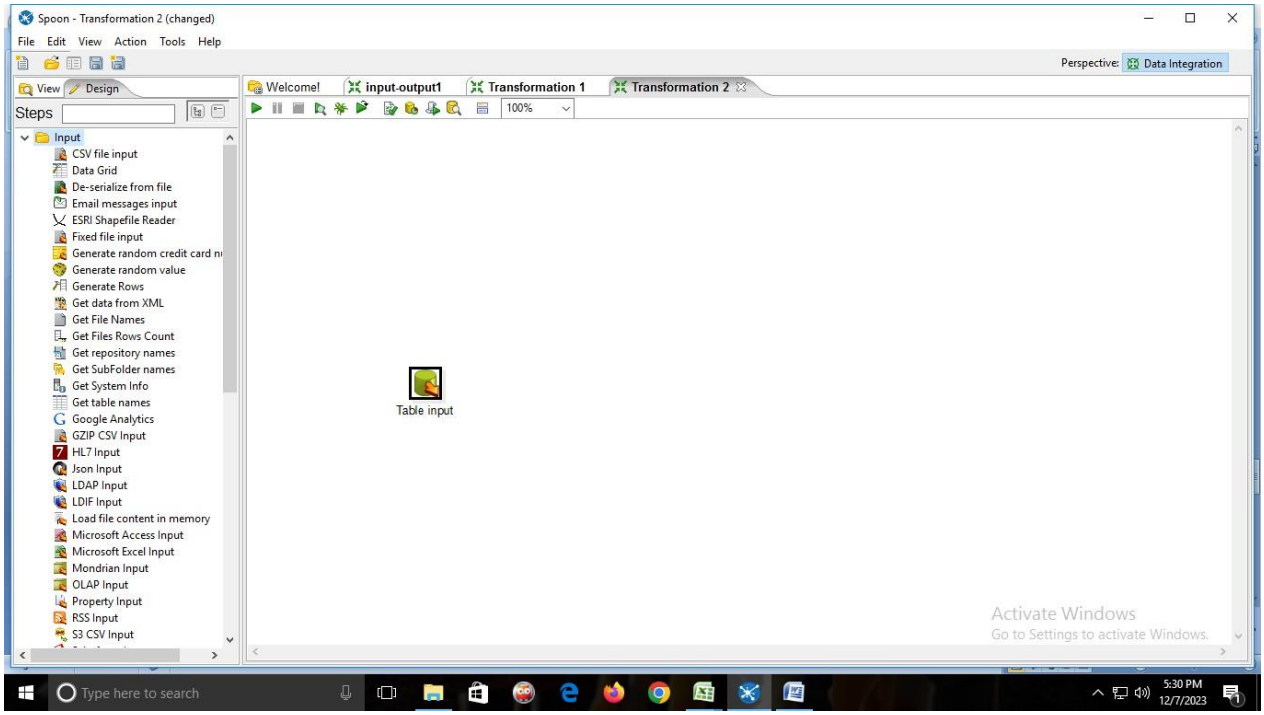
# OUTPUT:


Spoon - nullif
File Edit View Action Tools Help
Perspective: Data Integration
View Design
Steps
Input
Output
Transform
Utility
Change file encoding
Clone row
Delay row
Edi to XML
Execute a process
If field value is null
Mail
Metadata structure of stream
Null if...
Process files
Run SSH commands
Send message to Syslog
Table Compare
Write to log
Zip file
Flow
Scripting
Lookup
Joins
Data Warehouse
Validation
Statistics
Big Data
Agile
Cryptography
Palo
Open ERP
Job
Welcome!
excel
excel2
nullif
100%
Table input
Null if...
Execution Results
Execution History
Logging
Step Metrics
Performance Graph
Metrics
Preview data
First rows
Last rows
Off
# EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
1 7369 SMITH CLERK 7902 1980-12-17 00:00:00.000000000 800 <null> 20
2 7499 ALLEN SALESMAN 7698 1981-02-20 00:00:00.000000000 1600 300 30
3 7521 WARD SALESMAN 7698 1981-02-22 00:00:00.000000000 1250 500 30
4 7566 JONES MANAGER 7839 1981-04-02 00:00:00.000000000 2975 <null> 20
5 7654 MARTIN SALESMAN 7698 1981-09-28 00:00:00.000000000 1250 1400 30
6 7698 BLAKE MANAGER 7839 1981-05-01 00:00:00.000000000 2850 <null> 30
7 7782 CLARK MANAGER 7839 1981-06-09 00:00:00.000000000 2450 <null> 10
8 7788 SCOTT ANALYST 7566 1987-04-19 00:00:00.000000000 <null> <null> 20
9 7839 KING PRESIDENT <null> 1981-11-17 00:00:00.000000000 5000 <null> 10
10 7844 TURNER SALESMAN 7698 1981-09-08 00:00:00.000000000 1500 0 30
Activate Windows
Go to Settings to activate Windows.
Type here to search
4:38 PM
12/14/2023

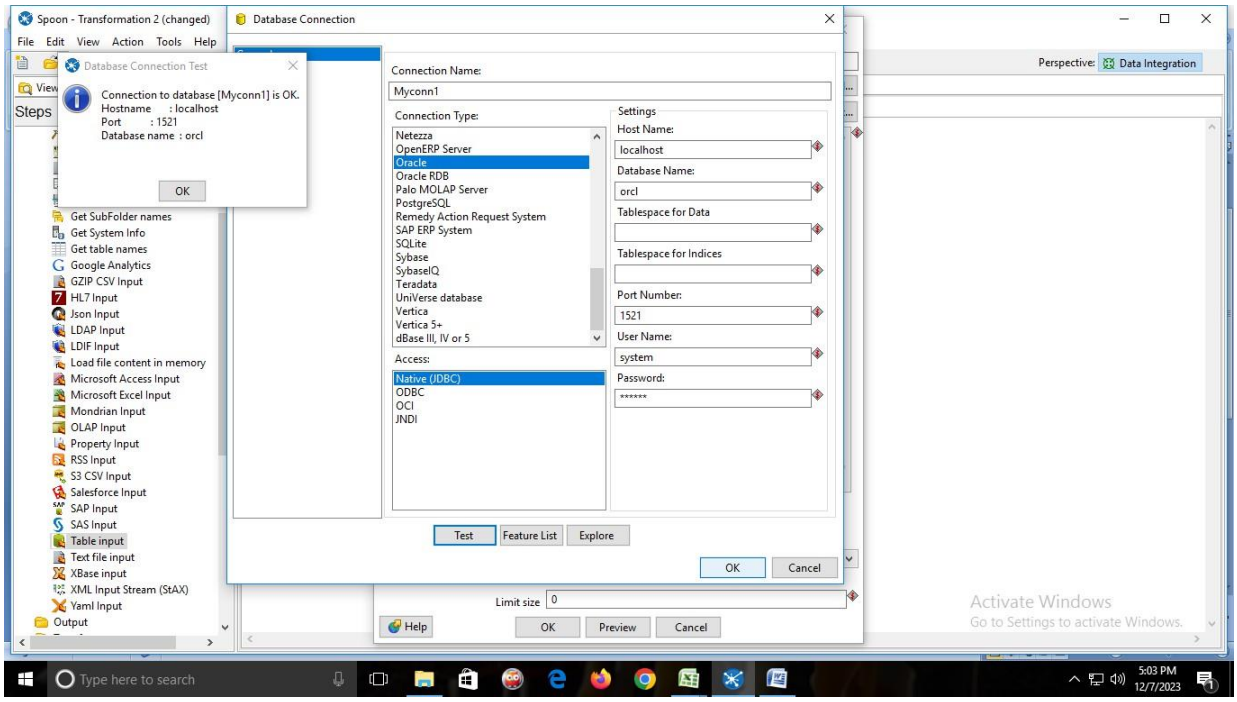
# I)AIM: Implementation of filter operations with Pentaho

## FILTER

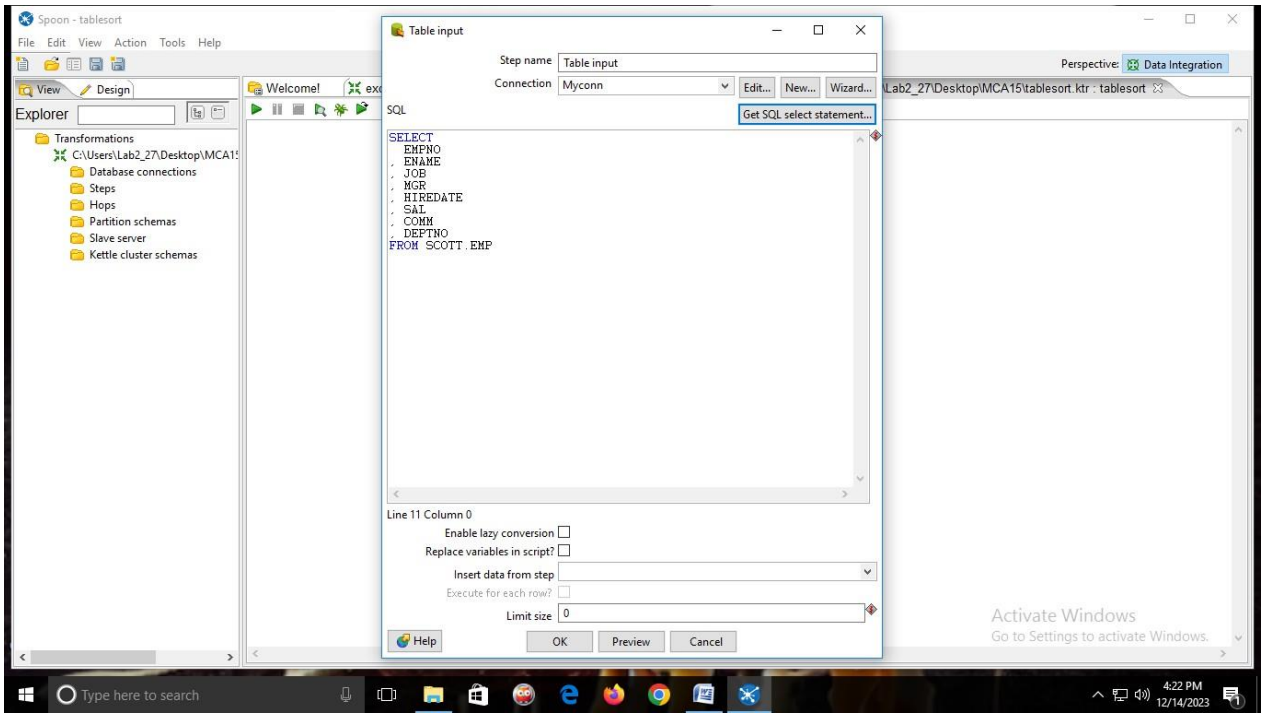
- from input folder→drag & drop table input.



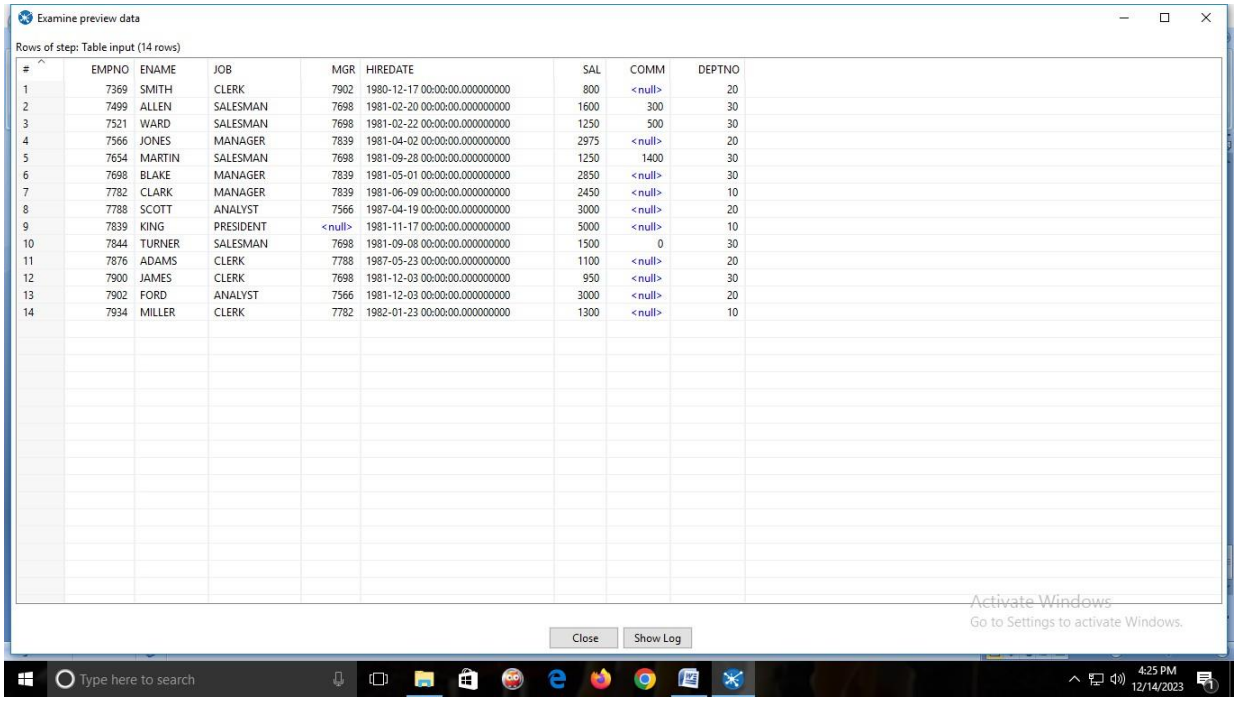

- New→connection name



- Click on→GET SQL SELECT STATEMENT→and select the table→then click on preview .

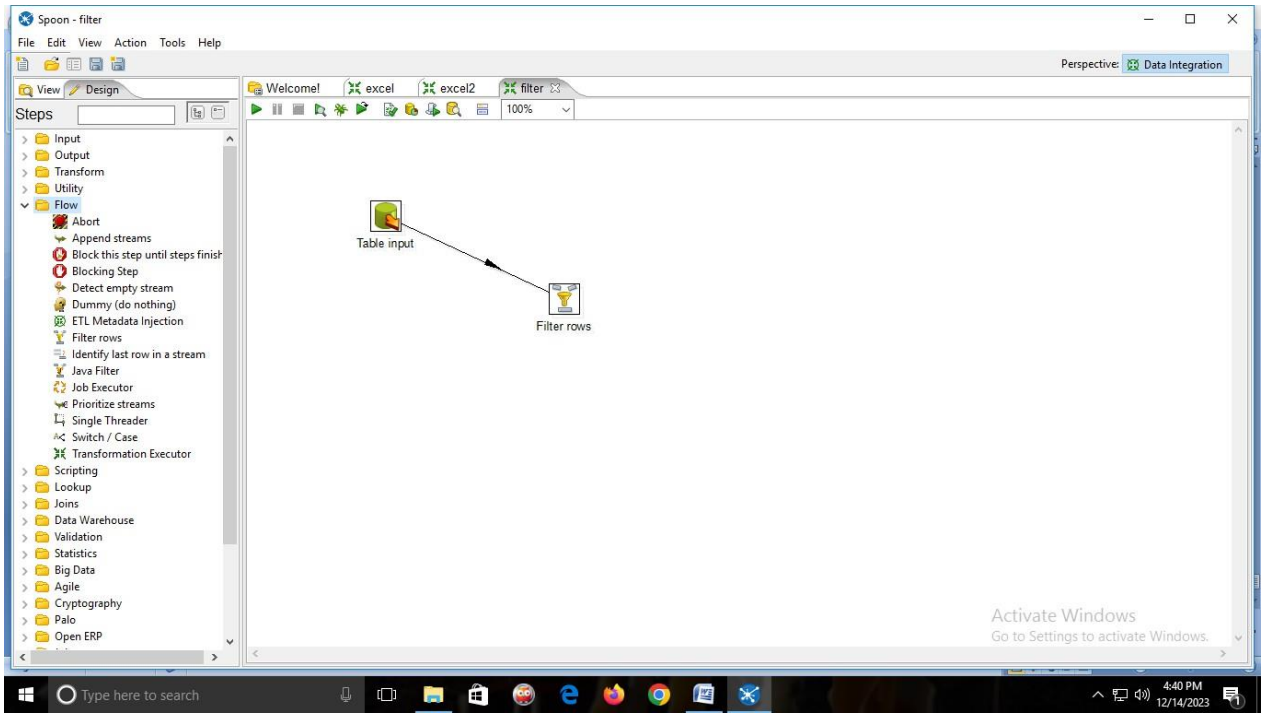



- →Preview tbale

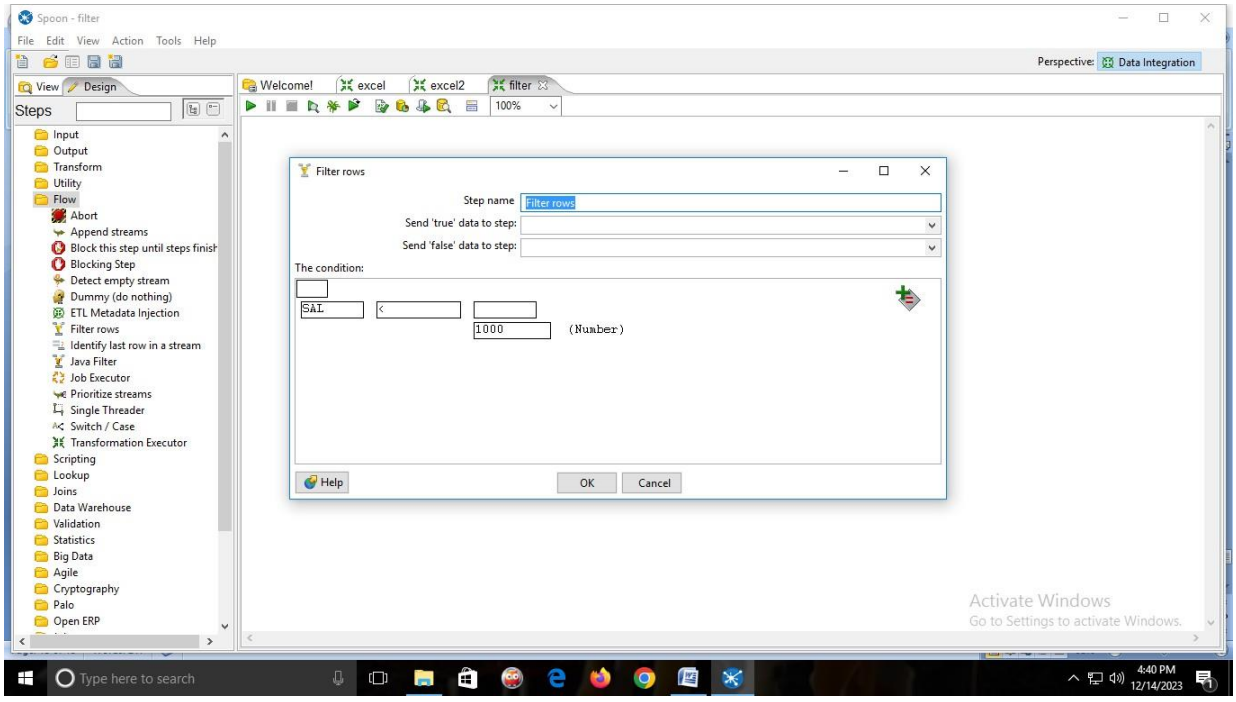



| # | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|---|
| 1 | 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00.000000000 | 800 | <null> | 20 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000000000 | 1600 | 300 | 30 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000000000 | 1250 | 500 | 30 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000000000 | 2975 | <null> | 20 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000000000 | 1250 | 1400 | 30 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000000000 | 2850 | <null> | 30 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000000000 | 2450 | <null> | 10 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00.000000000 | 3000 | <null> | 20 |
| 9 | 7839 | KING | PRESIDENT | <null> | 1981-11-17 00:00:00.000000000 | 5000 | <null> | 10 |
| 10 | 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00.000000000 | 1500 | 0 | 30 |
| 11 | 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00.000000000 | 1100 | <null> | 20 |
| 12 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000000000 | 950 | <null> | 30 |
| 13 | 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00.000000000 | 3000 | <null> | 20 |
| 14 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000000000 | 1300 | <null> | 10 |

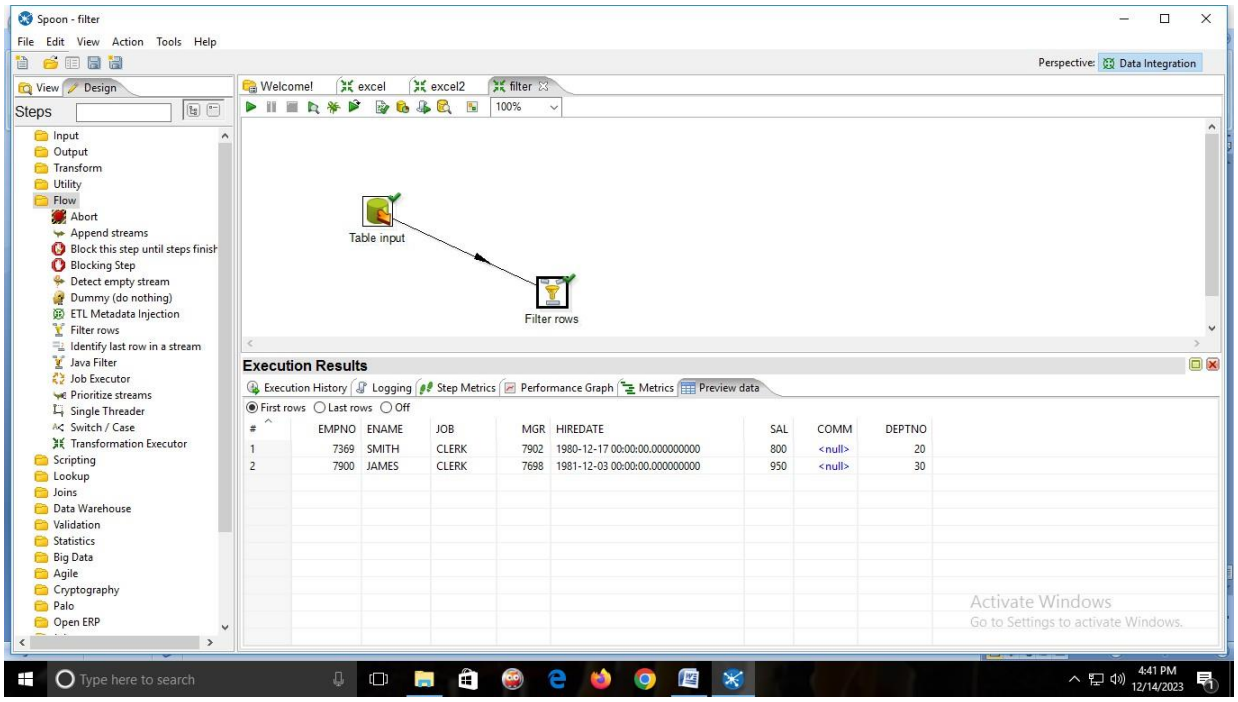
- From Flow folder → drag & drop FILTER ROWS and connect table input to filter rows.




- Double click on filter rows and fill the values and click on OK.



# OUTPUT:


Spoon - filter
File Edit View Action Tools Help
Perspective: Data Integration
View Design
Steps
Input
Output
Transform
Utility
Flow
Abort
Append streams
Block this step until steps finish
Blocking Step
Detect empty stream
Dummy (do nothing)
ETL Metadata Injection
Filter rows
Identify last row in a stream
Java Filter
Job Executor
Prioritize streams
Single Threader
Switch / Case
Transformation Executor
Scripting
Lookup
Joins
Data Warehouse
Validation
Statistics
Big Data
Agile
Cryptography
Palo
Open ERP
Welcome!
excel
excel2
filter
100%
Table input
Filter rows
Execution Results
Execution History
Logging
Step Metrics
Performance Graph
Metrics
Preview data
First rows
Last rows
Off
# EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
1 7369 SMITH CLERK 7902 1980-12-17 00:00:00.000000000 800 <null> 20
2 7900 JAMES CLERK 7698 1981-12-03 00:00:00.000000000 950 <null> 30
Activate Windows
Go to Settings to activate Windows.
Type here to search
4:41 PM
12/14/2023

# Practical 6

## Aim:Implementation of Basic R commands data acquisition

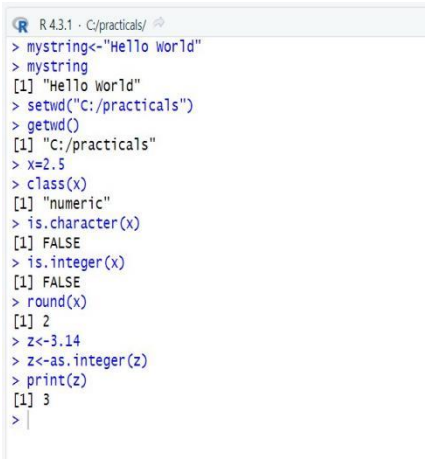
mystring<-"Hello World" mystring setwd("C:/practicals") getwd() x=2.5

class(x) is.character(x) is.integer(x)

round(x) z<-3.14 z<-

as.integer(z) print(z)

## Output:

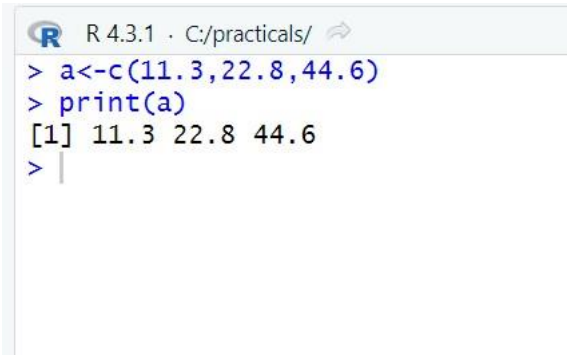



## ➢ using c()function()

a<-c(11.3,22.8,44.6)

print(a)

**output:**



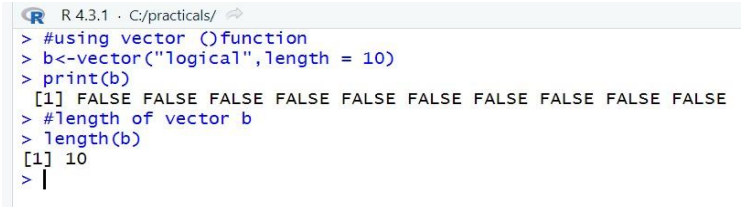

## ➢ using vector ()

function b<-vector("logical",length = 10) print(b)

**#length of vector b**

length(b)

**output:**



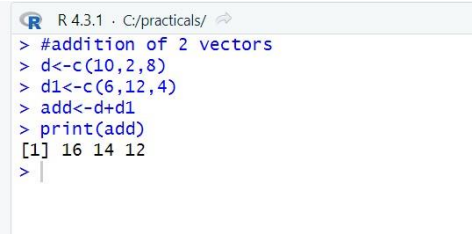

## ➢ addition of 2 vectors

d<-c(10,2,8) d1<-c(6,12,4)

add<-d+d1

print(add)

**Output:**



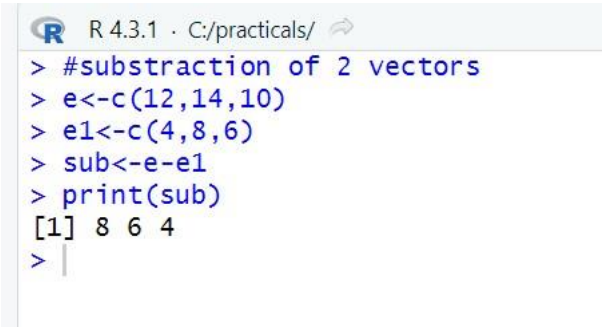

## ➢ subtraction of 2 vectors

e<-c(12,14,10)

e1<-c(4,8,6)

sub<-e-e1 print(sub)

**Output:**



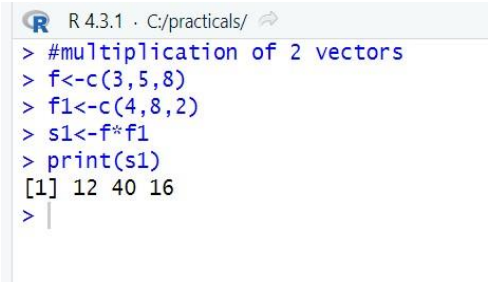

## ➢ multiplication of 2 vectors

f<-c(3,5,8)

f1<-c(4,8,2)

s1<-f*f1 print(s1)

**Output:**




## ➢ division of 2 vectors

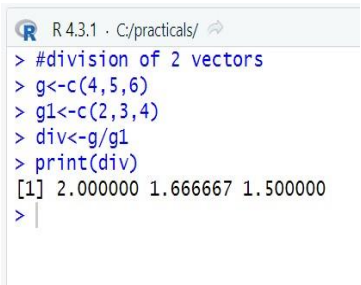
g<-c(4,5,6)

g1<-c(2,3,4)

div<-g/g1

print(div)

**Output:**

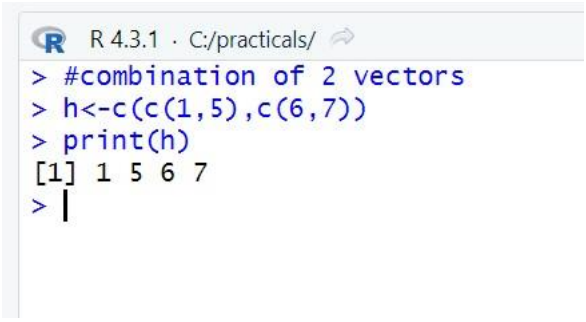



## ➢ combination of 2 vectors

h<-c(c(1,5),c(6,7))

print(h)

**output:**

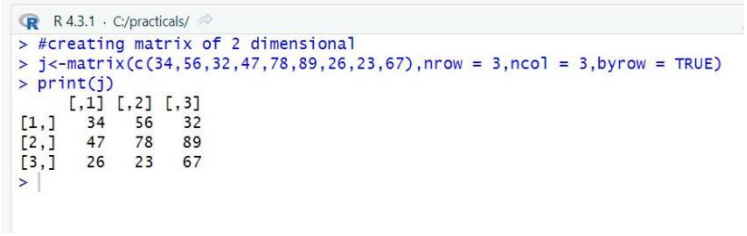



## ➢ creating matrix of 2 dimensional

j<-matrix(c(34,56,32,47,78,89,26,23,67),nrow = 3,ncol = 3,byrow = TRUE)

print(j)

**output:**

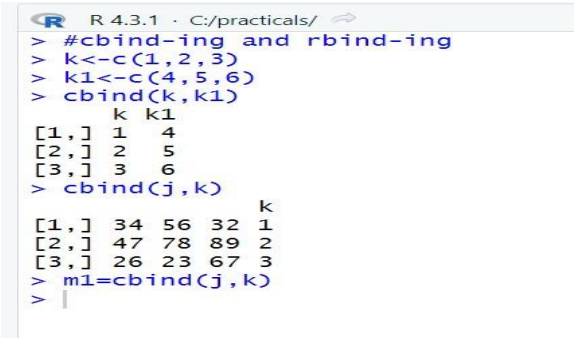



## ➢ cbind-ing and rbind-ing

k<-c(1,2,3) k1<-c(4,5,6) cbind(k,k1) cbind(j,k)

m1=cbind(j,k)

**output:**



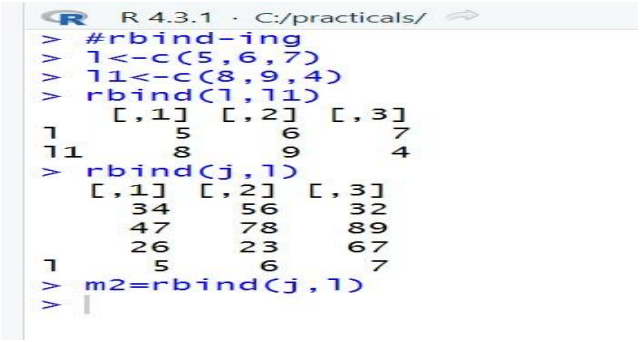

## ➢ rbind-ing

l<-c(5,6,7) l1<-c(8,9,4)

rbind(l,l1) rbind(j,l)

m2=rbind(j,l)

**output:**



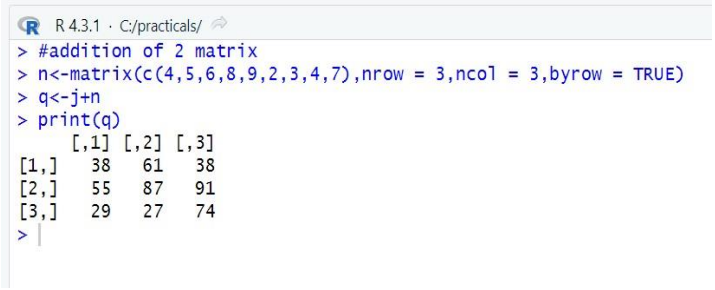

## ➢ addition of 2 matrix

n<-matrix(c(4,5,6,8,9,2,3,4,7),nrow = 3,ncol = 3,byrow =TRUE)

q<-j+n

print(q)

**output:**




## ➢ multiplication of matrix

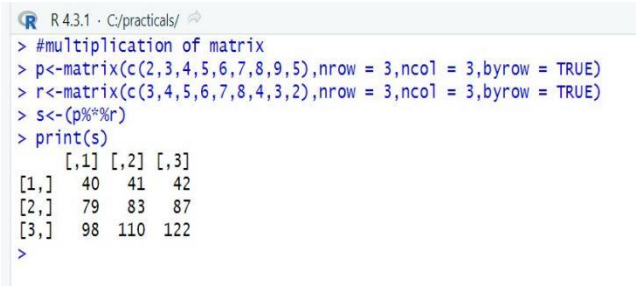
p<-matrix(c(2,3,4,5,6,7,8,9,5),nrow = 3,ncol = 3,byrow = TRUE)

r<-matrix(c(3,4,5,6,7,8,4,3,2),nrow = 3,ncol = 3,byrow = TRUE)

s<-(p%*%r)

print(s)

**output:**

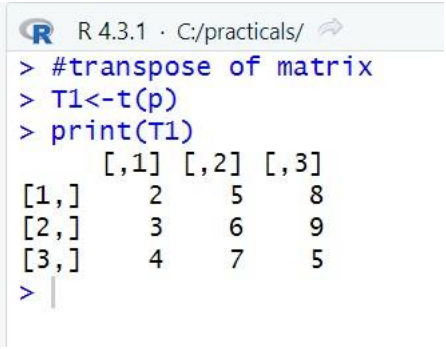



## ➢ transpose of matrix

T1<-t(p)

print(T1)

**output:**



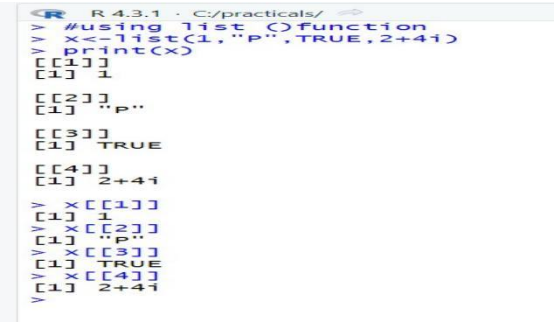

## ➢ Using list () function

x<-list(1,"P",TRUE,2+4i) print(x)

x[[1]] x[[2]] x[[3]]

x[[4]]

**output:**




## ➢ status is vector

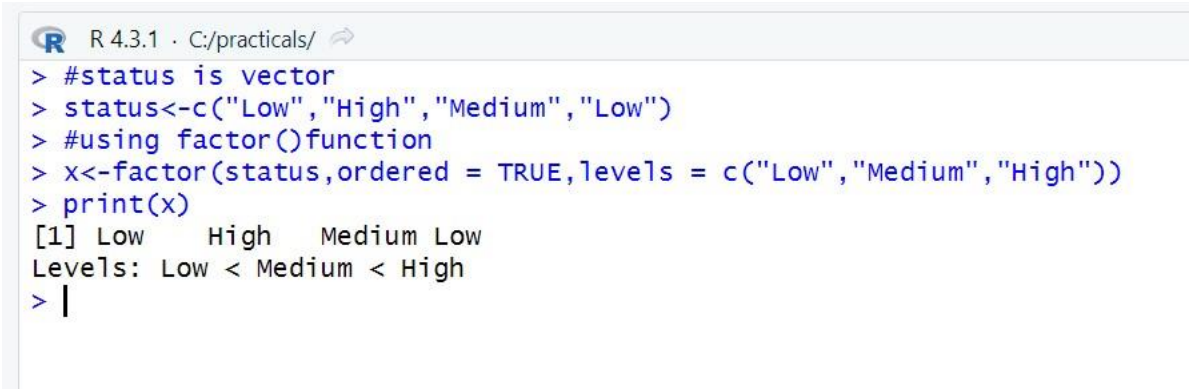
status<-c("Low","High","Medium","Low")

#using factor()function x<-factor(status,ordered = TRUE,levels =

c("Low","Medium","High"))

print(x)

**output:**




## ➢ using data frame ()function

student_id<-c(1,2,3) student_names<-c("Gita","Sita","Rita") position<-

c("First","Second","Third") data<-

data.frame(student_id,student_names,position)

print(data)

## ➢ accessing particular column

data$student_id data$student_names data$position

#no. of rows in

data nrow(data)

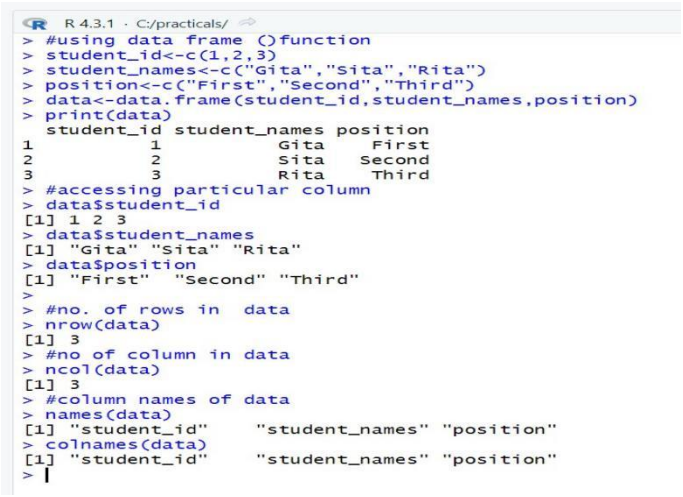
#no of column in data

ncol(data)

#column names of data

names(data) colnames(data)

**output:**




## ➢ reading and writing data from csv

setwd("C:/Program Files/R")

getwd()

data=read.csv("studentinfo.csv")

print(data)

#2nd method to read the file

dataT<-read.table("studentinfo.csv",sep = ",",header = T)

print(dataT)

#to print the 1 st and 2nd name

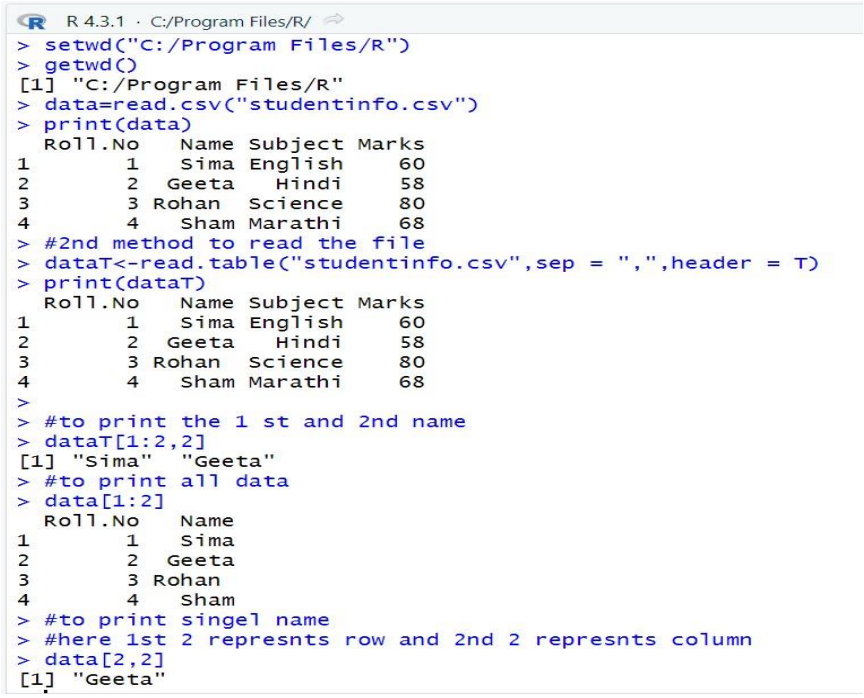
dataT[1:2,2] #to print all data  data[1:2]

#to print single name

#here 1st 2 represnts row and 2nd 2 represnts column

data[2,2]

**output:**




# Practical 7

## Aim:Implementation of Data pre-processing

- ## Data processing

setwd("C:/Program Files/R")

getwd() data=read.csv("mtcars.csv")

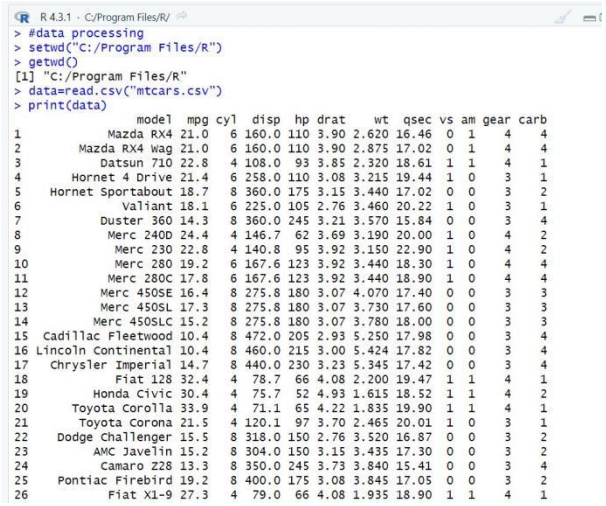
print(data)

head(data,8)

tail(data,6)

#for having the 1st 5 rows and 5colums  data[1:5,1:5]

**output:**



```
R 4.3.1 · C:/Program Files/R/
> #data processing
> setwd("C:/Program Files/R")
> getwd()
[1] "C:/Program Files/R"
> data=read.csv("mtcars.csv")
> print(data)
                 model  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1            Mazda RX4 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2        Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3           Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
4       Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
5    Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
6              Valiant 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
7           Duster 360 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
8            Merc 240D 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
9             Merc 230 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
10            Merc 280 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
11           Merc 280C 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
12          Merc 450SE 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
13          Merc 450SL 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
14         Merc 450SLC 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
15  Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
16 Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
17   Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
18            Fiat 128 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
19         Honda Civic 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
20      Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
21       Toyota Corona 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
22    Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
23         AMC Javelin 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
24          Camaro Z28 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
25    Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
26           Fiat X1-9 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
```

# ➢ Writing data in csv

setwd("C:/Users/ANIKET S RAUL/Desktop/wbs")

getwd()

data=data.frame(a=5,b=10,c=pi)

write.csv(data,file="data.csv")

install.packages("XLConnect")

library(XLConnect)

install.packages("readxl")

library(readxl)

```
install.packages("writexl")

library(writexl)

install.packages("openxlsx")

library(openxlsx)


data2=XLConnect::readWorksheetFromFile("C:/Users/ANIKET S

RAUL/Desktop/wbs/studentinfo.xlsx",sheet=1)

data2

data3=data2[1:2]

data3

write.xlsx(data2,"Book2.xlsx")
```

# Output:

```
> data2=XLConnect::readWorksheetFromFile("C:/Users/ANIKET S RAUL/Desktop/wbs/studentinfo.xlsx",sheet=1)
> data2
  Roll  Name Subject Marks
1    1 Rohan   maths    33
2    2 Suraj   maths    42
3    3 Rahul   maths    53
4    4 Siraj   maths    42
5    5 Rohit   maths    56
6    6   Sam   maths    45
>
```

```
> data3=data2[1:2]
> data3
  Roll  Name
1    1 Rohan
2    2 Suraj
3    3 Rahul
4    4 Siraj
5    5 Rohit
6    6   Sam
>
> write.xlsx(data2,"Book2.xlsx")
```

# ➢ Read mtcars file and perform other operations

**Code:**

```
setwd("C:/Program Files/R")

getwd()

my_data<-mtcars

 head(my_data,5)

 my_data1<-my_data[1:6,1:5]

my_data1
```

```
#making changes in any column or to add the column we use dplyr along with the function rename require(dplyr)

install.packages("dplyr")
 my_data1<-my_data[1:6,1:5]

#printing column my_data1
my_data1=rename(my_data1,horse_power = hp)

#to change the name of column  my_data1
#adding new variable

my_data1$new_hp1<-my_data1$horse_power * 0.5 colnames(my_data1)

 # Data preprocessing
#create vector v with 1 NA (not applicable )value
v<-c(1,2,NA,3)
#median with and without NA value
median(v)
 #on removing NA
#apply is.na() to vector (is used to check if vector is having na value)
is.na(v)



#removing the NA values by using logical indexing
# naVals is new variable to store values

 naVals<-is.na(v)

#get values that are not NA v[!naVals]
#subsetting with complete case - values that re not NA v[complete.cases(v)]
#subsetting a data frame with complete cases

 data<-read.csv(file="C:/Program Files/R/Data.csv")
data
```

```
#removing the NA value
 dataCompleteCases<-data[complete.cases(data),]
dataCompleteCases

#imputation (whenever you have missing values we have to use that data (so we assign/or replace some value to NA)
#impuation of mean ,median # IMPUTATION = estimating or derving missing value
install.packages("Hmisc")
library(Hmisc)
 #create a vector
 x=c(1,2,3,NA,4,4,NA)

#mean imputation - from pacakage, mention
 x<-impute(x, fun=mean)
x
#median imputation
 x<-impute(x,fun=median)
x

#convert character into factor(categorical data)
#create gender vector
 gender_vector<-c("Male","Female","Female","Male","Male")
class(gender_vector)
#convert gender vector to factor
factor_gender_vector<-factor(gender_vector)
class(factor_gender_vector)
```

## Output:

**1.To read mtcars csv file**

```
R 4.3.1 · ~/
> my_data<-mtcars
> head(my_data,5)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
> my_data1<-my_data[1:6,1:5]
> my_data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
>
```

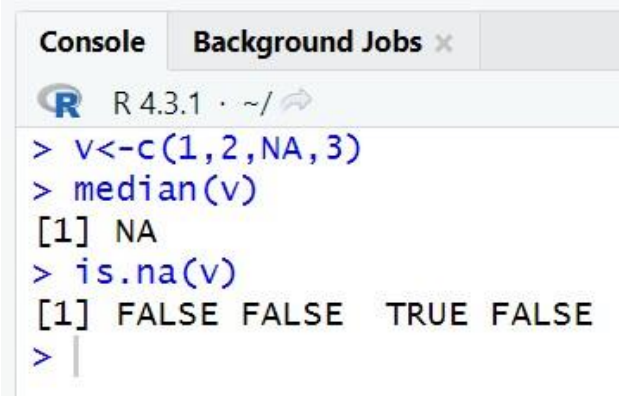
**2.To install dplyr package and replacing hp to horse_power**

```
Console  Background Jobs
R 4.3.1 · ~/
> require(dplyr)
> my_data1<-my_data[1:6,1:5]#printing column
> my_data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
> my_data1=rename(my_data1,horse_power = hp)#to change the name of column
> my_data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
>
```

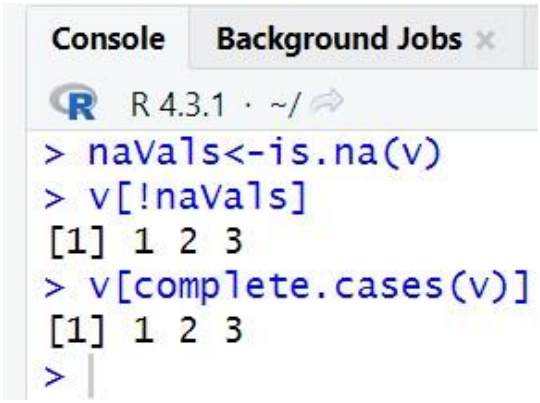
**3.Adding new variable /column as new_hp1**

```
Console  Background Jobs
R 4.3.1 · ~/
> my_data1$new_hp1<-my_data1$horse_power * 0.5
> colnames(my_data1)
[1] "mpg"         "cyl"         "disp"        "horse_power" "drat"        "new_hp1"
>
```

**4.Creating new vector V with NA value and finding is median and applying is.na to check if.na(v) value is there or not**




**5.Removing the NA values and then getting the values that are not NA and subsetting with complete case - values that re not NA**




**6. subsetting a data frame with complete cases and Removing the NA values**

```
Console  Background Jobs
R 4.3.1 · ~/
> data<-read.csv(file="C:/Program Files/R/Data.csv")
> data
  SerialNo   Name Salary  Department
1        1  Disha 50,000 Programming
2        2   Riya 55,000          IT
3        3 Drisha   <NA>  Oprational
4        4 Srisha 45,000  Backoffice
5        5   Siya 65,000          IT
> dataCompleteCases<-data[complete.cases(data),]
> dataCompleteCases
  SerialNo   Name Salary  Department
1        1  Disha 50,000 Programming
2        2   Riya 55,000          IT
4        4 Srisha 45,000  Backoffice
5        5   Siya 65,000          IT
>
```

**7. imputation**

**1.Creating vector and Finding is mean and median from package mention**

```
Console  Background Jobs
R 4.3.1 · ~/
> library(Hmisc)
> x=c(1,2,3,NA,4,4,NA)
> x<-impute(x, fun=mean)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
> x<-impute(x,fun=median)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
>
```

**8. convert character into factor(categorical data)**

**1. create gender vector and convert gender vector to factor**

```
Console  Background Jobs
R 4.3.1 · ~/
> gender_vector<-c("Male","Female","Female","Male","Male")
> class(gender_vector)
[1] "character"
> factor_gender_vector<-factor(gender_vector)
> class(factor_gender_vector)
[1] "factor"
>
```

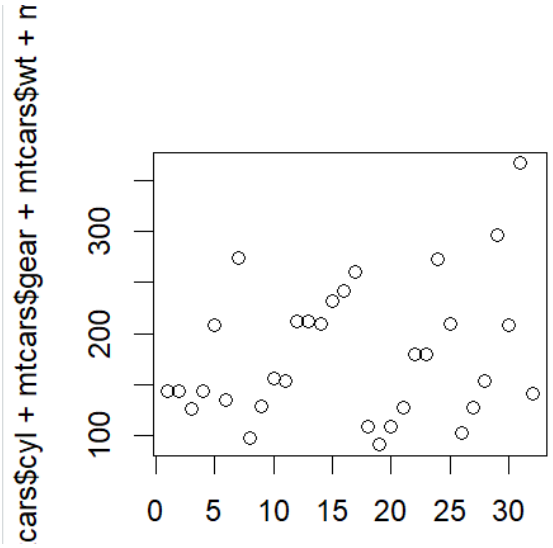
# Practical 8

## Aim:Implementation of Linear Regression

```
setwd("C:/Users/ANIKET S RAUL/Desktop/wbs")
getwd()
my_data<-mtcars
my_data
names(my_data)
dim(my_data)

#randomize
my_data<-my_data[sample(nrow(my_data),),]
head(my_data)

TrainData<-my_data[1:20,]
TrainData
TestData<-my_data[21:32,]
TestData

#Linear Model
fit = lm(mpg~hp,data=mtcars)
summary(fit)
preds<-predict(fit,newdata = TestData)
df1<-data.frame(preds,TestData$mpg)
head(df1)
#install ggplot packages
install.packages("ggplot2")
library(ggplot2)



#correaltion
```

```
cor(preds,TestData$mpg)
plot(mtcars$hp,mtcars$mpg)


#bettermodel ?
lmmodell<-lm(mpg~hp+cyl+gear+wt , data= TrainData)
summary(lmmodell)


preds_new<-predict(lmmodell,newdata = TestData)
df2<-data.frame(preds_new,TestData$mpg)
head(df2)
cor( TestData$mp,preds_new)
plot(mtcars$hp+mtcars$cyl+mtcars$gear+mtcars$wt+mtcars$mpg)
```

## Output:




```
> preds<-predict(fit,newdata = TestData)
> df1<-data.frame(preds,TestData$mpg)
> head(df1)
                  preds TestData.mpg
Ferrari Dino   18.15891         19.7
Merc 280       21.70678         19.2
Merc 280C      21.70678         17.8
Hornet 4 Drive 22.59375         21.4
Mazda RX4      22.59375         21.0
Lotus Europa   22.38907         30.4
```

```
> cor( TestData$mp,preds_new)
[1] 0.9497174
```

# Practical 9

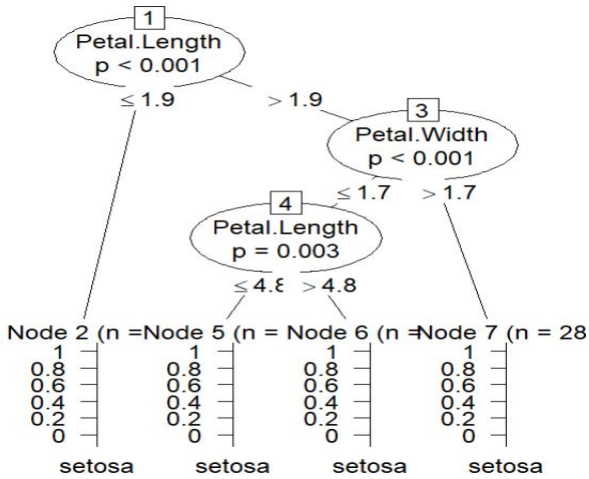
## Aim: Implementation of Decision Tree

```
#decison tree
data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/Iris1.csv")
data("iris")

dim(data)
str(data)

set.seed(789)
sampled.data<-sample(2,nrow(iris),replace=TRUE ,prob = c(0.7,0.3))
head(sampled.data)
tail(sampled.data)
trainData<- iris[sampled.data==1,]
dim(trainData)

testData<-iris[sampled.data==2,]
dim(testData)

install.packages("party")
library(party)
# task is to predict the species class given all other variable
myformula<-Species ~Sepal.Length +Sepal.Width + Petal.Length+Petal.Width
#ctree() function  from party package  builds the decision tree model
iris_ctree<-ctree(myformula,data=trainData)
plot(iris_ctree)

plot(iris_ctree,type="simple")

#now predict classfication on test data
testPred<-predict(iris_ctree,newdata=testData)
tab<-table(testPred,testData$Species)
accuracy<-function (x){sum(diag(x)/(sum(rowSums(x))))*100}
accuracy(tab)
```

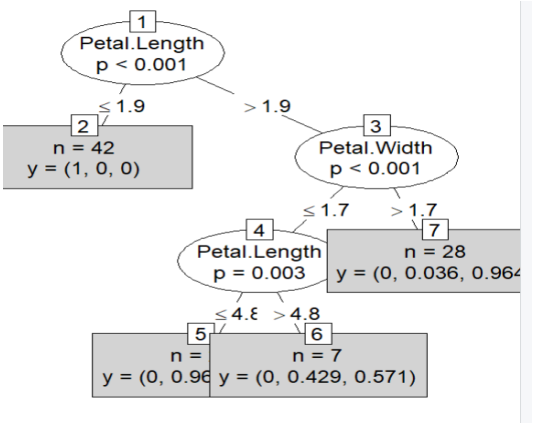
## output:

```
> data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/Iris1.csv")
> data("iris")
>
> dim(data)
[1] 150   6
> str(data)
'data.frame':   150 obs. of  6 variables:
 $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ SepalLengthCm: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ SepalWidthCm : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ PetalLengthCm: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ PetalWidthCm : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : chr  "Iris-setosa" "Iris-setosa" "Iris-setosa" "Iris-setosa" ...
`
```

```
> set.seed(789)
> sampled.data<-sample(2,nrow(iris),replace=TRUE ,prob = c(0.7,0.3))
> head(sampled.data)
[1] 1 1 1 1 1 1
> tail(sampled.data)
[1] 2 1 1 1 1 1
> trainData<- iris[sampled.data==1,]
> dim(trainData)
[1] 105   5
>
> testData<-iris[sampled.data==2,]
> dim(testData)
[1] 45  5
```




```
> #now predict classfication on test data
> testPred<-predict(iris_ctree,newdata=testData)
> tab<-table(testPred,testData$Species)
>
> accuracy<-function (x){sum(diag(x)/(sum(rowSums(x))))*100}
> accuracy(tab)
[1] 97.77778
>
```



# Practical 10

## Aim:Implementation of Naive Bayesian

**Code:**

**# Install and load required packages**

```
 install.packages("e1071") install.packages("MASS")
install.packages("lattice") install.packages("ggplot2")
install.packages("caret")

library(e1071) library(MASS)
library(lattice) library(ggplot2)
library(caret)

data <- read.csv("C:/Program Files/R/iris1.csv")
```

**# Split the data into training and testing sets**

```
index <- sample(nrow(data), floor(nrow(data) * 0.7))
train1 <- data[index, ] test <- data[-index, ]

xtrain1 <- train1[, -5] ytrain1 <-
train1$Species
xtest <- test[, -5]
ytest <- test$Species
```

**# Create a train control object for cross-validation**

```
train_control <- trainControl(method = 'cv', number = 10)
```

**# Train the Naive Bayes model using caret**

model <- train(x = xtrain1, y = ytrain1, method = 'nb', trControl = train_control)

**# Print the trained model**

print(model)

**# Make predictions on the test set**

predictions <- predict(model, newdata = xtest)

**# Display the confusion matrix**

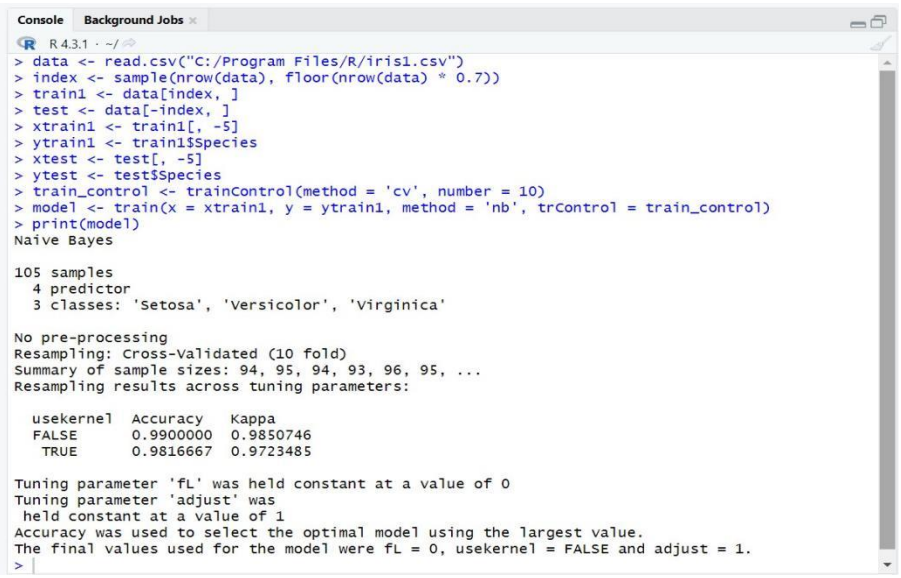
conf_matrix <- table(predictions, ytest) print(conf_matrix)

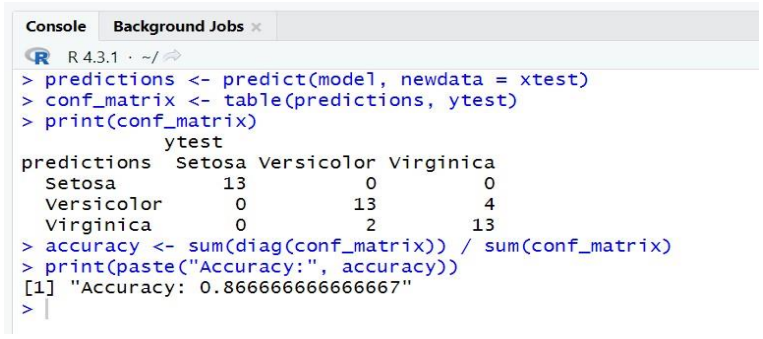
**# Calculate the accuracy**

accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)

print(paste("Accuracy:", accuracy))

## output:



# Practical 11

## Aim: Implementation of K-Nearest Neighbour(KNN)

## CODE:

```
data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/Iris1.csv")
#reading data
data
head(data,5)
df<-data(iris)
head(iris)
ran=sample(1:nrow(iris),0.9*nrow(iris))
nor=function(x){(x-min(x))/(max(x))}
iris_norm=as.data.frame(lapply(iris[,c(1,2,3,4)],nor))
summary(iris_norm)

#extract traning set
iris_train<-iris_norm[ran,]
dim(iris_train)

#extracting testing set
iris_test<-iris_norm[-ran,]
dim(iris_test)

#extract the 5th column of train dataset because it wiil be used
#as'cl'argument in knn function
iris_target_category<-iris[ran,5]

#extract 5th column if test data set  #to measure the accuracy
iris_test_category<-iris[-ran,5]




#load the pacakage class
```

install.packages("class")

library(class)

#run knn function

pr<-knn(iris_train,iris_test,cl=iris_target_category,k=13)

**#create confusion matrix**

tab<-table(pr,iris_test_category)

tab

**#this function divides the correct prediction by total number #of prediction that tell us how accurate the model is**

accuracy<-function (x){sum(diag(x)/(sum(rowSums(x))))*100}

accuracy(tab)

# output:

```
> data
     Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm         Species
1     1           5.1          3.5           1.4          0.2     Iris-setosa
2     2           4.9          3.0           1.4          0.2     Iris-setosa
3     3           4.7          3.2           1.3          0.2     Iris-setosa
4     4           4.6          3.1           1.5          0.2     Iris-setosa
5     5           5.0          3.6           1.4          0.2     Iris-setosa
6     6           5.4          3.9           1.7          0.4     Iris-setosa
7     7           4.6          3.4           1.4          0.3     Iris-setosa
8     8           5.0          3.4           1.5          0.2     Iris-setosa
9     9           4.4          2.9           1.4          0.2     Iris-setosa
10   10           4.9          3.1           1.5          0.1     Iris-setosa
11   11           5.4          3.7           1.5          0.2     Iris-setosa
12   12           4.8          3.4           1.6          0.2     Iris-setosa
13   13           4.8          3.0           1.4          0.1     Iris-setosa
```

```
> df<-data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> ran=sample(1:nrow(iris),0.9*nrow(iris))
> nor=function(x){(x-min(x))/(max(x))}
> iris_norm=as.data.frame(lapply(iris[,c(1,2,3,4)],nor))
> summary(iris_norm)
  Sepal.Length     Sepal.Width      Petal.Length      Petal.Width
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
 1st Qu.:0.1013   1st Qu.:0.1818   1st Qu.:0.08696   1st Qu.:0.0800
 Median :0.1899   Median :0.2273   Median :0.48551   Median :0.4800
 Mean   :0.1954   Mean   :0.2403   Mean   :0.39971   Mean   :0.4397
 3rd Qu.:0.2658   3rd Qu.:0.2955   3rd Qu.:0.59420   3rd Qu.:0.6800
 Max.   :0.4557   Max.   :0.5455   Max.   :0.85507   Max.   :0.9600
>
> #extract traning set
> iris_train<-iris_norm[ran,]
> dim(iris_train)
[1] 135   4
```

**#extracting testing set:**

```
> #extracting testing set
> iris_test<-iris_norm[-ran,]
> dim(iris_test)
[1] 15  4
```

**#extract the 5th column of train data set: extract 5th column if test data set: load the pacakage class:**

```
#extract the 5th column of train dataset because it will be used
#as'cl'argument in knn function
iris_target_category<-iris[ran,5]

#extract 5th column if test data set  #to measure the accuracy
iris_test_category<-iris[-ran,5]
pr<-knn(iris_train,iris_test,cl=iris_target_category,k=13)
```

**#run knn function:  create confusion matrix:**

```
> #create confusion matrix
> tab<-table(pr,iris_test_category)
> tab
            iris_test_category
pr           setosa versicolor virginica
  setosa          4          0         0
  versicolor      0          5         0
  virginica       0          1         5
```

**#this function divides the correct prediction by total number:**

```
> #this function divides the correct prediction by total number
el is
> accuracy<-function (x){sum(diag(x)/(sum(rowSums(x))))*100}
> accuracy(tab)
[1] 93.33333
```

# Practical 12

# Aim: Implementation of algorithm using Market Basket Analysis (APRIORI)

## CODE:

```
data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/data_apriori.csv")
#reading
data
trans=split(data$Products, data$Customer_Id,"Transations")
trans
head(trans)
#install arules library
install.packages("Matrix", version = "1.4.0", dependencies=TRUE)

# Load the updated Matrix package
library(Matrix)

# Install or load the arules package again
install.packages("arules", dependencies=TRUE)
library(arules)
rules=apriori(trans,parameter = list(support=0.5,confidence=0.9,maxlen=3, minlen=2))
inspect(rules)
```

## output:

```
> data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/data_apriori.csv")
> #reading
> data
   Customer_Id  Products
1            1     bread
2            1    butter
3            1      eggs
4            1      milk
5            4     bread
6            4    butter
7            4      eggs
8            4      milk
```

```
> trans=split(data$Products, data$Customer_Id,"Transations")
> trans
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"

$`4`
[1] "bread"  "butter" "eggs"   "milk"   "soda"
```

```
> head(trans)
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"
```

```
> rules=apriori(trans,parameter = list(support=0.5,confidence=0.9,maxlen=3, minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.9    0.1    1 none FALSE            TRUE       5     0.5      2      3  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 7

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[15 item(s), 15 transaction(s)] done [0.00s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [11 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> inspect(rules)
     lhs               rhs      support   confidence coverage  lift     count
[1]  {eggs}         => {milk}   0.6000000 1          0.6000000 1.666667 9
[2]  {milk}         => {eggs}   0.6000000 1          0.6000000 1.666667 9
[3]  {butter}       => {bread}  0.6000000 1          0.6000000 1.250000 9
[4]  {butter, eggs} => {milk}   0.5333333 1          0.5333333 1.666667 8
[5]  {butter, milk} => {eggs}   0.5333333 1          0.5333333 1.666667 8
[6]  {bread, eggs}  => {milk}   0.5333333 1          0.5333333 1.666667 8
[7]  {bread, milk}  => {eggs}   0.5333333 1          0.5333333 1.666667 8
[8]  {butter, eggs} => {bread}  0.5333333 1          0.5333333 1.250000 8
[9]  {bread, eggs}  => {butter} 0.5333333 1          0.5333333 1.666667 8
[10] {butter, milk} => {bread}  0.5333333 1          0.5333333 1.250000 8
[11] {bread, milk}  => {butter} 0.5333333 1          0.5333333 1.666667 8
>
```

# Practical 13

# Aim: Implementation of K means Clustering algorithm

## CODE:

```
data=read.csv("C:/Users/ANIKET S RAUL/Desktop/wbs/Iris1.csv")
head(iris)
library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
set.seed(20)
irisCluster<- kmeans(iris[, 3:4], 3, nstart = 20)
irisCluster
table(irisCluster$cluster, iris$Species)
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```

## output:

```
Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [50] 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [99] 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2
[148] 2 2 2

Within cluster sum of squares by cluster:
[1] 13.05769 16.29167  2.02200
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
1       0          48         4
2       0           2        46
3      50           0         0
```


```
K-means clustering with 3 clusters of sizes 52, 48, 50

Cluster means:
  Petal.Length Petal.Width
1     4.269231    1.342308
2     5.595833    2.037500
3     1.462000    0.246000
```