**Practical No. 1**

**Aim: Implementation of Data partitioning through Rang and List partitioning**

**Objectives:** Understand partitioning concept with Range and List partitioning method

**Theory:**

- What is Partitioning?

  Partitioning enables tables and indexes to be subdivided into individual smaller pieces. Each piece of the database object is called a partition. A partition has its own name, and may optionally have its own storage characteristics

- Key features and benefits of partitioning

  Partitioning allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity.

- Types of Partitioning: -
  - ➢ Range Partitioning: The data is distributed based on a range of values of the partitioning key.
  - ➢ List Partitioning: The data distribution is defined by a discrete list of values of the partitioning key

**Range Partitioning:**

- ➢ Create table sales_range with fields salesman_id, salesman_name, sales_amount, sales_date. Partition the table into 4 partitions using range partitioning.

```
SQL> CREATE TABLE sales_range
  2   (salesman_id NUMBER(5),
  3   salesman_name VARCHAR2(30),
  4   sales_amount NUMBER(10),
  5   sales_date DATE)
  6   PARTITION BY RANGE(sales_date)
  7   (
  8   PARTITION sales_jan2000 VALUES LESS THAN(TO_DATE('01/02/2000','DD/MM/YYYY')),
  9   PARTITION sales_feb2000 VALUES LESS THAN(TO_DATE('01/03/2000','DD/MM/YYYY')),
 10   PARTITION sales_mar2000 VALUES LESS THAN(TO_DATE('01/04/2000','DD/MM/YYYY')),
 11   PARTITION sales_apr2000 VALUES LESS THAN(TO_DATE('01/05/2000','DD/MM/YYYY'))
 12   )
 13   ;

Table created.
```

- ➢ Check to See If The Partitions Are Correctly Built

```
SQL> SELECT TABLE_NAME,PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE
  2  TABLESPACE_NAME='USERS';

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
SALES_RANGE                    SALES_JAN2000
SALES_RANGE                    SALES_FEB2000
SALES_RANGE                    SALES_MAR2000
SALES_RANGE                    SALES_APR2000
```

- ➢ Insert data in sales_range table

```
SQL> insert into sales_range values(1,'Janaki Joshi',5000,TO_DATE('23/02/2000','DD/MM/YYYY'));

1 row created.

SQL> insert into sales_range values(1,'Anand Sharma',15000,TO_DATE('12/03/2000','DD/MM/YYYY'));

1 row created.

SQL> insert into sales_range values(2,'Sonal Shah',35000,TO_DATE('17/04/2000','DD/MM/YYYY'));

1 row created.

SQL> insert into sales_range values(3,'Swati Chaudhary',14000,TO_DATE('24/01/2000','DD/MM/YYYY'));

1 row created.

SQL> insert into sales_range values(4,'Saniya Sane',4000,TO_DATE('16/08/1998','DD/MM/YYYY'));

1 row created.
```

- ➢ Display data from sales_range table

```
SQL> select * from sales_range;

SALESMAN_ID SALESMAN_NAME                  SALES_AMOUNT SALES_DAT
----------- ------------------------------ ------------ ---------
          3 Swati Chaudhary                       14000 24-JAN-00
          4 Saniya Sane                            4000 16-AUG-98
          1 Janaki Joshi                           5000 23-FEB-00
          1 Anand Sharma                          15000 12-MAR-00
          2 Sonal Shah                            35000 17-APR-00
```

- ➢ Display data from sales_jan2000 partition of sales_range table

```
SQL> select * from sales_range partition(sales_jan2000);

SALESMAN_ID SALESMAN_NAME                  SALES_AMOUNT SALES_DAT
----------- ------------------------------ ------------ ---------
          3 Swati Chaudhary                       14000 24-JAN-00
          4 Saniya Sane                            4000 16-AUG-98
```

**List Partitioning:**

- ➢ Create table sales_list with fields saleman_id, salesman_name, sales_state, sales_amount, sales_date. Apply list partitioning on the table.

```
SQL> CREATE TABLE sales_list
  2   (salesman_id NUMBER(5),
  3   salesman_name VARCHAR2(30),
  4   sales_state VARCHAR2(20),
  5   sales_amount NUMBER(10),
  6   sales_date DATE)
  7   PARTITION BY LIST(sales_state)
  8   (
  9   PARTITION sales_west VALUES('California', 'Hawaii'),
 10   PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida'),
 11   PARTITION sales_central VALUES('Texas', 'Illinois'),
 12   PARTITION sales_other VALUES(DEFAULT)
 13   )
 14   enable row movement
 15   ;

Table created.
```

(NOTE: enable row movement: This clause either enables or disables the migration of a row to a new partition if its key is updated. The default is **DISABLE ROW MOVEMENT**. In existing row if we try to update partition col value which causes change of partition is only possible if this option is enable)

- Check to See If the Partitions Are Correctly Built

```
SQL>  SELECT TABLE_NAME,PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE
  2  TABLESPACE_NAME='USERS' ;

SALES_RANGE                    SALES_JAN2000
SALES_RANGE                    SALES_FEB2000
SALES_RANGE                    SALES_MAR2000
SALES_RANGE                    SALES_APR2000
SALES_LIST                     SALES_WEST
SALES_LIST                     SALES_EAST

TABLE_NAME                     PARTITION_NAME
------------------------------ ---------------
SALES_LIST                     SALES_CENTRAL
SALES_LIST                     SALES_OTHER
```

- Insert data in sales_list table

```
SQL> insert into sales_list values(1,'Jayesh Patel','New York',2300,TO_DATE('23/02/2020','DD/MM/YYYY') );

1 row created.

SQL> insert into sales_list values(2,'Yash Pal','California',12300,TO_DATE('17/10/2020','DD/MM/YYYY') );

1 row created.

SQL> insert into sales_list values(3,'Sumit Rao','Texas',60000,TO_DATE('05/12/2021','DD/MM/YYYY') );

1 row created.

SQL> insert into sales_list values(4,'Sayli Joshi','Florida',18000,TO_DATE('09/09/2022','DD/MM/YYYY') );

1 row created.

SQL> insert into sales_list values(5,'Mina Shetty','Banglore',24000,TO_DATE('19/07/2022','DD/MM/YYYY') );

1 row created.
```

- Display data from sales_list table

```
SQL> select * from sales_list;

SALESMAN_ID SALESMAN_NAME                  SALES_STATE          SALES_AMOUNT
----------- ------------------------------ -------------------- ------------
SALES_DAT
---------
          2 Yash Pal                       California                  12300
17-OCT-20

          1 Jayesh Patel                   New York                     2300
23-FEB-20

          4 Sayli Joshi                    Florida                     18000
09-SEP-22


SALESMAN_ID SALESMAN_NAME                  SALES_STATE          SALES_AMOUNT
----------- ------------------------------ -------------------- ------------
SALES_DAT
---------
          3 Sumit Rao                      Texas                       60000
05-DEC-21

          5 Mina Shetty                    Banglore                    24000
19-JUL-22
```

- Display data from sales_east partition of sales_list table

```
SQL> select * from sales_list partition(sales_east);

SALESMAN_ID SALESMAN_NAME                  SALES_STATE          SALES_AMOUNT
----------- ------------------------------ -------------------- ------------
SALES_DAT
---------
          1 Jayesh Patel                   New York                     2300
23-FEB-20

          4 Sayli Joshi                    Florida                     18000
09-SEP-22
```

- Display data from sales_west partition of sales_list table

```
SQL> select * from sales_list partition(sales_west);

SALESMAN_ID SALESMAN_NAME                  SALES_STATE          SALES_AMOUNT
----------- ------------------------------ -------------------- ------------
SALES_DAT
---------
          2 Yash Pal                       California                  12300
17-OCT-20
```

# Practical No. 2

**Aim: Implementation of Analytical quries like ROLL_UP, CUBE, RANK, DENSE_RANK, LEAD, LAG, FIRST and LAST.**

**Objective:** Understand different types of Analytical functions

**Theory:**

**Analytical functions:-** Analytic functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group.

**Syntax:-**

analytic_function([arguments])OVER(analytic_clause)

The analytic_clause breaks down into the following optional elements.

[query_partition_clause][order_by_clause]

**ROLL_UP:-** The ROLLUP is an extension of the GROUP BY clause.The ROLLUP calculates multiple levels of subtotals across a group of columns (or dimensions) along with the grand total. the ROLLUP extension produces group subtotals from right to left and a grand total. If "n" is the number of columns listed in the ROLLUP, there will be n+1 levels of subtotals.

**CUBE:-** In addition to the subtotals generated by the ROLLUP extension, the CUBE extension will generate subtotals for all combinations of the dimensions specified. If "n" is the number of columns listed in the CUBE, there will be $2^n$ subtotal combinations.

**ROW_NUMBER():-** It is an analytical function and unlike NTILE this function assigns a unique sequential number to each row of the result set.

**RANK():-** The RANK() function is an analytic function that calculates the rank of a value in a set of values.The RANK() function returns the same rank for the rows with the same values.

**DENSE_RANK():-** The DENSE_RANK() function is an analytic function that calculates the rank of a value in a set of values.

The DENSE_RANK() function returns the same rank for the rows with the same values. DENSE_RANK() does not have anygap in rankings

**LEAD() and LAG():-** The LEAD function is used to access data from SUBSEQUENT rows along with data from the current row. The LAG function is used to access data from PREVIOUS rows along with data from the current row

**Syntax:-** LEAD(expression,[offset],[default]) over ([query_partition_clause] order_by_clause)

**FIRST and LAST :-** FIRST is an analytic function as the name suggests is used to provide the value of the first row in an ordered set of rows.LAST is also an analytical function which is used to get the value of the last row in an ordered set of rows.

**Implementation:-**

- Create table employee with fields emp_no,dep_no,emp_name,dob,salary,comm,job.

```
SQL> CREATE TABLE employee
  2  (emp_no NUMBER(10),
  3  dep_no  NUMBER(10),
  4  emp_name  VARCHAR(25),
  5  dob  DATE,
  6  salary  NUMBER(10),
  7  comm  NUMBER(10),
  8  job VARCHAR(25)
  9  );
Table created.
```

```
SQL> CREATE TABLE Employee
  2  (EMP_NO NUMBER(5),
  3   DEP_NO NUMBER(5),
  4   EMPLOYEE_NAME VARCHAR2(10),
  5   BDATE DATE,
  6   SALARY NUMBER(10),
  7   COMM NUMBER(10),
  8   JOB VARCHAR2(15));

Table created.
```

- Insert 10 records

```
SQL> insert into employee
VALUES(101,10,'TEJAS',TO_DATE('12/01/82','DD/MM/YYYY'),22000,1000,'CLERK');
1 row created.
```

- Display all the records.

Name: Roll No.

```
SQL> select * from employee;

    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
------------------------
       101         10 TEJAS                     12-JAN-82      22000       1000
CLERK

       102         10 MANISH                    12-FEB-83      52000       2000
CLERK

       103         10 SACHIN                    12-MAR-84      25000       4000
CLERK


    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
------------------------
       104         20 VANCHITA                  12-APR-85      35000       5000
manager

       105         20 SWARUPA                   12-MAY-86      45000       6000
manager

       106         20 ASHAWINI                  12-JUN-87      25000       2000
manager


    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
------------------------
       107         10 SUDIP                     12-JUL-88      55000       6000
manager

       108         20 AKSHAY                    12-AUG-89      35000       6000
clerk


8 rows selected.
```

## Roll_up

- Display dep_no,job,job count, sum of salary group them up using a roll up function in the order of dep_no,job.

SQL> select dep_no,job,count(*),sum(salary)
2 from employee
3 group by rollup(dep_no,job);

```
    DEP_NO JOB                       COUNT(*) SUM(SALARY)
---------- ------------------------- ---------- -----------
        10 CLERK                              3       99000
        10 manager                            1       55000
        10                                    4      154000
        20 clerk                              1       35000
        20 manager                            3      105000
        20                                    4      140000
                                              8      294000

7 rows selected.
```

## Cube:

- Display dep_no,job,job count, sum of salary group them up using a cube function in the order of dep_no,job.

SQL> select dep_no,job,count(*),sum(salary)
2 from employee
3 group by cube(dep_no,job);

```
    DEP_NO JOB                       COUNT(*) SUM(SALARY)
---------- ------------------------- ---------- -----------
                                              8      294000
           CLERK                              3       99000
           clerk                              1       35000
           manager                            4      160000
        10                                    4      154000
        10 CLERK                              3       99000
        10 manager                            1       55000
        20                                    4      140000
        20 clerk                              1       35000
        20 manager                            3      105000

10 rows selected.
```

- Display emp_no,emp_name and salary from employee table and give numbers to each row.

SQL> select rownum,emp_no,emp_name,salary from employee;

```
SQL> select rownum,emp_no,emp_name,salary from employee;

    ROWNUM     EMP_NO EMP_NAME                      SALARY
---------- ---------- ------------------------- ----------
         1        101 TEJAS                          22000
         2        102 MANISH                         52000
         3        103 SACHIN                         25000
         4        104 VANCHITA                       35000
         5        105 SWARUPA                        45000
         6        106 ASHAWINI                       25000
         7        107 SUDIP                          55000
         8        108 AKSHAY                         35000

8 rows selected.
```

SQL> select rownum,emp_no,emp_name,salary from employee order by salary ;

```
SQL> select rownum,emp_no,emp_name,salary from employee order by salary ;

    ROWNUM     EMP_NO EMP_NAME                      SALARY
---------- ---------- ------------------------- ----------
         1        101 TEJAS                          22000
         6        106 ASHAWINI                       25000
         3        103 SACHIN                         25000
         8        108 AKSHAY                         35000
         4        104 VANCHITA                       35000
         5        105 SWARUPA                        45000
         2        102 MANISH                         52000
         7        107 SUDIP                          55000

8 rows selected.
```

- Display emp_no, emp_name and descending order of salary from employee table and give numbers to each row.

SQL> select rownum,emp_no,emp_name,salary from employee order by salary desc;

```
SQL> select rownum,emp_no,emp_name,salary from employee order by salary desc;

    ROWNUM     EMP_NO EMP_NAME                      SALARY
---------- ---------- ------------------------- ----------
         7        107 SUDIP                          55000
         2        102 MANISH                         52000
         5        105 SWARUPA                        45000
         8        108 AKSHAY                         35000
         4        104 VANCHITA                       35000
         6        106 ASHAWINI                       25000
         3        103 SACHIN                         25000
         1        101 TEJAS                          22000
```

## Row_number():

- Use Row_number() analytical function to give numbering according to salary.

SQL>select row_number() over(order by salary),emp_no,emp_name,salary from employee order by salary desc;

```
SQL> select row_number()over(order by salary),emp_no,emp_name,salary from employee order by salary desc;

ROW_NUMBER()OVER(ORDERBYSALARY)     EMP_NO EMP_NAME                   SALARY
------------------------------- ---------- ------------------------- ----------
                              8        107 SUDIP                          55000
                              7        102 MANISH                         52000
                              6        105 SWARUPA                        45000
                              4        108 AKSHAY                         35000
                              5        104 VANCHITA                       35000
                              2        106 ASHAWINI                       25000
                              3        103 SACHIN                         25000
                              1        101 TEJAS                          22000
```

## Rank():

- Display eno, ename and salary from employee table and rank them according to ascending order of salary.

SQL>select emp_no,emp_name,salary,rank() over(order by salary) from employee order by salary;

```
SQL> select emp_no,emp_name,salary,rank() over(order by salary) from employee order by salary;

    EMP_NO EMP_NAME                   SALARY RANK()OVER(ORDERBYSALARY)
---------- ------------------------- ---------- -------------------------
       101 TEJAS                          22000                         1
       106 ASHAWINI                       25000                         2
       103 SACHIN                         25000                         2
       108 AKSHAY                         35000                         4
       104 VANCHITA                       35000                         4
       105 SWARUPA                        45000                         6
       102 MANISH                         52000                         7
       107 SUDIP                          55000                         8

8 rows selected.
```

## Dense_rank() :

- Display eno, name, salary from employee table and rank them according to ascending order of salary using dense_rank()

```
SQL> select emp_no,emp_name,salary,dense_rank() over(order by salary) from employee order by salary;

    EMP_NO EMP_NAME                   SALARY DENSE_RANK()OVER(ORDERBYSALARY)
---------- ------------------------- ---------- -------------------------------
       101 TEJAS                          22000                               1
       106 ASHAWINI                       25000                               2
       103 SACHIN                         25000                               2
       108 AKSHAY                         35000                               3
       104 VANCHITA                       35000                               3
       105 SWARUPA                        45000                               4
       102 MANISH                         52000                               5
       107 SUDIP                          55000                               6

8 rows selected.
```

- Display three highest salaried person.

```
SQL> select * from(select dense_rank() over(order by salary desc)top,emp_name,emp_no,salary from employee) where top < = 3;

       TOP EMP_NAME                   EMP_NO     SALARY
---------- ------------------------- -------- ----------
         1 SUDIP                          107      55000
         2 MANISH                         102      52000
         3 SWARUPA                        105      45000
```

- Partition by department_number

```
SQL> select emp_no,dep_no,salary,comm,rank() over(partition by dep_no order by salary)as Rank from employ

    EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       101         10      22000       1000          1
       103         10      25000       4000          2
       102         10      52000       2000          3
       107         10      55000       6000          4
       106         20      25000       2000          1
       108         20      35000       6000          2
       104         20      35000       5000          2
       105         20      45000       6000          4
```

## Update employee salary

```
SQL> update Employee
  2  set salary=33000
  3  where emp_no=101;
1 row updated.
SQL>  update Employee
  2  set salary=44000
  3  where emp_no=102;
1 row updated.
```

```
SQL> select * from employee;

    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
-------------------------
       101         10 TEJAS                     12-JAN-82      33000       1000
CLERK

       102         10 MANISH                    12-FEB-83      44000       2000
CLERK

       103         10 SACHIN                    12-MAR-84      25000       4000
CLERK


    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
-------------------------
       104         20 VANCHITA                  12-APR-85      35000       5000
manager

       105         20 SWARUPA                   12-MAY-86      45000       6000
manager

       106         20 ASHAWINI                  12-JUN-87      25000       2000
manager


    EMP_NO     DEP_NO EMP_NAME                   DOB          SALARY       COMM
---------- ---------- ------------------------- --------- ---------- ----------
JOB
-------------------------
       107         10 SUDIP                     12-JUL-88      55000       6000
manager

       108         20 AKSHAY                    12-AUG-89      35000       6000
clerk
```

- Inserting null value in commission

SQL>insert into employee values(101,10,'TEJAS',TO_DATE('12/01/82','DD/MM/YYYY'),22000,null,'CLERK');

- Replace null values of commission by 1000

SQL>selectemp_no,emp_name,salary,nvl(comm,1000)new_comm from employee order by comm desc;

```
SQL> Select emp_no, emp_nmae,salary,nvl(comm,1000) new_comm from employee order by comm desc;

    EMP_NO EMP_NAME                   SALARY   NEW_COMM
---------- ------------------------- ---------- ----------
       101 TEJAS                          22000       1000
       105 SWARUPA                        45000       6000
       108 AKSHAY                         35000       6000
       107 SUDIP                          55000       6000
       104 VANCHITA                       35000       5000
       103 SACHIN                         25000       4000
       106 ASHAWINI                       25000       2000
       102 MANISH                         44000       2000
       101 TEJAS                          33000       1000
```

- Display details of employee and give ranking only for employee in dept_no 10

SQL> select dense_rank()over(partition by dep_no order by salary)Rank,dep_no,emp_name,salary from employee where dep_no=10;

```
SQL> select emp_no,dep_no,salary,comm,dense_rank() over(partition by dep_no order by salary)as Rank from employee where dep_no = 10;

   EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       101         10      22000       1000          1
       103         10      25000       4000          2
       102         10      52000       2000          3
       107         10      55000       6000          4

      RANK     DEP_NO EMP_NAME                   SALARY
---------- ---------- ------------------------- ----------
         1         10 TEJAS                          22000
         2         10 SACHIN                         25000
         3         10 TEJAS                          33000
         4         10 MANISH                         44000
         5         10 SUDIP                          55000
```

Name: Roll No.

- Display name, job & salary and Rank the salary job wise

SQL> select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee order by job;

```
      RANK JOB                       EMP_NAME                       SALARY
---------- ------------------------- ------------------------- ----------
         1 CLERK                     TEJAS                          22000
         2 CLERK                     SACHIN                         25000
         3 CLERK                     TEJAS                          33000
         4 CLERK                     MANISH                         44000
         1 clerk                     AKSHAY                         35000
         1 manager                   ASHAWINI                       25000
         2 manager                   VANCHITA                       35000
         3 manager                   SWARUPA                        45000
         4 manager                   SUDIP                          55000

9 rows selected.
```

- Display information of employee & rank them for employee working as manager

SQL> select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee where job='manager';

```
      RANK JOB                       EMP_NAME                       SALARY
---------- ------------------------- ------------------------- ----------
         1 manager                   ASHAWINI                       25000
         2 manager                   VANCHITA                       35000
         3 manager                   SWARUPA                        45000
         4 manager                   SUDIP                          55000
```

- Display first five records of employee in descending order of salary

SQL>select*from(select emp_no, emp_name, salary, dense_rank()over(order by salary desc)rank from employee) where rank<=5;

```
EMP_NO EMP_NAME                   SALARY       RANK
------ ------------------------- ---------- ----------
   107 SUDIP                          55000          1
   105 SWARUPA                        45000          2
   102 MANISH                         44000          3
   104 VANCHITA                       35000          4
   108 AKSHAY                         35000          4
   101 TEJAS                          33000          5
```

**LEAD()**

- Display Employee details using Lead() analytical function.

SQL> select emp_no,dob,lead(dob,1)over(order by dob)as "next" from employee;

```
 EMP_NO DOB       next
------- --------- ---------
    101 12-JAN-82 12-JAN-82
    101 12-JAN-82 12-FEB-83
    102 12-FEB-83 12-MAR-84
    103 12-MAR-84 12-APR-85
    104 12-APR-85 12-MAY-86
    105 12-MAY-86 12-JUN-87
    106 12-JUN-87 12-JUL-88
    107 12-JUL-88 12-AUG-89
    108 12-AUG-89
```

**LAG():**

- Display Employee details using Lag() analytical function.

SQL> select emp_no,dob,lag(dob,1)over(order by dob)as "previous" from employee;

```
 EMP_NO DOB       previous
------- --------- ---------
    101 12-JAN-82
    101 12-JAN-82 12-JAN-82
    102 12-FEB-83 12-JAN-82
    103 12-MAR-84 12-FEB-83
    104 12-APR-85 12-MAR-84
    105 12-MAY-86 12-APR-85
    106 12-JUN-87 12-MAY-86
    107 12-JUL-88 12-JUN-87
    108 12-AUG-89 12-JUL-88
```

**First:**

SQL>select dep_no, salary, max(salary) keep (DENSE_RANK FIRST ORDER BY salary desc)over(PARTITION BY dep_no)"max" from employee;

```
 DEP_NO     SALARY        max
------- ---------- ----------
     10      25000      55000
     10      55000      55000
     10      44000      55000
     10      33000      55000
     10      22000      55000
     20      35000      45000
     20      25000      45000
     20      35000      45000
     20      45000      45000
```

**Last:**

SQL>select dep_no,salary,min(salary)keep(DENSE_RANK LAST ORDER BY salary desc)over(PARTITION BY dep_no)"min" from employee;

```
SQL> select dep_no,salary,min(salary)keep(DENSE_RANK LAST ORDER BY salary desc) over (partition
 by dep_no)as min from employee;

    DEP_NO     SALARY        MIN
---------- ---------- ----------
        10      25000      22000
        10      55000      22000
        10      44000      22000
        10      33000      22000
        10      22000      22000
        20      35000      25000
        20      25000      25000
        20      35000      25000
        20      45000      25000

9 rows selected.
```

## Practical No.3

**Aim: Implementation of ORDBMS using ADT (Abstract Data Types), References, etc.**

**Objectives:** Understand implementation of ORDBMS concept using ADT(Abstract Data Types) and References.

**Theory:** An object relational database management system (ORDBMS) is a database management system with that is similar to a relational database, except that it has an object-oriented database model. This system supports objects, classes and inheritance in database schemas and query language.

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations.

**Implementation:**

- Create ADT for name and address

create type type_name13 As object(fname varchar(20), mname varchar(20),lname varchar(20));
/
Type created

```
SQL> create type type_name13 As object
  2  (
  3  fname varchar(20),mname varchar(20),lname varchar(20)
  4  );
  5  /

Type created.
```

create type type_address13 As object(street varchar(20), city varchar(20), pincode number(10));
/
Type created

```
SQL> create type type_address13 As object
  2  (
  3  street varchar(20),
  4  city varchar(20),
  5  pincode number(10)
  6  );
  7  /

Type created.
```

- Create table customer with fields cid(number), cname(type_name), cadd(type_add), cphone(number)

Creating table :
create table customer013
(
c_id number(5) primary key,c_name type_name,
c_add type_address,c_phno number(10)
);
Table created.

```
SQL> create table customer013
  2  (
  3  c_id number(5) primary key,
  4  c_name type_name13,
  5  c_add type_address13,
  6  c_phno number(10)
  7  );

Table created.
```

Insert data:
insert into customer013
values(1,type_name('varsha','s','atul'),
type_address('Sainagar','Mumbai',400042),123456789);

```
SQL> insert into customer013
  2  values(2,type_name13('simran','p','gupta'),
  3  type_address13('Sainagar','Mumbai',480042),9434945924);

1 row created.

SQL> insert into customer013
  2  values(3,type_name13('sailee','s','kadam'),
  3  type_address13('virar','Mumbai',407042),1212346789);

1 row created.

SQL> insert into customer013
  2  values(4,type_name13('sumit','v','gawde'),
  3  type_address13('kandiwali','Mumbai',407042),1212346789);

1 row created.

SQL> insert into customer013
  2  values(5,type_name13('pratik','a','ahirao'),
  3  type_address13('borivali','Mumbai',407042),1212346789);

1 row created.
```

Selecting data from table:

select * from customer013;

```
SQL> select * from customer013;

      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------
    C_PHNO
----------
         1
TYPE_NAME13('aadarsh', 'p', 'choudhari')
TYPE_ADDRESS13('virar', 'Mumbai', 400042)
 123456789


      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------
    C_PHNO
----------
         2
TYPE_NAME13('simran', 'p', 'gupta')
TYPE_ADDRESS13('Sainagar', 'Mumbai', 480042)
9434945924


      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------
    C_PHNO
----------
         3
TYPE_NAME13('sailee', 's', 'kadam')
TYPE_ADDRESS13('virar', 'Mumbai', 407042)
1212346789
```

➢ Select street value of customer whose cid=1

SQL> select c.c_add.street from customer1 c where c_id=1;

```
SQL> select c.c_add.street from customer013 c where c_id=1;

C_ADD.STREET
--------------------
virar
```

➢ Select fname of customer whose cid=1

SQL> select c.c_name.fname from customer1 c where c_id=1;

```
SQL> select c.c_name.fname from customer013 c where c_id=1;

C_NAME.FNAME
--------------------
aadarsh
```

➢ Select lname of customer whose cid=1

```
SQL> select c_id,c.c_name.lname from customer013 c;

      C_ID C_NAME.LNAME
---------- --------------------
         1 choudhari
         2 gupta
         3 kadam
         4 gawde
         5 ahirao
```

➢ Select cname of customers.

SQL> select c_name from customer1;

```
SQL>  select c_name from customer013;

C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------
TYPE_NAME13('aadarsh', 'p', 'choudhari')
TYPE_NAME13('simran', 'p', 'gupta')
TYPE_NAME13('sailee', 's', 'kadam')
TYPE_NAME13('sumit', 'v', 'gawde')
TYPE_NAME13('pratik', 'a', 'ahirao')
```

➢ Select full name of customers.

SQL> select c.c_name.fname||' '||c.c_name.mname||' '||c.c_name.lname from customer1 c;

```
SQL> select c.c_name.fname||' '||c.c_name.mname||' '||c.c_name.lname
  2  from customer013 c;

C.C_NAME.FNAME||''||C.C_NAME.MNAME||''||C.C_NAME.LNAME
--------------------------------------------------------------------
aadarsh p choudhari
simran p gupta
sailee s kadam
sumit v gawde
pratik a ahirao
```

**REF and DREF function:**

Creating Object tables

SQL> create or replace type ANIMAL_TY12 as object(Breed varchar2(25),
Name varchar2(25),BirthDate DATE);
/
Type created.

```
SQL> create or replace type ANIMAL_TY12 as object (Breed varchar2(25),
  2  Name varchar2(25), BirthDate DATE);
  3  /

Type created.
```

SQL> create table ANIMAL12 of ANIMAL_TY12;
Table created.

```
SQL> create table ANIMAL12 of ANIMAL_TY12;

Table created.
```

Inserting Rows into Object Tables

SQL> insert into ANIMAL12 values( ANIMAL_TY12('MULE','FRANCES','01-APR-02'));
 insert into ANIMAL12 values( ANIMAL_TY12('DOG','BENJI','03-SEP-01'));
1 row created.

```
SQL> insert into ANIMAL12 values( ANIMAL_TY12('MULE','FRANCES','01-APR-02'));

1 row created.

SQL>
SQL>  insert into ANIMAL12 values( ANIMAL_TY12('DOG','BENJI','03-SEP-01'));

1 row created.
```

The REF Function

SQL> select REF(A) from ANIMAL12 A;

```
SQL> select REF(A) from ANIMAL12 A;

REF(A)
--------------------------------------------------------------------------------
0000280209E1F34FF0BDE64B8BA6317510F72BD625BF03A2BED4A04A6880D2F0B3D3874139010002
4D0000

0000280209D827A5504ABE4AFBBFB98B476DF07686BF03A2BED4A04A6880D2F0B3D3874139010002
4D0001
```

Using the DEREF Function

SQL> create table KEEPER12 (KeeperName varchar2(25), AnimalKept REF ANIMAL_TY12);

Table created.

```
SQL> create table KEEPER12 (KeeperName varchar2(25), AnimalKept REF ANIMAL_TY12);

Table created.
```

SQL> describe KEEPER12

```
SQL> describe KEEPER12
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 KEEPERNAME                                         VARCHAR2(25)
 ANIMALKEPT                                         REF OF ANIMAL_TY12
```

SQL> insert into KEEPER12 select 'CATHERINE', REF(A)  from ANIMAL12 A where Name='BENJI';
1 row created.

```
SQL> insert into KEEPER12  select 'CATHERINE', REF(A) from ANIMAL12 A where Name='BENJI';

1 row created.
```

SQL> select * from KEEPER12;

```
SQL> select * from KEEPER12;

KEEPERNAME
-------------------------
ANIMALKEPT
--------------------------------------------------------------------------------
CATHERINE
0000220208D827A5504ABE4AFBBFB98B476DF07686BF03A2BED4A04A6880D2F0B3D3874139
```

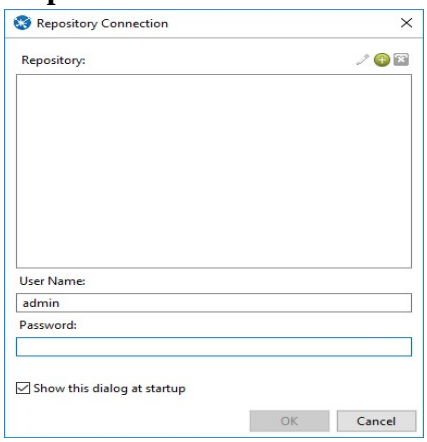
SQL> select KeeperName, DEREF(K.AnimalKept)from KEEPER12 K;

```
SQL> select KeeperName, DEREF(K.AnimalKept) from KEEPER12 K;

KEEPERNAME
-------------------------
DEREF(K.ANIMALKEPT)(BREED, NAME, BIRTHDATE)
--------------------------------------------------------------------------------
CATHERINE
ANIMAL_TY12('DOG', 'BENJI', '03-SEP-01')
```

**ETL Implementation through Pentaho**

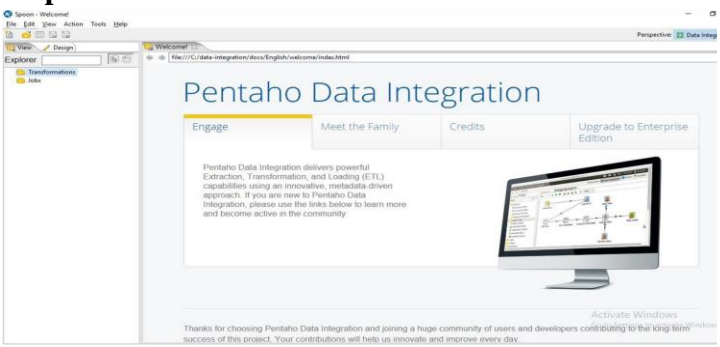
**Aim :** Perform Transformation on source Table and store the data to Output table in SQL.
Starting with Pentaho

**Step 1. Open Data Integration folder (C:\data-integration)**
**Step 2. Double click on spoon(Windows Batch file)**
**Step 3. Click on cancel.**

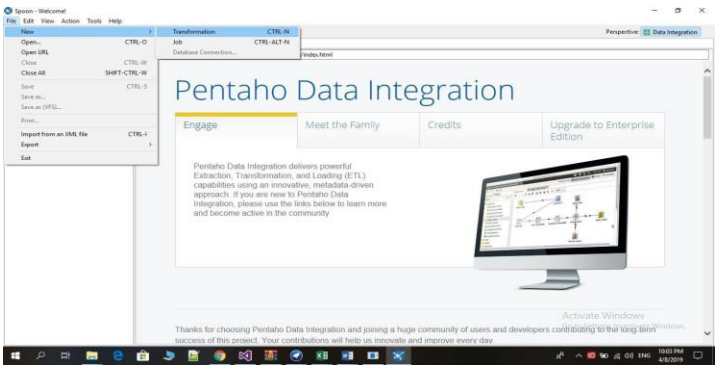



1. Transforming Source Table and storing to Output Table in SQL

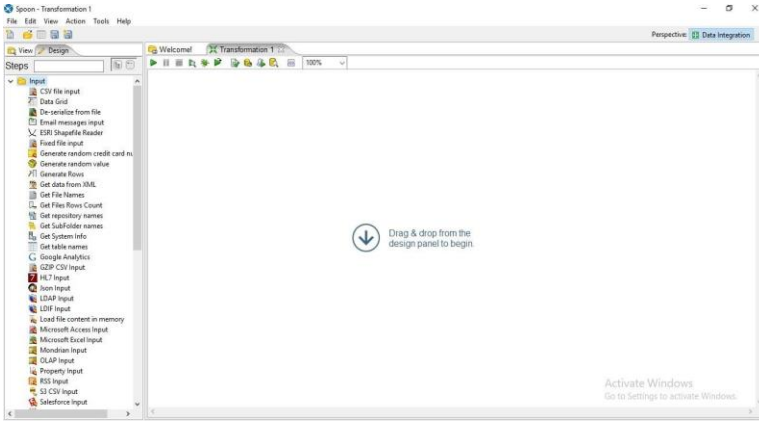
**Step1: Go to Pentaho**



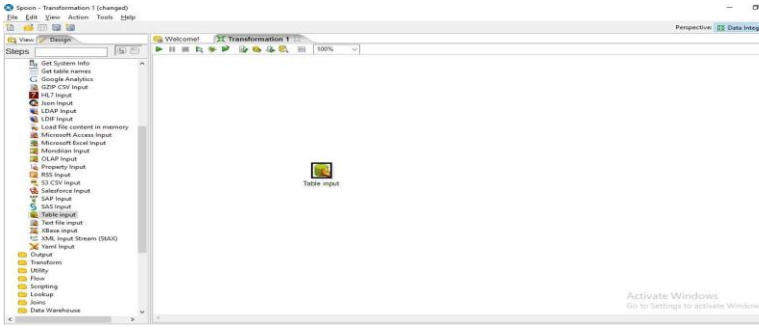

**Step2: Go to file->New->Transformation.**




**Step 3: Click on Input**

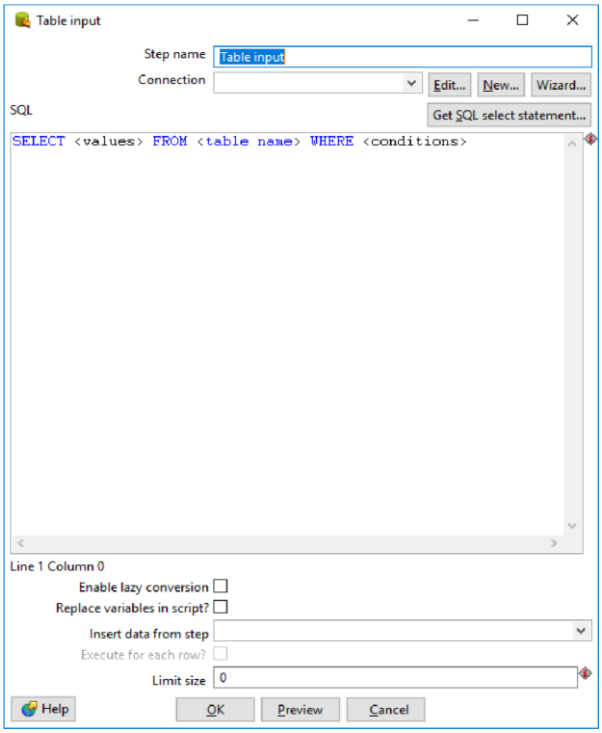



**Step 4: Drag Table Input on the panel**

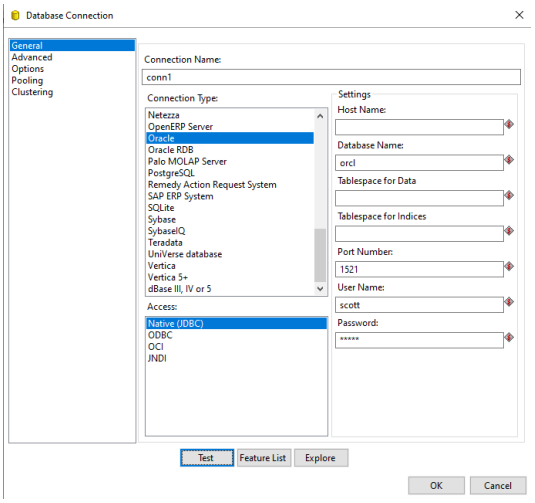


**Step 5: Double Click on Table Input and the following tab will appear.**

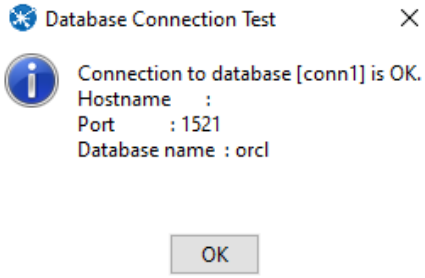
a. Click On New



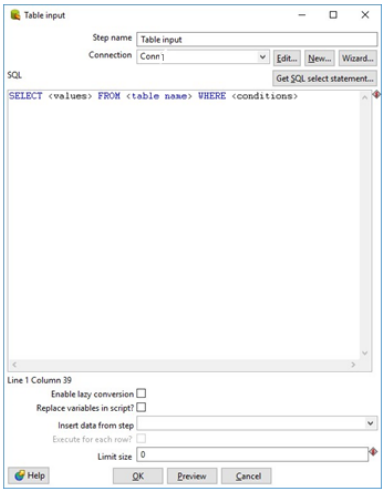

b. Select Oracle in Connection Type and enter Connection Name, Database Name, User Name and Password and click on Test.



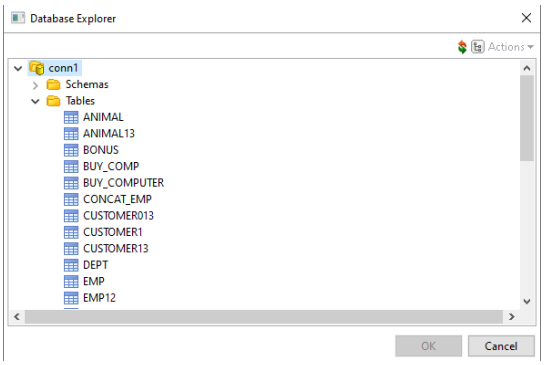

Connection Successful



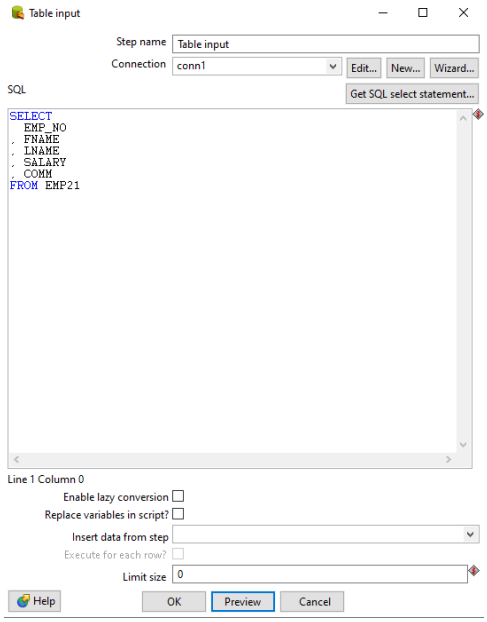

c. Click on get SQL select statement

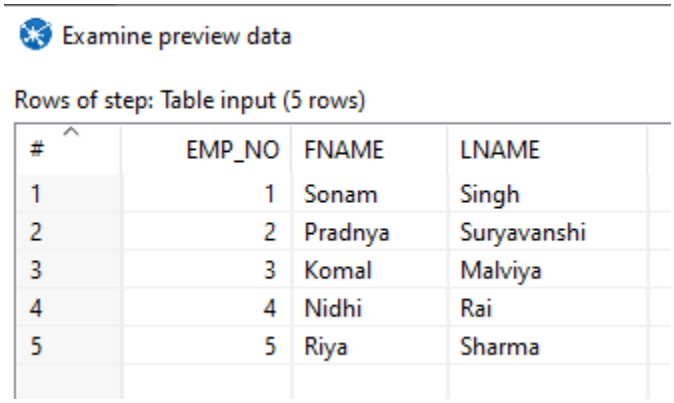



d. Click on database -> table and select the table name.



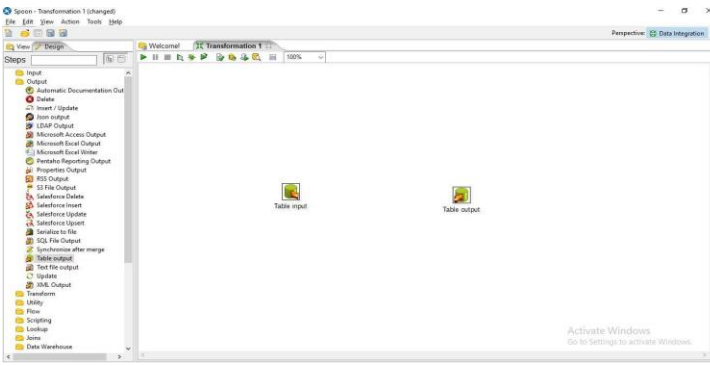
e. Click on get Preview.



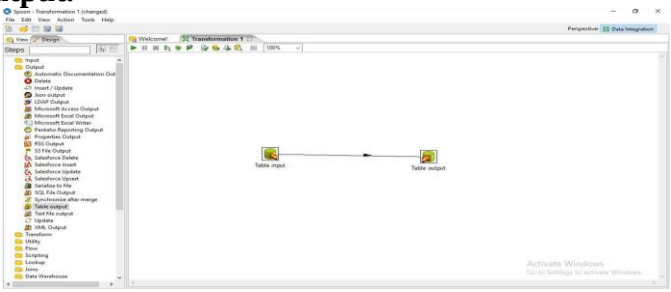

| # | EMP_NO | FNAME | LNAME |
|---|---|---|---|
| 1 | 1 | Sonam | Singh |
| 2 | 2 | Pradnya | Suryavanshi |
| 3 | 3 | Komal | Malviya |
| 4 | 4 | Nidhi | Rai |
| 5 | 5 | Riya | Sharma |

**Step 6: Click on Output and drag table Output.**



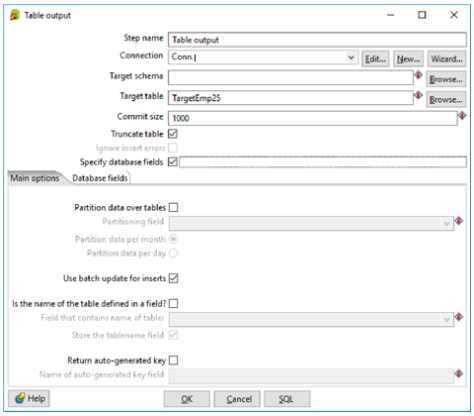

+

**Step 7: Hold the mouse Pointer on table input and select and drag the Output connector to the Table output.**

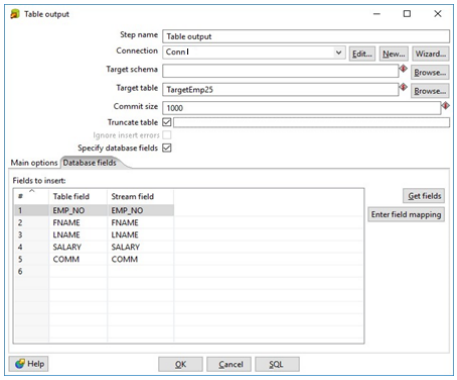


**Step 8: Double Click on target table**

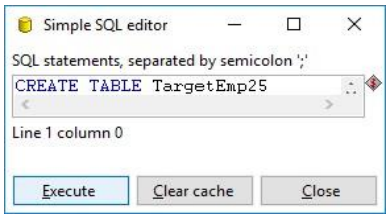
a. Enter the Target table name and check the truncate table and Specify database fields Check Box.



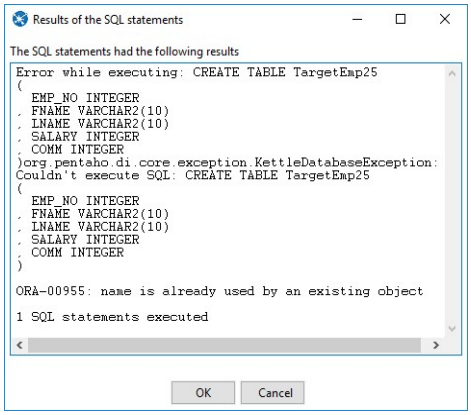

b. Click On Database fields and click on Get fields.



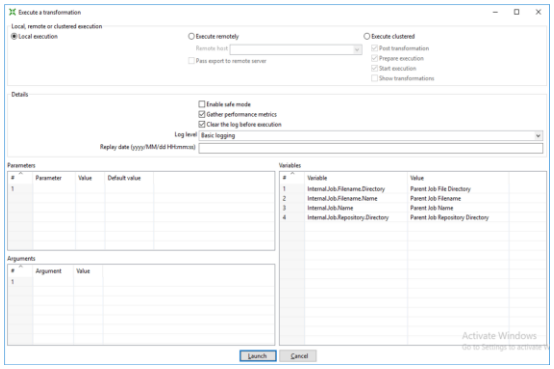

c. Click on SQL and click on Execute.



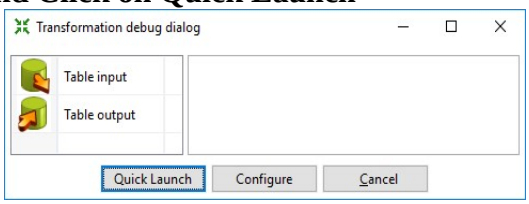

d. Click Ok




**Step 9: Click on Run Transformation and Click on Launch**




**Step 10: Click on Debug Transformation(Spider) and Click on Quick Launch**

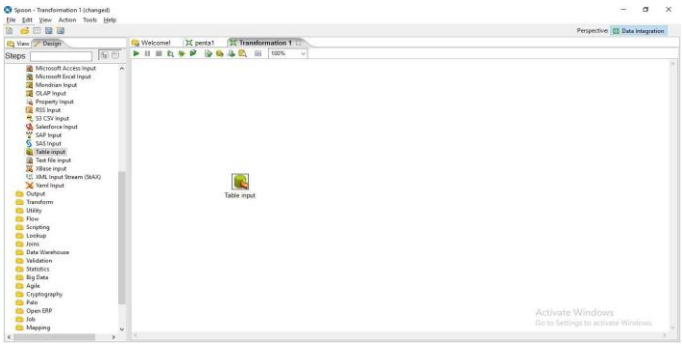



**Output:**

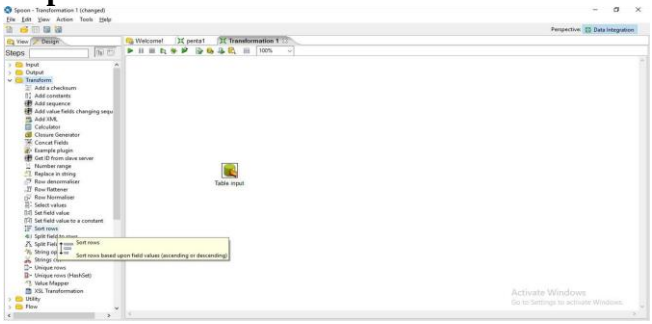
The Green ticks on the table input and table output shows successful transformation.

**2. Sorting Operation and adding Sequence to Output Table.**

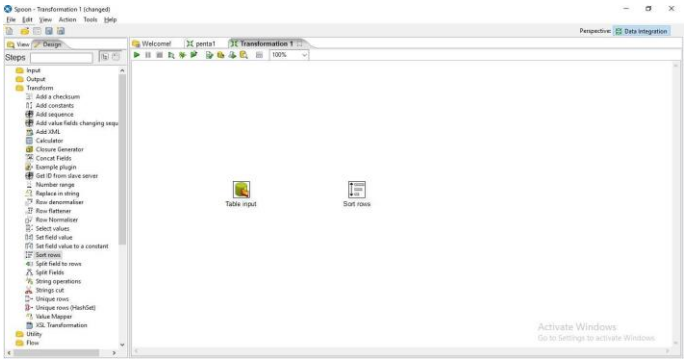
**Step 1: Perform the first 5 steps same as practical 1.**



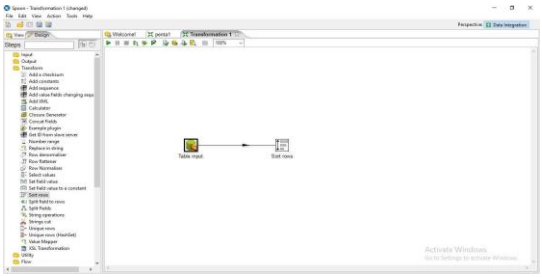
**Step 2: Click on Transform in the left**



**Step 3: Drag sort rows on the Panel**

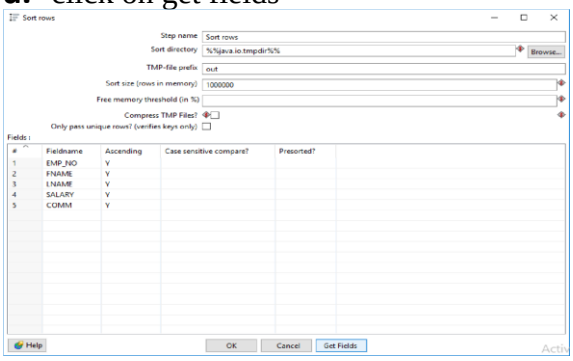


**Step 4: Hold the mouse pointer on the Table Input and then drag the output connector to the sort rows.**

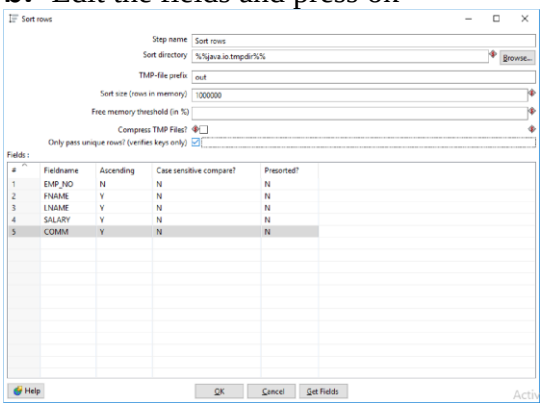


**Step 5: Double click on the sort rows**

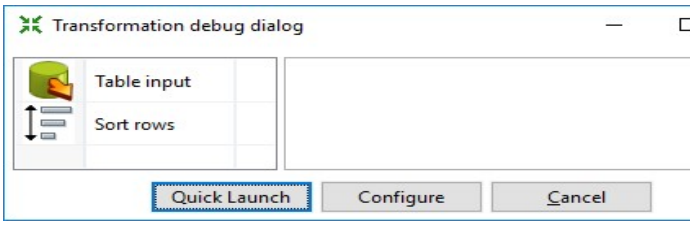
**a.** click on get fields



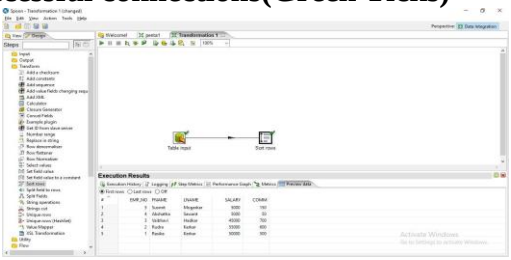
**b.** Edit the fields and press ok



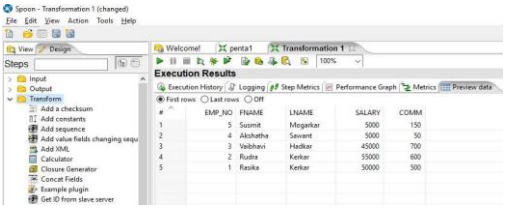
**Step 6: Click on Debug the Transformation and Click on Quick Launch.**



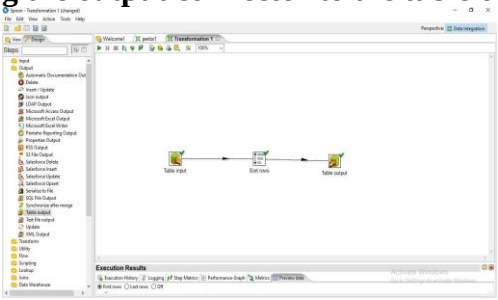

**Successful connections(Green Ticks)**



**Sorted Data**

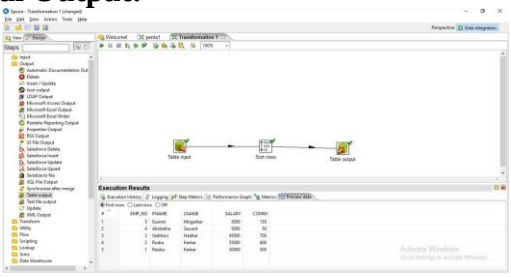


**Step 7: hold the cursor on the sort row and then drag the output connector to the table output.**



**Step 8:now perform step no 8 to steo 10 from practical 1.**

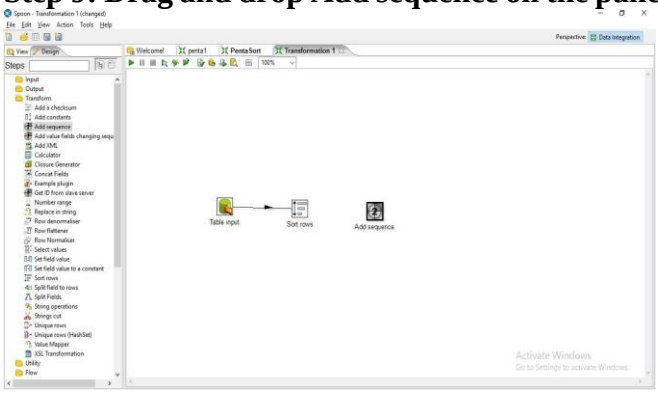
**Final Output:**



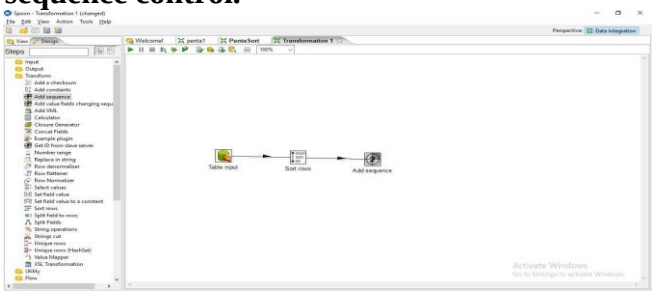
**SQL> select * from emppp13;**

```
SQL> select * from emp_target_concat;

    EMP_NO FNAME                          LNAME
---------- ------------------------------ ------------
    SALARY       COMM FULLNAME
---------- ---------- ------------------------------
         1 Sonam                          Singh
     25000        500 Sonam     _Singh

         2 Pradnya                        Suryavanshi
     20000        700 Pradnya   _Suryavansh

         3 Komal                          Malviya
     17000       1000 Komal     _Malviya


    EMP_NO FNAME                          LNAME
---------- ------------------------------ ------------
    SALARY       COMM FULLNAME
---------- ---------- ------------------------------
         4 Nidhi                          Rai
     30000        900 Nidhi     _Rai

         5 Riya                           Sharma
     15000       1100 Riya      _Sharma
```

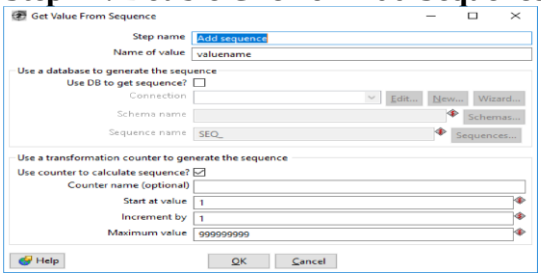
**Step 9: Drag and drop Add sequence on the panel**



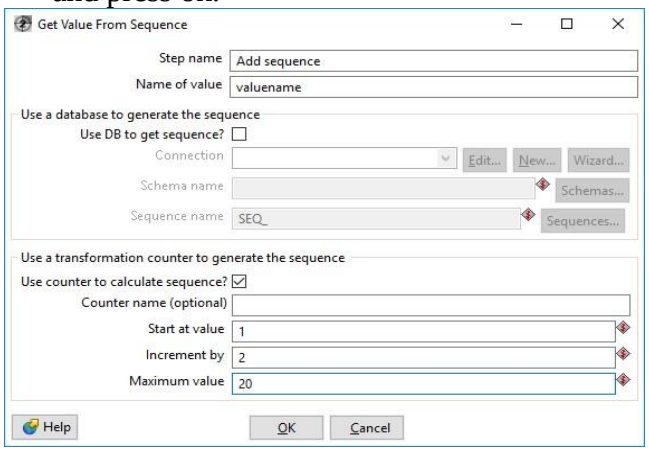
**Step 10: Hold the mous pointer on the sort rows and drag the output connector to the Add sequence control.**



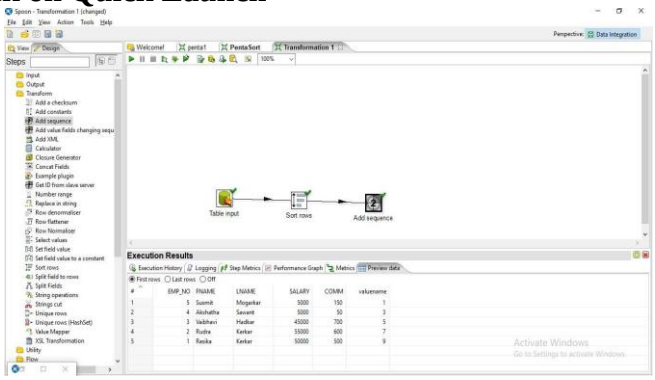
**Step 11: Double Click on Add Sequence Control**



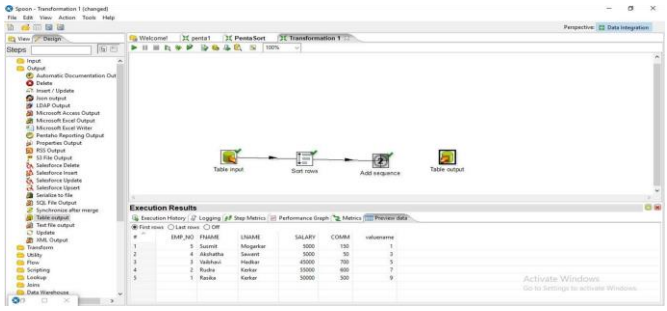

a. Enter the values for Start at Values,Increment by and Maximum Values and press ok.



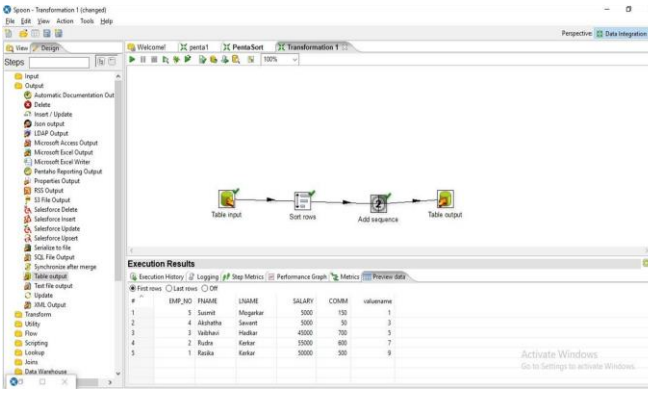

**Step 12: Click on Debug the Transformation and click on Quick Launch**



**Step 13: Drag and drop the table output on the panel**

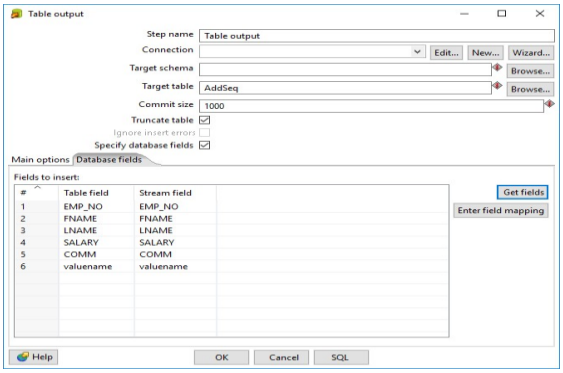


**Step 14: Hold the mouse pointer on the add sequence control and drag the output connector to the table output.**

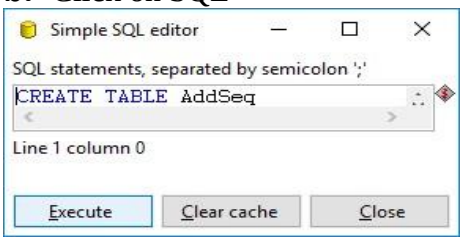


**Step 15: Double Click on the table output.**

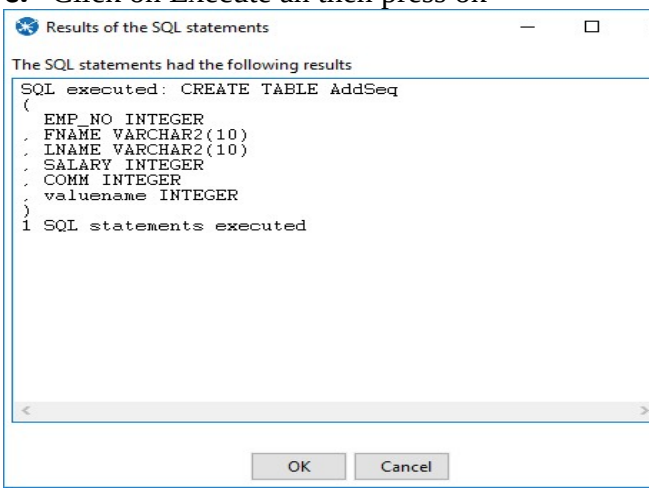
**a.** Click on Database Fields



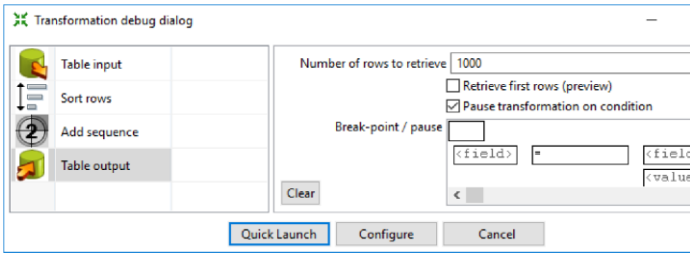

**b.** Click on SQL



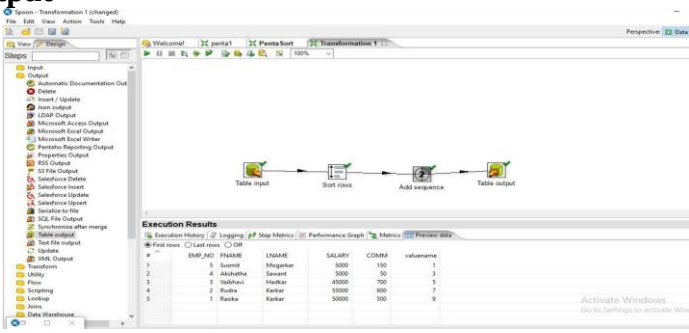

**c.** Click on Execute an then press ok




**Step 16: Click on debug Transformation and then quick Launch**




**Output**

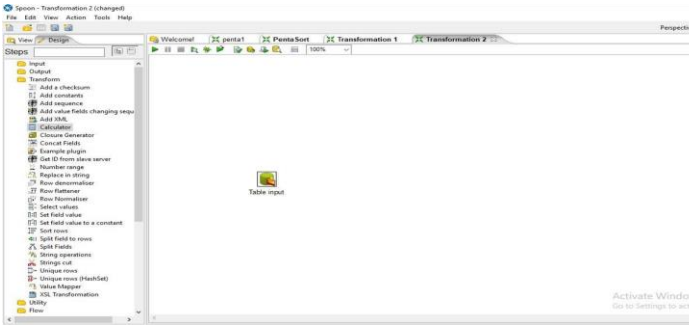


**SQL> select * from empp13;**

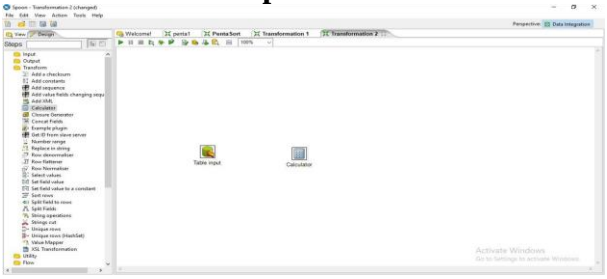
```
SQL> select * from empp13;

       ENO FNAME      LNAME          SALARY       COMM  VALUENAME
---------- ---------- ---------- ---------- ---------- ----------
         5 smit       bole           160000      12000          1
         4 prateek    ahirrao        150000      15000          3
         2 aadarsh    choudhari      200000      74000          5
         1 simran     gupta          100000      44000          7
```

**3. Calculator Operation**

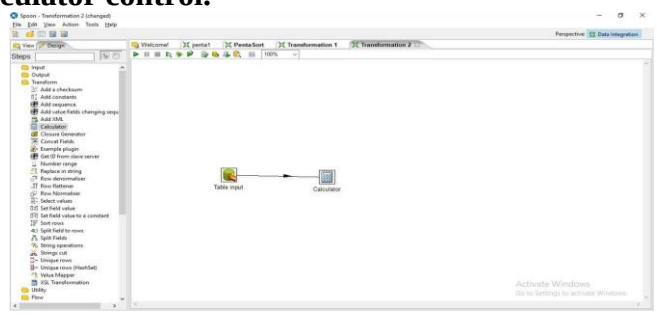
**Step 1: Perform first 5 steps same as practical 1.**



**Step 2: Drag and Drop Calculator from transformation on the panel.**



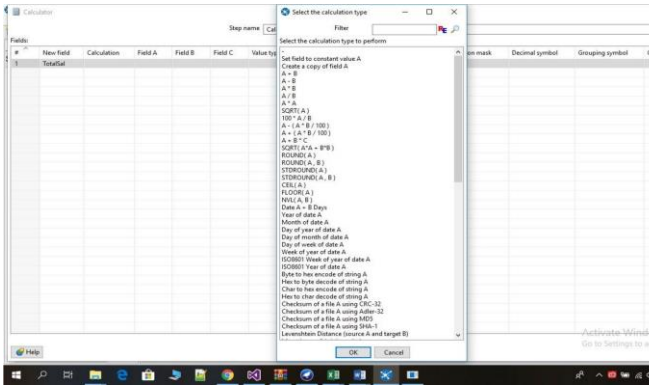
**Step 3: Hold the mouse pointer on the Table input and drag the output connector to the calculator control.**



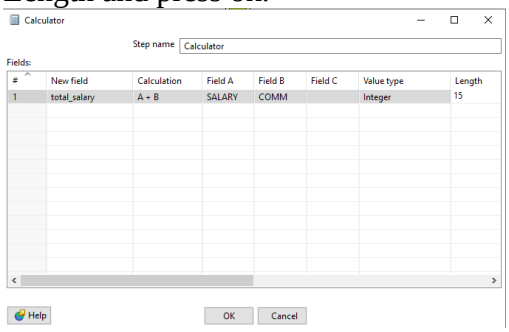
**Step 4: Double Click on the Calculator**

**a.** specify the new field name

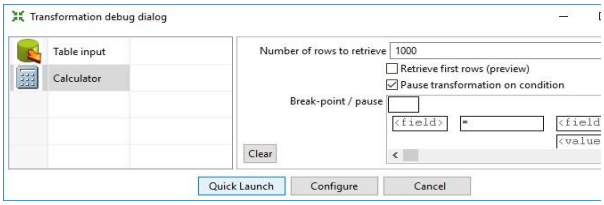
**b.** select the calculation function.



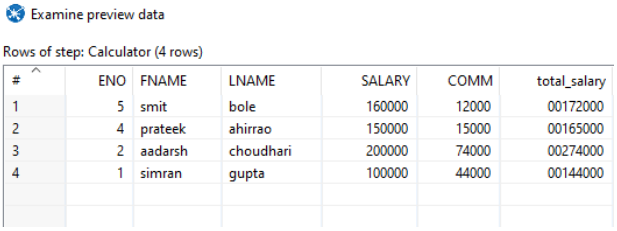
**c.** Enter Field A, Field B, Value type and Length and press ok.




**Step 5: Click on Debug transformation and click on Quick Launch.**

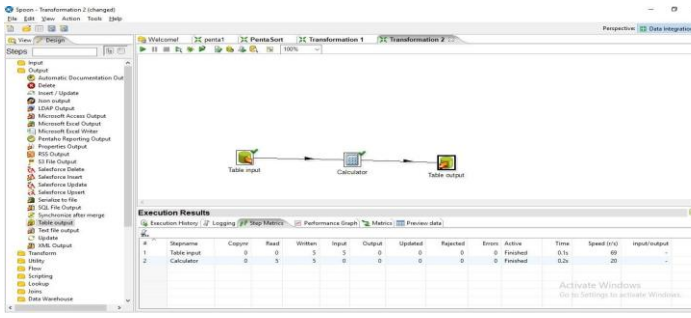



**Preview**

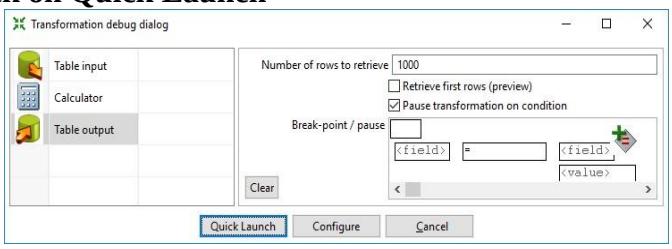



| # | ENO | FNAME | LNAME | SALARY | COMM | total_salary |
|---|---|---|---|---|---|---|
| 1 | 5 | smit | bole | 160000 | 12000 | 00172000 |
| 2 | 4 | prateek | ahirrao | 150000 | 15000 | 00165000 |
| 3 | 2 | aadarsh | choudhari | 200000 | 74000 | 00274000 |
| 4 | 1 | simran | gupta | 100000 | 44000 | 00144000 |

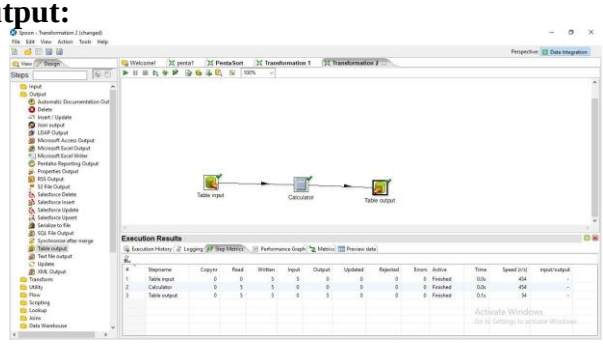
**Step 6: Now Perform the steps 6-10 same as practical 1.**



**Step 7: Click on Debug Transformation and then click on Quick Launch**




**Output:**

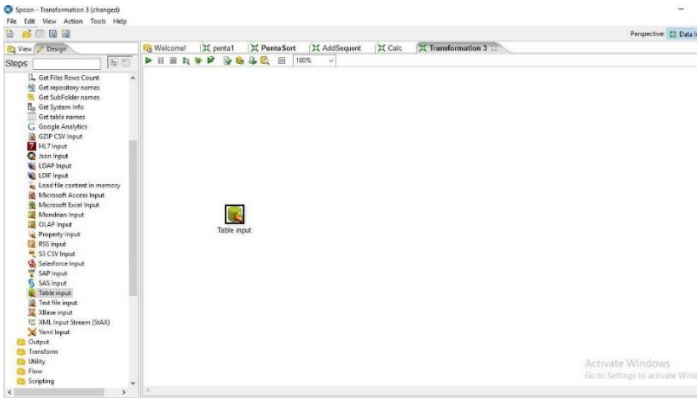


**SQL> select * from emppp13;**

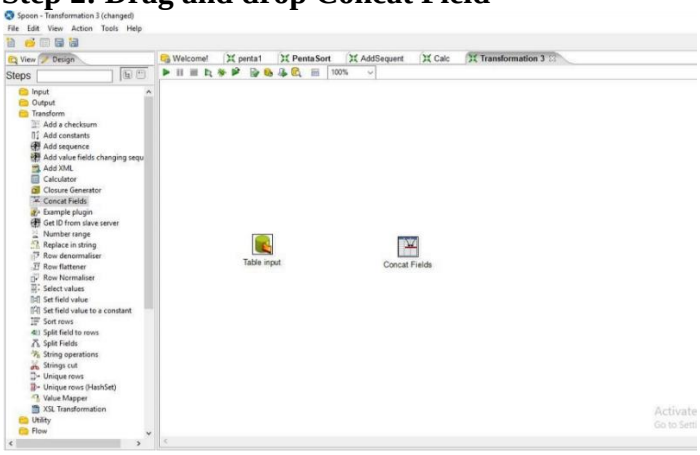
```
SQL> select * from emppp13;

       ENO FNAME      LNAME          SALARY       COMM TOTAL_SALARY
---------- ---------- ---------- ---------- ---------- ------------
         1 simran     gupta          100000      44000       144000
         2 aadarsh    choudhari      200000      74000       274000
         4 prateek    ahirrao        150000      15000       165000
         5 smit       bole           160000      12000       172000
```

**d. Concatenation Operation**

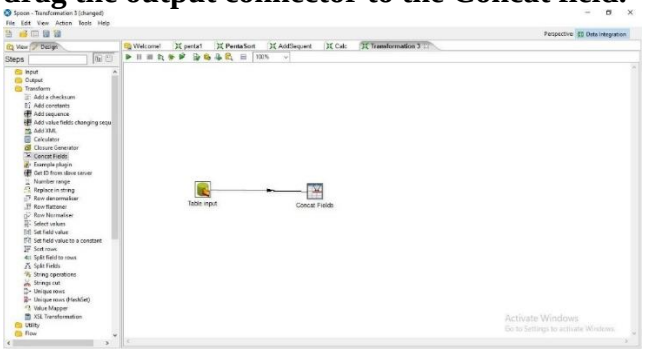
**Step 1: Perform first 6 steps same as practical 1.**



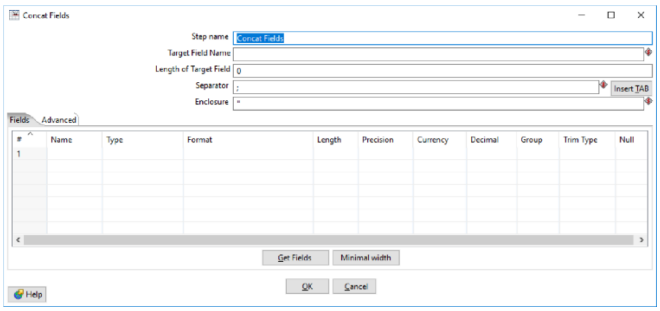
**Step 2: Drag and drop Concat Field**



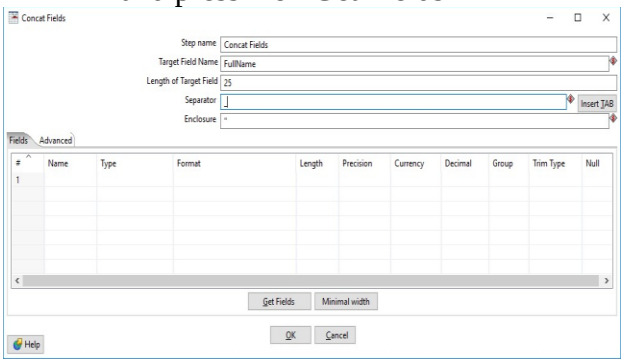
**Step 4: Hold mouse pointer on input table and drag the output connector to the Concat field.**



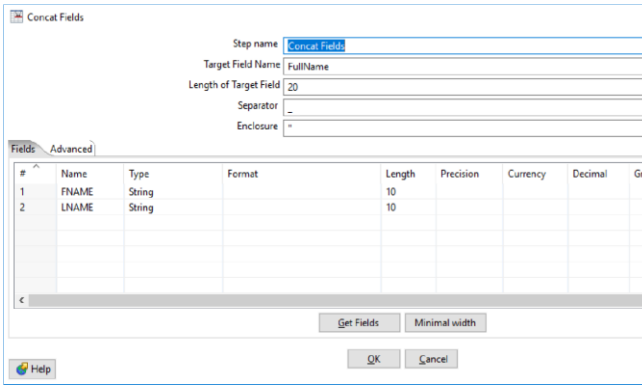
**Step 5: Double Click on Cncat Field**



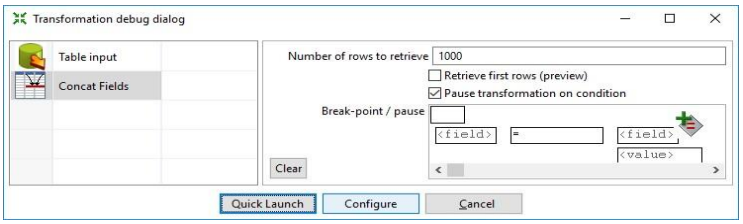
**a.** Enter the values for Target Field Name, Length of the Target Field and the Seperator and press on Get Fields

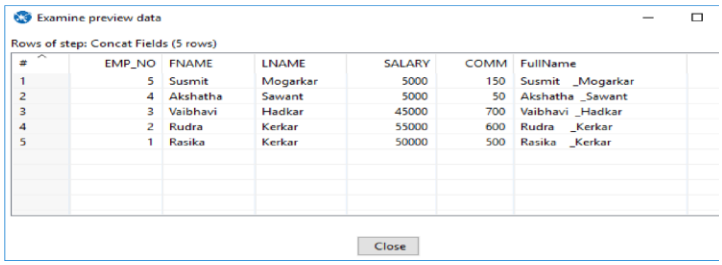


**b.** Select the columns that you want to concat

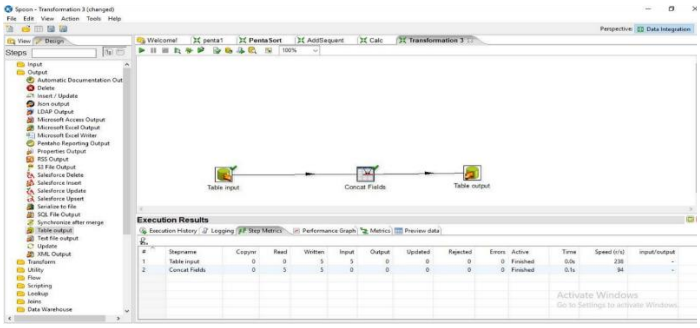


**Step 6: Click on Debug transformation and click on Quick Launch.**

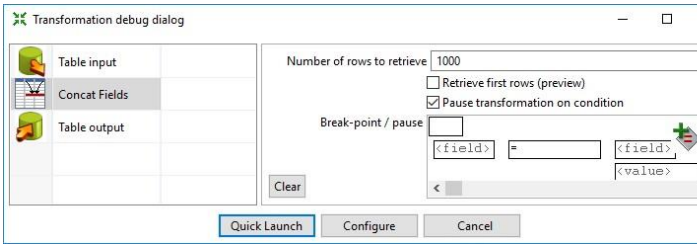




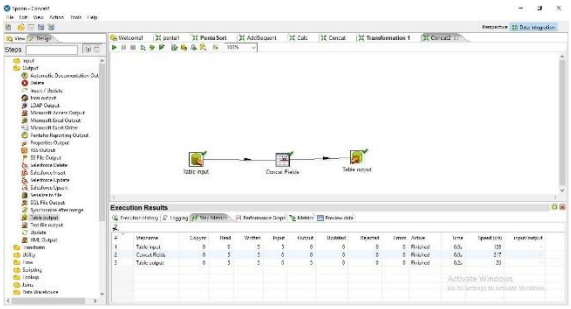

**Step 7:Now Perform the steps 6-10 same as practical 1.**

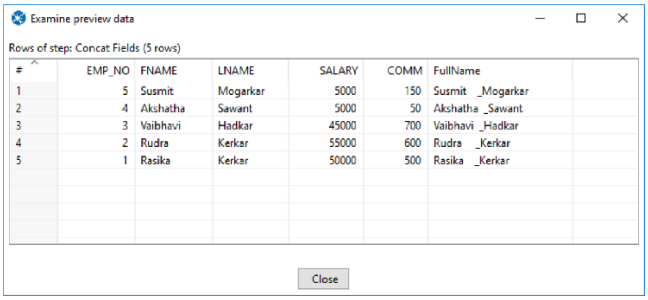



**Step 8: Click on debug Transformation and click on Quick Launch**




**Output:**

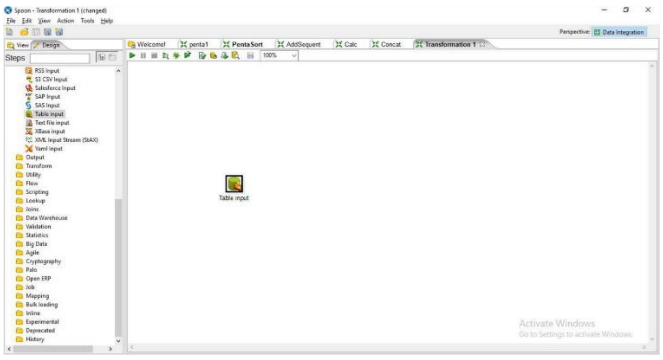




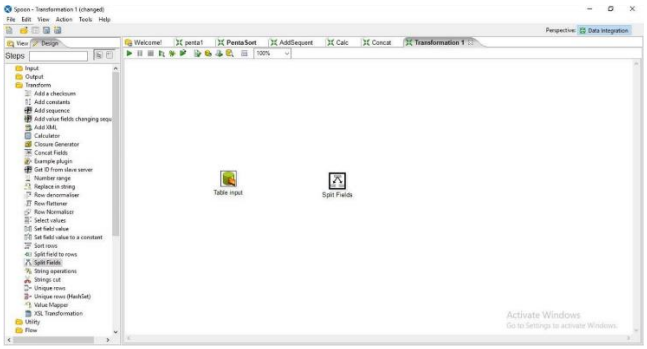

**SQL> select * from Concat;**

```
SQL> select * from emp_target_concat;

    EMP_NO FNAME                          LNAME
---------- ------------------------------ ----------
    SALARY       COMM FULLNAME
---------- ---------- ------------------------------
         1 Sonam                          Singh
     25000        500 Sonam      _Singh

         2 Pradnya                        Suryavanshi
     20000        700 Pradnya    _Suryavansh

         3 Komal                          Malviya
     17000       1000 Komal      _Malviya


    EMP_NO FNAME                          LNAME
---------- ------------------------------ ----------
    SALARY       COMM FULLNAME
---------- ---------- ------------------------------
         4 Nidhi                          Rai
     30000        900 Nidhi      _Rai

         5 Riya                           Sharma
     15000       1100 Riya       _Sharma
```

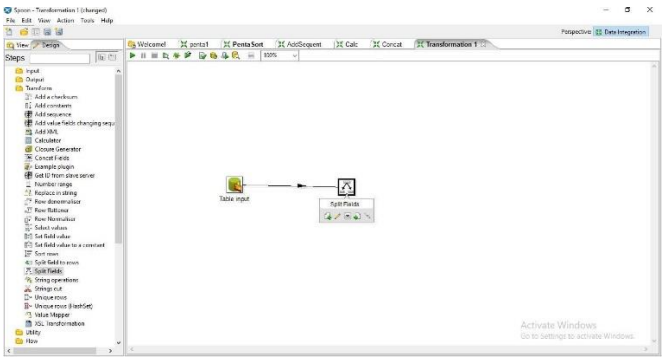
**Step 1: Perform first 6 steps same as practical 1.**



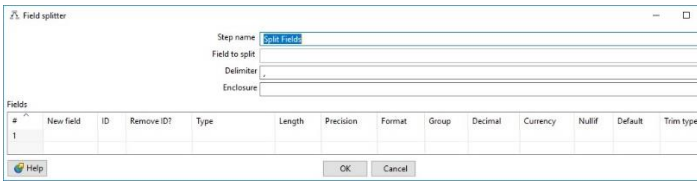

**Step 2: Drag and drop Split Field on the Panel**



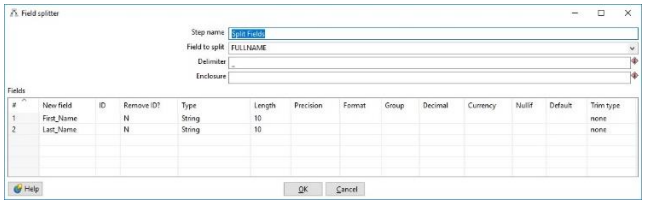

**Step 3: Hold the mouse pointer og table input and drag it to the Split Field.**



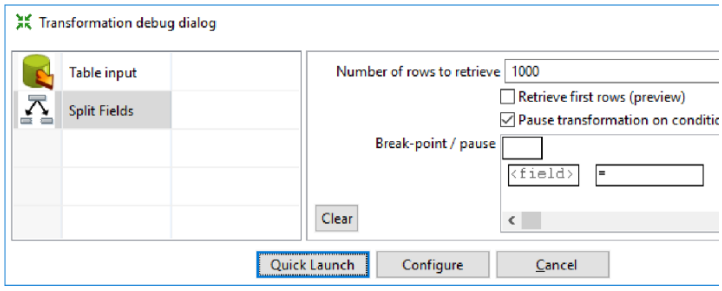
**Step 4: Double Click On Split Field.**



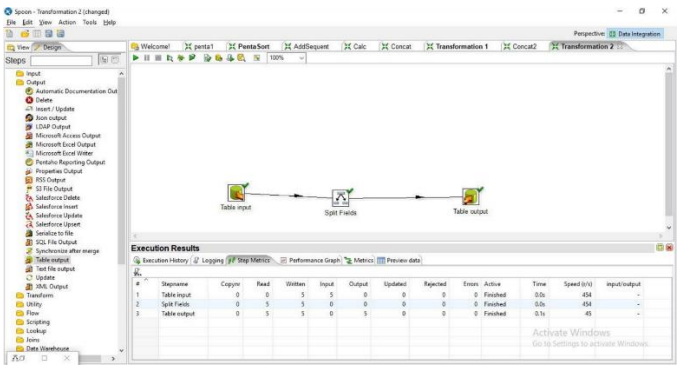
a. **Enter the values For Field to Split, Delimiter and the field names and press ok**



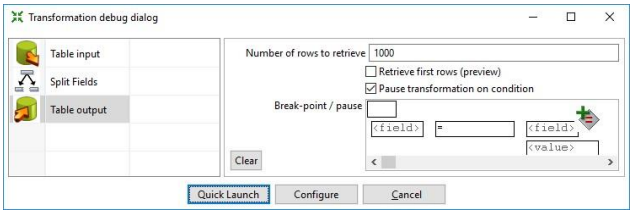
**Step 6: Click on Debug transformation and Click on Quick Launch.**



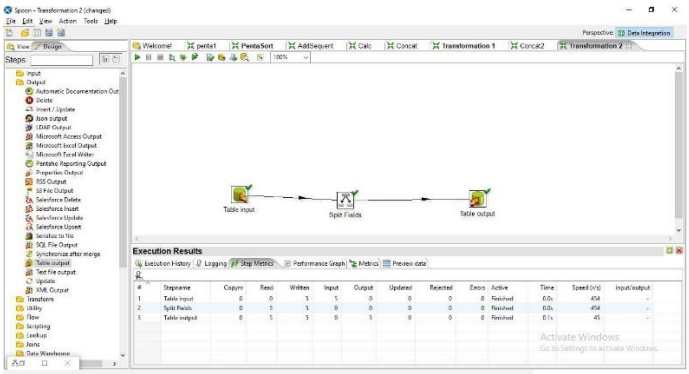

**Step 7: Perform steps 6-10 same as practical 1.**



**Step 8: Click on Debug transformation and Click on Quick Launch.**



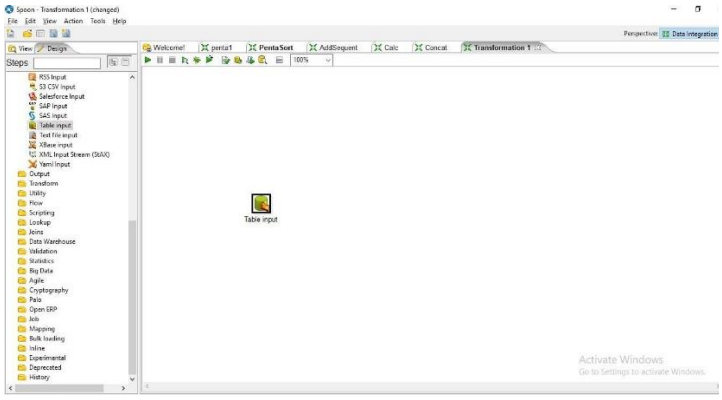

**Output:**



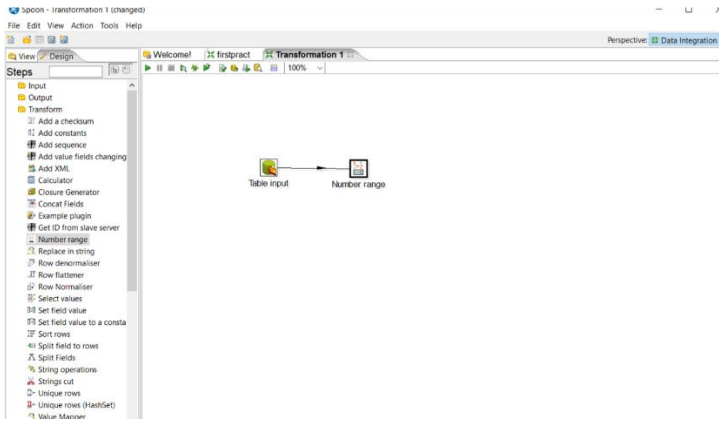
```
SQL> select * from emp_target_split;

    EMP_NO FNAME                          LNAME
---------- ------------------------------ --------------
    SALARY       COMM FIRSTNAME  LASTNAME
---------- ---------- ---------- ----------
         1 Sonam                          Singh
     25000        500 Sonam      Singh

         2 Pradnya                        Suryavanshi
     20000        700 Pradnya    Suryavansh

         3 Komal                          Malviya
     17000       1000 Komal      Malviya


    EMP_NO FNAME                          LNAME
---------- ------------------------------ --------------
    SALARY       COMM FIRSTNAME  LASTNAME
---------- ---------- ---------- ----------
         4 Nidhi                          Rai
     30000        900 Nidhi      Rai

         5 Riya                           Sharma
     15000       1100 Riya       Sharma
```

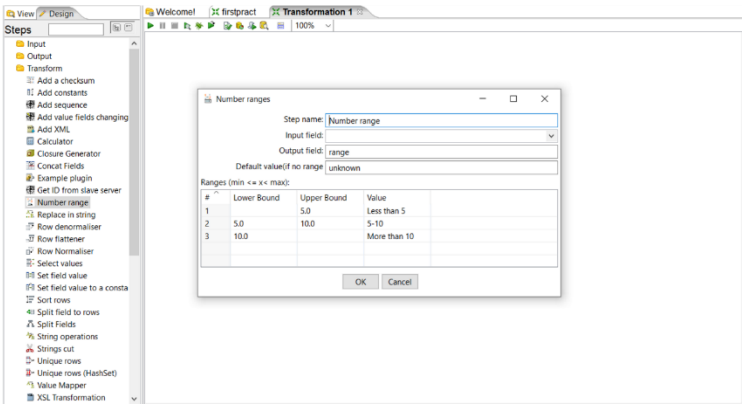
```
SQL> select * from student13;

   ROLL_NO PERCENTAGE
---------- ----------
       101         55
       102         59
       103         68
       104         75
       105         83
       106         35
       107         88
```

**Step 1: Perform first 6 steps same as practical 1.**

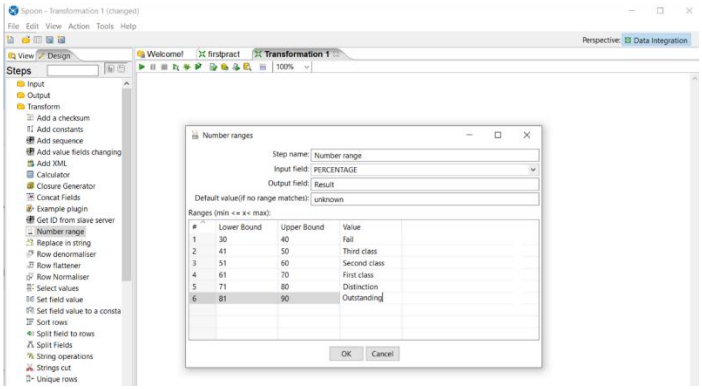


Step2: Drag and drop Number Range transformation and double click on it.

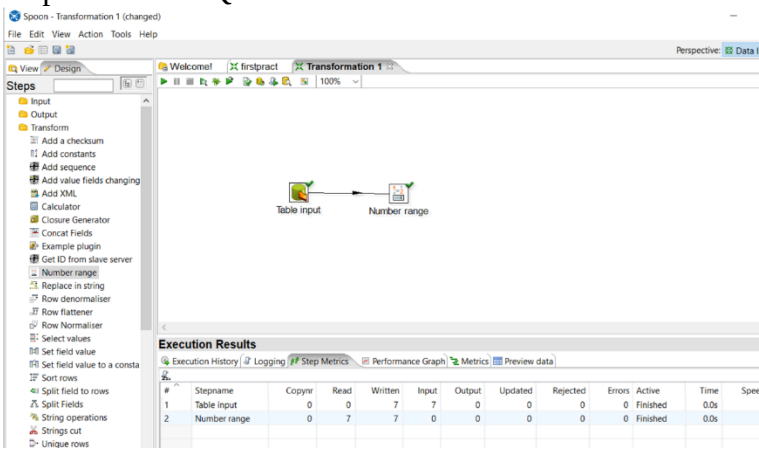




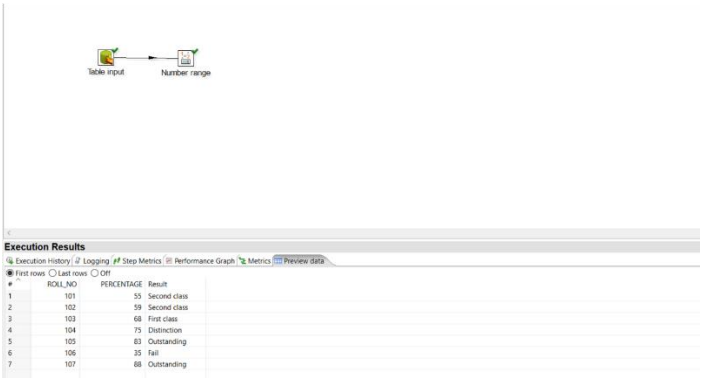
Step3: Set values for Input field, Output field, Lower bound, Upper bound and value according to the values specified in percentage column.



Step4: Click on Quick launch



Step 5: Click on Preview data

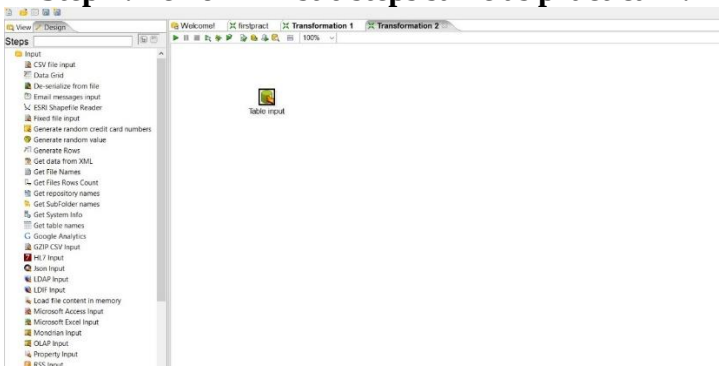



## 7. String Operation:

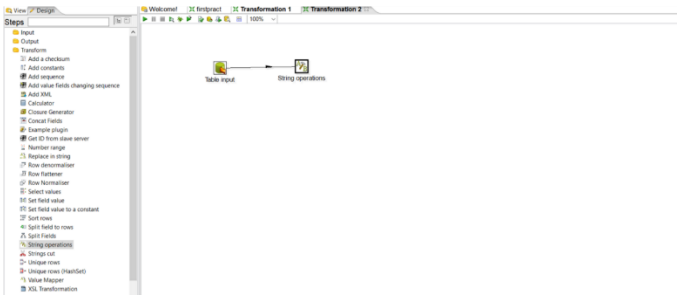
## Required SQL table:

```
SQL> select * from newemp13;

EMPN       ENAME                          DEPT
---------- ------------------------------ ------
JOB                            SALARY
------------------------------ ----------
E001       JOHN                           D002
MANAGER                         50000

E002       SMIT                           D001
HR                              40000

E003       STEVEKING                      D002
                                30000


EMPN       ENAME                          DEPT
---------- ------------------------------ ------
JOB                            SALARY
------------------------------ ----------
E004       LEX HEAN                       D003
MANAGER                         50000

E005       BRUCE                          D003
MANAGER                         50000

E006       KEVINNASH                      D002
SALES                           50000
```

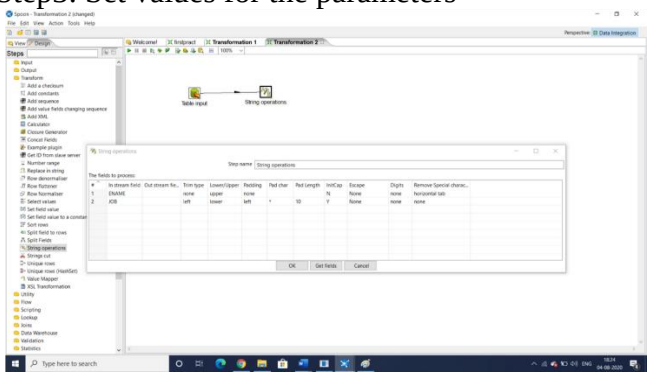
**Step 1: Perform first 6 steps same as practical 1.**



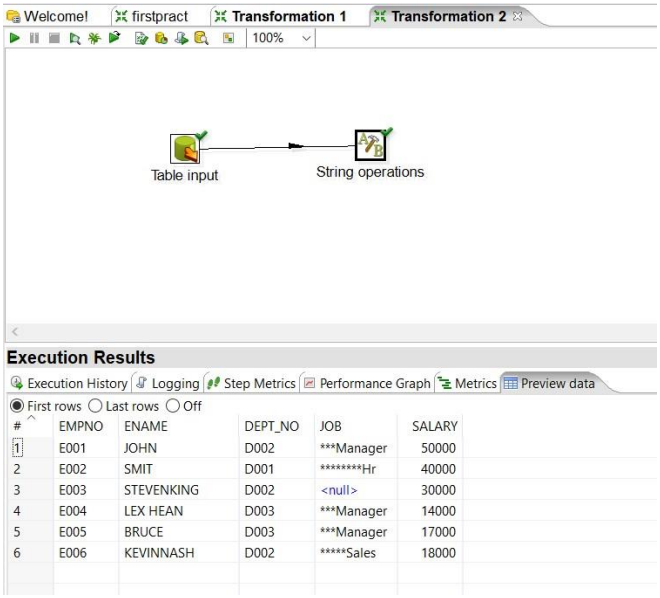

Step 2: Drag and drop String operation transformation, make connection between table input and string operation and double click on it.



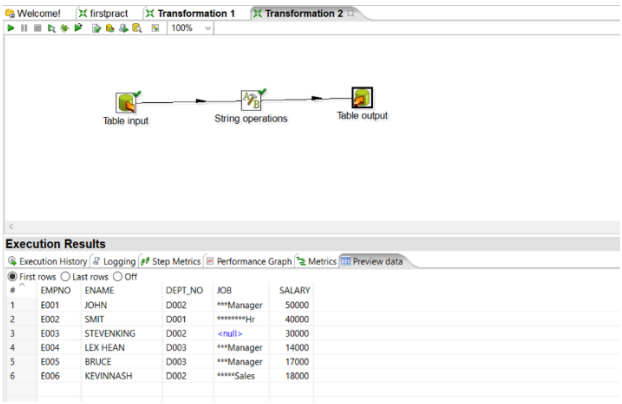

Step3: Set values for the parameters



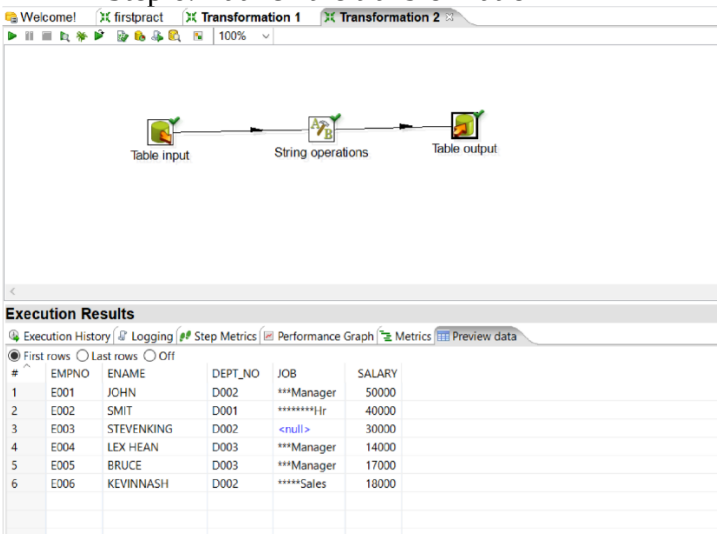

Step 4: Launch the transformation




Step 5: Drag and drop output table




Step 6: Launch the transformation



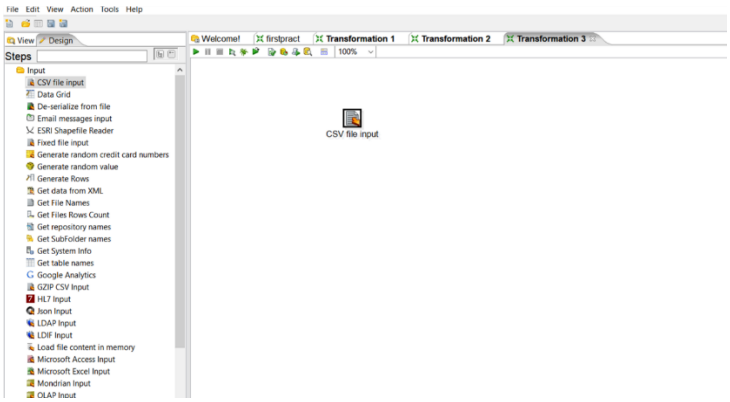

Step 7: In SQL output table

```
SQL>  select * from stringgg;

EMPN       ENAME                     DEPT
---------- ------------------------- ------------
JOB                           SALARY
----------------------------- ----------
E001       JOHN                      D002
***Manager                     50000

E002       SMIT                      D001
********Hr                     40000

E003       STEVEKING                 D002
                               30000


EMPN       ENAME                     DEPT
---------- ------------------------- ------------
JOB                           SALARY
----------------------------- ----------
E004       LEX HEAN                  D003
***Manager                     50000

E005       BRUCE                     D003
***Manager                     50000

E006       KEVINNASH                 D002
*****Sales                     50000
```

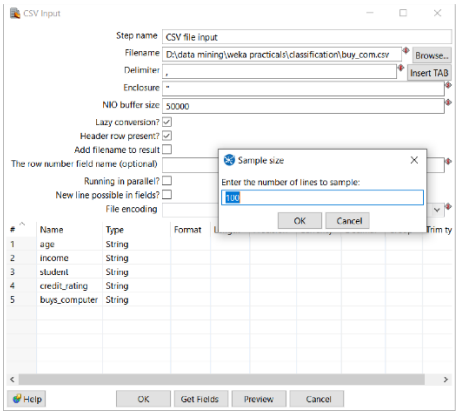
6 rows selected.

8. Copy contents of.CSV file to the target table

Step1:Drag and drop CSV file input on a panel

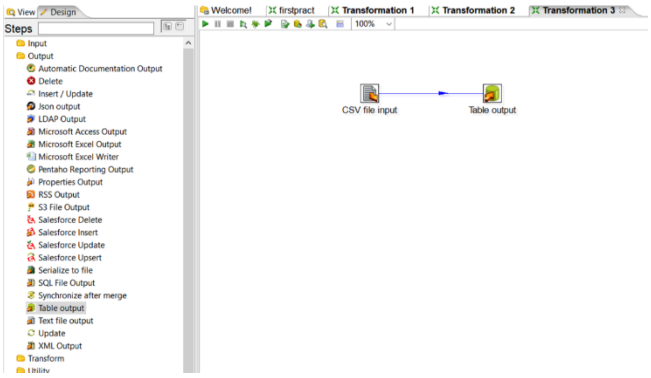



Step2: Double click on it and open respective CSV file, Lazy conversation and
Header row present? Options should be checked, click on Getfield and then ok.

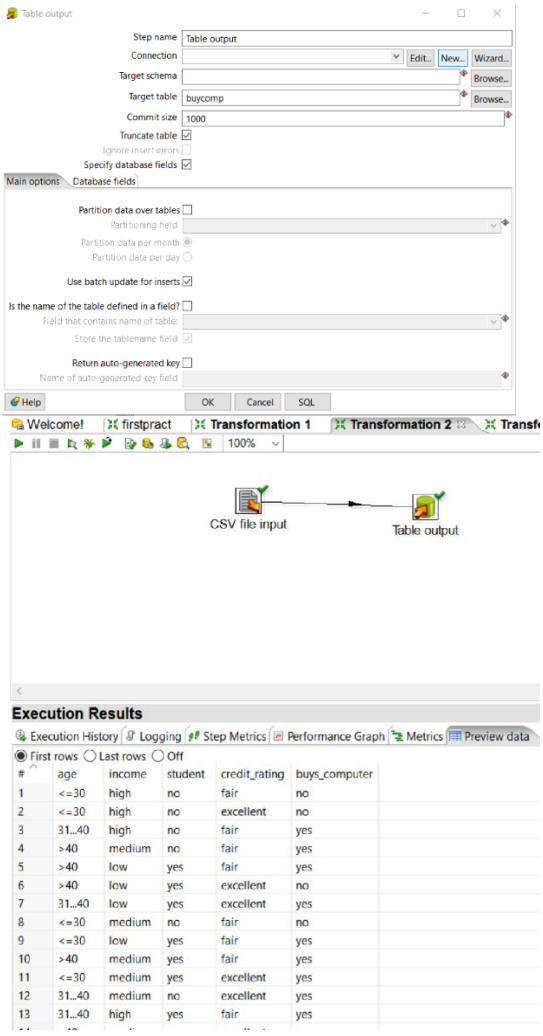



In this step we can change data type and format of CSV files after doing setting click on ok

Step3: Drag and drop Table output and connect it with CSV file.




Step4: Double click on Table output and create new connection and follow same stapes like before and launch the transformation.

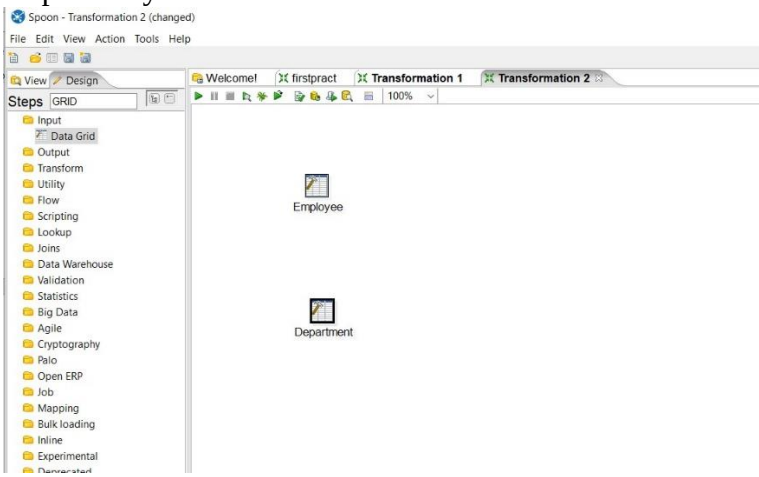



| # | age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | high | no | fair | no |
| 2 | <=30 | high | no | excellent | no |
| 3 | 31...40 | high | no | fair | yes |
| 4 | >40 | medium | no | fair | yes |
| 5 | >40 | low | yes | fair | yes |
| 6 | >40 | low | yes | excellent | no |
| 7 | 31...40 | low | yes | excellent | yes |
| 8 | <=30 | medium | no | fair | no |
| 9 | <=30 | low | yes | fair | yes |
| 10 | >40 | medium | yes | fair | yes |
| 11 | <=30 | medium | yes | excellent | yes |
| 12 | 31...40 | medium | no | excellent | yes |
| 13 | 31...40 | high | yes | fair | yes |

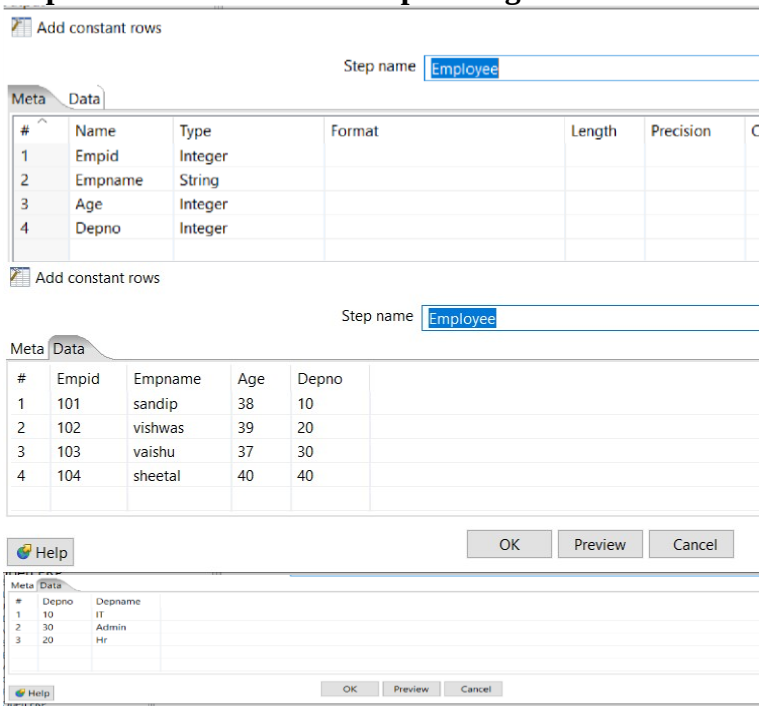
Step 5: Output table created in SQL

```
SQL> select * from csv13;

AGE   INCOME STU CREDIT_RA BUY
----- ------ --- --------- ---
<=30  high   no  fair      no
<=31  high   no  excellent no
31à40 high   no  fair      yes
>40   medium no  fair      yes
>40   low    yes fair      yes
>40   low    yes excellent no
31à40 low    yes excellent yes
<=30  medium no  fair      no
<=30  low    yes fair      yes
>40   medium yes fair      yes
<=30  medium yes excellent yes

AGE   INCOME STU CREDIT_RA BUY
----- ------ --- --------- ---
31à40 medium no  excellent yes
31à40 high   yes fair      yes
>40   medium no  excellent no

14 rows selected.
```

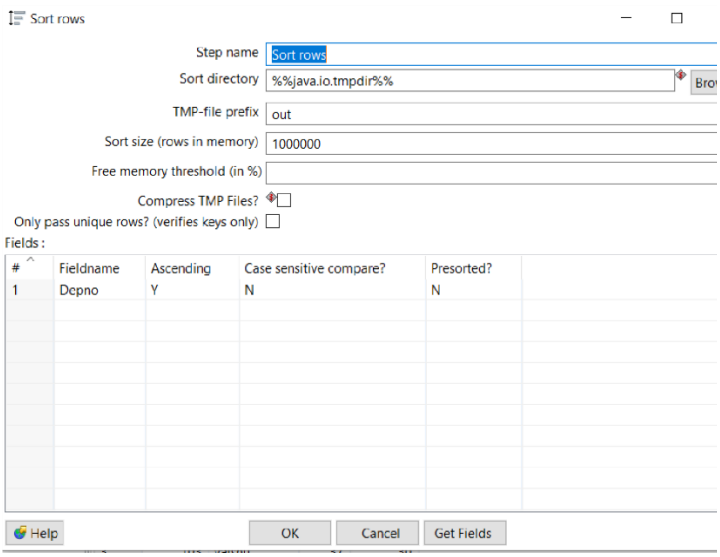
9. **Implement the merge join transformation on tables** Step1: Drag two Data grid on panel rename them with Employee and Department respectively.

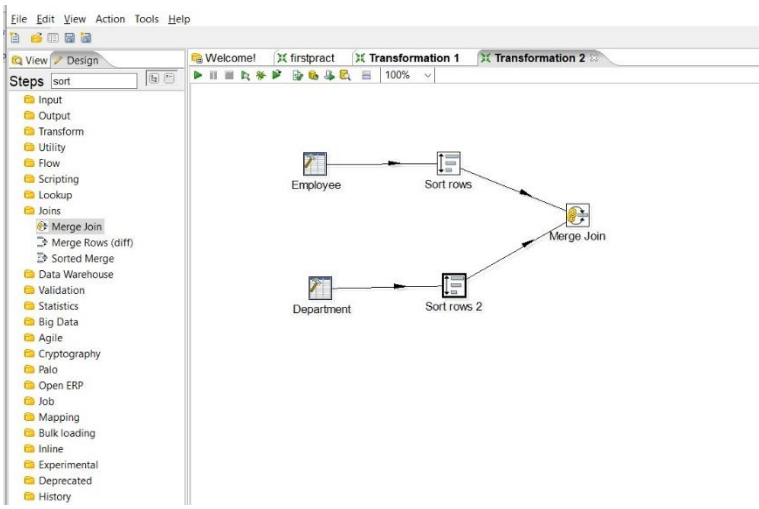



**Step2: Insert records into respective grids.**

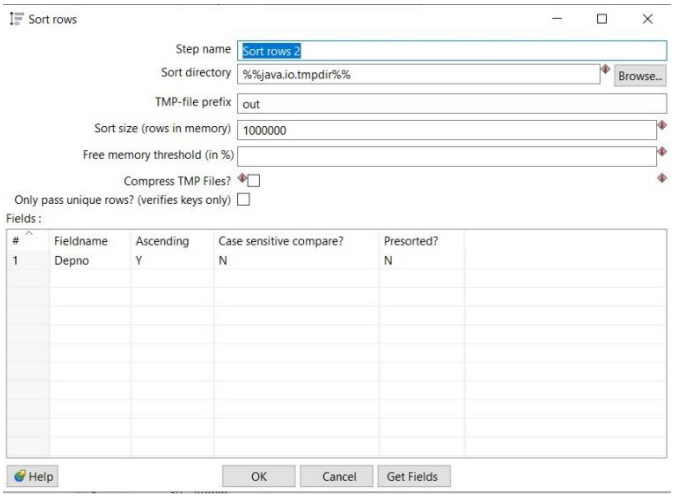


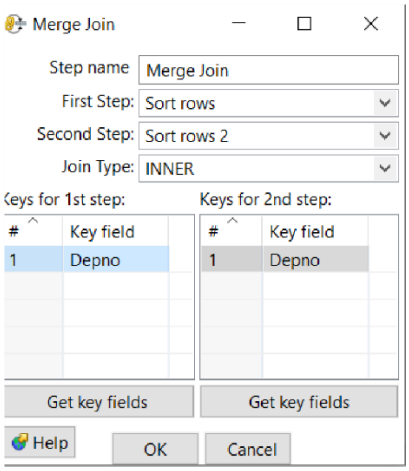

Step3: Insert sort rows transformation(on Empid) for both Employee and Department.



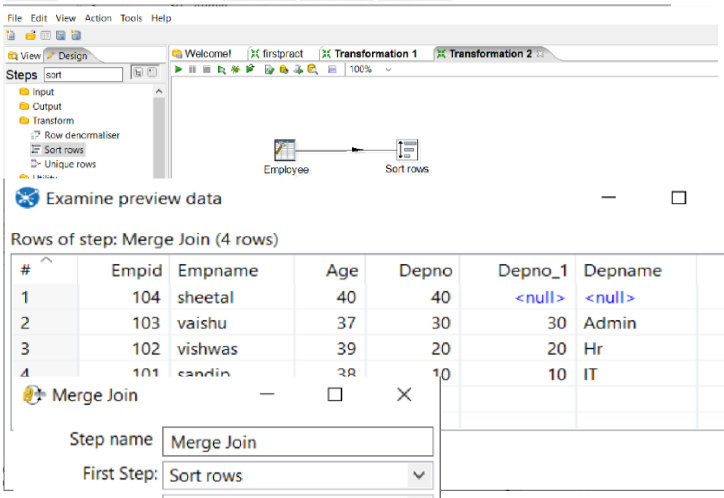




Step5: Double click on Merge join and made necessary settings First select First_step, then Second_step, select Join type INNER, and from get fields select Depno only, after that quick launch the transformation.



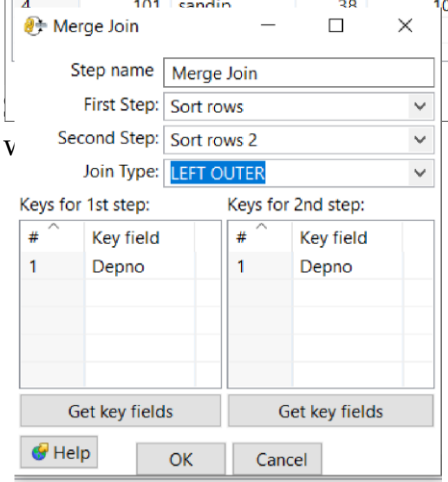



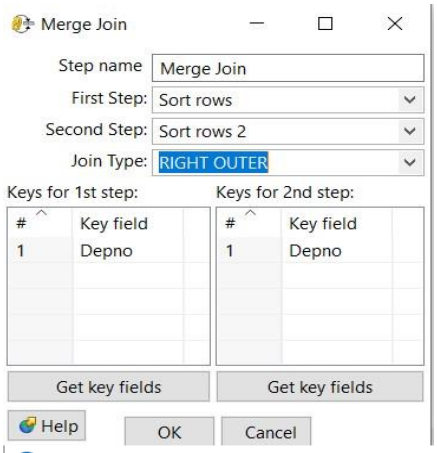

**Step 6: Select Join Type LeftOuter,then Right outer and finally Full outer join**

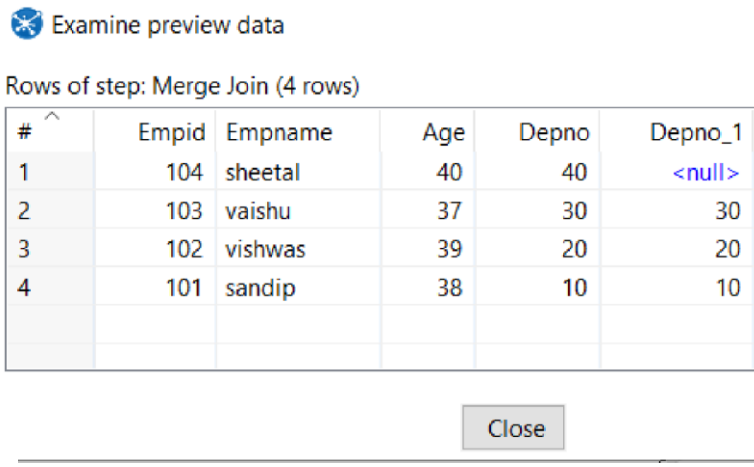


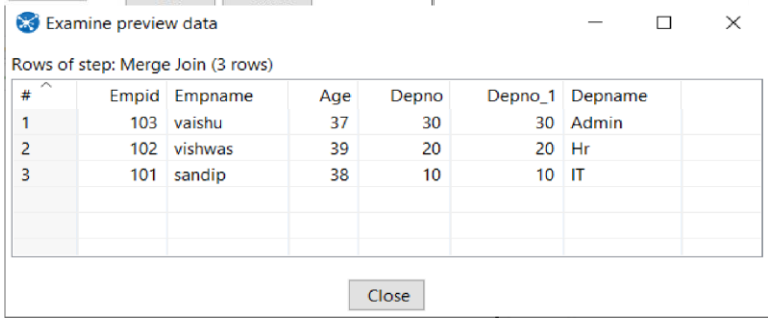


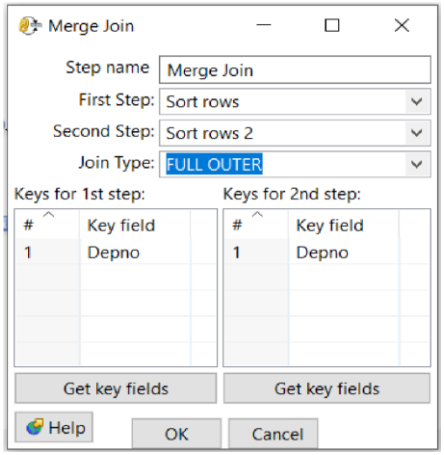




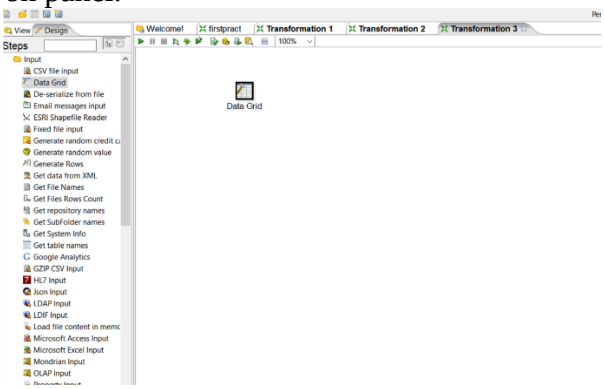


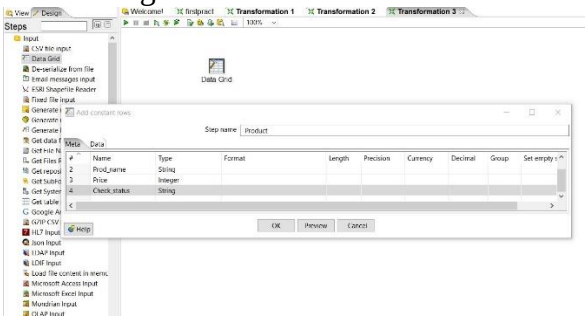



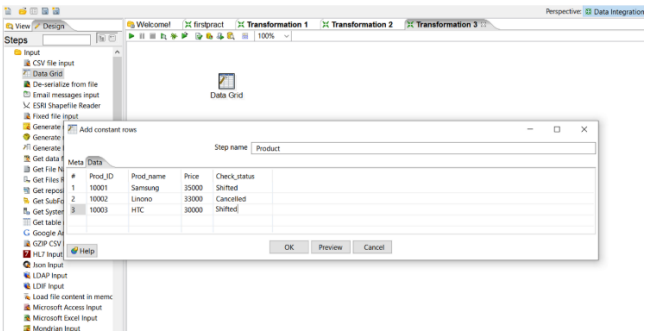




0. **Implement ifferent data alidations on able.** Step1: Drag nd drop Data grid on panel.

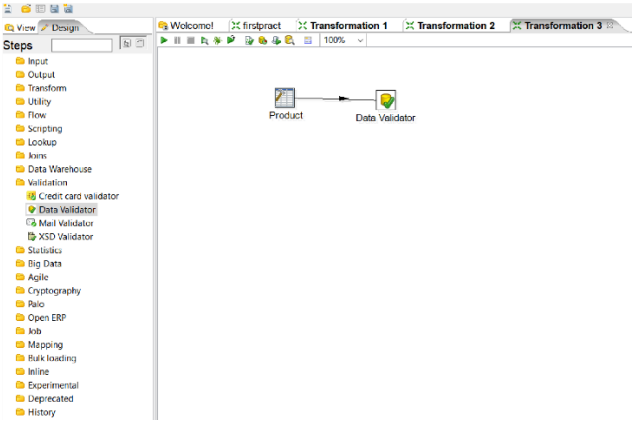



Step2: Double click on Data grid and create product table in Meta tab and enter records into it using Data tab.



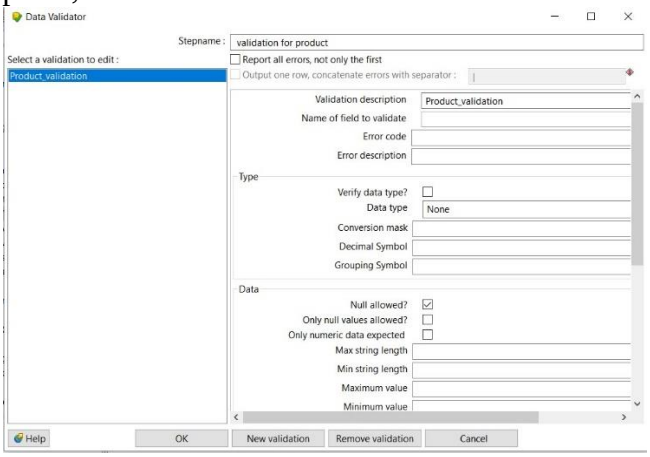


Step3: Go to Validation steps and select, Drag and Drop Data validator on panel and double click on it.

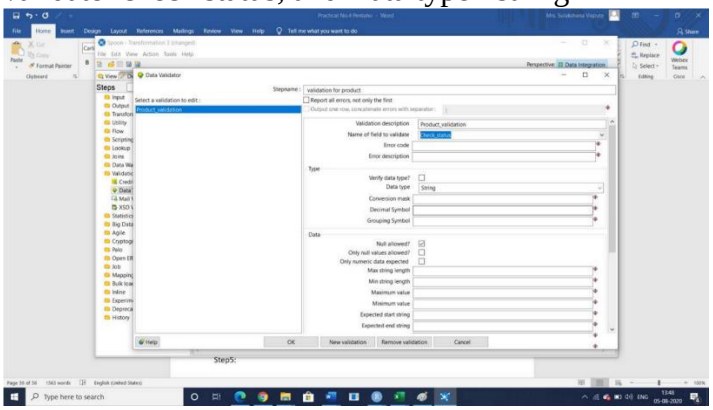


Step4: Click on new validation button and give new name to validation.

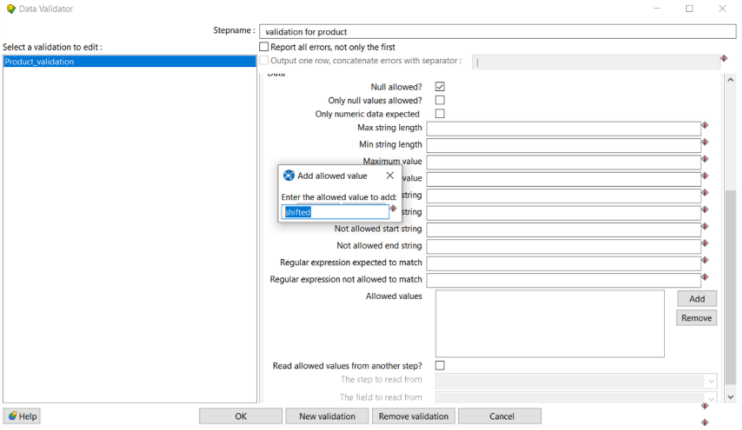
Once new validation is created it will appear on left panel, double click on it to set data validation



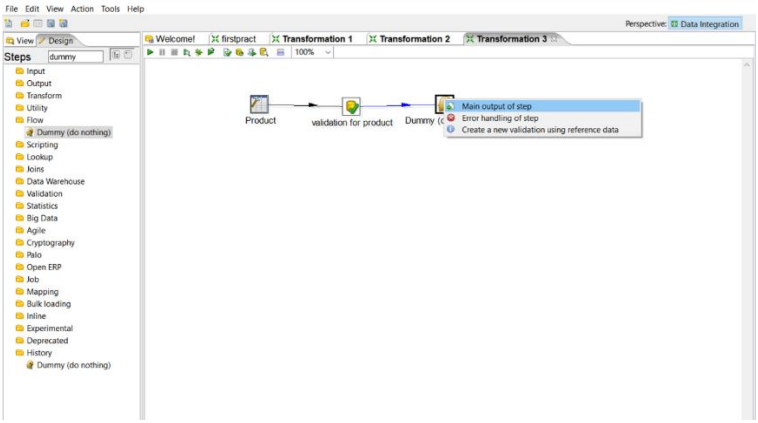
Step5: Set values for Name of field to validate=check status, and Data type =string



Step6: Click on add to set validation, set it65 to Shifted and press Enter and click on ok.

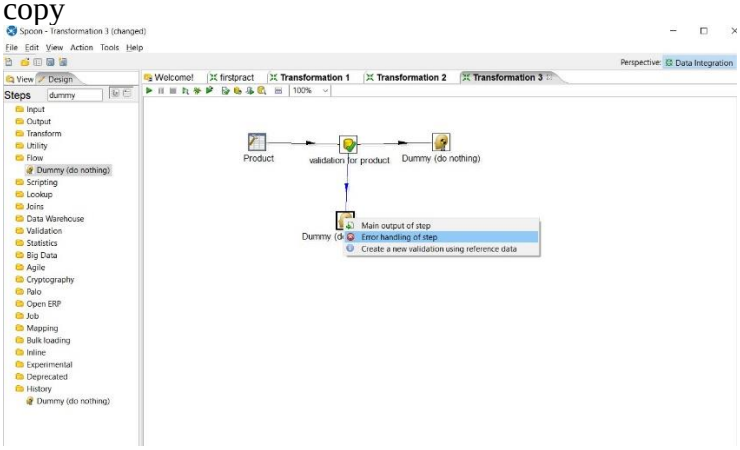


Step7: Select Dummy file from Flow to hold output and connect it with Data validation, while connecting select option " Main Output of step"

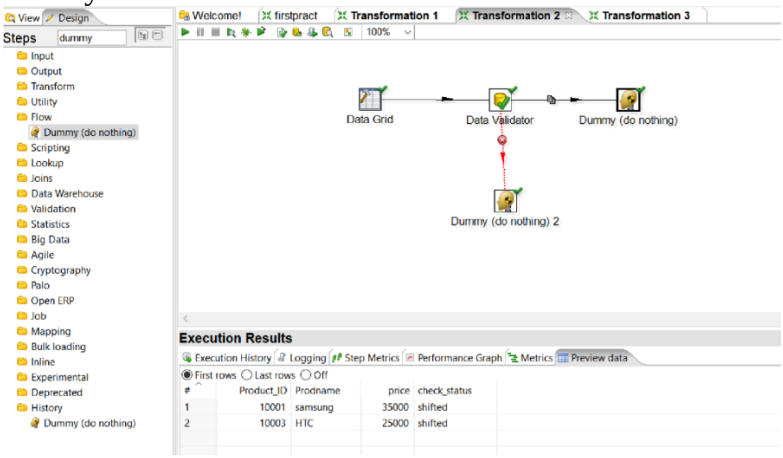


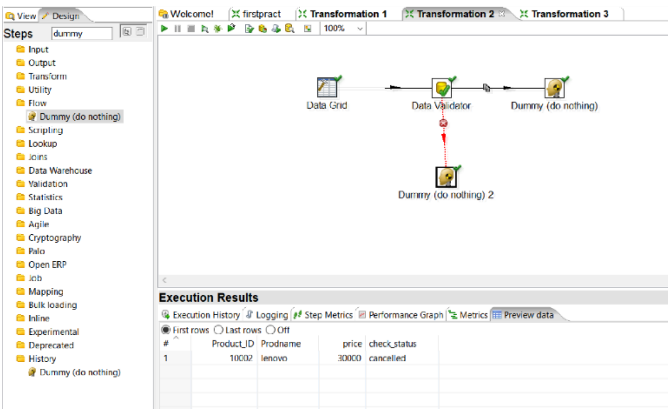
Step8: Select another dummy and connect it to 'Validation for product' and while selecting select

‘Error handling of step and in next window click on copy




Step 9: launch transformation  by selecting one dummy file each.



# Practical No. 5

**Aim: -** Introduction to R programming and Data acquisition.

**Objective: -** To understand the concept of R Programming. and evaluate different data mining techniques like classification, prediction, clustering and association rule mining in R

**R Commands:-**

```
myString <- "Hello, World!"
print ( myString)
[1] "Hello, World!"
#setwd() - sets working directory.
setwd("D:/58")
getwd() ##getwd() - gets current working directory.
[1] "D:/58"
dir() ## dir() - lists the contents of current working directory.
[1] "data_apriori.csv"  "missing_col1.csv"  "missing_col1.xlsx" "na_data.csv"       "PRACT5.R"
>
ls() ##ls() - lists names of objects in R environment
 [1] "accuracy"            "clusterCut"          "clusters"            "df"
 [5] "df2"                 "fit"                 "index"               "iris"
 [9] "iris_norm"           "iris_target_category" "iris_test"          "iris_test_category"
[13] "iris_train"          "irisCluster"         "lmmodel1"            "model"
[17] "my_data"             "myString"            "nor"                 "pr"
[21] "preds"               "preds_new"           "ran"                 "tab"
[25] "test"                "TestData"            "train"               "TrainData"
[29] "xTest"               "xTrain"              "yTest"               "yTrain"
#Creating and assigning to a variable:
x<-1
class(x)
[1] "numeric"
#Printing a variable:
#auto-printing
print(x) #explicit printing
[1] 1
x<-'c'
is.character(x) #check if character
[1] TRUE
is.integer(x) #check if integer
[1] FALSE
y<-'2.14'
as.integer(y)
[1] 2
###Creating Vector: contains objects of same class.
y<-vector("logical",length=10)
length(x)
[1] 3
y<-c(4,5,6)
5*x
[1]  56.5 137.5 169.0
###Creating Matrix: Two-dimensional array having elements of same class.
m<-matrix(c(11,12,13,55,60,65,66,72,78), nrow=3,ncol=3)
m
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
>
dim(m)
[1] 3 3
attributes(m)
$dim
[1] 3 3
matrix(c(11,12,13,55,60,65,66,72,78),nrow= 3,ncol=3,byrow = TRUE)
m
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
> m
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
>
x<-c(1,2,3) y<-c(11,12,13)
cbind(x,y)
          x y
[1,] 11.3 4
[2,] 27.5 5
[3,] 33.8 6
>
rbind(x,y)
  [,1] [,2] [,3]
x 11.3 27.5 33.8
y  4.0  5.0  6.0
>
p<-3*m
p
```

```
     [,1] [,2] [,3]
[1,]   33  165  198
[2,]   36  180  216
[3,]   39  195  234
> |
```

n<-
matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)
q<- m+n
q

```
     [,1] [,2] [,3]
[1,]   15   69   90
[2,]   17   75   97
[3,]   19   81  104
> |
```

o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)
o

```
     [,1] [,2]
[1,]    4   14
[2,]    5   15
[3,]    6   16
> |
```

r<-m%*% o
r

```
     [,1] [,2]
[1,]  715 2035
[2,]  780 2220
[3,]  845 2405
> |
```

mdash<-t(m)
mdash

```
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
> |
```

s<-matrix(c(4,5,6,14,15,16,24,25,26),
nrow=3,ncol=3,byrow=TRUE)
s_det<-det(s)
s_det

```
[1] 1.110223e-14
```

###List: A special type of vector containing elements of different classes. ####Elements of list can be accessed by giving element index or name in [[]].

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

```
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

[[4]]
[1] 2+4i
```

###Factor: Represents categorical data. Can be ordered or unordered.
status<-
c("low","high","medium","high","low")
x<-factor(status,
ordered=TRUE,levels=c("low","medium","high"))
x

```
[1] low    high   medium high   low
Levels: low < medium < high
> |
```

###Data frame: Used to store tabular data. Can contain different classes.
student_id<-c(1,2,3)
student_names<-
c("Ram","Shyam","Laxman")
position<-
c("First","Second","Third")
data<-
data.frame(student_id,student_names,position)
data

```
  student_id student_names position
1          1           Ram    First
2          2         Shyam   Second
3          3        Laxman    Third
> |
```

data$student_id

```
[1] 1 2 3
```

nrow(data)

```
[1] 3
```

ncol(data)

```
[1] 3
```

names(data)

```
[1] "student_id"    "student_names" "position"
```

smoke <-
matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
colnames(smoke) <-
c("High","Low","Middle")
rownames(smoke) <-

c("current","former","never")
smoke <- as.table(smoke)
smoke

```
         High Low Middle
current    51  43      22
former     92  28      21
never      68  22       9
> |
```

Reading and writing data from csv
dataT <- read.table("na_data.csv", sep =",", header = T)
dataT

```
   X1       Rick  X623.3 X01.01.2012         IT
1   2        Dan  515.20  23/09/2013 Operations
2   3  Michelle   611.00  15/11/2014         IT
3   4       Ryan  729.00  11/05/2014         HR
4  NA       Gary  843.25  27/03/2015    Finance
5   6      Nina       NA  21/05/2013         IT
6   7    Simon    632.80  30/07/2013 Operations
7   8       Guru  722.50  17/06/2014   Finance
8   9      John       NA  21/05/2012
9  10       Rock  600.80  30/07/2013         HR
10 11      Brad  1032.80  30/07/2013 Operations
11 12       Ryan  729.00  11/05/2014         HR
> |
```

dim(dataT)

```
[1] 11  5
```

head(dataT, 2)

```
  X1       Rick X623.3 X01.01.2012         IT
1  2        Dan  515.2  23/09/2013 Operations
2  3  Michelle   611.0  15/11/2014         IT
> |
```

tail(dataT, 2)

```
   X1  Rick X623.3 X01.01.2012         IT
10 11 Brad  1032.8  30/07/2013 Operations
11 12  Ryan  729.0  11/05/2014         HR
> |
```

z <- data.frame(a = 5, b = 10, c = pi)
write.csv(z,file="data.csv")

| A | B | C | D |
|---|---|---|---|
| | a | b | c |
| 1 | 5 | 10 | 3.141593 |

Reading and writing data from Excel using XLConnect

dataX <- XLConnect:: readWorksheetFromFile("employee_info.xls",sheet=1)
dataX

```
   X1       Rick  X623.3 X2012.01.01.00.00.00         IT
1   2        Dan  515.20           23/09/2013 Operations
2   3  Michelle   611.00           15/11/2014         IT
3   4       Ryan  729.00  2014-11-05 00:00:00         HR
4  NA       Gary  843.25           27/03/2015    Finance
5   6      Nina       NA           21/05/2013         IT
6   7    Simon    632.80           30/07/2013 Operations
7   8       Guru  722.50           17/06/2014   Finance
8   9      John       NA           21/05/2012       <NA>
9  10       Rock  600.80           30/07/2013         HR
10 11      Brad  1032.80           30/07/2013 Operations
11 12       Ryan  729.00  2014-11-05 00:00:00         HR
> |
```

dataY<- dataX[1:2,]
dataY

```
  X1       Rick X623.3 X2012.01.01.00.00.00         IT
1  2        Dan  515.2           23/09/2013 Operations
2  3  Michelle   611.0           15/11/2014         IT
> |
```

Reading and writing data from Excel using readXL and writeXL
data <-data.frame(Name=character(), Age=numeric())

# Practical No. 6

**Aim: - Implementation of Data preprocessing techniques like,**

1. Naming and Renaming variables, adding a new variable.

2. Dealing with missing data.

3. Dealing with categorical data.

4. Data reduction using sub setting.

**R Commands:-**

#my_data

```
> setwd("D:/46_fymca")
> getwd()
[1] "D:/46_fymca"
> data<-mtcars
> head(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
> data1<-data[1:6,1:5]
> 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
```

## Renaming columns with dplyr::rename()

```
> require(dplyr)
> install.packages("dplyr")
Error in install.packages : Updating loaded packages
> install.packages("dplyr")
WARNING: Rtools is required to build R packages but is not currently ins
 download and install the appropriate version of Rtools before proceedir

https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  package 'dplyr' is in use and will not be installed
> data1=rename(data1,horse_power=hp)
> 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
```

## Adding new variable

```
> data1$new_hp1 <- data1$horse_power*0.5
> colnames(data1)
[1] "mpg"         "cyl"         "disp"        "horse_power" "drat"
> data1
                   mpg cyl disp horse_power drat new_hp1
Mazda RX4         21.0   6  160         110 3.90    55.0
Mazda RX4 Wag     21.0   6  160         110 3.90    55.0
Datsun 710        22.8   4  108          93 3.85    46.5
Hornet 4 Drive    21.4   6  258         110 3.08    55.0
Hornet Sportabout 18.7   8  360         175 3.15    87.5
Valiant           18.1   6  225         105 2.76    52.5
```

#naming variable

Create missing_col1.csv file with following data.

| | | | | |
|---|---|---|---|---|
| 1 | Rick | 623.3 | 01/01/2012 | IT |
| 2 | Dan | 515.2 | 23/09/2013 | Operations |
| 3 | Michelle | 611 | 15/11/2014 | IT |
| 4 | Ryan | 729 | 11/05/2014 | HR |
| | Gary | 843.25 | 27/03/2015 | Finance |
| 6 | Nina | | 21/05/2013 | IT |
| 7 | Simon | 632.8 | 30/07/2013 | Operations |
| 8 | Guru | 722.5 | 17/06/2014 | Finance |
| 9 | John | | 21/05/2012 | |
| 10 | Rock | 600.8 | 30/07/2013 | HR |
| 11 | Brad | 1032.8 | 30/07/2013 | Operations |
| 12 | Ryan | 729 | 11/05/2014 | HR |

#Reading with read.table() assumes no headers by default. First few lines :

```
> data2 = read.table(file= "D:/46_fymca/missing_col1.csv", sep = ",")
> data2=read.table(file="D:/46_fymca/na_data.csv",sep = ",")
> data2
   V1          V2      V3          V4          V5
1   1        Rick  623.30 01/01/2012          IT
2   2         Dan  515.20 23/09/2013  Operations
3   3   Michelle   611.00 15/11/2014          IT
4   4        Ryan  729.00 11/05/2014          HR
5  NA        Gary  843.25 27/03/2015     Finance
6   6        Nina      NA 21/05/2013          IT
7   7       Simon  632.80 30/07/2013  Operations
8   8        Guru  722.50 17/06/2014     Finance
9   9        John      NA 21/05/2012
10 10        Rock  600.80 30/07/2013          HR
11 11       Brad  1032.80 30/07/2013  Operations
12 12        Ryan  729.00 11/05/2014          HR
```

#V1, V2, V3.. are given as default names (titles) by R

```
> data2=read.csv(file="D:/46_fymca/na_data.csv",col.names=c("sn
o","Name","Salary","DateofJoining","Department"))
> data2
   sno       Name  Salary DateofJoining Department
1    2        Dan  515.20    23/09/2013 Operations
2    3   Michelle  611.00    15/11/2014         IT
3    4       Ryan  729.00    11/05/2014         HR
4   NA       Gary  843.25    27/03/2015    Finance
5    6       Nina      NA    21/05/2013         IT
6    7      Simon  632.80    30/07/2013 Operations
7    8       Guru  722.50    17/06/2014    Finance
8    9       John      NA    21/05/2012
9   10       Rock  600.80    30/07/2013         HR
10  11       Brad 1032.80    30/07/2013 Operations
11  12       Ryan  729.00    11/05/2014         HR
```

Error Detection and Correction NA: Not Available - Known as missing values

Works as a place holder for something that is 'missing'

Most basic operations (addition, subtraction, multiplication, etc.) in R deal with it without crashing and return NA if one of the inputs is NA

is.na(VALUE) is used to check if the input value is NA or not. Returns a TRUE/FALSE vector Whereas in case of Excel like utilities for numeric computations it's assumed to be 0.

# Operation with NA

```
NA+4
[1]NA
```

# Create a vector V with 1 NA value
v <- c(1,2,NA,3)

# Median with and without NA (remove NA)

```
> median(v)
[1] NA
```

# On removing NAs

```
> median(v,na.rm = T)
[1] 2
```

# Apply is.na() to vector
>is.n(V)
[1] FALSE FALSE TRUE FALSE

# Removing the NA values by using logical indexing

```
> naVals <- is.na(v)
```

# Get values that are not NA
V[!naVals]
[1] 1 2 3

# Subsetting with complete cases - values that are not NA

V[complete.cases(V)]

[1] 1 2 3

#Create na_data.csv file with following data.

| | | | | |
|---|---|---|---|---|
| 1 | Rick | 623.3 | 01/01/2012 | IT |
| 2 | Dan | 515.2 | 23/09/2013 | Operations |
| 3 | Michelle | 611 | 15/11/2014 | IT |
| 4 | Ryan | 729 | 11/05/2014 | HR |
| | Gary | 843.25 | 27/03/2015 | Finance |
| 6 | Nina | | 21/05/2013 | IT |
| 7 | Simon | 632.8 | 30/07/2013 | Operations |
| 8 | Guru | 722.5 | 17/06/2014 | Finance |
| 9 | John | | 21/05/2012 | |
| 10 | Rock | 600.8 | 30/07/2013 | HR |
| 11 | Brad | 1032.8 | 30/07/2013 | Operations |
| 12 | Ryan | 729 | 11/05/2014 | HR |

# Subsetting a data frame with complete cases
# Complete Data of Prime Ministers. Notice NAs

```
> dataC <- read.csv(file ="D:/46_fymca/na_data.csv", na.strings = "")
> dataC
   X1       Rick  X623.3 X01.01.2012         IT
1   2        Dan  515.20  23/09/2013 Operations
2   3  Michelle   611.00  15/11/2014         IT
3   4       Ryan  729.00  11/05/2014         HR
4  NA       Gary  843.25  27/03/2015    Finance
5   6       Nina      NA  21/05/2013         IT
6   7     Simon    632.80  30/07/2013 Operations
7   8       Guru  722.50  17/06/2014    Finance
8   9      John       NA  21/05/2012       <NA>
9  10       Rock  600.80  30/07/2013         HR
10 11      Brad  1032.80  30/07/2013 Operations
11 12       Ryan  729.00  11/05/2014         HR
```

# Subset only the rows without NA

```
> dataCompleteCases <- dataC[complete.cases(dataC),]
> dataCompleteCases
   X1       Rick X623.3 X01.01.2012         IT
1   2        Dan  515.2  23/09/2013 Operations
2   3  Michelle   611.0  15/11/2014         IT
3   4       Ryan  729.0  11/05/2014         HR
6   7     Simon    632.8  30/07/2013 Operations
7   8       Guru  722.5  17/06/2014    Finance
9  10       Rock  600.8  30/07/2013         HR
10 11      Brad  1032.8  30/07/2013 Operations
11 12       Ryan  729.0  11/05/2014         HR
```

Imputation
The process of estimating or deriving missing values There are various methods for imputation
– Imputation of the mean
– Imputation of the median
– Imputation using linear regression models
• Package Hmisc implments many imputation methods, few examples :

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

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/ADMIN/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/Hmisc_4.7-2.zip'
Content type 'application/zip' length 3302837 bytes (3.1 MB)
downloaded 3.1 MB

package 'Hmisc' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ADMIN\AppData\Local\Temp\RtmpCiYL4J\downloaded_packages
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units

Warning messages:
1: package 'Hmisc' was built under R version 4.1.3
2: package 'ggplot2' was built under R version 4.1.3
```

## create a vector

```
> x=c(1,2,3,NA,4,4,NA)
```

# mean imputation - from package, mention name of function to be used

```
> 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*
```

#median imputation

```
> 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*
```

** Categorical Data **

## Factors are variables in R which take on a limited number of different values; such variables are often referred to as categorical variables.

#Convert Character into Factor(categorical data)

# Create gender vector

```
> gender_vector <- c("Male", "Female",
> class(gender_vector)
[1] "character"
```

# Convert gender_vector to a factor

```
> factor_gender_vector <- factor(gende
> class(factor_gender_vector)
[1] "factor"
```

# Create Ordinal categorical vector

```
> day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
```

# Convert `day_vector` to a factor with ordered level

```
> factor_day <- factor(day_vector, order = TRUE, levels =c('morning','midday', 'afternoo
n','evening','midnight'))
```

# Print the new variable

```
> factor_day
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
```

# Convert Numeric to Factor

# Creating vectors

```
> age <- c(40, 49, 48, 40, 67, 52, 53)
> salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)
> gender <- c("male","male", "transgender", "female", "male", "female", "transgender")
```

# Creating data frame named employee

```
> employee<-data.frame(age, salary, gender)
> employee
  age salary      gender
1  40 103200        male
2  49 106200        male
3  48 150200 transgender
4  40  10606      female
5  67  10390        male
6  52  14070      female
7  53  10220 transgender
```

# Creating a factor corresponding to age with labels

```
> wfact = cut(employee$age, 3,labels=c('Young',
> table(wfact)
wfact
 Young Medium   Aged
     4      2      1
>
```

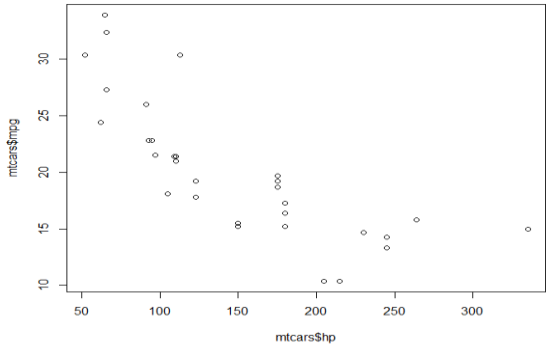
## Practical 7

**Aim: -** Implementation and analysis of Linear regression through graphical methods

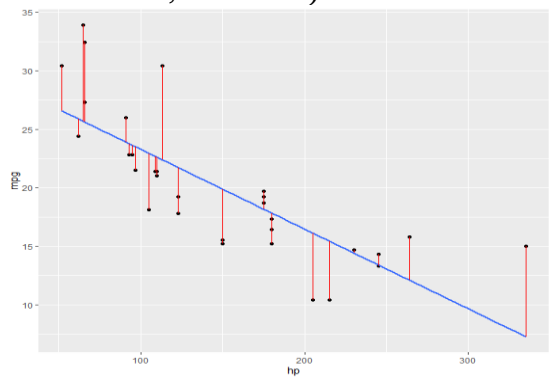
**Theory: -** Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

**R Commands:**

```
# setwd("D:/53 FYMCA/prac7")
#getwd()
[1] "D:/53 FYMCA/prac7"
# my_data <-mtcars
# names(my_data)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "q
# dim(my_data)
------------ RANDOMIZE----------------
# my_data <- my_data[sample(nrow(my_data),
), ]
# head(my_data)
                  mpg cyl  disp  hp drat
Honda Civic      30.4   4  75.7  52 4.93 1.6
Lotus Europa     30.4   4  95.1 113 3.77 1.5
Merc 280         19.2   6 167.6 123 3.92 3.4
Merc 230         22.8   4 140.8  95 3.92 3.1
Hornet 4 Drive   21.4   6 258.0 110 3.08 3.2
AMC Javelin      15.2   8 304.0 150 3.15 3.4
# TrainData <- my_data[1:20,]
# TestData <- my_data[21:32,]
----------Linear Model-------------
# fit = lm(mpg ~ hp, data=mtcars)
# summary(fit)
Call:
lm(formula = mpg ~ hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-5.7121 -2.1122 -0.8854  1.5819  8.2360

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
hp          -0.06823    0.01012  -6.742 1.79e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,    Adjusted R-squared:  0.5892
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
# preds <- predict(fit, newdata = TestData)
# df1 <-data.frame(preds,TestData$mpg)
# head(df1)
                      preds TestData.mpg
Fiat X1-9          25.59579         27.3
Lincoln Continental 15.42978         10.4
Hornet Sportabout  18.15891         18.7
Toyota Corona      23.48072         21.5
Chrysler Imperial  14.40636         14.7
Mazda RX4          22.59375         21.0
#Correlation :- cor(preds,TestData$mpg)
 [1] 0.8784059
# plot(mtcars$hp, mtcars$mpg)
```




```
# ggplot(fit, aes(hp, mpg)) + geom_point() +
 stat_smooth(method = lm, se = FALSE) +
 geom_segment(aes(xend = hp, yend = .fitted),
color = "red", size = 0.3)
```



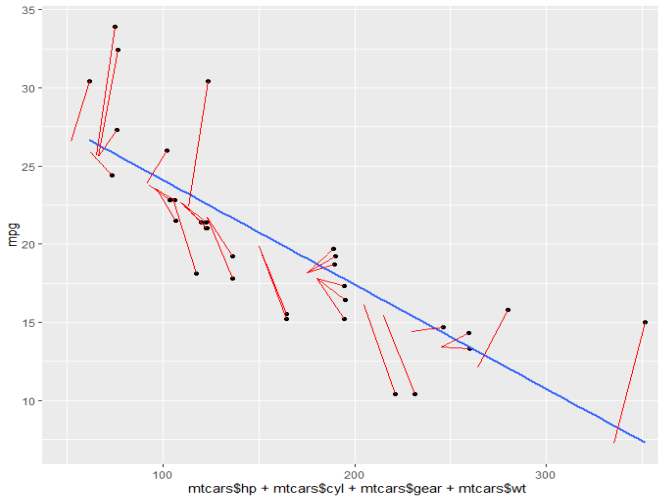

```
------
----Better MODEL------------------
# lmmodel1 <- lm(mpg ~ hp+cyl+gear+wt, data
=TrainData)
# summary(lmmodel1)
```

```
Call:
lm(formula = mpg ~ hp + cyl + gear + wt, data = TrainData)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9431 -1.1795 -0.4437  0.4191  4.1244

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 50.719998   6.657175   7.619 1.56e-06
hp          -0.001659   0.015130  -0.110 0.914155
cyl         -1.684467   0.760197  -2.216 0.042588
gear        -1.310123   1.008465  -1.299 0.213515
wt          -4.667897   1.082578  -4.312 0.000617

(Intercept) ***
hp
cyl         *
gear
wt          ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.976 on 15 degrees of freedom
Multiple R-squared:  0.9138,    Adjusted R-squared:  0.8908
F-statistic: 39.73 on 4 and 15 DF,  p-value: 8.187e-08
```

# preds_new <- predict(lmmodel1, newdata = TestData)
# df2 <-data.frame(preds_new,TestData$mpg)
# head(df2)

| | preds_new | TestData.mpg |
|---|---|---|
| Fiat X1-9 | 29.599785 | 27.3 |
| Lincoln Continental | 7.638601 | 10.4 |
| Hornet Sportabout | 16.966057 | 18.7 |
| Toyota Corona | 28.384503 | 21.5 |
| Chrysler Imperial | 7.982484 | 14.7 |
| Mazda RX4 | 22.960359 | 21.0 |

----------Corelation---------
# cor(preds_new,TestData$mpg)
[1] 0.8891933
# ggplot(fit,
aes(mtcars$hp+mtcars$cyl+mtcars$gear+mtcars $wt, mpg)) + geom_point() +
# stat_smooth(method = lm, se = FALSE) +
# geom_segment(aes(xend = hp, yend = .fitted), color = "red", size = 0.3)



# Practical 8

**Aim: -** Implementation and analysis of Classification algorithms like
1. Naive Bayesian,
2. K-Nearest Neighbor.

**Theory:-**

Naive Bayes

• Based on the Bayes theorem

• Predicts based on probabilities from training data

P(B|A) = P(A|B) P(B)/P(A) Gives posterior probability of 'B' given 'A' using prior probability of 'B' prior probability of 'A' and conditional probability of 'A' given 'B'

•Takes two step approach

– Calculates the posterior probability of the Class given the input - for every class

– Assigns the class with higher posterior probability

• More suited when dimensionality of input is high the - widely used for document classification

• Also good for the multiclass classifications

• Works well with less datasets also, but the assumption that predictor variables are independent should hold ##NaiveBayes .

**R Commands:-**

setwd("E:/R Orientation")

library("caret")

library(ggplot2)

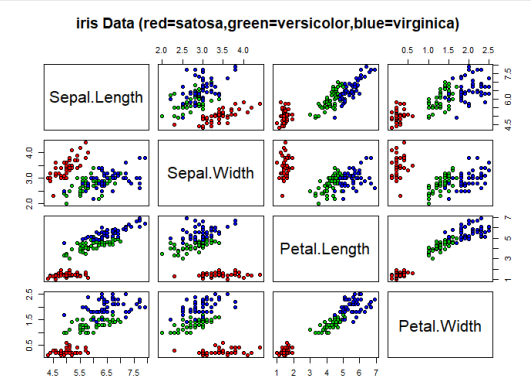
data(iris)

head(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
> |
```

unique(iris$Species)

pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)", pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])




index = sample(nrow(iris), floor(nrow(iris) * 0.7))

train = iris[index,] test = iris[-index,]

xTrain = train[,-5]

yTrain = train$Species

xTest = test[,-5]

yTest =test$Species

model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))

model

```
> model
Naive Bayes

105 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 96, 95, 94, 93, 94, 94, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9336364  0.9003204
   TRUE      0.9436364  0.9155895

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
> |
```

prop.table(table(predict(model$finalModel,xTest)$class,yTest))

```
> prop.table(table(predict(model$finalModel,xTest)$class,yTest))
            yTest
                setosa versicolor  virginica
  setosa     0.31111111 0.00000000 0.00000000
  versicolor 0.00000000 0.26666667 0.02222222
  virginica  0.00000000 0.00000000 0.40000000
> |
```

K-Nearest Neighbor

df <- data(iris)

head(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)-
min(x)) }
iris_norm <-
as.data.frame(lapply(iris[,c(1,2,3,4)], nor))
summary(iris_norm)
```

```
> summary(iris_norm)
  Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333
 Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000
 Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.45806
 3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
>
```

```
iris_train <-iris_norm[ran,]
iris_test <-iris_norm[-ran,]
iris_target_category <- iris[ran,5]
iris_test_category <- iris[-ran,5]
library(class)
pr <-
knn(iris_train,iris_test,cl=iris_target_category,k
=13)
tab <-table(pr,iris_test_category)
accuracy <-
function(x){sum(diag(x)/(sum(rowSums(x)))) *
100}
accuracy(tab)
```

```
>  accuracy(tab)
[1] 93.33333
>
```

# Practical No: 9

**Aim: -** Implementation and analysis of Apriori Algorithm using Market Basket Analysis.

**Theory: -**

**Apriori algorithm** is used for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

**R Commands:**

```
setwd("D:/13_Fymca")
getwd()
mba_data<-read.csv("data_apriori.csv")
trans<-
split(mba_data$Products,mba_data$Customer_I
d,"transactions")
head(trans)
```

```
> head(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"

$`5`
[1] "buns"    "chips"   "beer"    "mustard" "pickels" "s

$`6`
[1] "bread"     "butter"    "chocolate" "eggs"      "mil
```

```
install.packages("arules")
library(arules)
rules=apriori(trans,parameter =
list(support=0.5,confidence=0.9,maxlen=3,minl
en=2))
inspect(rules)
```

```
> 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 No: 10

**Aim: -** Implementation and analysis of clustering algorithms like K-Means , Agglomerative.

**Theory:**

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

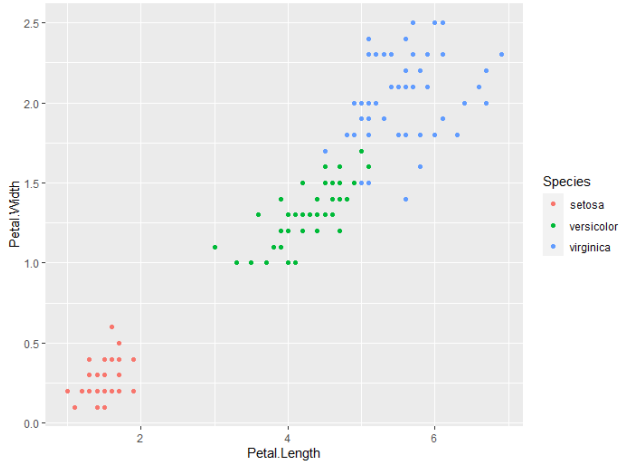
**R Commands:-**

head(iris)

install.packages("ggplot2",dependencies = TRUE)

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

```
> irisCluster
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

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
 [49] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 [97] 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
[145] 2 2 2 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
> |
```

irisCluster$cluster <- as.factor(irisCluster$cluster)

ggplot(iris, aes(Petal.Length, Petal.Width, color =irisCluster$cluster)) + geom_point()

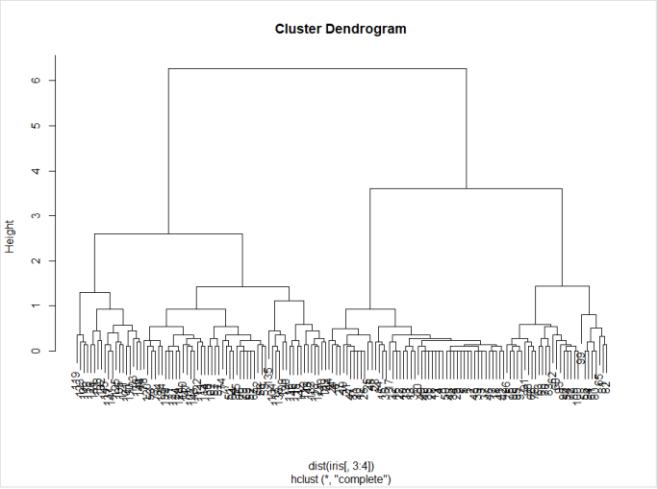

#Agglomerative Clustering

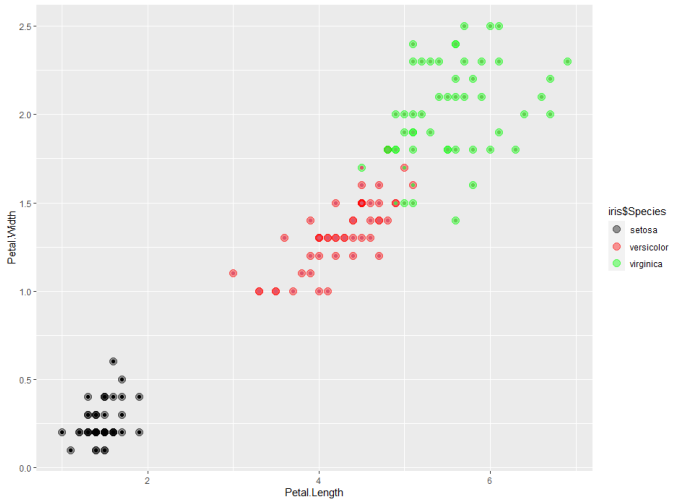
Agglomerative Clustering is Also known as bottom-up approach or hierarchical agglomerative clustering . A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters

head(iris)

clusters <- hclust(dist(iris[, 3:4]))

plot(clusters)\

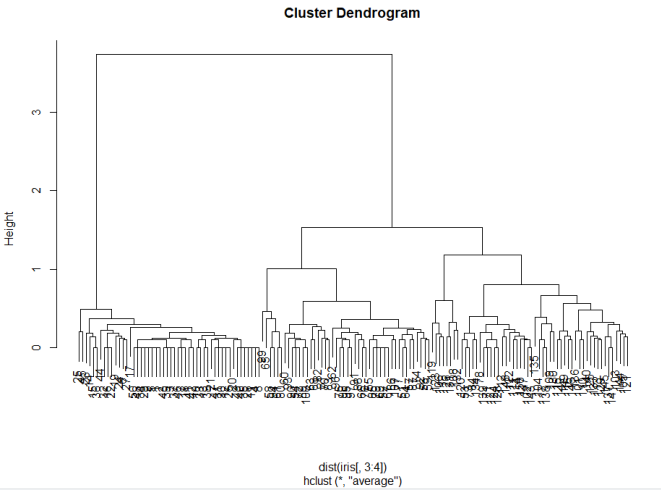



clusterCut<- cutree(clusters, 3)
table(clusterCut, iris$Species)

```
> table(clusterCut, iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         21        50
         3      0         29         0
>
```

clusters <- hclust(dist(iris[, 3:4]), method = 'average')
plot(clusters)




clusterCut <-cutree(clusters, 3)
table(clusterCut, iris$Species)
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$Species)) +geom_point(alpha = 0.4, size = 3.5) + geom_point(col = clusterCut) + scale_color_manual(values = c('black','red', 'green'))