NAME: SHETTY DIKSHA B

DECCAN EDUCATION SOCIETY'S
NAVINCHANDRA MEHTA
INSTITUTE OF TECHNOLOGY AND
DEVELOPMENT.

STANDARD : F.Y.M.C.A
DIV: B

ROLL NUMBER:C21112
SUBJECT: ADVANCED
DATABASE MANAGEMENT
SYSTEM

Deccan Education Society's

Navinchandra Mehta Institute of

Technology and Development

C E R T I F I C A T E

This is to certify that Miss. **SHETTY DIKSHA B** of M.C.A.

Semester I with Roll No.**C21112** has completed 9 practical's of ADVANCED DATABASE MANAGEMENT SYSTEM under Dr. Sulakshana Vispute supervision in this college

during the year 2021-2022.

CO	R1 (Attendance)	R2 (Performance during lab session)	R3 (Innovation in problem-solving technique)	R4 (Mock Viva)	R5 (Variation in implementation of learned topics on projects)
CO1					
CO2					
CO3					
CO4					

Practical-in-charge

Head of Department

MCA Department (NMITD)

Sr.No	Title	Date
1	Implementation of different types of Partitions : Range, List and composite partitions.	28-01-2022
2	Implementation of Analytical queries like Roll_UP, CUBE, First, Last , Lead ,Lag,Rank AND Dense Rank	09-02-2022
3	Implementation of ORDBMS concepts like ADT(Abstract Data Types), Reference	16-02-2022
4	Implementation of ETL transformation with Pentaho like Copy data from Source (Table/Excel/ Oracle) and store it to Target (Table/Excel/ Oracle) , Adding sequence,Adding Calculator Concatenation of two fields Splitting of two fields Number Range String Operations Sorting data Implement the merge join transformation on tables Implement data validations on the table data.	18-02-2022

5	**Introduction to R programming and Data acquisition** **Install packages , Loading packages** **Data types, checking type of variable, printing variable and objects** **(Vector, Matrix, List, Factor, Data frame, Table)** **cbind-ing and rbind-ing**	**23-02-2022**

	Reading and Writing data. setwd(), getwd(), data(), rm(), Attaching and Detaching data. Reading data from the console. **Loading data from different data sources.(CSV, Excel).**	
6	**Implementation of Data preprocessing techniques like, Naming and Renaming variables, adding a new variable. Dealing with missing data. Dealing with categorical data. Data reduction using subsetting**	**02-03-2022**
7	**Implementation and analysis of Classification algorithms like Naive Bayesian, K-Nearest** **Neighbor, ID3 , C4.5**	**11-03-2022**
8	**Implementation and analysis of Apriori Algorithm using Market Basket** **Analysis.**	**16-03-2022**
9	**Implementation and analysis of clustering algorithms like** **K-Means , Agglomerative**	**19-03-2022**

1. Implementation of different types of Partitions : Range, List and composite partitions.

- RANGE PARTITION

  - **Creating Table**

```
SQL> CREATE TABLE sale_range
  2  ( salesman_id NUMBER(10),
  3  salesman_name VARCHAR2(50),
  4  sales_amount NUMBER(10),
  5  sales_date DATE)
  6  PARTITION BY RANGE(sales_date)
  7  (
  8  PARTITION sales_jan2021 VALUES LESS
THAN(TO_DATE('01/02/2021','DD/MM/YYYY')),
  9  PARTITION sales_feb2021 VALUES LESS
THAN(TO_DATE('01/03/2021','DD/MM/YYYY')),
10  PARTITION sales_mar2021 VALUES LESS
THAN(TO_DATE('01/04/2021','DD/MM/YYYY')),
11  PARTITION sales_apr2021 VALUES LESS
THAN(TO_DATE('01/05/2021','DD/MM/YYYY'))
12  )
13  ;

Table created.
```

- **To Check if the partitions are Built Correctly**

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

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
SALE_RANGE                     SALES_JAN2021
SALE_RANGE                     SALES_FEB2021
SALE_RANGE                     SALES_MAR2021
SALE_RANGE                     SALES_APR2021
```

- **To Insert Data in the Table**

```
SQL> insert into sale_range values(02,'Yash
Dikshaput',20000,TO_DATE('16/02/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into sale_range values(03,'Laxmi
Rai',30000,TO_DATE('13/01/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into sale_range values(04,'Ashwath
Kapoor',25000,TO_DATE('21/04/2021','DD/MM/YYYY'));
```

1 row created.

SQL> insert into sale_range values(05,'Swara Mestry',30000,TO_DATE('02/04/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into sale_range values(06,'Ayushi Hegde',20000,TO_DATE('29/01/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into sale_range values(07,'Keertana Shetty',15000,TO_DATE('11/02/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into sale_range values(08,'Rachan Kaur',35000,TO_DATE('10/03/2021','DD/MM/YYYY'));

1 row created.

- **To Check That Data is Inserted Correctly**

SQL> SELECT * FROM sale_range;

SALESMAN_ID SALESMAN_NAME SALES_AMOUNT

---------- ---------- ----------

SALES_DAT

---

3 Laxmi Rai 30000

13-JAN-21

6 Ayushi Hegde 20000

29-JAN-21

2 Yash Dikshaput 20000

16-FEB-21

SALESMAN_ID SALESMAN_NAME SALES_AMOUNT

---------- ---------- ----------

SALES_DAT

---

7 Keertana Shetty 15000

11-FEB-21

1 Diksha Shetty 25000

06-MAR-21

```
        8 Rachan Kaur                    35000
10-MAR-21



SALESMAN_ID SALESMAN_NAME                    SALES_AMOUNT
----------- ---------------------------------------------- ------------
SALES_DAT
---------
        4 Ashwath Kapoor                  25000
21-APR-21


        5 Swara Mestry                    30000
02-APR-21
8 rows selected.
```

- **Display Data Using PARTITION**

```
SQL> SELECT * FROM sale_range PARTITION(sales_mar2021);

SALESMAN_ID SALESMAN_NAME                    SALES_AMOUNT
----------- ---------------------------------------------- ------------
SALES_DAT
---------
        1 Diksha Shetty                   25000
06-MAR-21


        8 Rachan Kaur                     35000
```

10-MAR-21

➢ LIST PARTITION

- **To Create Table:**

```
SQL> CREATE TABLE invoices
  2  ( invoice_id NUMBER(10),
  3  invoice_name VARCHAR2(50),
  4  invoice_amount NUMBER(10),
  5  invoice_location VARCHAR2(50),
  6  invoice_date DATE)
  7  PARTITION BY LIST(invoice_location)
  8  (
  9  PARTITION invoice_a VALUES('Bhayander','Andheri'),
 10  PARTITION invoice_b VALUES('Worli','Parel'),
 11  PARTITION invoice_c VALUES('Daiser','TilakNagar'),
 12  PARTITION invoice_d VALUES(DEFAULT)
 13  )
 14  enable row movement
 15  ;

Table created.
```

- **To Check if the partitions are Built Correctly**

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


TABLE_NAME                     PARTITION_NAME

------------------------------ ------------------------------

INVOICES                       INVOICE_C

INVOICES                       INVOICE_D

INVOICES                       INVOICE_A

INVOICES                       INVOICE_B
```

- **To Insert Data**

```
SQL> insert into invoices
values(1,'Diksha',15000,'Bhayander',TO_DATE('06/03/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into invoices
values(2,'Ross',25000,'Parel',TO_DATE('26/03/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into invoices
values(3,'Rachel',35000,'Worli',TO_DATE('26/04/2021','DD/MM/YYYY'));

1 row created.

SQL> insert into invoices
values(4,'Pheobe',30000,'Daiser',TO_DATE('02/02/2021','DD/MM/YYYY'));

1 row created.

SQL>

SQL> insert into invoices
values(5,'Chandler',30000,'TilakNagar',TO_DATE('12/02/2021','DD/MM/YYYY'));

1 row created.
```

```
SQL> insert into invoices
values(6,'Monica',30000,'Andheri',TO_DATE('09/02/2021','DD/MM/YYYY'));


1 row created.


SQL> insert into invoices
values(7,'Joey',32000,'Virar',TO_DATE('20/06/2021','DD/MM/YYYY'));


1 row created.


SQL> insert into invoices
values(8,'Ashwath',25000,'Bhayander',TO_DATE('06/06/2021','DD/MM/YYYY'));


1 row created.


SQL>  select * from invoices;


INVOICE_ID INVOICE_NAME                         INVOICE_AMOUNT
---------- -------------------------------------------------- --------------
INVOICE_LOCATION                   INVOICE_D
-------------------------------------------------- ---------
         1 Diksha                                  15000
Bhayander                          06-MAR-21


         6 Monica                                  30000
Andheri                            09-FEB-21
```

8 Ashwath 25000

Bhayander 06-JUN-21

INVOICE_ID INVOICE_NAME INVOICE_AMOUNT

---------- ---------- ----------

INVOICE_LOCATION INVOICE_D

---------- ---------

2 Ross 25000

Parel 26-MAR-21

3 Rachel 35000

Worli 26-APR-21

4 Pheobe 30000

Daiser 02-FEB-21

INVOICE_ID INVOICE_NAME INVOICE_AMOUNT

---------- ---------- ----------

INVOICE_LOCATION INVOICE_D

---------- ---------

5 Chandler 30000

TilakNagar 12-FEB-21

```
         7 Joey                                32000
Virar                          20-JUN-21


8 rows selected.
```

- Display data using Partition

```
SQL> SELECT * FROM invoices PARTITION(invoice_b);

INVOICE_ID INVOICE_NAME                          INVOICE_AMOUNT
---------- ---------------------------------------------- --------------
INVOICE_LOCATION                    INVOICE_D
------------------------------------------------ ---------
         2 Ross                                25000
Parel                          26-MAR-21

         3 Rachel                              35000
 Worli                          26-APR-21
```

- **Display Data from the PARTITION having Default value**

```
SQL> SELECT * FROM invoices PARTITION(invoice_d);
```

INVOICE_ID INVOICE_NAME INVOICE_AMOUNT

---------- ---------- --------------

INVOICE_LOCATION INVOICE_D

---------- ---------

7 Joey 32000

Virar 20-JUN-21

- **HASH PARTITION**

  - **To Create Table**

```
SQL> CREATE TABLE salesh
  2  ( sales_id NUMBER(10),
  3   sales_name VARCHAR2(50),
  4   sales_amount NUMBER(10),
  5  week_no NUMBER(10))
  6  PARTITION BY HASH(sales_id)
  7  PARTITIONS 4
  8  ;
```

Table created.

  - **To Check if Partition is Built Properly**

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

TABLE_NAME	PARTITION_NAME
SALESH	SYS_P21
SALESH	SYS_P22
SALESH	SYS_P23
SALESH	SYS_P24

  - **To Insert Data in the Table**

```
SQL> insert into salesh values(01,'Luffy',40000,12);
```

1 row created.

```
SQL> insert into salesh values(02,'Nami',30000,8);

1 row created.

SQL> insert into salesh values(03,'Zoro',30000,5);

1 row created.

SQL> insert into salesh values(04,'Ussop',20000,11);

1 row created.

SQL> insert into salesh values(05,'Sanji',23000,10);

1 row created.

SQL> insert into salesh values(06,'Chopper',35000,9);

1 row created.

SQL> insert into salesh values(07,'Robin',35000,6);

1 row created.

SQL> insert into salesh values(08,'Frankie',35000,7);

1 row created.
```

- **To Display Inserted Data**

```
SQL>  select * from salesh
  2  ;

 SALES_ID SALES_NAME                                SALES_AMOUNT
---------- -------------------------------------------------- ------------
  WEEK_NO
----------
```

6 Chopper 35000

9

2 Nami 30000

8

5 Sanji 23000

10

SALES_ID SALES_NAME SALES_AMOUNT

---------- -- ------------

WEEK_NO

----

8 Frankie 35000

7

1 Luffy 40000

12

3 Zoro 30000

5

SALES_ID SALES_NAME SALES_AMOUNT

---------- ---------- ---------- ----------

WEEK_NO

---

4 Ussop 20000

11

7 Robin 35000

6

8 rows selected.

- **To Display Data Based on Partition**

SQL> select * from salesh partition(sys_p21);

SALES_ID SALES_NAME SALES_AMOUNT

---------- ---------- ---------- ----------

WEEK_NO

---

6 Chopper 35000

9

SQL> select * from salesh partition(sys_p23);

```
 SALES_ID SALES_NAME                                        SALES_AMOUNT

---------- -------------------------------------------------- ------------

 WEEK_NO

----------

         2 Nami                                               30000

         8


         5 Sanji                                              23000

        10


         8 Frankie                                            35000

         7
```

2. Implementation of Analytical queries

- **To Create Table**

```
SQL> Create Table ssb1

_2  (emp_no integer,

_3  dep_no integer,
```

```
_4  bdate date,
_5  salary integer,
_6  comm integer,
_7  job varchar2(20));
```

Table created.

- **Inserting data**

```
SQL> insert into ssb1
  2  values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job');
Enter value for emp_no: 101
Enter value for dep_no: 01
Enter value for bdate: 06-jun-00
Enter value for salary: 50000
Enter value for comm: 3000
Enter value for job: Analyst
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(101,01,'06-jun-00',50000,3000,'Analyst')

1 row created.

SQL> r
  1  insert into ssb1
  2* values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
```

```
Enter value for emp_no: 102
Enter value for dep_no: 02
Enter value for bdate: 04-mar-99
Enter value for salary: 30000
Enter value for comm: 1500
Enter value for job: Manager
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(102,02,'04-mar-99',30000,1500,'Manager')

1 row created.

SQL> r
  1  insert into ssb1
  2* values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
Enter value for emp_no: 103
Enter value for dep_no: 03
Enter value for bdate: 25-jan-98
Enter value for salary: 25000
Enter value for comm: 1000
Enter value for job: Clerk
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(103,03,'25-jan-98',25000,1000,'Clerk')

1 row created.
SQL> r
```

```
  1  insert into ssb1
  2* values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
Enter value for emp_no: 104
Enter value for dep_no: 02
Enter value for bdate: 30-Sep-95
Enter value for salary: 35000
Enter value for comm: 1500
Enter value for job: Assistant_Manger
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(104,02,'30-Sep-95',35000,1500,'Assistant_Manger')

1 row created.

SQL> r
  1  insert into ssb1
  2* values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
Enter value for emp_no: 105
Enter value for dep_no: 01
Enter value for bdate: 18-Dec-94
Enter value for salary: 45000
Enter value for comm: 2500
Enter value for job: Programmer
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(105,01,'18-Dec-94',45000,2500,'Programmer')
```

1 row created.

SQL> r

```
  1  insert into ssb1
  2* values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
Enter value for emp_no: 106
Enter value for dep_no: 03
Enter value for bdate: 09-May-01
Enter value for salary: 25000
Enter value for comm: 1000
Enter value for job: Peon
old   2: values(&emp_no,&dep_no,'&bdate',&salary,&comm,'&job')
new   2: values(106,03,'09-May-01',25000,1000,'Peon')
```

1 row created.

- **Display the Table**

SQL> select * from ssb1;

EMP_NO	DEP_NO	BDATE	SALARY	COMM	JOB
101	1	06-JUN-00	50000	3000	Analyst
102	2	04-MAR-99	30000	1500	Manager
103	3	25-JAN-98	25000	1000	Clerk
104	2	30-SEP-95	35000	1500	Assistant_Manger
105	1	18-DEC-94	45000	2500	Programmer

106 3 09-MAY-01 25000 1000 Peon

6 rows selected.

❖ ROLLUP

```
SQL> SELECT dep_no,job,count(*),sum(salary) from ssb1
  2   group by rollup(dep_no,job);

   DEP_NO JOB                  COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
         1 Analyst                     1       50000
         1 Programmer                  1       45000
         1                             2       95000
         2 Manager                     1       30000
         2 Assistant_Manger            1       35000
         2                             2       65000
         3 Peon                        1       25000
         3 Clerk                       1       25000
         3                             2       50000
                                       6      210000

10 rows selected.

SQL> select dep_no,job,sum(salary) from ssb1
  2  where dep_no in(01,02)
  3  group by dep_no, rollup(job);
```

```
   DEP_NO JOB                  SUM(SALARY)
---------- -------------------- -----------
         1 Analyst                    50000
         1 Programmer                 45000
         1                            95000
         2 Manager                    30000
         2 Assistant_Manger           35000
         2                            65000

6 rows selected.

SQL>  SELECT dep_no,job,count(*),sum(salary) from ssb1
  2  group by job,rollup(dep_no);

   DEP_NO JOB                  COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
         3 Peon                        1       25000
           Peon                        1       25000
         3 Clerk                       1       25000
           Clerk                       1       25000
         1 Analyst                     1       50000
           Analyst                     1       50000
         2 Manager                     1       30000
           Manager                     1       30000
         1 Programmer                  1       45000
```

Programmer 1 45000

2 Assistant_Manger 1 35000

DEP_NO JOB COUNT(*) SUM(SALARY)

---------- ---------- ---------- ----------

Assistant_Manger 1 35000

12 rows selected.

- **CUBE**

SQL> SELECT dep_no,job,count(*),sum(salary) from ssb1

2 group by cube(dep_no,job);

DEP_NO JOB COUNT(*) SUM(SALARY)

---------- ---------- ---------- ----------

6 210000

Peon 1 25000

Clerk 1 25000

Analyst 1 50000

Manager 1 30000

Programmer 1 45000

Assistant_Manger 1 35000

1 2 95000

1 Analyst 1 50000

1 Programmer 1 45000

```
         2                    2       65000


   DEP_NO JOB                 COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
         2 Manager                     1       30000
         2 Assistant_Manger            1       35000
         3                    2       50000
         3 Peon                        1       25000
         3 Clerk                       1       25000


16 rows selected.
```

- **RANK**

SQL> select emp_no,dep_no,salary,comm, rank() over(partition by dep_no order by salary) as Rank from ssb1;

```
   EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       105          1      45000       2500          1
       101          1      50000       3000          2
       102          2      30000       1500          1
       104          2      35000       1500          2
       106          3      25000       1000          1
       103          3      25000       1000          1
```

6 rows selected.

- **LEAD**

```
SQL> select emp_no,bdate, lead(bdate,1) over(order by bdate) as "next birthdate"from ssb1;

   EMP_NO BDATE     next birt
---------- --------- ---------
       105 18-DEC-94 30-SEP-95
       104 30-SEP-95 25-JAN-98
       103 25-JAN-98 04-MAR-99
       102 04-MAR-99 06-JUN-00
       101 06-JUN-00 09-MAY-01
       106 09-MAY-01

6 rows selected.

SQL> select emp_no,bdate,
  2  lead(bdate,1) over(order by bdate) as "next birthdate"
  3  from ssb1 where dep_no=03;

   EMP_NO BDATE     next birt
---------- --------- ---------
       103 25-JAN-98 09-MAY-01
       106 09-MAY-01
```

- **LAG**

```
SQL>  select emp_no,bdate,
  2   lag(bdate,1) over(order by bdate) as "Previous"
  3  from ssb1 ;

   EMP_NO BDATE     Previous
---------- --------- ---------
       105 18-DEC-94
       104 30-SEP-95 18-DEC-94
       103 25-JAN-98 30-SEP-95
       102 04-MAR-99 25-JAN-98
       101 06-JUN-00 04-MAR-99
       106 09-MAY-01 06-JUN-00

SQL>  select emp_no,bdate,
  2   lag(bdate,1) over(order by bdate) as "Previous"
  3  from ssb1 where dep_no=02;

   EMP_NO BDATE     Previous
---------- --------- ---------
       104 30-SEP-95
       102 04-MAR-99 30-SEP-95
```

- **MAX**

```
SQL> select dep_no,salary,
```

```
  2   max(salary)keep(DENSE_RANK FIRST ORDER BY comm)
  3  over(PARTITION BY dep_no)"max"
  4  from ssb1;

   DEP_NO     SALARY        max
---------- ---------- ----------
         1      45000      45000
         1      50000      45000
         2      30000      35000
         2      35000      35000
         3      25000      25000
         3      25000      25000

6 rows selected.


SQL> select dep_no,salary,
  2  max(salary)keep(DENSE_RANK FIRST ORDER BY salary)
  3  over(PARTITION BY dep_no)"max"
  4  from ssb1;

   DEP_NO     SALARY        max
---------- ---------- ----------
         1      45000      45000
         1      50000      45000
```

2	30000	30000
2	35000	30000
3	25000	25000
3	25000	25000

6 rows selected.

- **DESCENDING**

```
select dep_no,salary,
  2  max(salary)keep(DENSE_RANK FIRST ORDER BY salary desc)
  3   over(PARTITION BY dep_no)"max"
  4  from ssb1;
```

DEP_NO	SALARY	max
1	45000	50000
1	50000	50000
2	30000	35000
2	35000	35000
3	25000	25000
3	25000	25000

6 rows selected.

- **MIN**

```
SQL> select dep_no,salary,
  2  min(salary)keep(DENSE_RANK FIRST ORDER BY salary)
  3  over(PARTITION BY dep_no)"min"
  4   from ssb1;

   DEP_NO     SALARY        min
---------- ---------- ----------
         1      45000      45000
         1      50000      45000
         2      30000      30000
         2      35000      30000
         3      25000      25000
         3      25000      25000

6 rows selected.
```

COMBINATION

```
SQL>  select dep_no,salary,
  2  max(salary)keep(DENSE_RANK FIRST ORDER BY salary desc)
  3  over(PARTITION BY dep_no)"max",
  4   max(salary)keep(DENSE_RANK LAST ORDER BY salary desc)
  5  over(PARTITION BY dep_no)"min"
  6  from ssb1;

   DEP_NO     SALARY        max        min
```

---------- ---------- ---------- ----------

1	45000	50000	45000
1	50000	50000	45000
2	30000	35000	30000
2	35000	35000	30000
3	25000	25000	25000
3	25000	25000	25000

6 rows selected.

LAST ORDER

```
SQL> select dep_no,salary,
  2  min(salary)keep(DENSE_RANK LAST ORDER BY salary desc)
  3  over(PARTITION BY dep_no)"min"
  4  from ssb1;
```

DEP_NO	SALARY	min
1	45000	45000
1	50000	45000
2	30000	30000
2	35000	30000
3	25000	25000

3 25000 25000

6 rows selected.

3. Implementation of ORDBMS concepts

```
SQL> Create type type_add as object
  2  (street varchar(20),
  3  city varchar(20),
  4  zip_code number(10));
  5  /

Type created.

SQL> Create type type_name as object
  2  (fname varchar(20),
  3  mname varchar(20),
  4  lname varchar(20));
  5  /

Type created.

SQL> Create table customer1
  2  (c_id number(5) primary key,
  3  c_name type_name,
  4  c_add type_add,
```

```
  5  c_phno number(10));


Table created.


SQL> insert into customer1

  2  values(01,type_name('Vyomesh','Ramkrishna','Shetty'), type_add('Salasar
nagar','Worli',104),7456321456);


1 row created.


SQL> insert into customer1

  2  values(02,type_name('Bala','Ramakant','Shinde'), type_add('Jaisel
park','Wadala',194),9874562132);


1 row created.


SQL> insert into customer1

  2  values(03,type_name('Shruti','Suresh','Suvarna'), type_add('Navghar
Road','Daiser',321),9966554477);


1 row created.


SQL> select * from customer1;


      C_ID
----------
```

C_NAME(FNAME, MNAME, LNAME)

C_ADD(STREET, CITY, ZIP_CODE)

C_PHNO

1

TYPE_NAME('Vyomesh', 'Ramkrishna', 'Shetty')

TYPE_ADD('Salasar nagar', 'Worli', 104)

7456321456

C_ID

C_NAME(FNAME, MNAME, LNAME)

C_ADD(STREET, CITY, ZIP_CODE)

C_PHNO

2

TYPE_NAME('Bala', 'Ramakant', 'Shinde')

TYPE_ADD('Jaisel park', 'Wadala', 194)

9874562132

C_ID

C_NAME(FNAME, MNAME, LNAME)

C_ADD(STREET, CITY, ZIP_CODE)

C_PHNO

3

TYPE_NAME('Shruti', 'Suresh', 'Suvarna')

TYPE_ADD('Navghar Road', 'Daiser', 321)

9966554477

SQL> desc customer1;

Name	Null?	Type
C_ID	NOT NULL	NUMBER(5)
C_NAME		TYPE_NAME
C_ADD		TYPE_ADD
C_PHNO		NUMBER(10)

SQL> set describe depth 2;

SQL> desc customer1;

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
C_ID                                      NOT NULL NUMBER(5)
C_NAME                                             TYPE_NAME
  FNAME                                            VARCHAR2(20)
  MNAME                                            VARCHAR2(20)
  LNAME                                            VARCHAR2(20)
C_ADD                                              TYPE_ADD
  STREET                                           VARCHAR2(20)
  CITY                                             VARCHAR2(20)
  ZIP_CODE                                         NUMBER(10)
C_PHNO                                             NUMBER(10)
```

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

C_ADD.CITY
--------------------
Worli
```

```
SQL> select c.c_name.fname from customer1 c where c_id=3;

C_NAME.FNAME
--------------------
Shruti
```

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

C.C_NAME.FNAME||"||C.C_NAME.MNAME||"||C.C_NAME.LNAME
------------------------------------------------------------
VyomeshRamkrishnaShetty
BalaRamakantShinde
ShrutiSureshSuvarna

SQL> alter type type_add
  2  add attribute(name type_name)cascade;

Type altered.

SQL> create table customer2
  2   (cust_no Integer,
  3  add1 type_add);

Table created.

SQL>  insert into customer2
  2  values(101,type_add('Borivali','Mumbai',102,type_name('santosh',' Jayram ',' Mestry ')));

1 row created.
```

```
SQL>  insert into customer2
  2  values(102,type_add('Bhayander','Thane',122,type_name('Dikshaendra',' Vijay ',' Dalvi ')));

1 row created.

SQL>  desc customer2;
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
CUST_NO                                            NUMBER(38)
ADD1                                               TYPE_ADD

SQL>  select cust_no from customer2;

   CUST_NO
----------
       101
       102

SQL>  select cust_no,c.add1.street,c.add1.name.fname from customer2 c;

   CUST_NO ADD1.STREET         ADD1.NAME.FNAME
---------- ------------------- -------------------
       101 Borivali            santosh
       102 Bhayander           Dikshaendra
```

```
SQL>  create type stud_type as object
  2  (roll_no number(5),
  3   name varchar2(30))
  4  ;
  5  /

Type created.

SQL> create table students of stud_type;

Table created.

SQL> insert into students values(stud_type(01,'Akshay'));

1 row created.

SQL> insert into students values(stud_type(02,'Yash'));

1 row created.

SQL> insert into students values(stud_type(03,'Naveen'));

1 row created.

SQL>

SQL>  insert into students values(stud_type(04,'Kumuda'));
```

1 row created.

SQL> select * from students;

ROLL_NO NAME

---------- ----------

1 Akshay

2 Yash

3 Naveen

4 Kumuda

SQL> select roll_no from students;

ROLL_NO

1

2

3

4

SQL> select s.roll_no from students s;

ROLL_NO

1

2

3

4

REF AND DEREF

```
SQL> create or replace type Animal_ty as object
  2  (Breed varchar2(25),
  3  Name varchar2(25),
  4  BirthDate DATE);
  5  /
```

Type created.

```
SQL>  create table ANIMAL of ANIMAL_TY;
```

Table created.

```
SQL>  insert into ANIMAL VALUES(
  2  ANIMAL_TY('Cat','Kiwi','06-Jun-06'));
```

1 row created.

```
SQL>  insert into ANIMAL VALUES(
  2  ANIMAL_TY('Dog','Candy','16-Mar-03'));
```

1 row created.

```
SQL>  insert into ANIMAL VALUES(
  2  ANIMAL_TY('Turtle','Turty','24-April-10'));

1 row created.

SQL> select REF(A) from ANIMAL A;

REF(A)
--------------------------------------------------------------------------------
00002802090F9C9DBCE1264ECF853A2B87F90DC0C298C5BB27BD074D0686D15737EDE
54426010002

5E0000

00002802091E86148CD640449EA3AEDBA64650353998C5BB27BD074D0686D15737EDE5
4426010002

5E0001

00002802096E319110A8AD478280DFD987D58B872298C5BB27BD074D0686D15737EDE54
426010002

5E0002

SQL> create table KEEPER
  2  (KeeperName varchar2(25),
  3  AnimalKept REF ANIMAL_TY);
```

Table created.

```
SQL> describe KEEPER
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
KEEPERNAME                                         VARCHAR2(25)
ANIMALKEPT                                         REF OF ANIMAL_TY
  BREED                                           VARCHAR2(25)
  NAME                                            VARCHAR2(25)
  BIRTHDATE                                       DATE
```

```
SQL>  insert into KEEPER
  2   select 'Rachael',
  3   REF(A)
  4   from ANIMAL A
  5   where Name='Kiwi';
```

1 row created.

```
SQL>  insert into KEEPER
  2   select 'Shreya',
  3   REF(A)
  4   from ANIMAL A
  5   where Name='Turty';
```

1 row created.

SQL> select * from KEEPER;

KEEPERNAME

ANIMALKEPT

Rachael

00002202080F9C9DBCE1264ECF853A2B87F90DC0C298C5BB27BD074D0686D15737EDE54426

Shreya

00002202086E319110A8AD478280DFD987D58B872298C5BB27BD074D0686D15737EDE54426

SQL> set describe depth 2;

SQL> describe KEEPER

Name Null? Type

KEEPERNAME VARCHAR2(25)

ANIMALKEPT REF OF ANIMAL_TY

BREED VARCHAR2(25)

NAME VARCHAR2(25)

BIRTHDATE DATE

SQL> select KeeperName,DEREF(K.AnimalKept)

2 from KEEPER K;

KEEPERNAME

DEREF(K.ANIMALKEPT)(BREED, NAME, BIRTHDATE)

Rachael

ANIMAL_TY('Cat', 'Kiwi', '06-JUN-06')

Shreya

ANIMAL_TY('Turtle', 'Turty', '24-APR-10')

INHERITANCE

SQL> create type PERSON_TY as object

2 (ssn number,

3 name varchar2(25),

4 gender char(1),

5 static function show_super(person_obj in person_ty)return varchar2,

6 member function show return varchar2)

7 NOT FINAL;

8 /

Type created.

```
SQL> create type TEACHER_TY UNDER PERSON_TY
  2  (id_of_joining DATE,
  3  salary number(7,2),
  4  courses_taught varchar2(50),
  5  OVERRIDING member function show return varchar2);
  6  /

Type created.

QL> create type STUDENT_TY UNDER PERSON_TY
  2  (grade char(1),
  3  yr_of_comp varchar2(20),
  4  courses_taken varchar2(50),
  5  OVERRIDING member function show return varchar2);
  6  /

Type created.

SQL> create type body PERSON_TY as
  2  --static function that can be called by subtypes
  3  static function show_super(person_obj in person_ty) return varchar2 is
  4  begin
  5  return 'SSN:'|| TO_CHAR(person_obj.ssn)||', Name:'||person_obj.Name||',
Gender:'||person_obj.gender;
  6  end;
  7  --funtion that can be overriden by subtypes
  8  member function show return varchar2 is
  9  begin
 10  return person_ty.show_super(SELF);
 11  end;
```

```
12  end;
13  /
```

Type body created.

```
SQL> create type body STUDENT_TY as
  2  OVERRIDING member function show return varchar2 is
  3  begin
  4  return person_ty.show_super(SELF)||'--
Grade:'||grade||',Yr_Of_Comp:'||yr_of_comp||',Courses Taken:'|| courses_taken;
  5  end;
  6  end;
  7  /
```

Type body created.

```
SQL> create type body TEACHER_TY as
  2  OVERRIDING member function show return varchar2 is
  3  begin
  4  return person_ty.show_super(SELF)||'--
Date_of_Joining:'||id_of_joining||',Salary:'||TO_CHAR(salary)||',Courses Taught:'||
courses_taught;
  5  end;
  6  end;
  7  /
```

Type body created.

CREATE AN OBJECT TABLE

```
SQL> create table person_obj_table of PERSON_TY;
```

Table created.

INSERT DATA INTO OBJECT TABLE

```
SQL> insert into person_obj_table values(PERSON_TY(1,'ABC','M'));
```

1 row created.

SQL> insert into person_obj_table values(TEACHER_TY(2,'XYZ','F','03 MAR 2011',25000,'CG,DS'));

1 row created.

SELECT DATA FROM OBJECT TABLE

SQL> select p.show() from person_obj_table p;

P.SHOW()

SSN:1, Name:ABC, Gender:M

SSN:2, Name:XYZ, Gender:F--Date_of_Joining:03-MAR-11,Salary:25000,Courses Taught :CG,DS

4. ETL-Implementation through Pentaho

4.ETL-Implementation through Pentaho

- **Create Table**

```
SQL> create table Empl01
  2  (
  3   emp_no numeric(5),
  4   fname varchar2(10),
  5  lname varchar2(10),
  6   salaray numeric(5),
  7  comm numeric(5)
  8  );
```

Table created.

SQL> insert into empl01 values(101,'Diksha','Shetty',50000,1500);

1 row created.

SQL> insert into empl01 values(102,'Harshal','Shetty',40000,1500);

1 row created.

SQL> insert into empl01 values(103,'Laxmi','Shetty',45000,1500);

1 row created.

SQL> insert into empl01 values(104,'Gopal','Shetty',35000,1500);

1 row created.

SQL> insert into empl01 values(105,'Nidhi','Shetty',30000,1500);

1 row created.

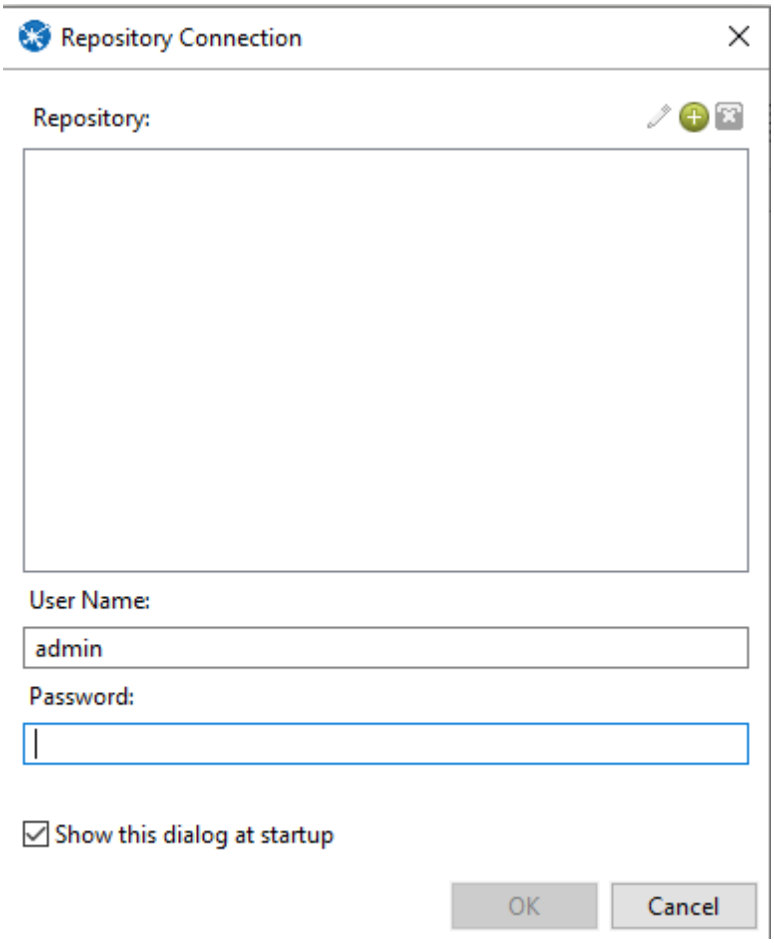
SQL> insert into empl01 values(106,'Skandha','Shetty',20000,1000);

1 row created.

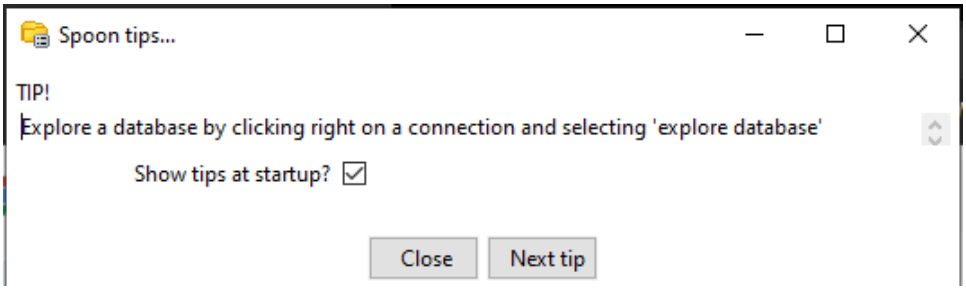
Start with Pentaho

1. **In Data Integration folder->Double click on spoon(Windows Batch file)**

2. **Click on cancel**

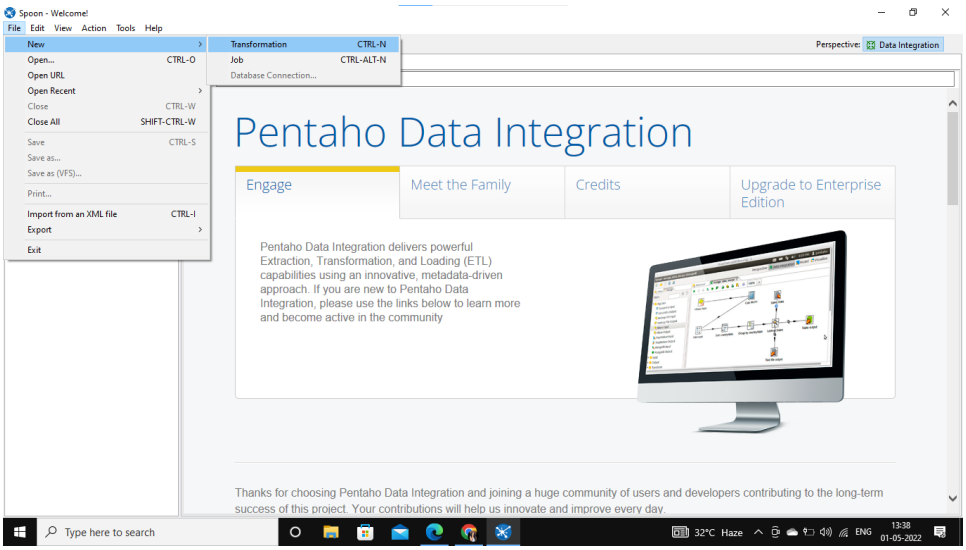


- **Click on Close**

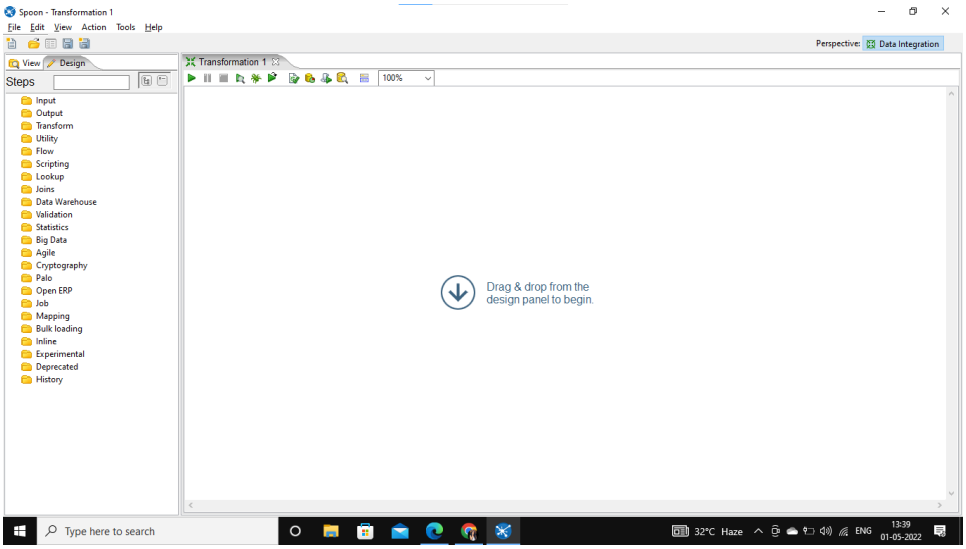


1. Transforming Source Table and storing to Output Table in SQL

- **Step 1:File>> New>> Transformation**

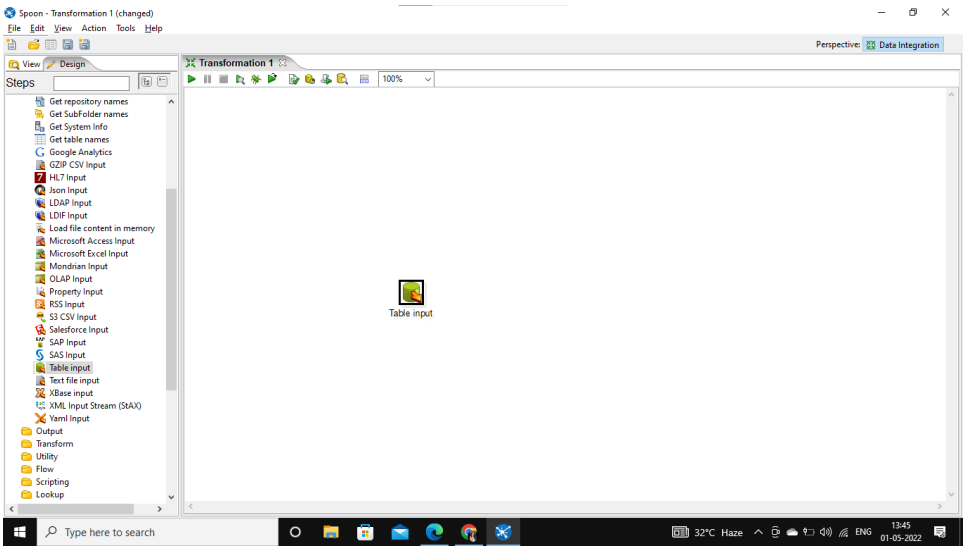


- **Transformation 1**

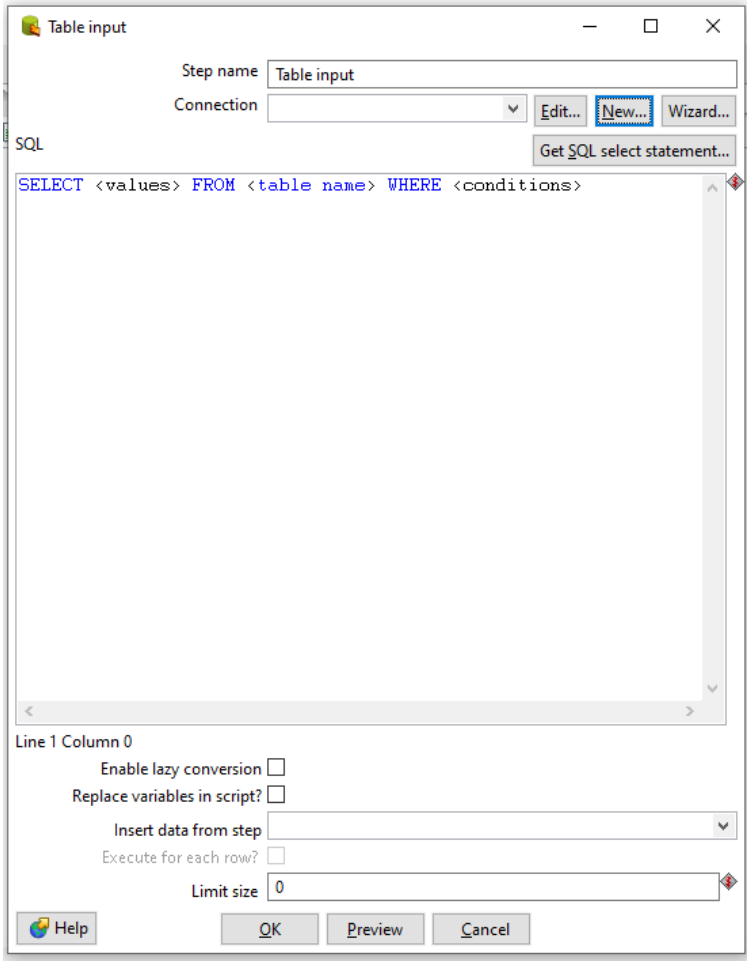


STEP 2

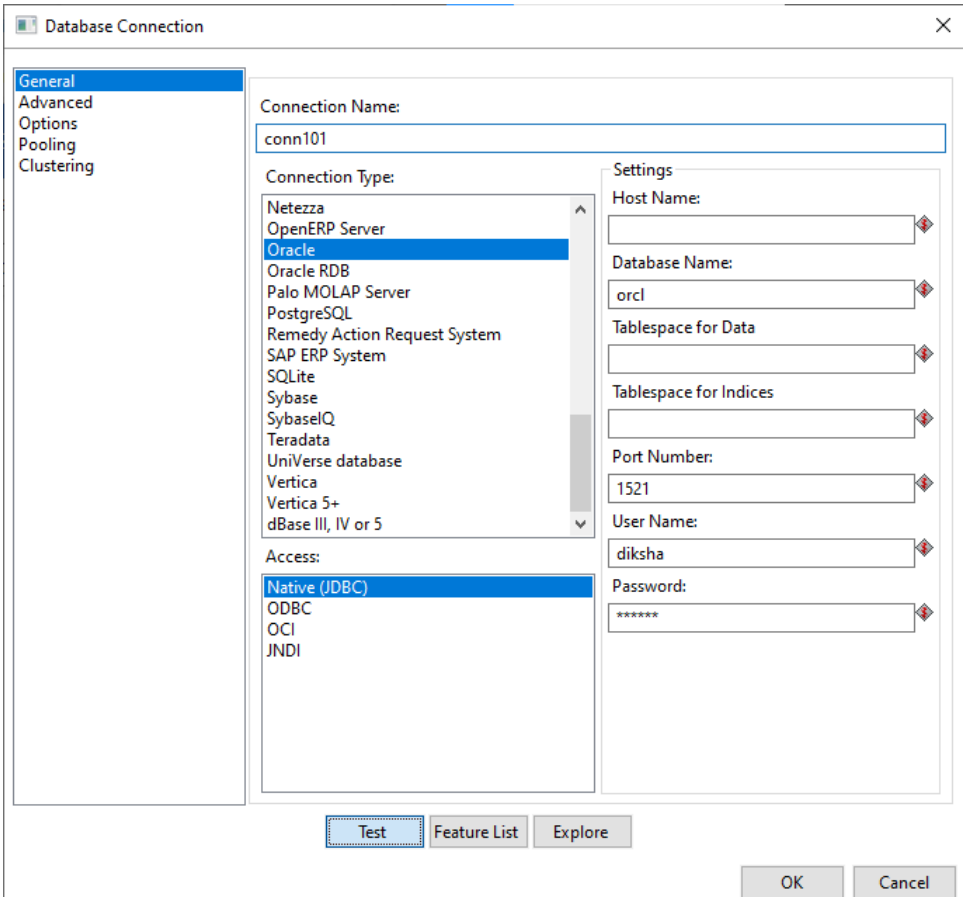
- **Click on Input**
- **Drag Table Input on the panel**



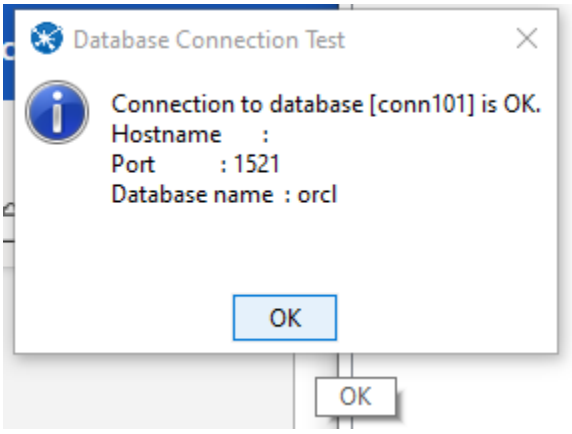
- **Double Click on Table Input and the following tab will appear.**
- **Click On New**



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

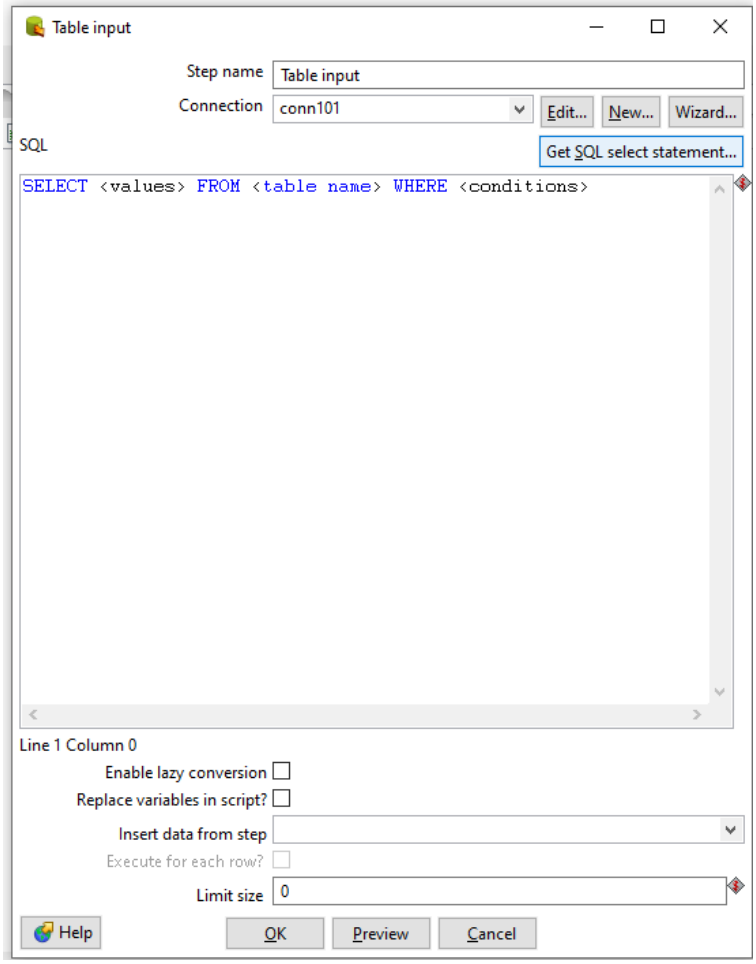


Connection Successful

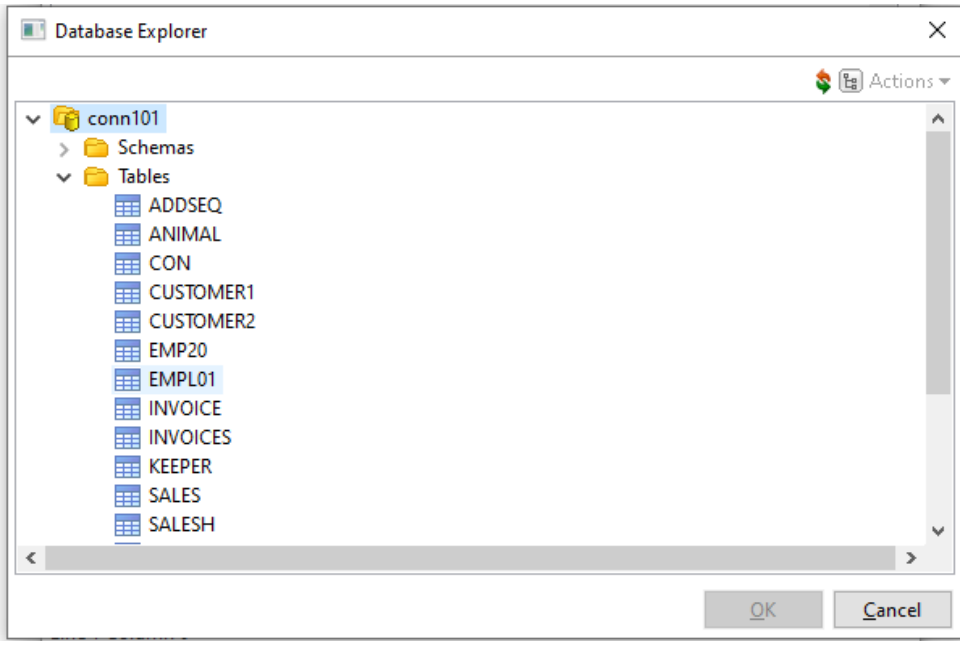


STEP 3:

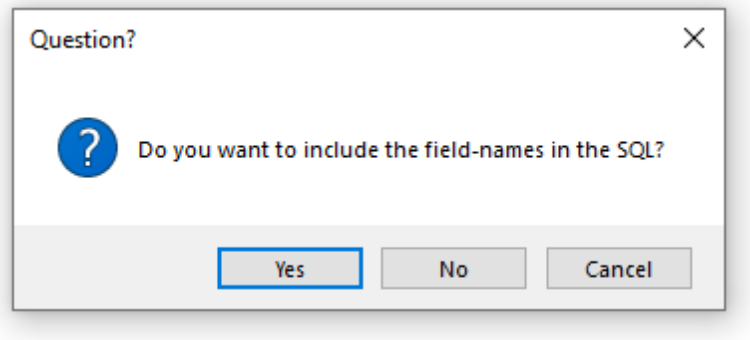
Click on get SQL select statement



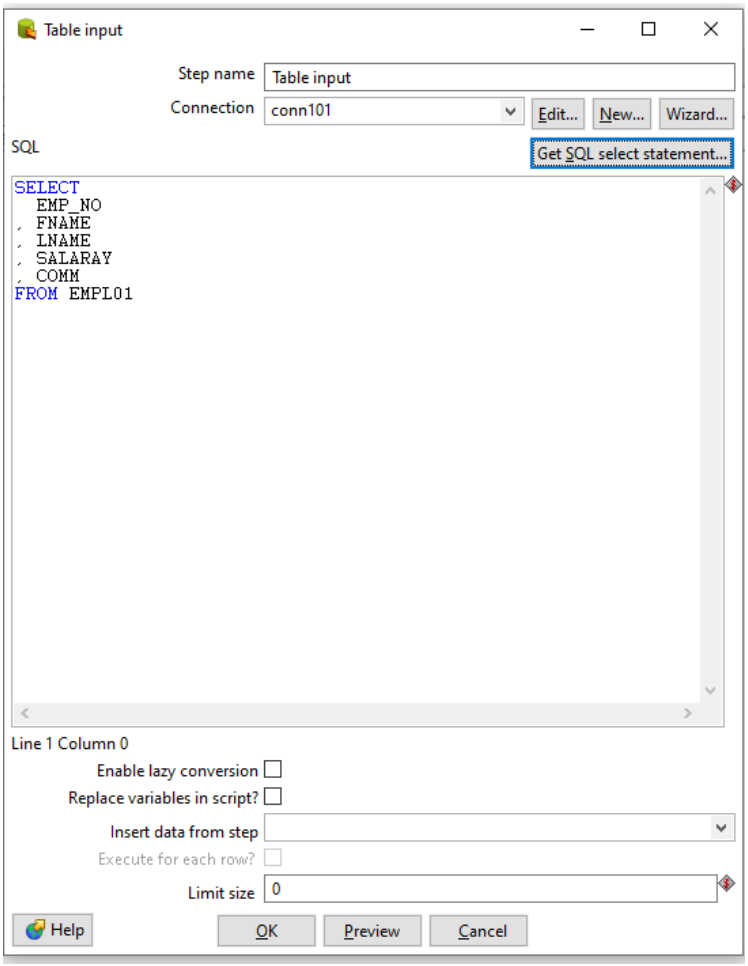
Click on database -> table and select the table name.



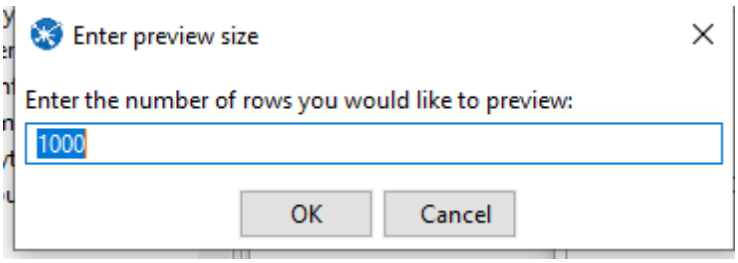
Click Yes



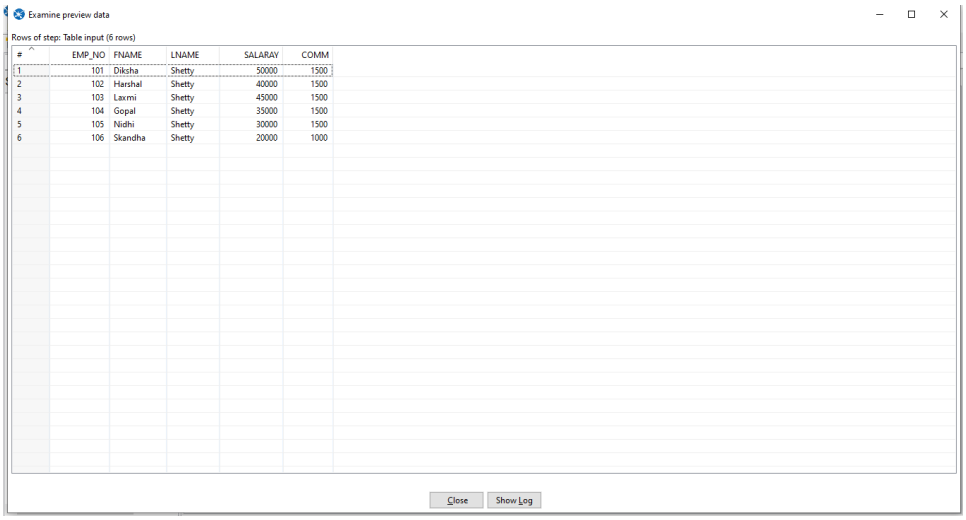
Click on Preview



Click Ok

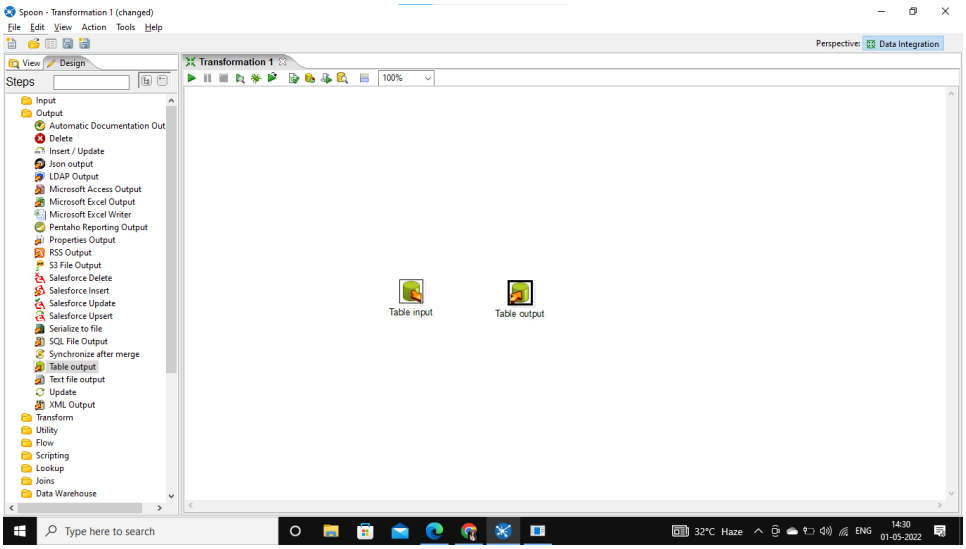


Preview Data



STEP 4:

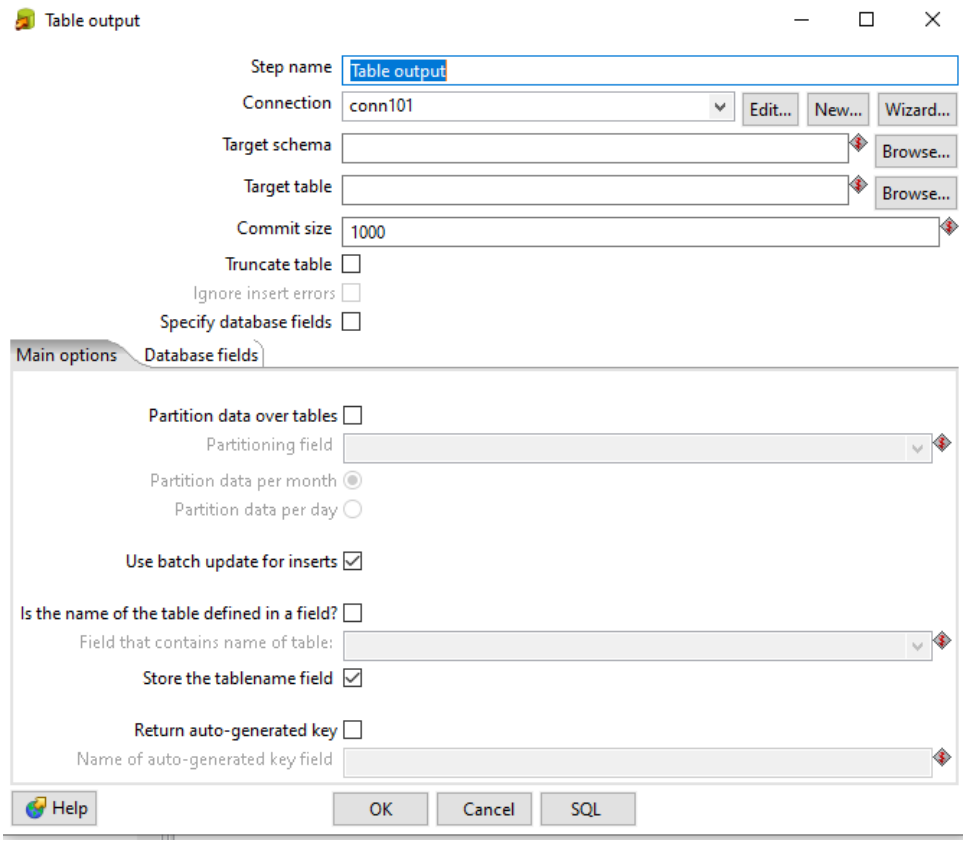
Click on Output and drag table Output.



STEP 5:

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

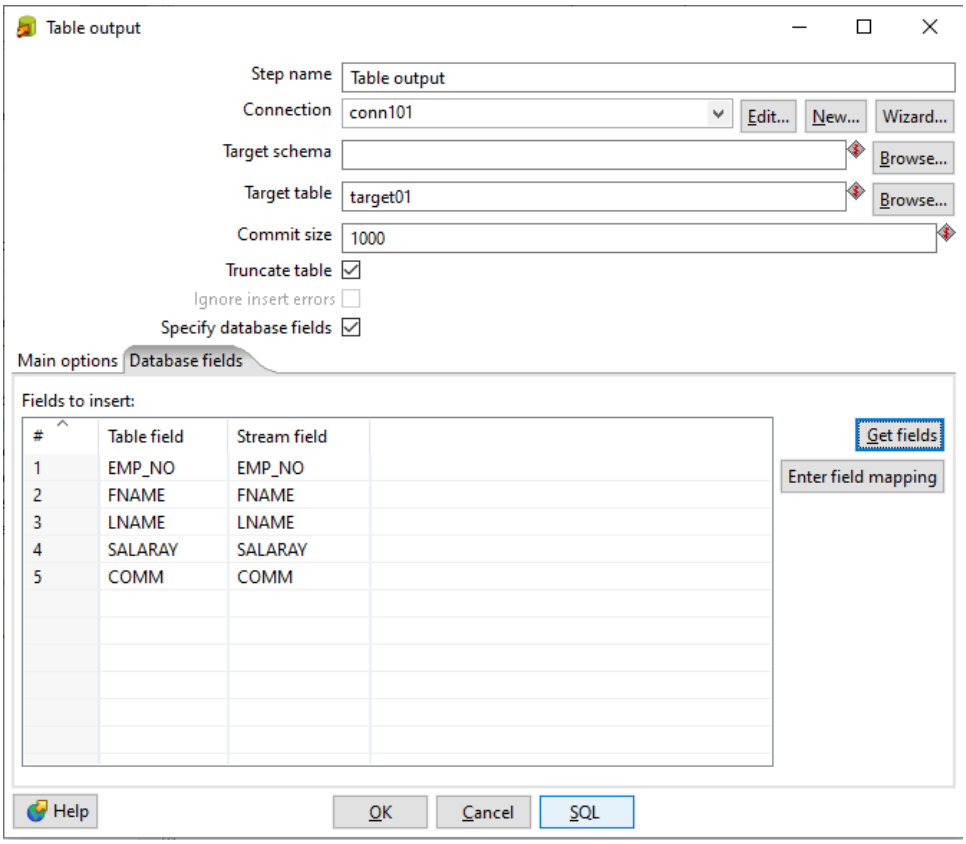
Double Click on target table



STEP 6:

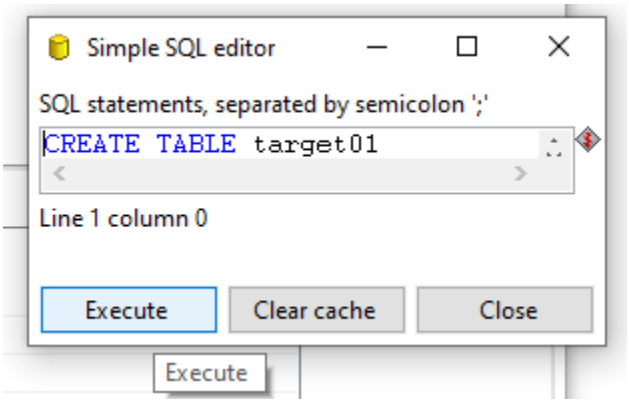
Enter the Target table name and check the truncate table and Specify database fields Check Box

Click On Database fields and click on Get fields.

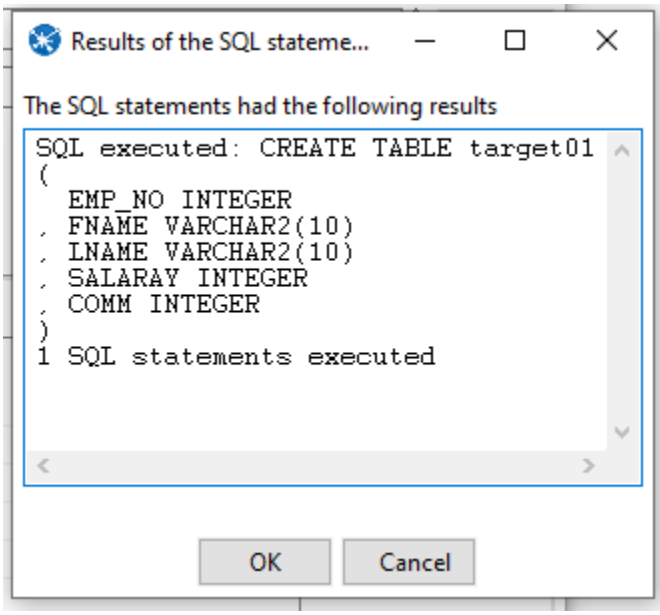


STEP 7:

Click on SQL and click on Execute.

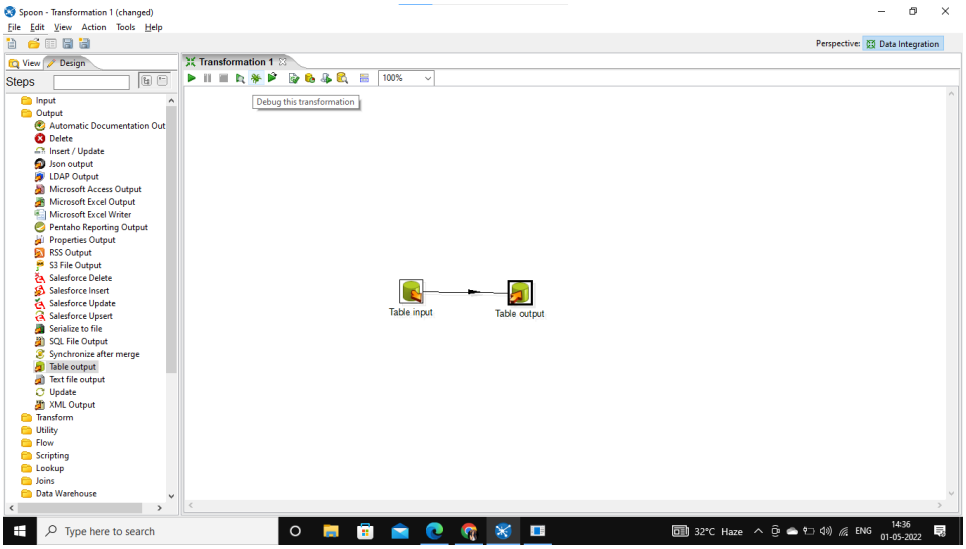


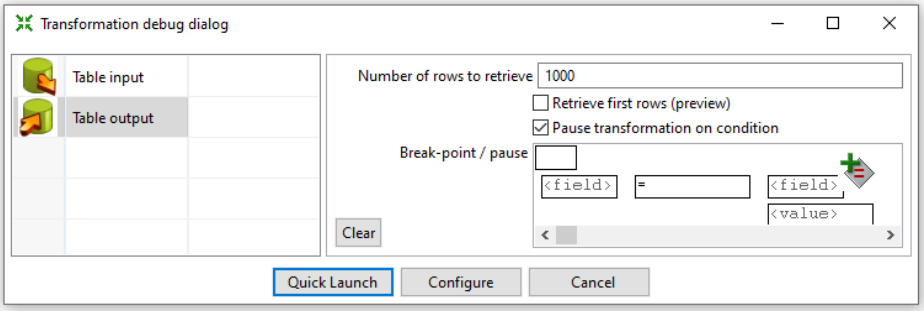
Click Ok

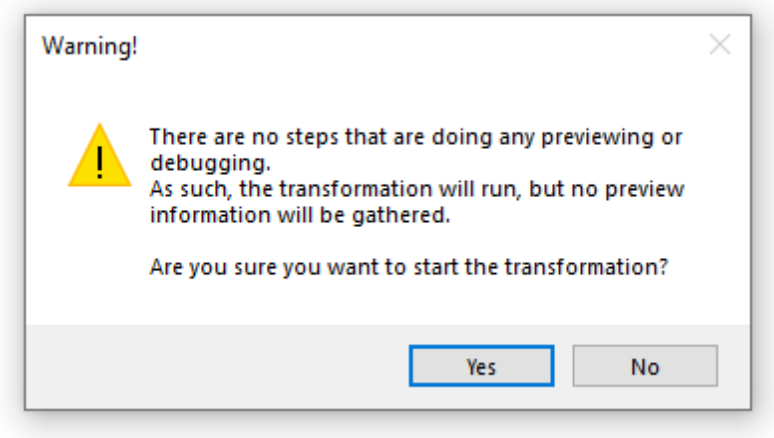


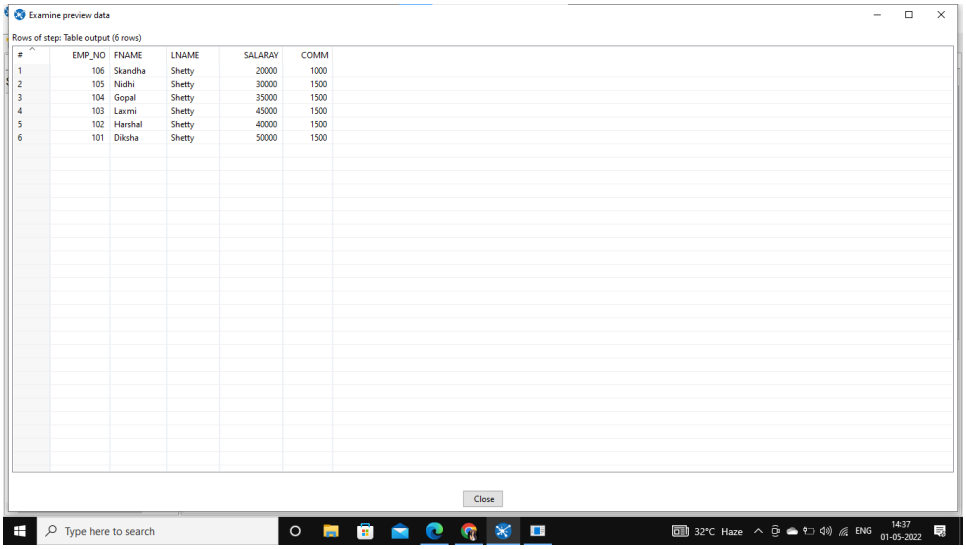
STEP 8:

Click on Debug Transformation(Spider) and Click on Quick Launch



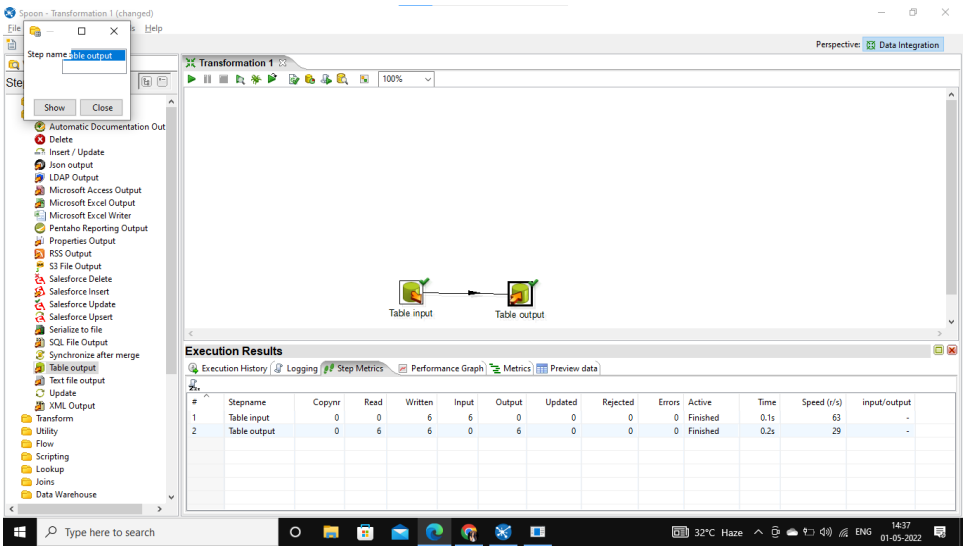






Output:

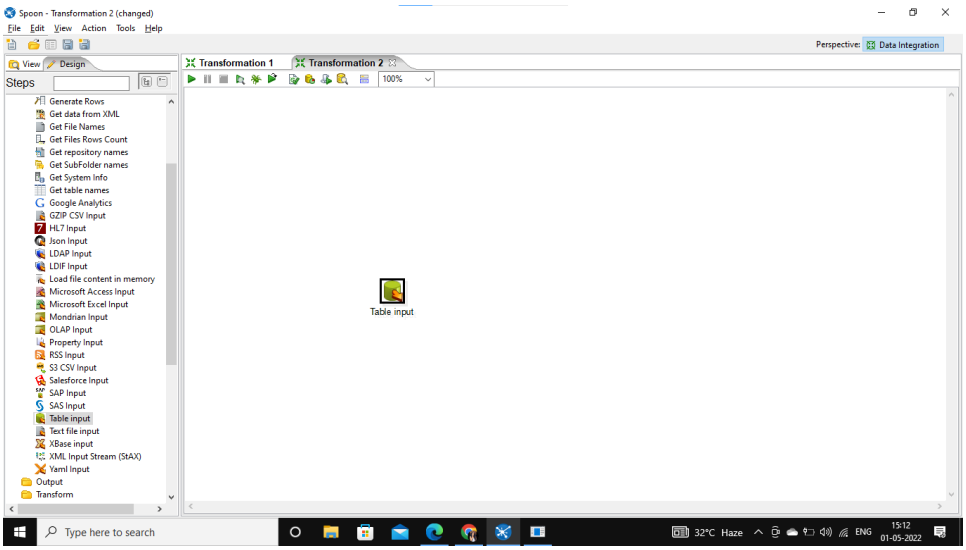
The Green ticks on the table input and table output shows successful transformation.

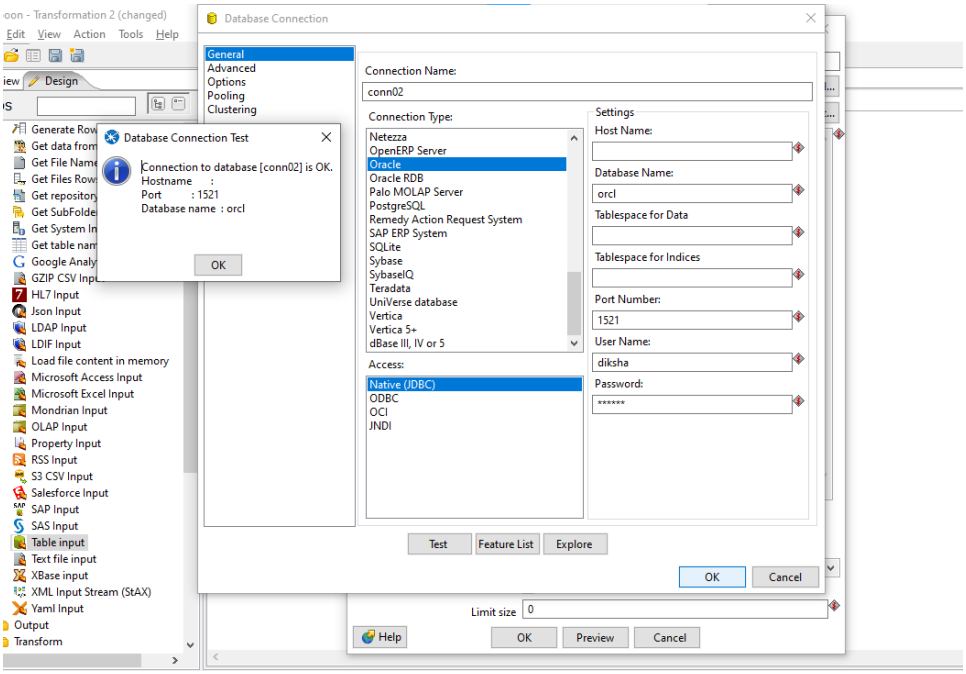


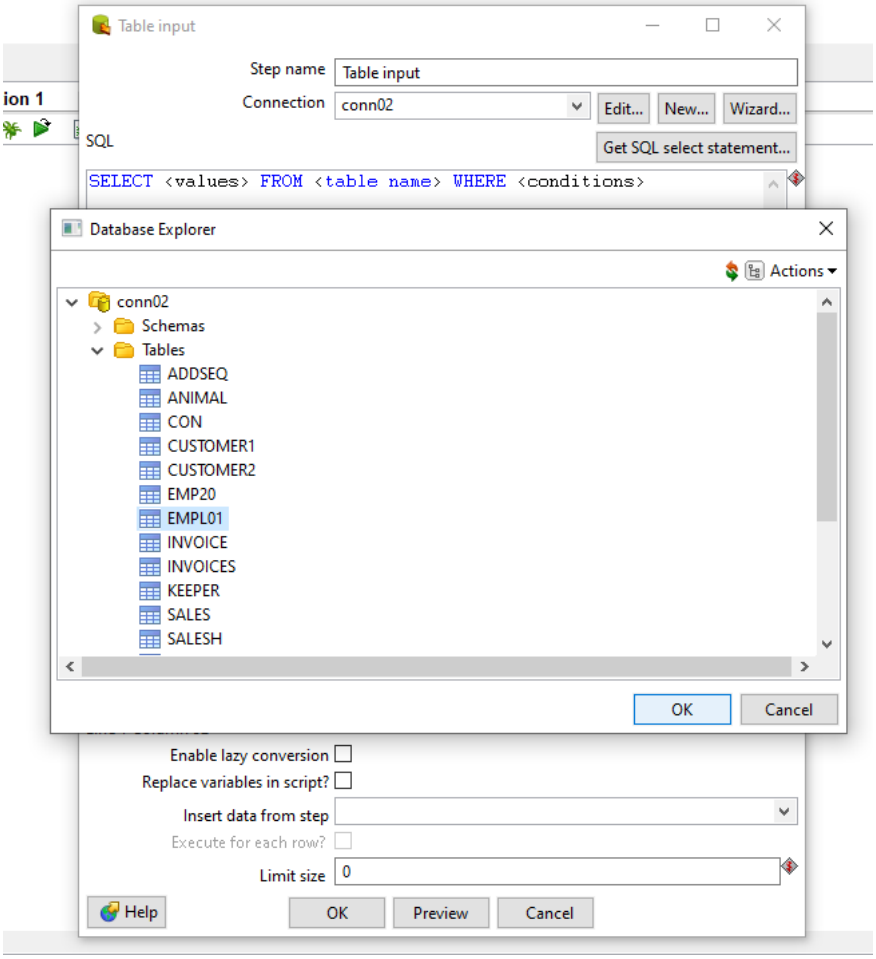

```
SQL> select * from target01;

    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
       101 Diksha     Shetty          50000       1500
       102 Harshal    Shetty          40000       1500
       103 Laxmi      Shetty          45000       1500
       104 Gopal      Shetty          35000       1500
       105 Nidhi      Shetty          30000       1500
       106 Skandha    Shetty          20000       1000

6 rows selected.
```

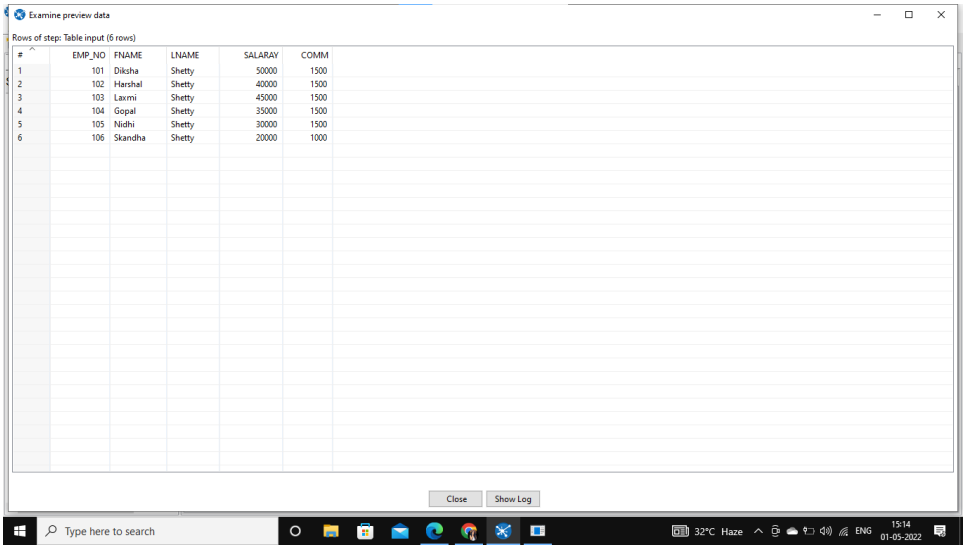
- **Transformation 2: Sequence and Sort Transformation**

Step 1: Perform the first 4 steps same as Trnasformation 1.


Spoon - Transformation 2 (changed)
File Edit View Action Tools Help
Perspective: Data Integration
Transformation 1
Transformation 2
Steps
Table input
Type here to search

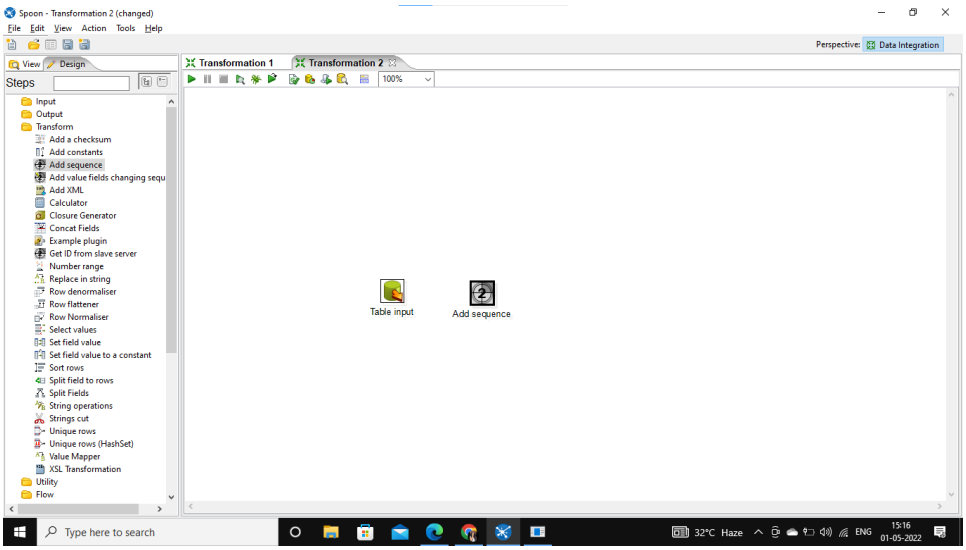

Database Connection
General
Advanced
Options
Pooling
Clustering
Connection Name:
conn02
Connection Type:
Oracle
Settings
Host Name:
Database Name:
orcl
Tablespace for Data
Tablespace for Indices
Port Number:
1521
User Name:
diksha
Password:
Access:
Native (JDBC)
Test
Feature List
Explore
OK
Cancel
Database Connection Test
Connection to database [conn02] is OK.
Hostname :
Port : 1521
Database name : orcl
OK
Limit size
0
Help
Preview

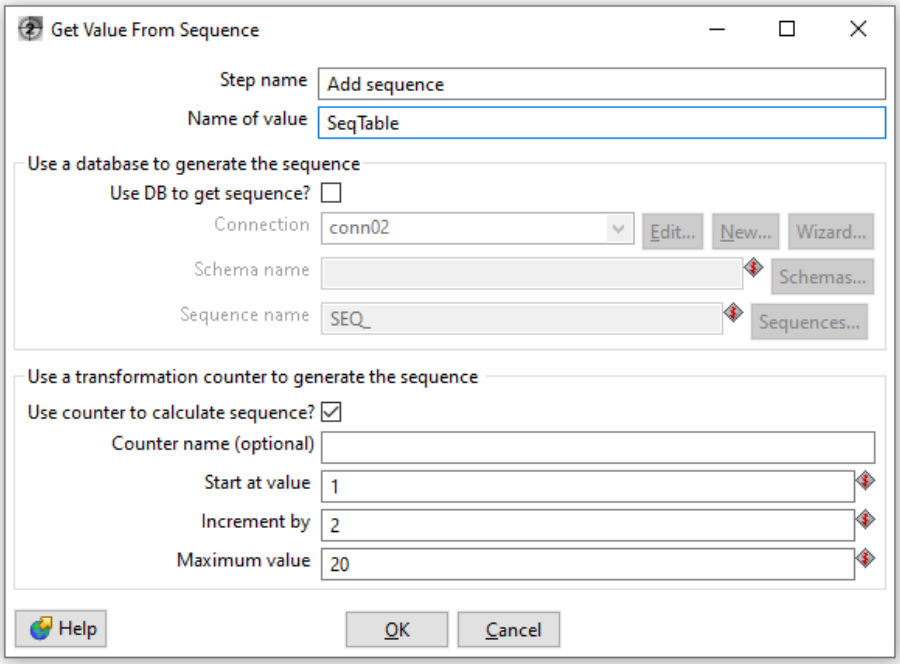


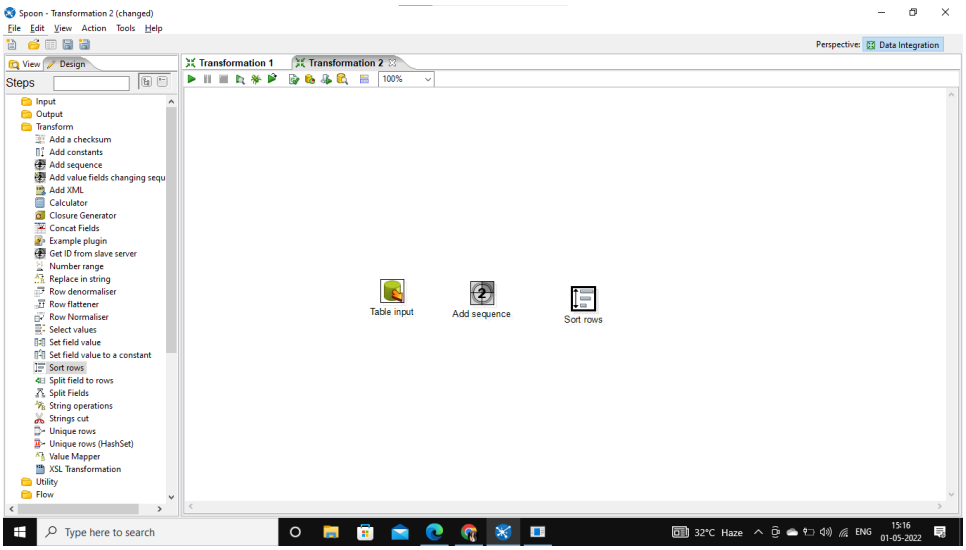


Step 2: Click on Transform in the left

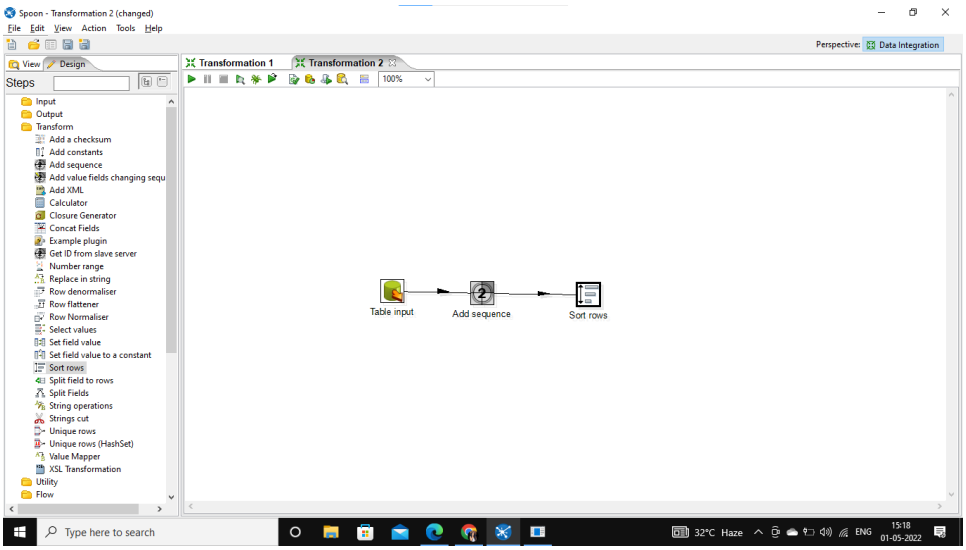
Step 3: Drag Add Seq and sort rows on the Panel


Spoon - Transformation 2 (changed)
File Edit View Action Tools Help
Perspective: Data Integration
View Design
Steps
Transformation 1
Transformation 2
100%
Input
Output
Transform
Add a checksum
Add constants
Add sequence
Add value fields changing sequ
Add XML
Calculator
Closure Generator
Concat Fields
Example plugin
Get ID from slave server
Number range
Replace in string
Row denormaliser
Row flattener
Row Normaliser
Select values
Set field value
Set field value to a constant
Sort rows
Split field to rows
Split Fields
String operations
Strings cut
Unique rows
Unique rows (HashSet)
Value Mapper
XSL Transformation
Utility
Flow
Table input
Add sequence
Type here to search
32°C Haze
ENG
15:16
01-05-2022


Get Value From Sequence
Step name
Add sequence
Name of value
SeqTable
Use a database to generate the sequence
Use DB to get sequence?
Connection
conn02
Edit...
New...
Wizard...
Schema name
Schemas...
Sequence name
SEQ_
Sequences...
Use a transformation counter to generate the sequence
Use counter to calculate sequence?
Counter name (optional)
Start at value
1
Increment by
2
Maximum value
20
Help
OK
Cancel

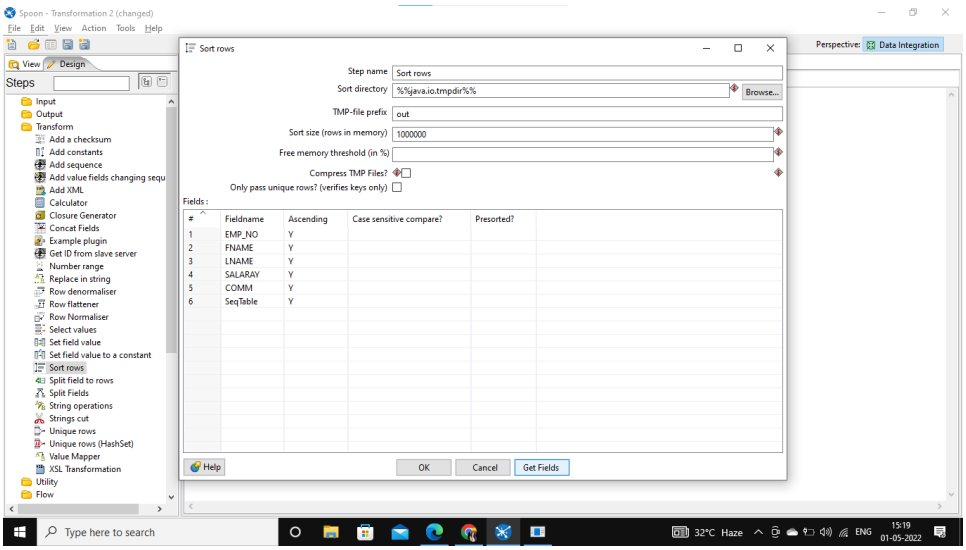


Step 4: Hold the mouse pointer on the Table Input and then drag the output connector to the sort rows and add seq.

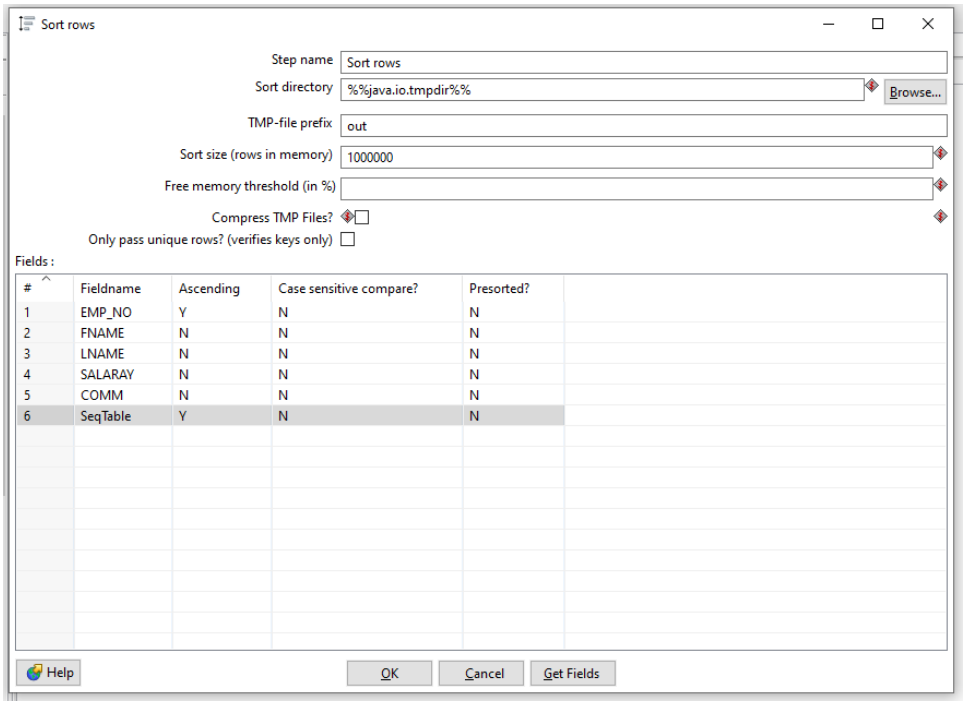


Step 5: Double click on the sort rows

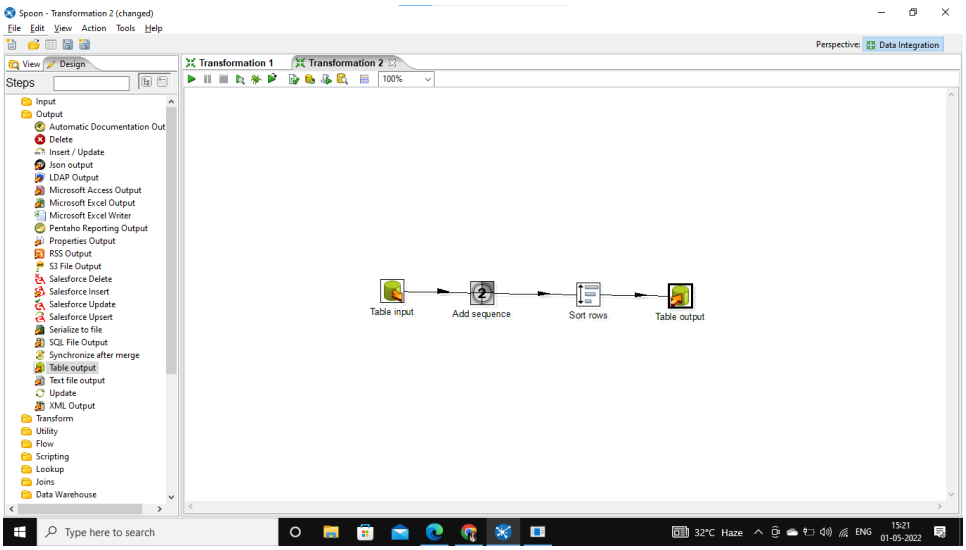
a. click on get fields

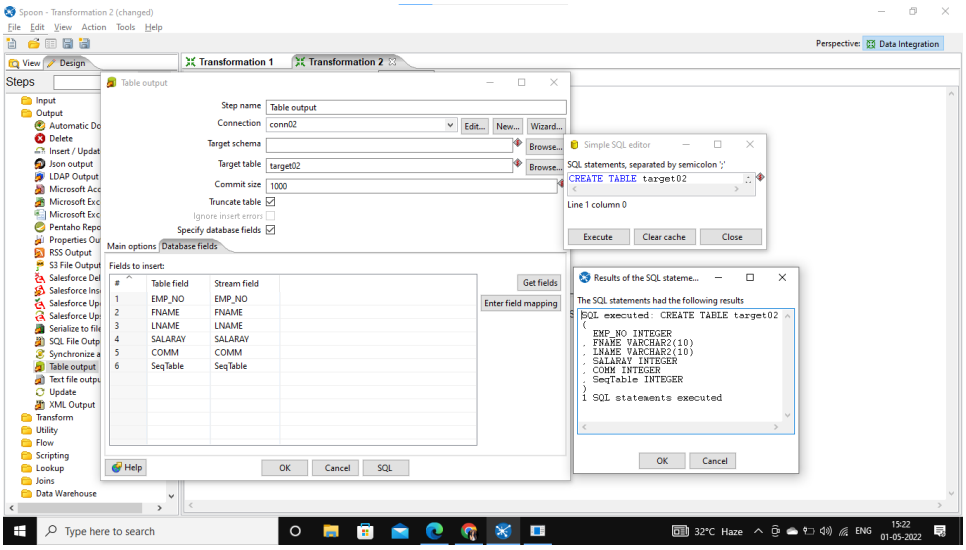


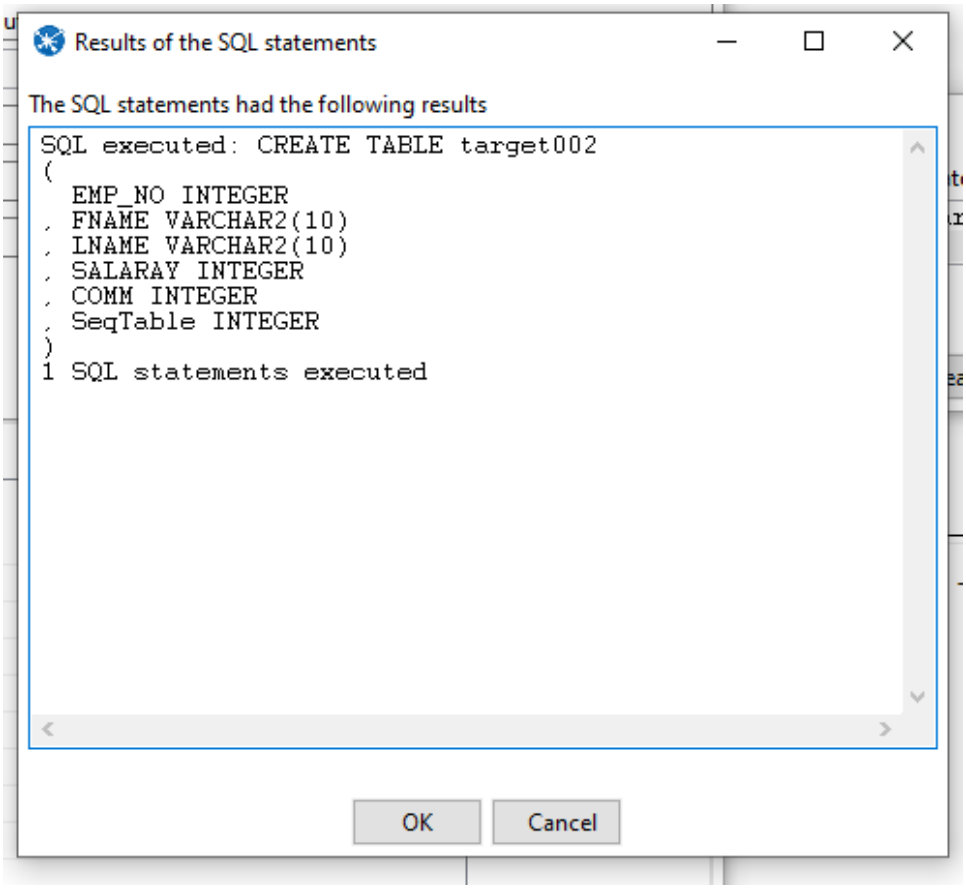
b. Edit the fields and press ok



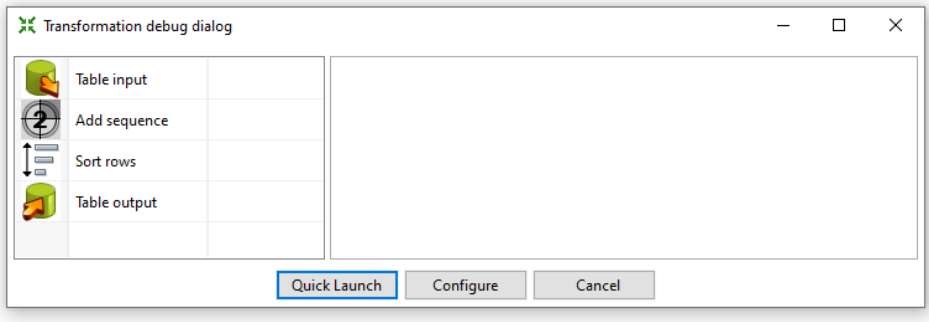
For table Output Follow same steps as transformation 1


Spoon - Transformation 2 (changed)
Transformation 1
Transformation 2
Table input
Add sequence
Sort rows
Table output

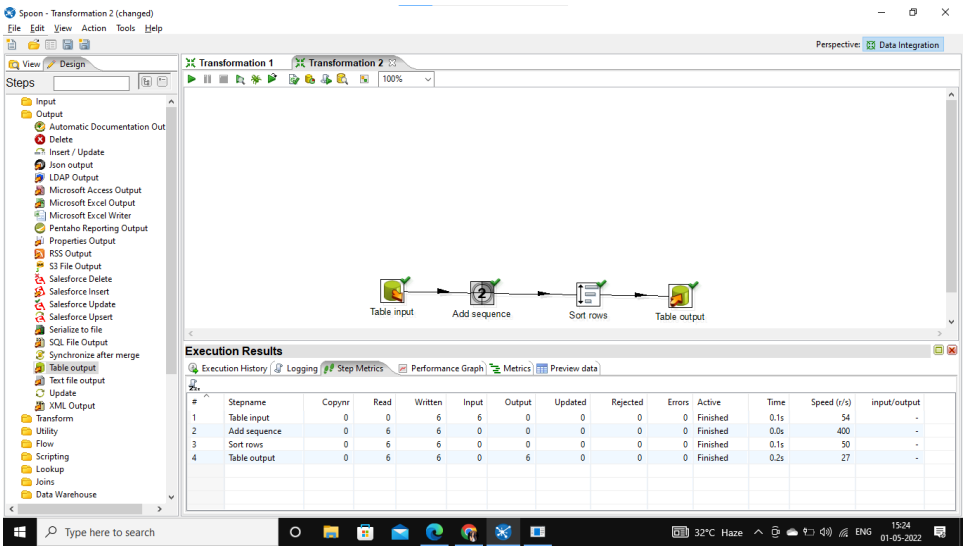

Spoon - Transformation 2 (changed)
Table output
Simple SQL editor
CREATE TABLE target02
Results of the SQL stateme...
1 SQL statements executed



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



Successful connections(Green Ticks)

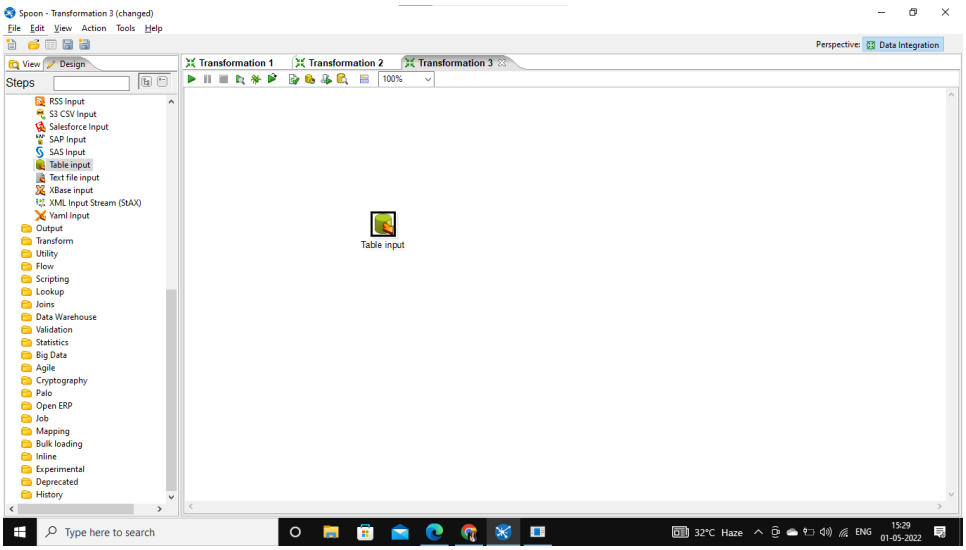



```
SQL> select * from target002;

    EMP_NO FNAME      LNAME          SALARAY       COMM    SEQTABLE
---------- ---------- ---------- ---------- ---------- ----------
       101 Diksha     Shetty          50000       1500          1
       102 Harshal    Shetty          40000       1500          3
       103 Laxmi      Shetty          45000       1500          5
       104 Gopal      Shetty          35000       1500          7
       105 Nidhi      Shetty          30000       1500          9
       106 Skandha    Shetty          20000       1000         11

6 rows selected.
```

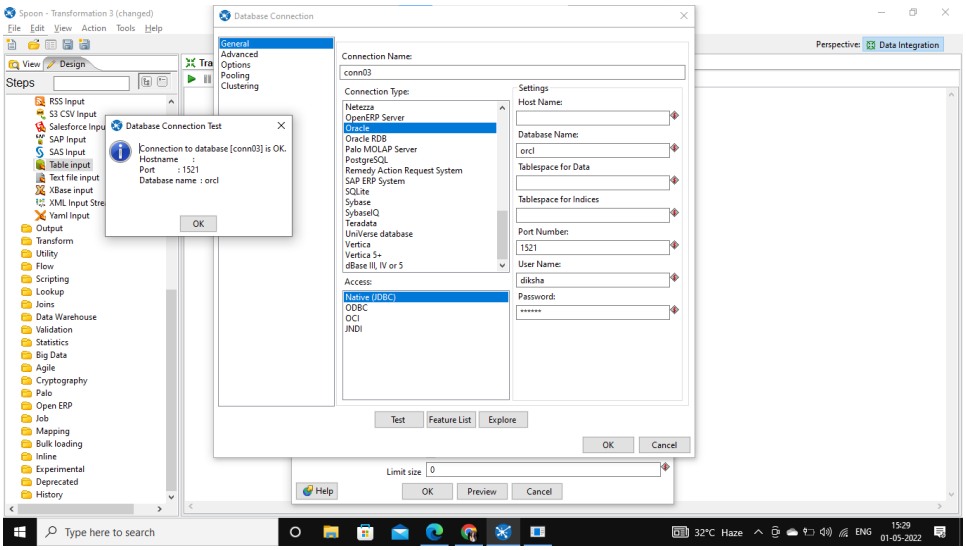
- **Transformation 3: Calculator**

Calculator

Drag and drop the table input -> Double click on Table input -> New Connection.



2) Fill up the connection details -> Test the connectivity -> OK.



Click on Get SQL Select Statement -> Conn03-> Tables -> EMPL01-> OK -> Get Fields ->YES.

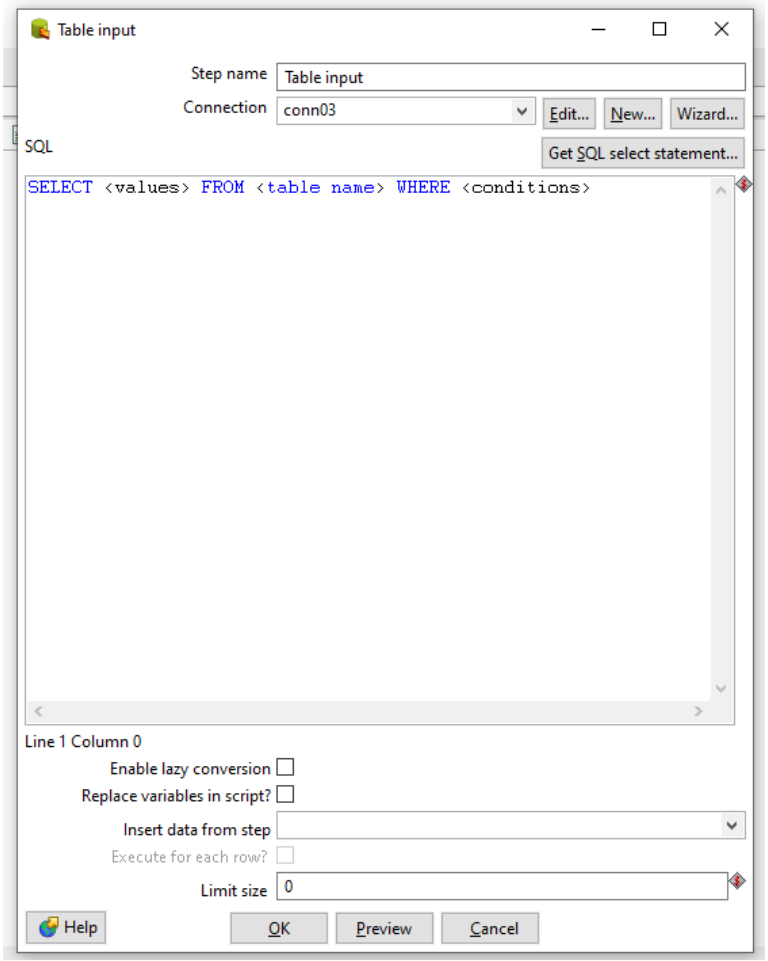
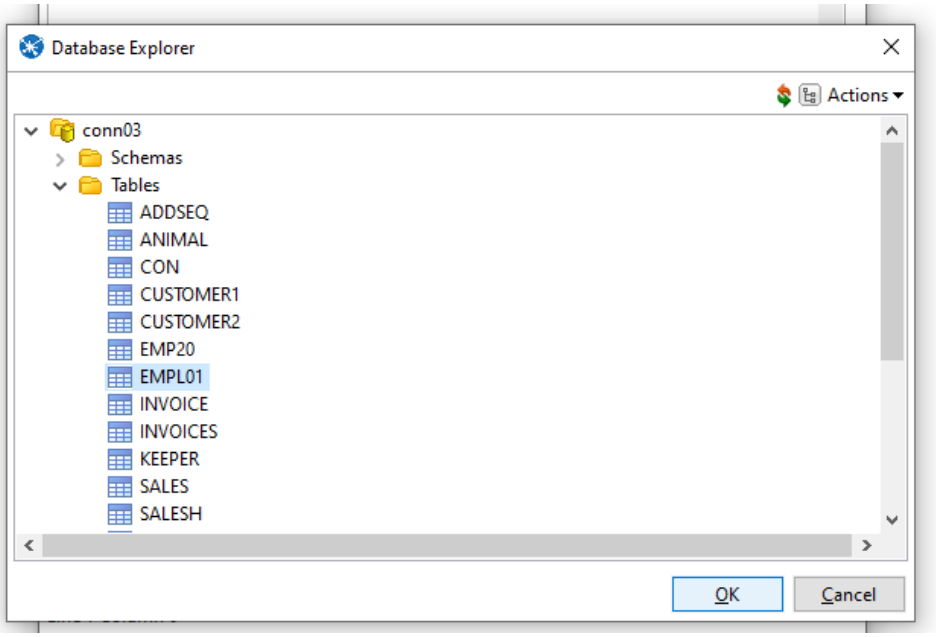
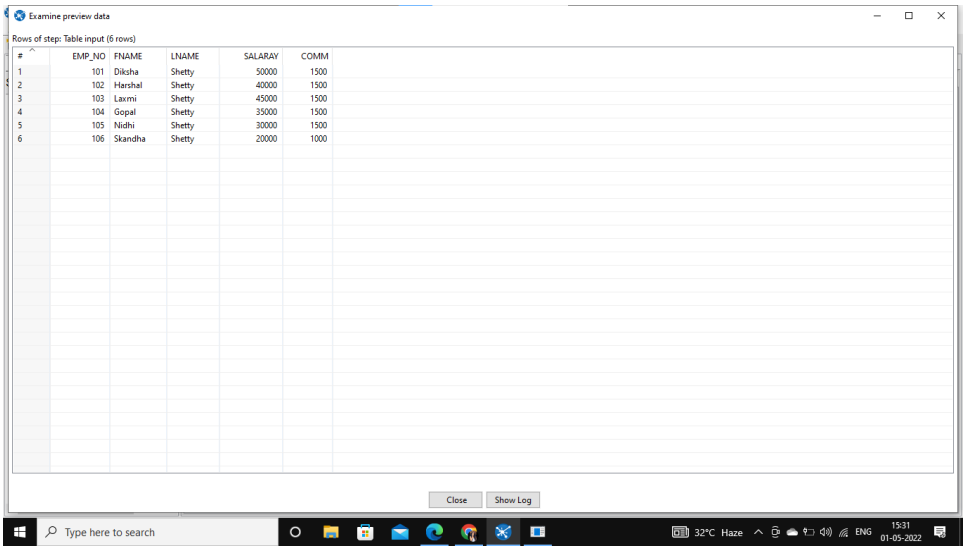


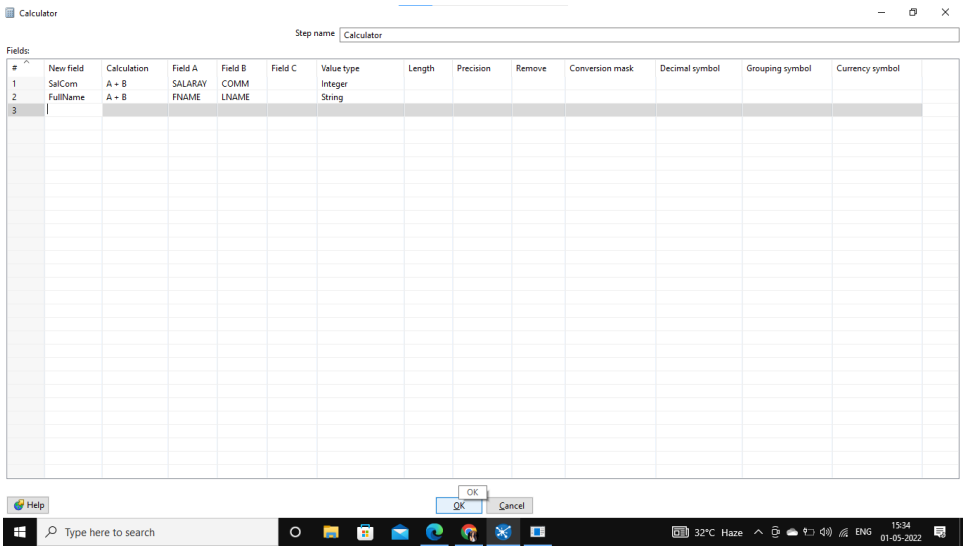
Table input
Step name
Table input
Connection
conn03
Edit...
New...
Wizard...
SQL
Get SQL select statement...
SELECT <values> FROM <table name> WHERE <conditions>
Line 1 Column 0
Enable lazy conversion
Replace variables in script?
Insert data from step
Execute for each row?
Limit size
0
Help
OK
Preview
Cancel


Database Explorer
Actions
conn03
Schemas
Tables
ADDSEQ
ANIMAL
CON
CUSTOMER1
CUSTOMER2
EMP20
EMPL01
INVOICE
INVOICES
KEEPER
SALES
SALESH
OK
Cancel

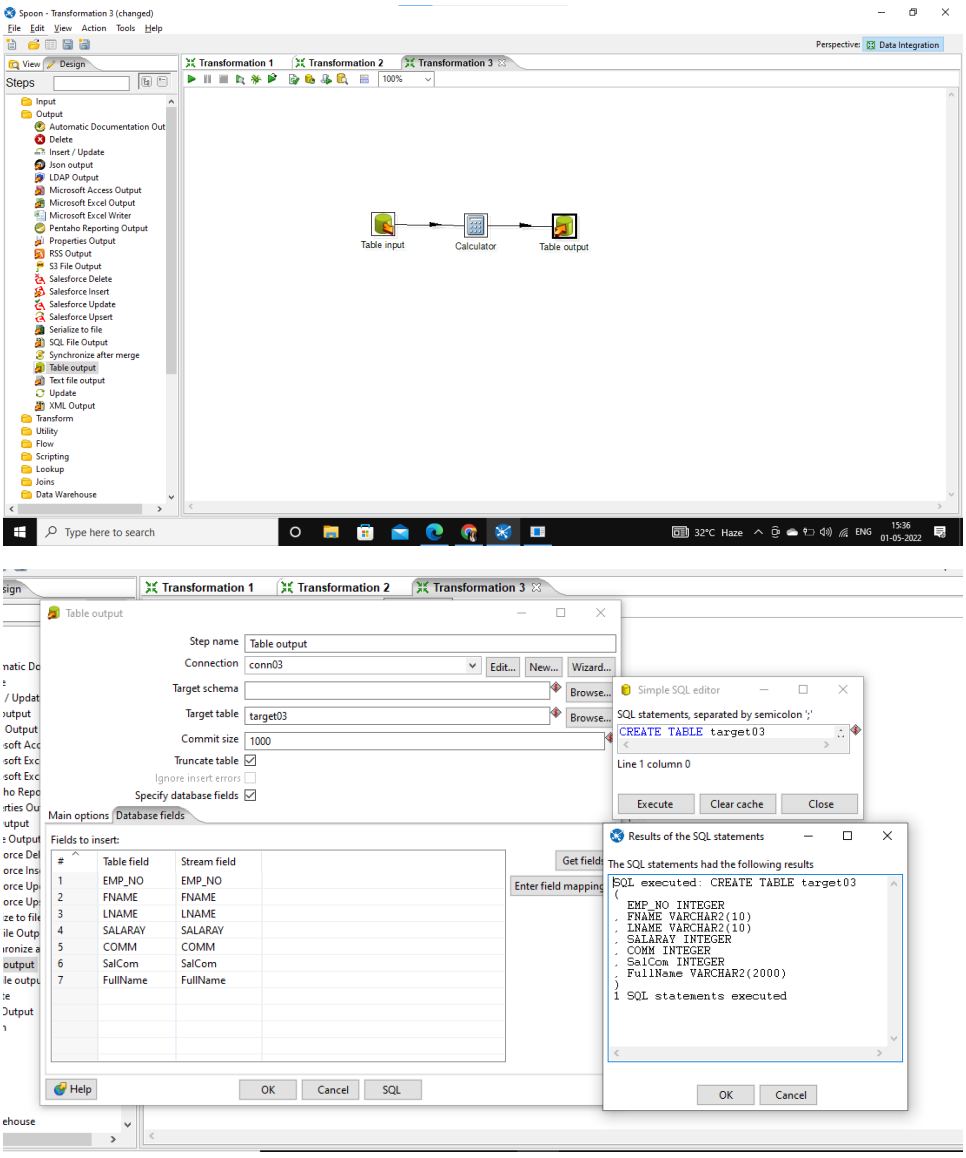
Click on preview data -> afterwards close button.



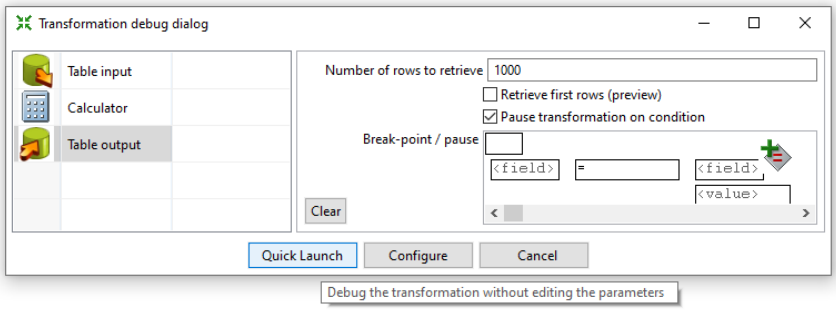
Drag and drop the calculator and double click on Calculator icon. Include the new field name, calculation what operation need to perform, fields column name, field value type.



6) **Drag and drop the table output -> Double click on Table output -> write the target table name -> Check the boxes -> Select the tab database fields -> Get fields-> Click on SQL -> Execute the statement -> OK .**



Click on spider icon for quick launch -> Quick Launch.

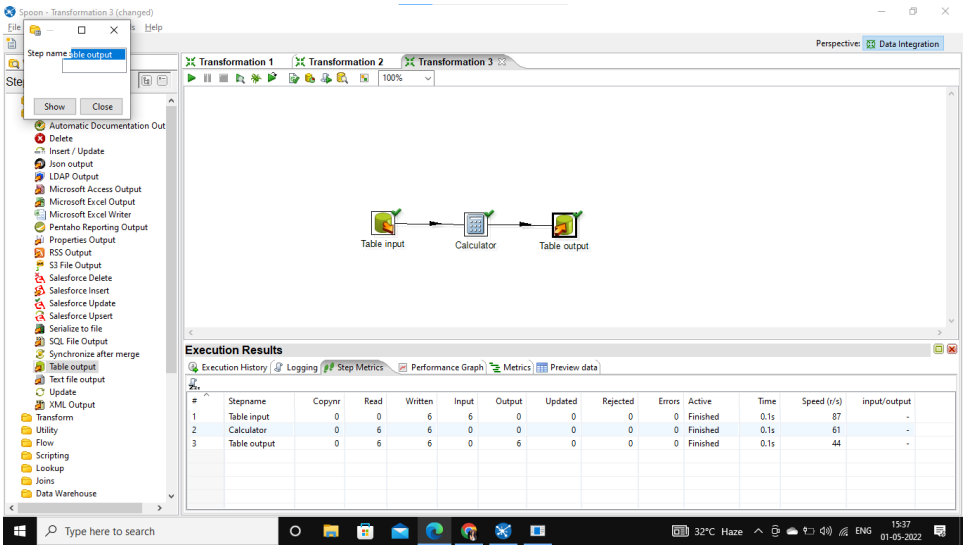


Examine preview data

Rows of step: Table output (6 rows)

#	EMP_NO	FNAME	LNAME	SALARAY	COMM	SalCom	FullName
1	106	Skandha	Shetty	20000	1000	21000	SkandhaShetty
2	105	Nidhi	Shetty	30000	1500	31500	NidhiShetty
3	104	Gopal	Shetty	35000	1500	36500	GopalShetty
4	103	Laxmi	Shetty	45000	1500	46500	LaxmiShetty
5	102	Harshal	Shetty	40000	1500	41500	HarshalShetty
6	101	Diksha	Shetty	50000	1500	51500	DikshaShetty

SUCCESSFUL

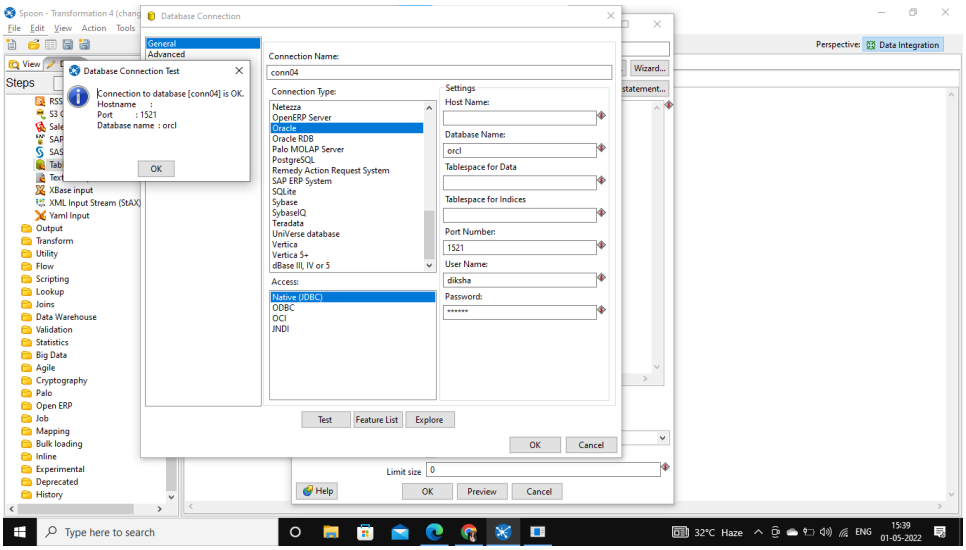


OUTPUT:

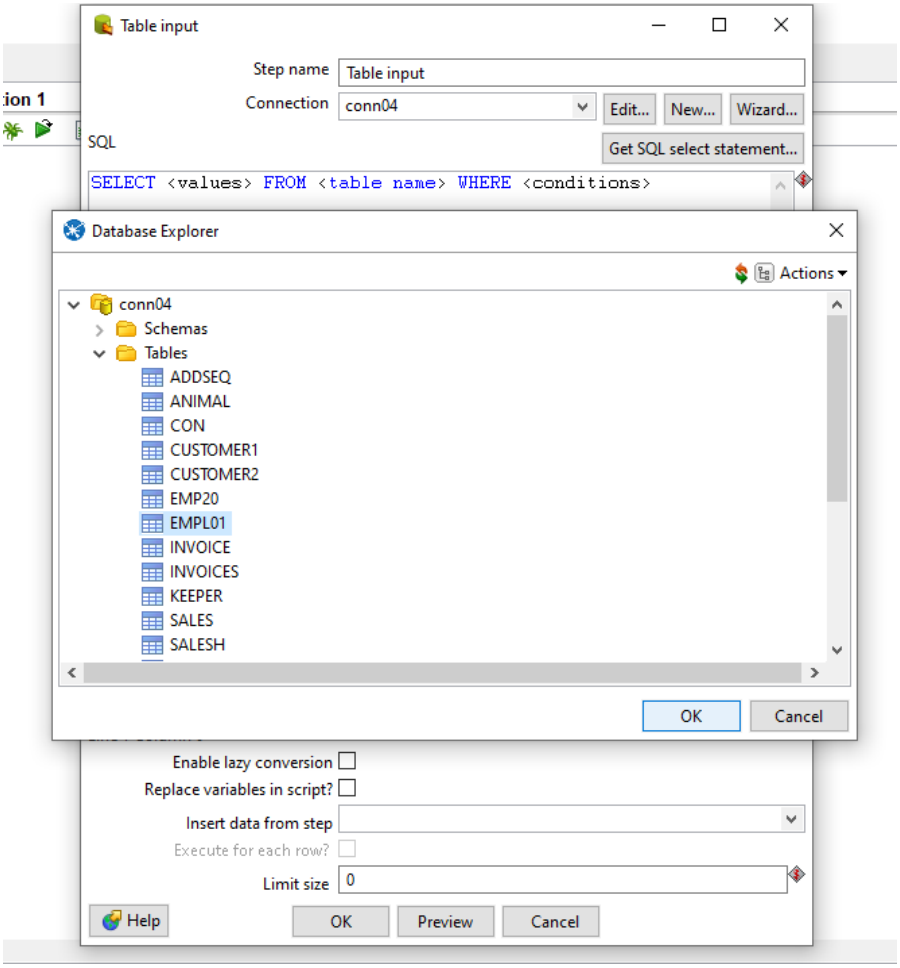
```
SQL> select * from target03;

    EMP_NO FNAME      LNAME          SALARAY       COMM      SALCOM
---------- ---------- ---------- ---------- ---------- ----------
FULLNAME
--------------------------------------------------------------------------------
       101 Diksha     Shetty          50000       1500      51500
DikshaShetty

       102 Harshal    Shetty          40000       1500      41500
HarshalShetty

       103 Laxmi      Shetty          45000       1500      46500
LaxmiShetty


    EMP_NO FNAME      LNAME          SALARAY       COMM      SALCOM
---------- ---------- ---------- ---------- ---------- ----------
FULLNAME
--------------------------------------------------------------------------------
       104 Gopal      Shetty          35000       1500      36500
GopalShetty

       105 Nidhi      Shetty          30000       1500      31500
NidhiShetty

       106 Skandha    Shetty          20000       1000      21000
SkandhaShetty


6 rows selected.
```

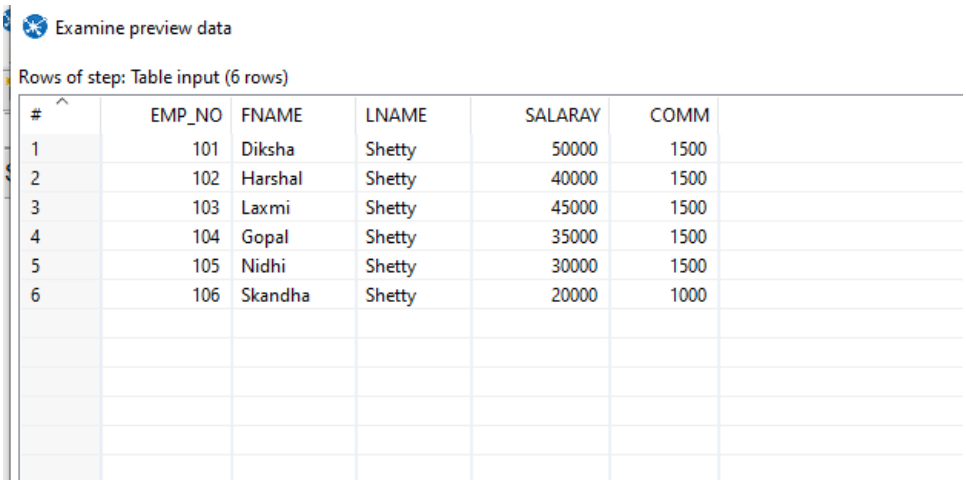
- **Transformation 4: Concat the Fields**

Drag and drop the input table -> double click on table input-> New Connection -> Fill up the details for connection -> Test the connectivity -> OK.



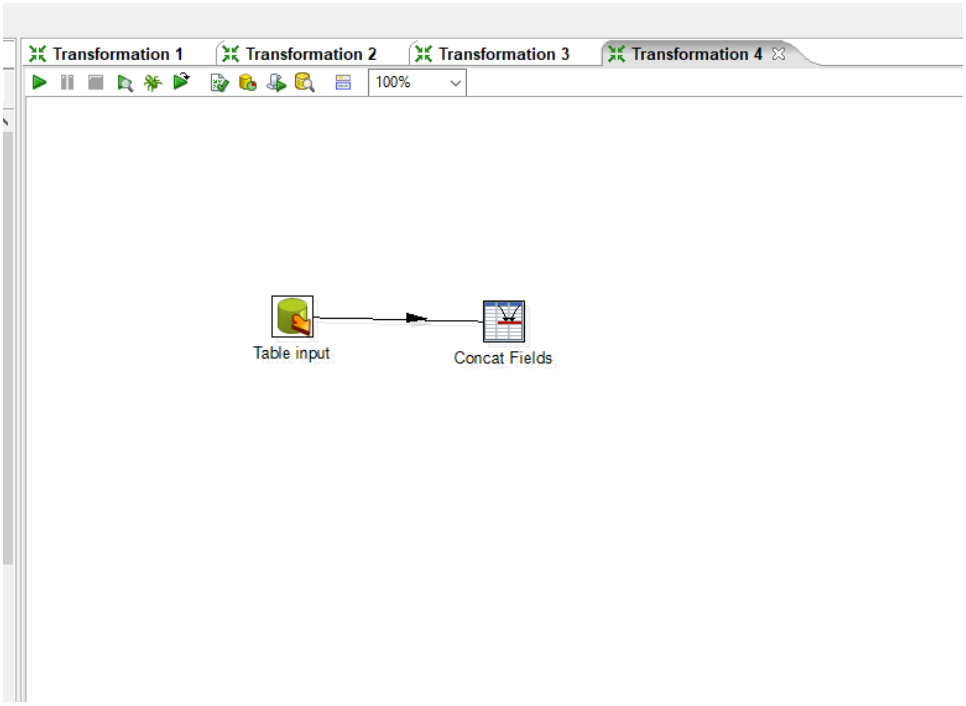
Click on GET SQL SELECT STATEMENT button -> Select Conn04-> tables-> empl01->OK.



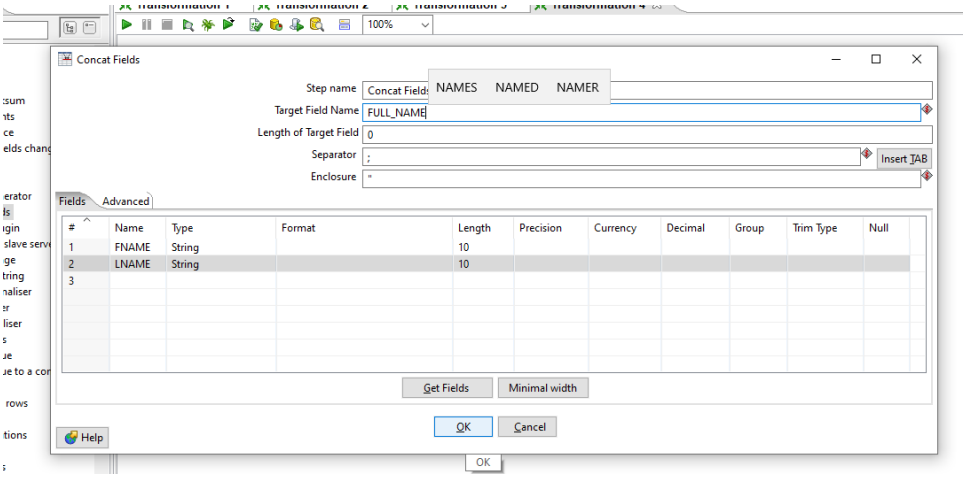


#	EMP_NO	FNAME	LNAME	SALARAY	COMM
1	101	Diksha	Shetty	50000	1500
2	102	Harshal	Shetty	40000	1500
3	103	Laxmi	Shetty	45000	1500
4	104	Gopal	Shetty	35000	1500
5	105	Nidhi	Shetty	30000	1500
6	106	Skandha	Shetty	20000	1000

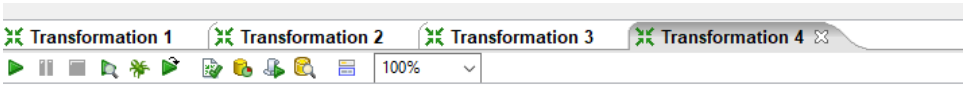
Drag and drop the concat fields and double click on concat fields. Write target step name as Full Name.

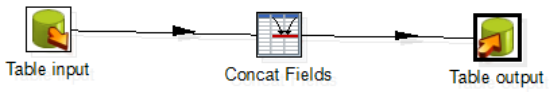


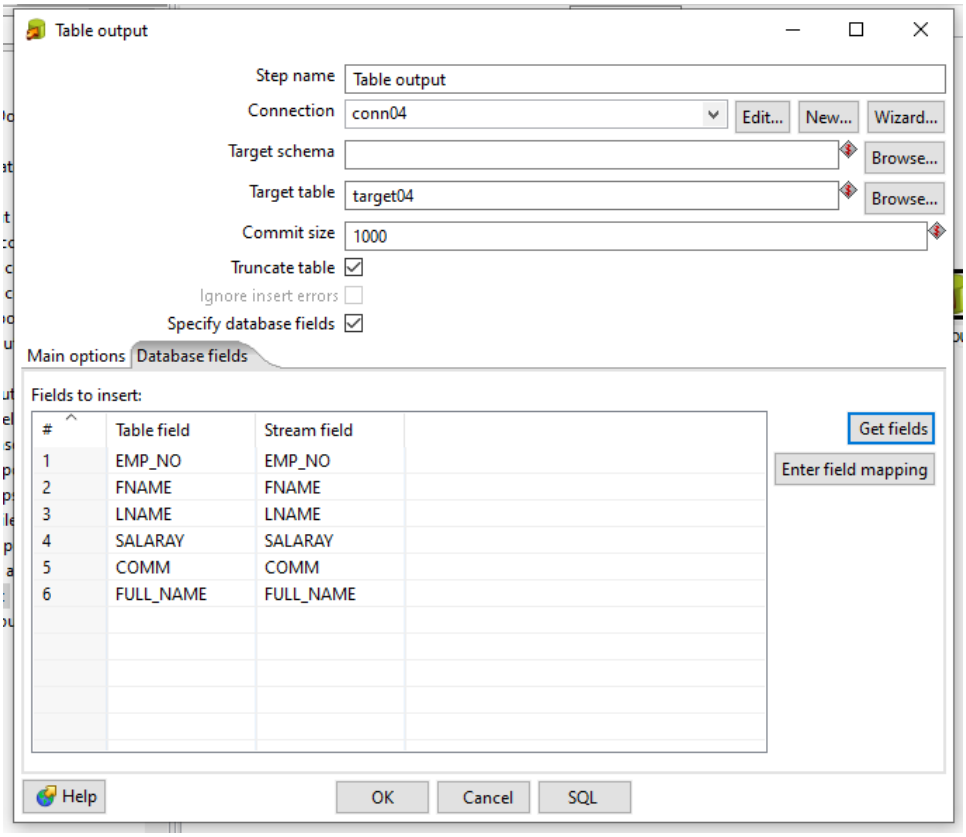
4) Click on get fields -> Fname and Lname are kept and others are discarded and their datatype would be STRING. They are separated by ';' and enclosed by ' " '. Then OK.

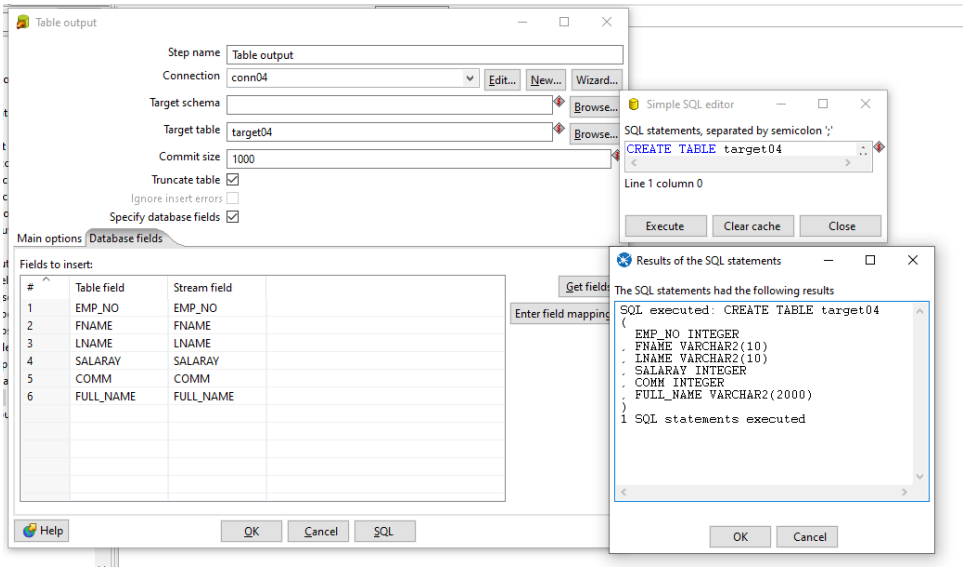


Drag and drop table output and double click on it. Check the boxes and select the tab database fields -> Get fields -> SQL -> Execute -> ok.

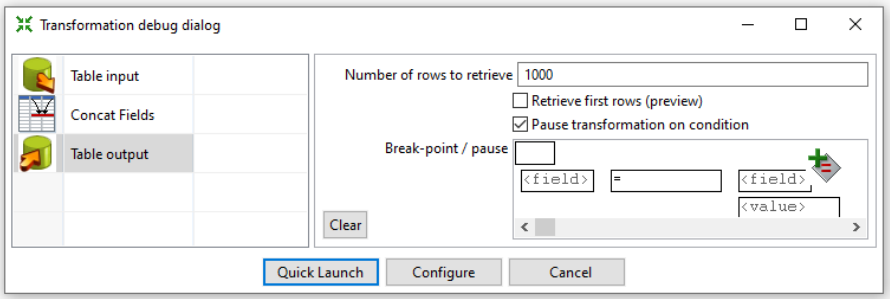




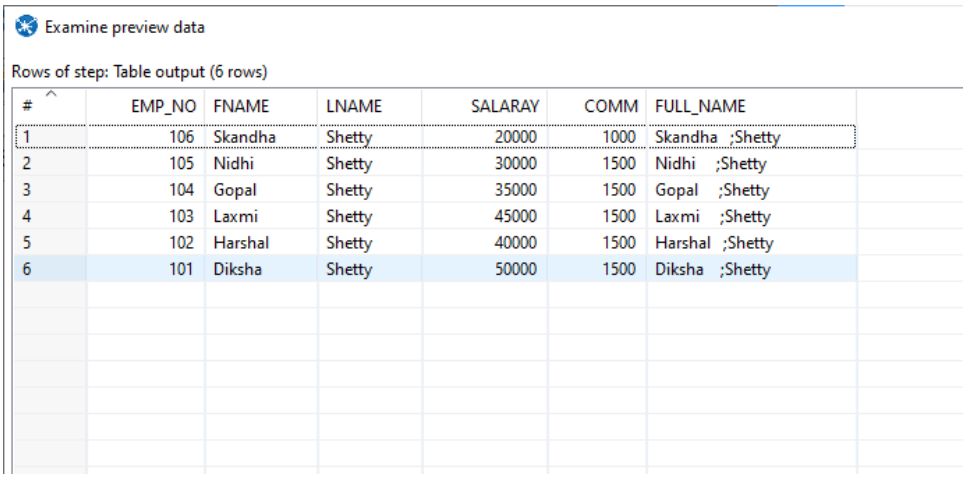




6) Click on spider icon for QUICK LAUNCH.



OUTPUT:

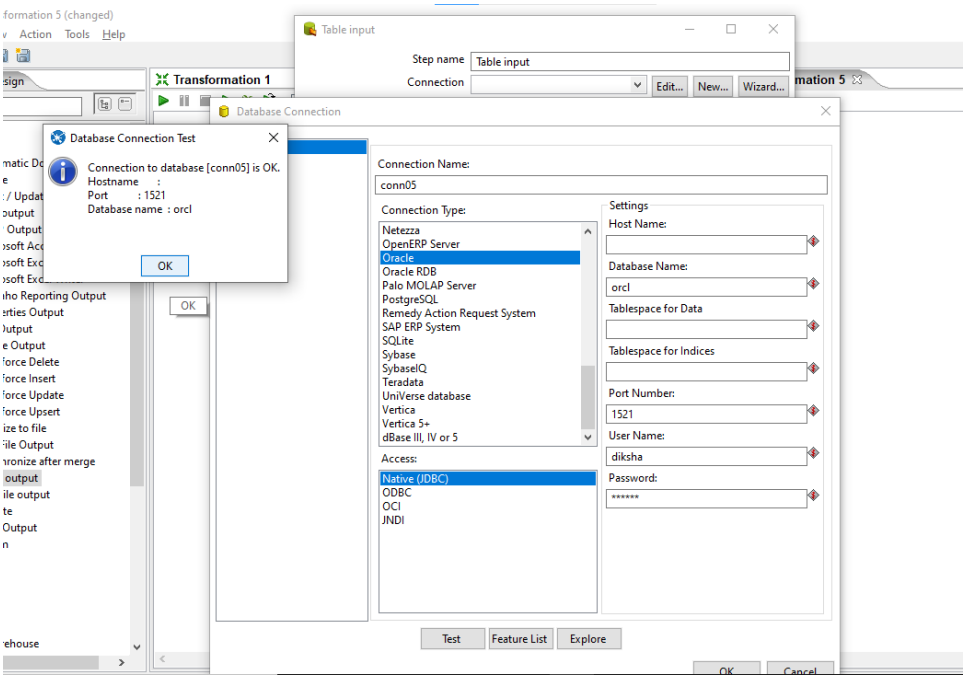


#	EMP_NO	FNAME	LNAME	SALARAY	COMM	FULL_NAME
1	106	Skandha	Shetty	20000	1000	Skandha ;Shetty
2	105	Nidhi	Shetty	30000	1500	Nidhi ;Shetty
3	104	Gopal	Shetty	35000	1500	Gopal ;Shetty
4	103	Laxmi	Shetty	45000	1500	Laxmi ;Shetty
5	102	Harshal	Shetty	40000	1500	Harshal ;Shetty
6	101	Diksha	Shetty	50000	1500	Diksha ;Shetty

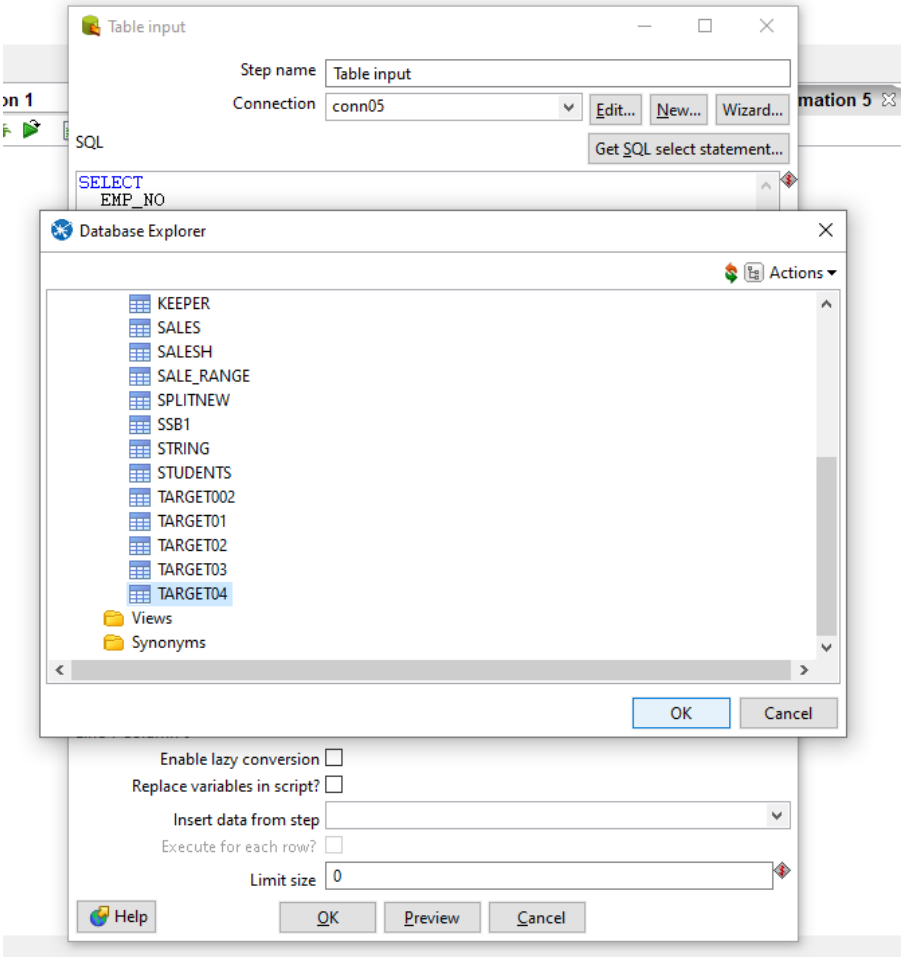
```
SQL> select * from target04;

    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
FULL_NAME
--------------------------------------------------------------------------------
       101 Diksha     Shetty          50000       1500
Diksha    ;Shetty

       102 Harshal    Shetty          40000       1500
Harshal   ;Shetty

       103 Laxmi      Shetty          45000       1500
Laxmi     ;Shetty


    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
FULL_NAME
--------------------------------------------------------------------------------
       104 Gopal      Shetty          35000       1500
Gopal     ;Shetty

       105 Nidhi      Shetty          30000       1500
Nidhi     ;Shetty

       106 Skandha    Shetty          20000       1000
Skandha   ;Shetty


6 rows selected.
```

- ## Transformation 5: Split rows

1) Drag and drop the table input -> double click on table input -> Fill up the connection details -> Test the connectivity -> OK.



2) Click on get sql select statement command -> Select the table target04 -> OK -> Get fields -> YES.



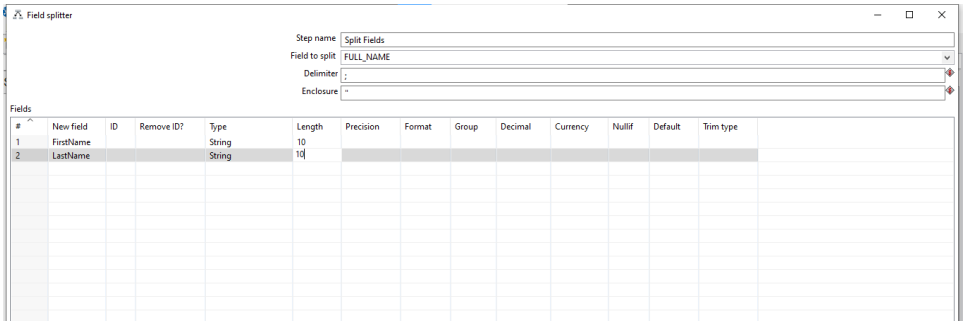
To preview the data, click on preview button.

Examine preview data

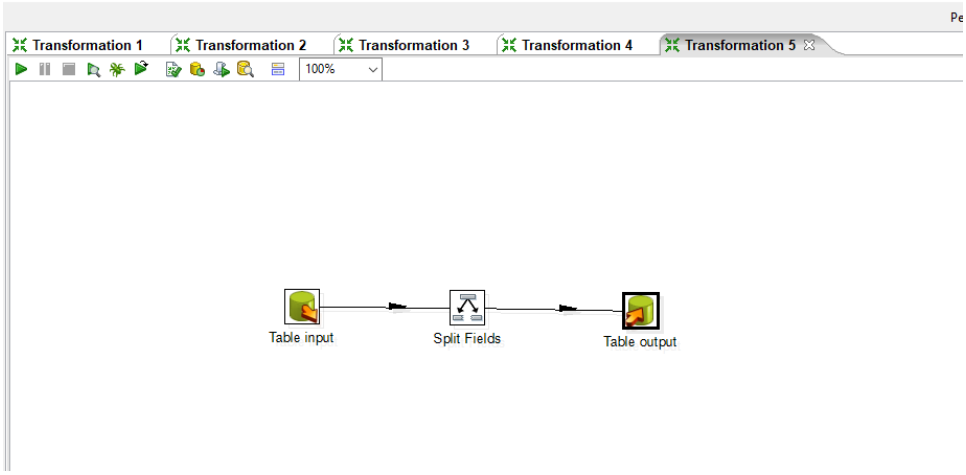
Rows of step: Table input (6 rows)

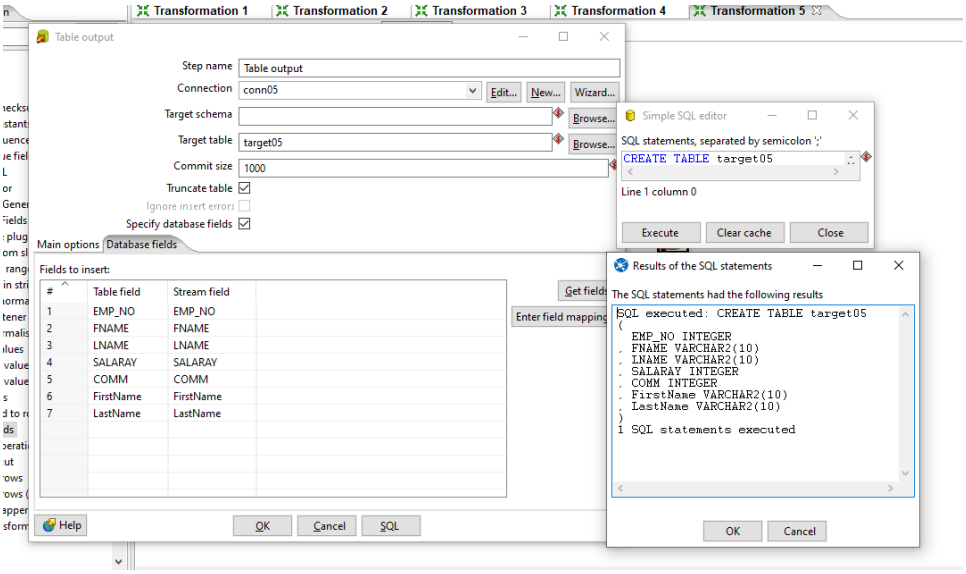
#	EMP_NO	FNAME	LNAME	SALARAY	COMM	FULL_NAME
1	101	Diksha	Shetty	50000	1500	Diksha ;Shetty
2	102	Harshal	Shetty	40000	1500	Harshal ;Shetty
3	103	Laxmi	Shetty	45000	1500	Laxmi ;Shetty
4	104	Gopal	Shetty	35000	1500	Gopal ;Shetty
5	105	Nidhi	Shetty	30000	1500	Nidhi ;Shetty
6	106	Skandha	Shetty	20000	1000	Skandha ;Shetty

Drag and drop the split fields -> Double click on it -> Field split is Full Name into first name and last name. Giving their Datatype and length of string.

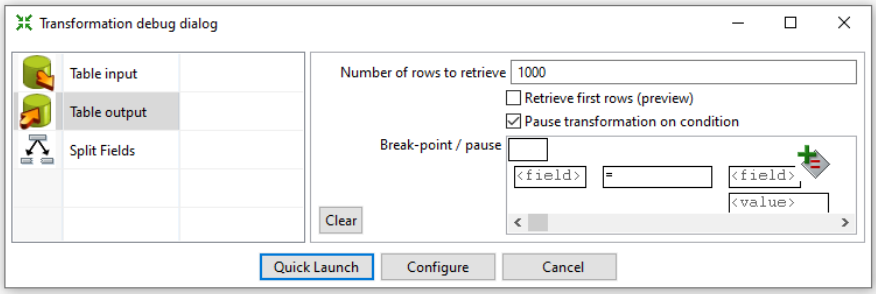


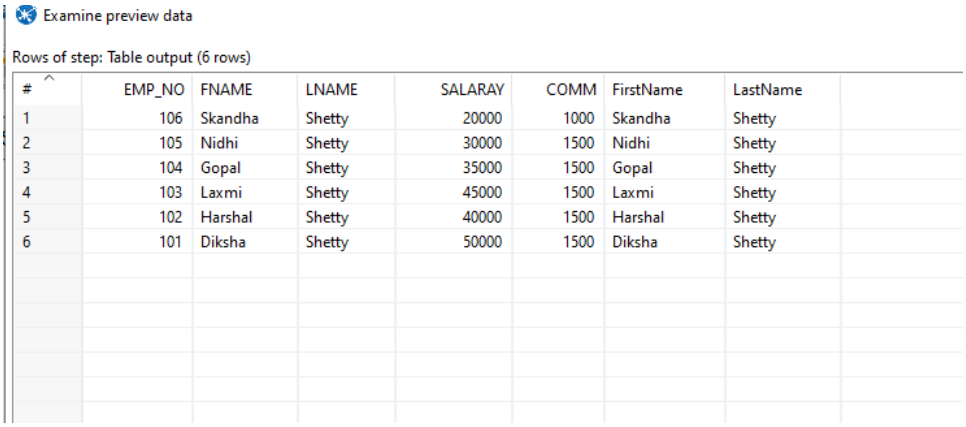
Drag and drop the table output -> Double click on it -> Write the target table name as target5 -> check the boxes and select the database fields -> Get fields .



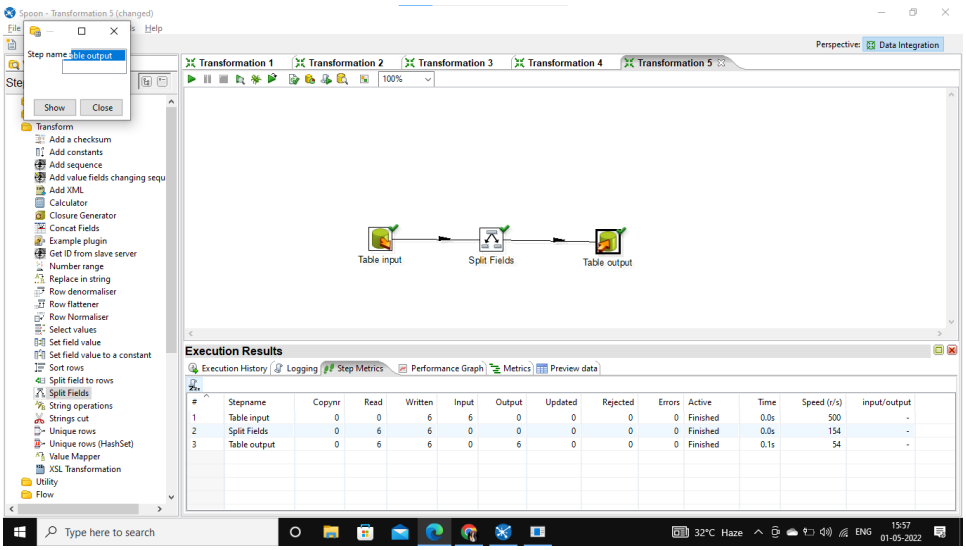


Click on quick launch icon. And the output are,





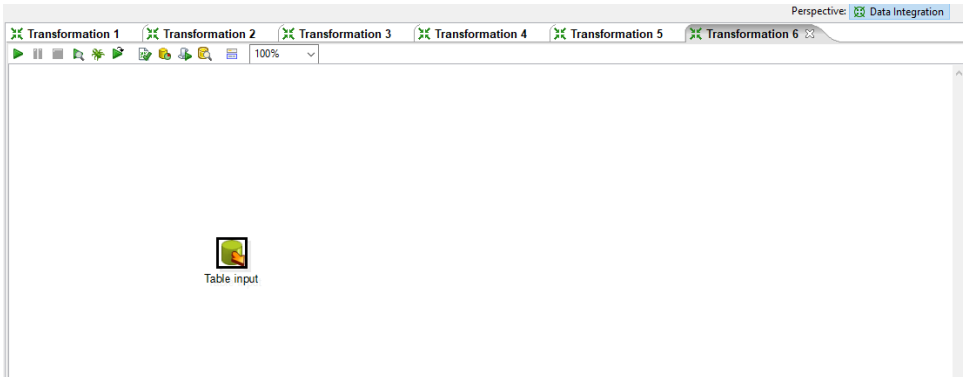
#	EMP_NO	FNAME	LNAME	SALARAY	COMM	FirstName	LastName
1	106	Skandha	Shetty	20000	1000	Skandha	Shetty
2	105	Nidhi	Shetty	30000	1500	Nidhi	Shetty
3	104	Gopal	Shetty	35000	1500	Gopal	Shetty
4	103	Laxmi	Shetty	45000	1500	Laxmi	Shetty
5	102	Harshal	Shetty	40000	1500	Harshal	Shetty
6	101	Diksha	Shetty	50000	1500	Diksha	Shetty

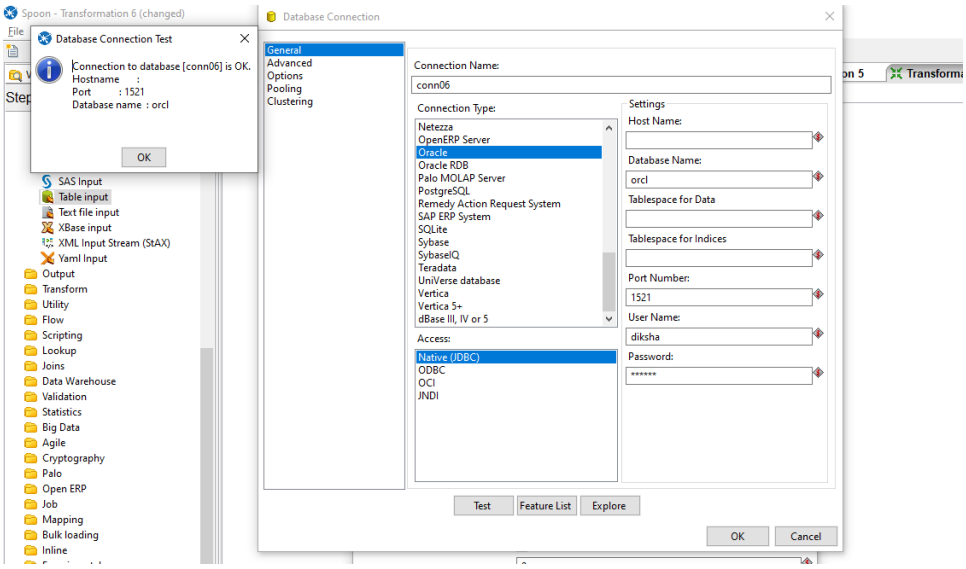



```
SQL> select * from target05;

    EMP_NO FNAME      LNAME          SALARAY       COMM FIRSTNAME  LASTNAME
---------- ---------- ---------- ---------- ---------- ---------- ----------
       101 Diksha     Shetty          50000       1500 Diksha     Shetty
       102 Harshal    Shetty          40000       1500 Harshal    Shetty
       103 Laxmi      Shetty          45000       1500 Laxmi      Shetty
       104 Gopal      Shetty          35000       1500 Gopal      Shetty
       105 Nidhi      Shetty          30000       1500 Nidhi      Shetty
       106 Skandha    Shetty          20000       1000 Skandha    Shetty

6 rows selected.
```

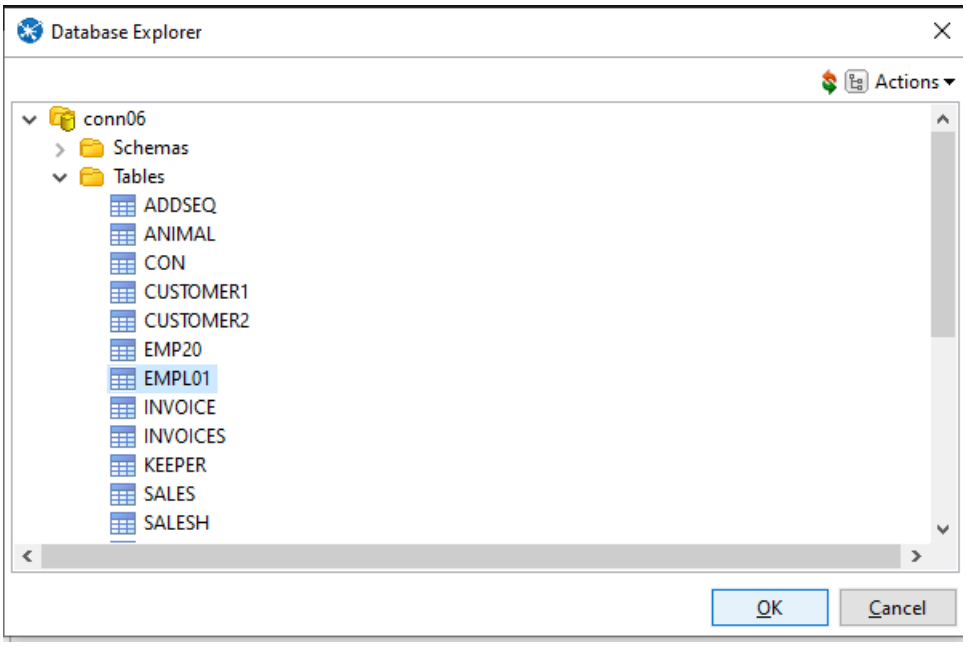
- ## Transformation 6: Number Range

Drag and drop the table input -> double click on it -> fill up the details for connection-> New.





3) Click on GET Select SQL Statement -> Select the table empl01 -> OK.



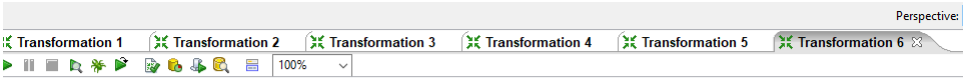
Click on preview data -> Close the log.

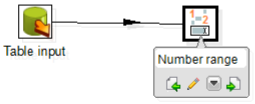
Examine preview data

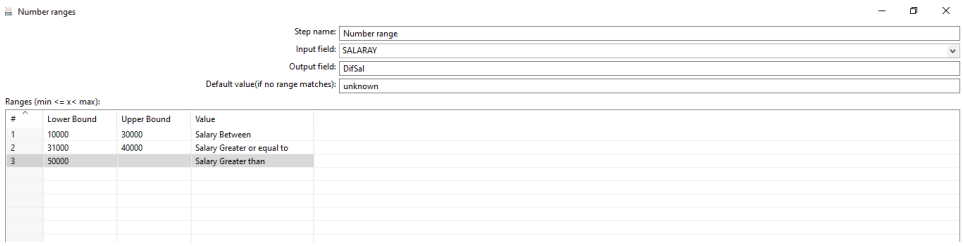
Rows of step: Table input (6 rows)

#	EMP_NO	FNAME	LNAME	SALARAY	COMM
1	101	Diksha	Shetty	50000	1500
2	102	Harshal	Shetty	40000	1500
3	103	Laxmi	Shetty	45000	1500
4	104	Gopal	Shetty	35000	1500
5	105	Nidhi	Shetty	30000	1500
6	106	Skandha	Shetty	20000	1000

5) Drag and drop the number range and double click on it. Input field as salary and output field as Differentiate Sal. Making conditions on table.



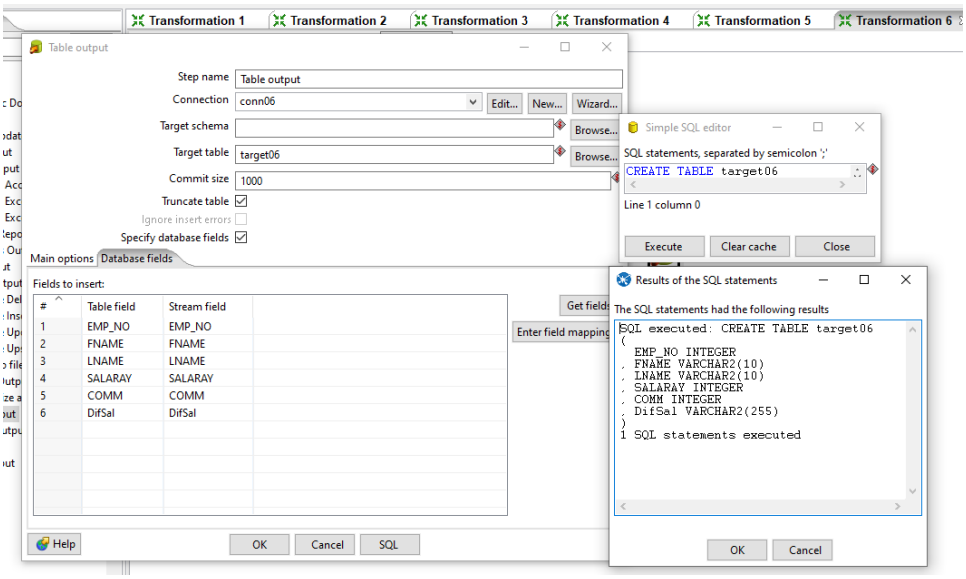




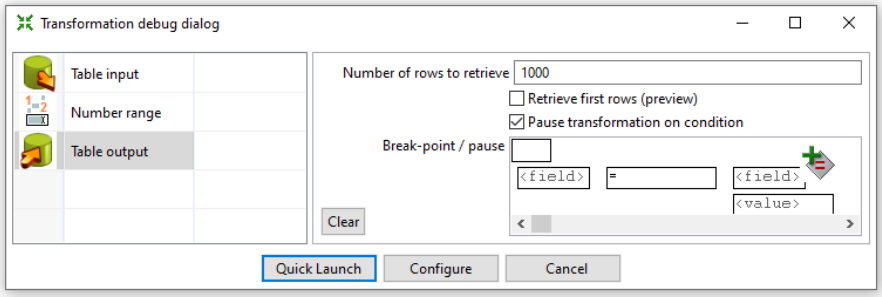
Drag and drop the table output ->double click on it.

Write the table name as target06 and check the boxes. Select the tab database fields -

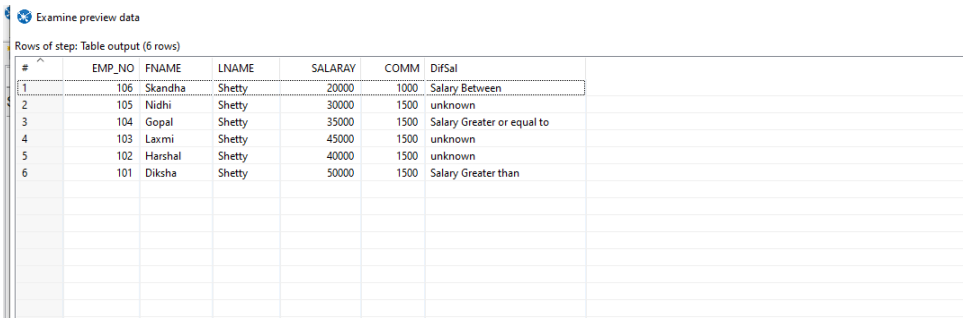
> Get fields -> SQL -> execute the statements -> Ok.



Click on quick launch icon.

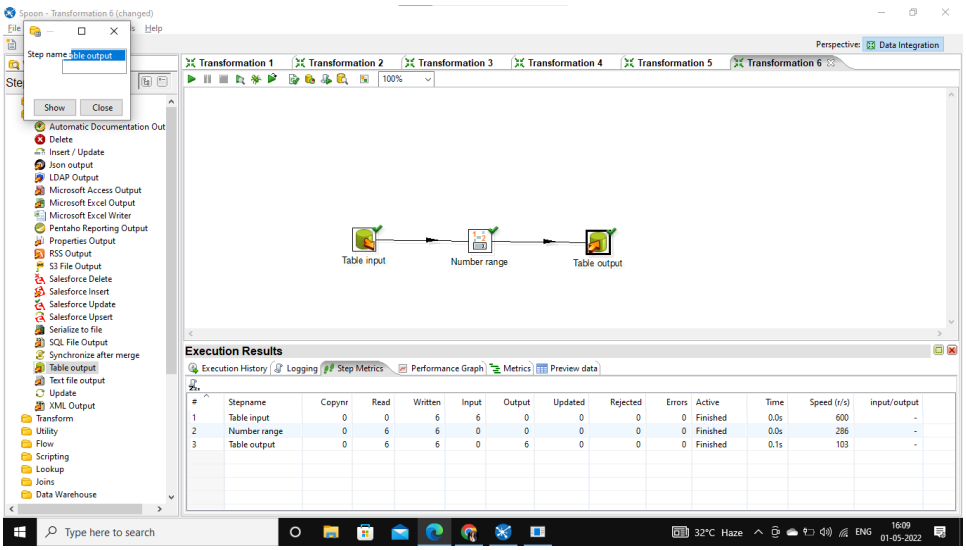


OUTPUT:



#	EMP_NO	FNAME	LNAME	SALARAY	COMM	DifSal
1	106	Skandha	Shetty	20000	1000	Salary Between
2	105	Nidhi	Shetty	30000	1500	unknown
3	104	Gopal	Shetty	35000	1500	Salary Greater or equal to
4	103	Laxmi	Shetty	45000	1500	unknown
5	102	Harshal	Shetty	40000	1500	unknown
6	101	Diksha	Shetty	50000	1500	Salary Greater than

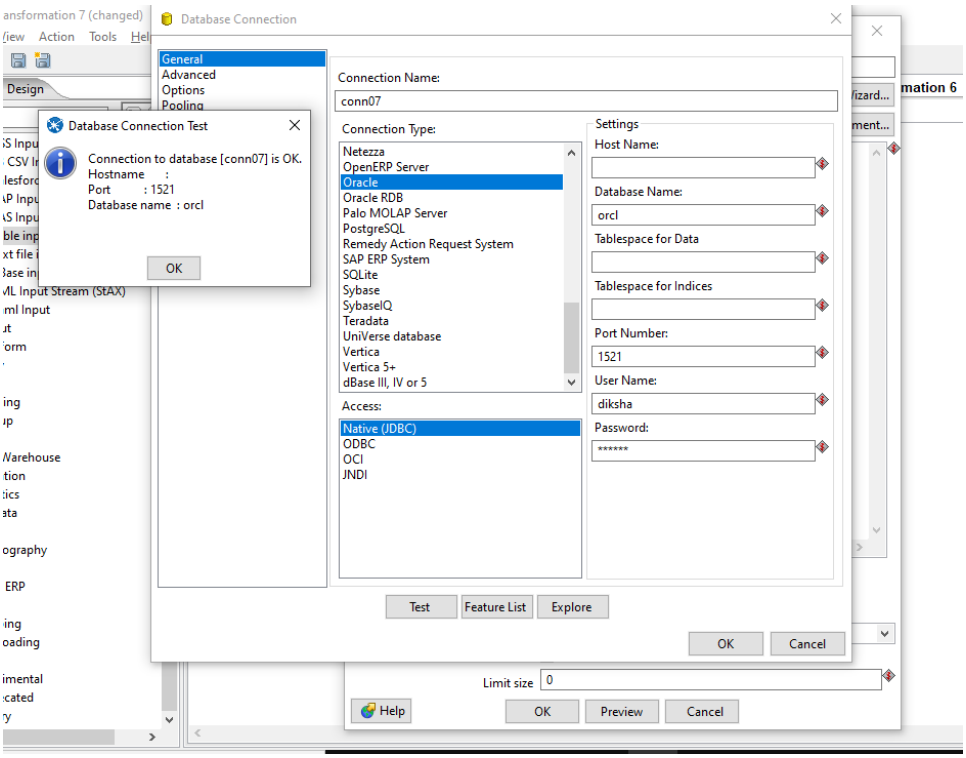
SUCCESS



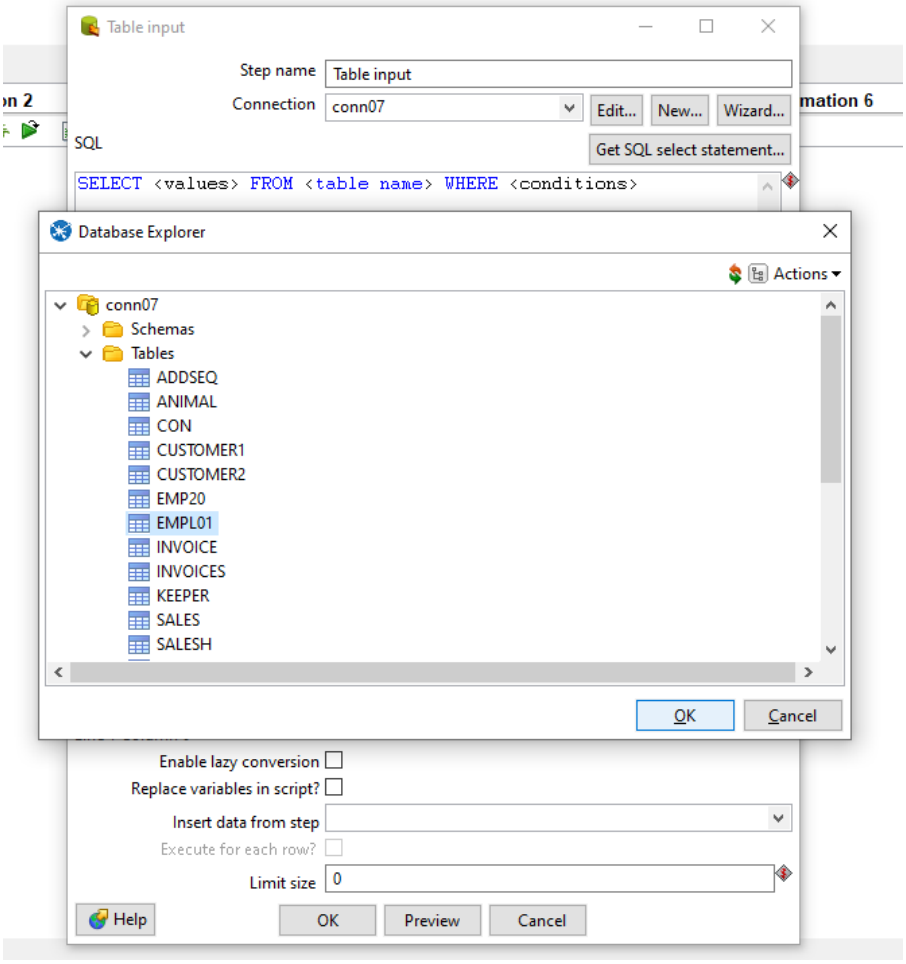

```
SQL> select * from target06;

    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
DIFSAL
--------------------------------------------------------------------------------
       101 Diksha     Shetty          50000       1500
Salary Greater than

       102 Harshal    Shetty          40000       1500
unknown

       103 Laxmi      Shetty          45000       1500
unknown


    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
DIFSAL
--------------------------------------------------------------------------------
       104 Gopal      Shetty          35000       1500
Salary Greater or equal to

       105 Nidhi      Shetty          30000       1500
unknown

       106 Skandha    Shetty          20000       1000
Salary Between


6 rows selected.
```

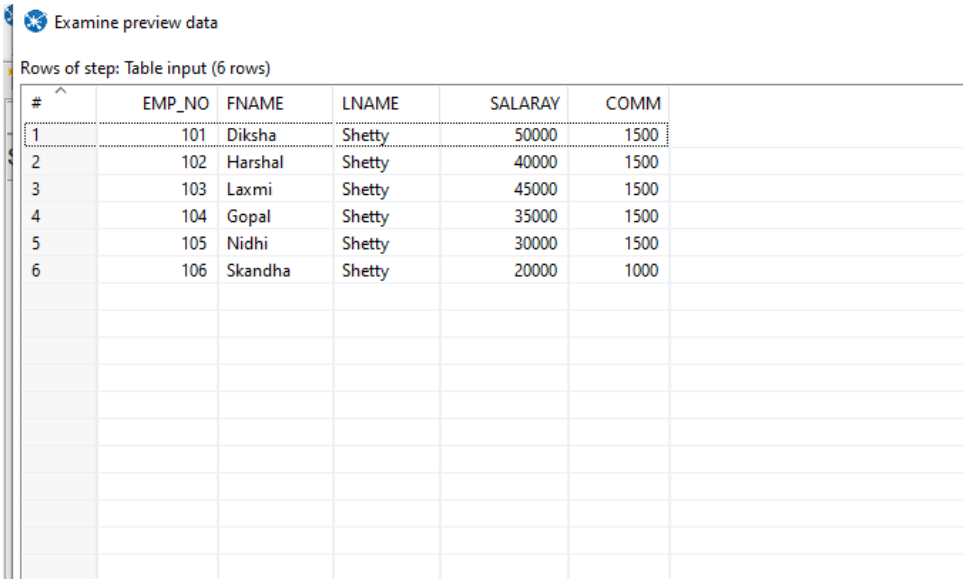
- ## Transformation 7: String Operations

Drag and drop the table input and double click on it. New Connection -> Fill up the details -> Test the connectivity -> OK.



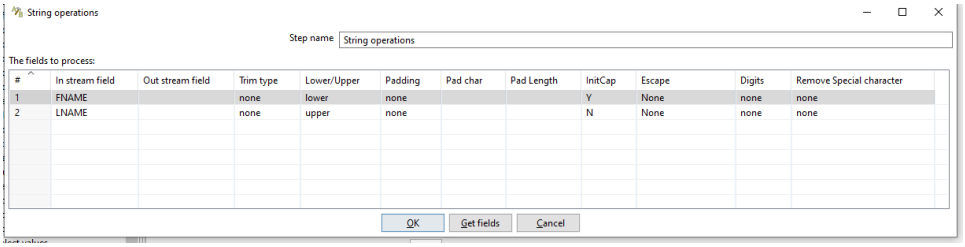
2) Click on Get SQL select statement -> Conn7->Tables-> empl01



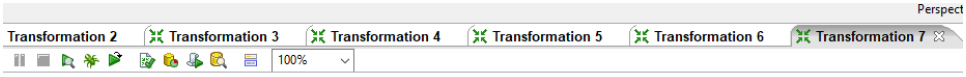


#	EMP_NO	FNAME	LNAME	SALARAY	COMM
1	101	Diksha	Shetty	50000	1500
2	102	Harshal	Shetty	40000	1500
3	103	Laxmi	Shetty	45000	1500
4	104	Gopal	Shetty	35000	1500
5	105	Nidhi	Shetty	30000	1500
6	106	Skandha	Shetty	20000	1000

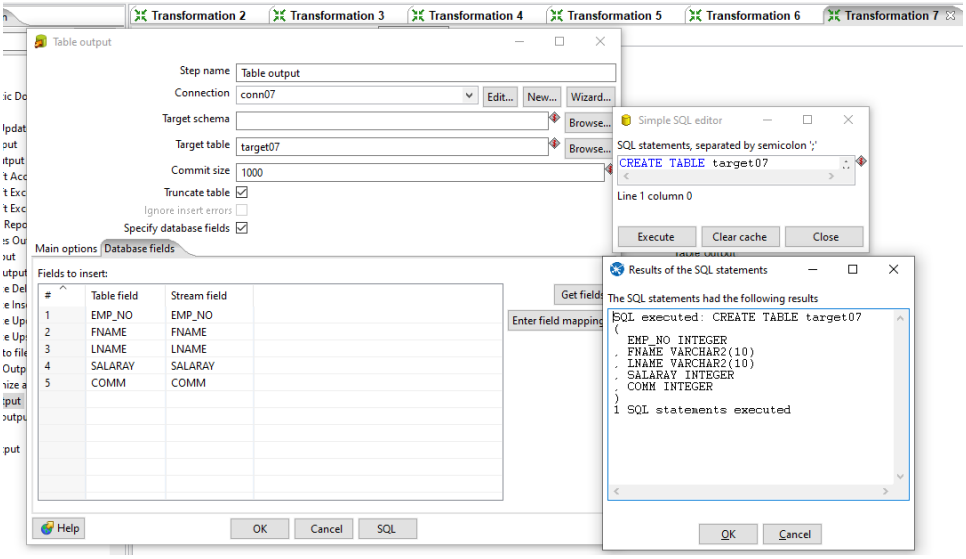
Drag and drop the string operations and double click on it. Performing the operations on Fname and Lname as Upper and lower Case .

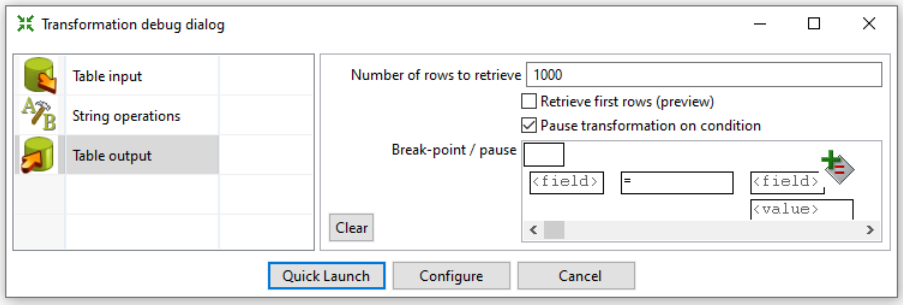


Drag and drop the table output and double click on it.



Write the table name as target07Check the boxes and select the tab database fields -> Click on Get fields. Execute the SQL statements -> OK.

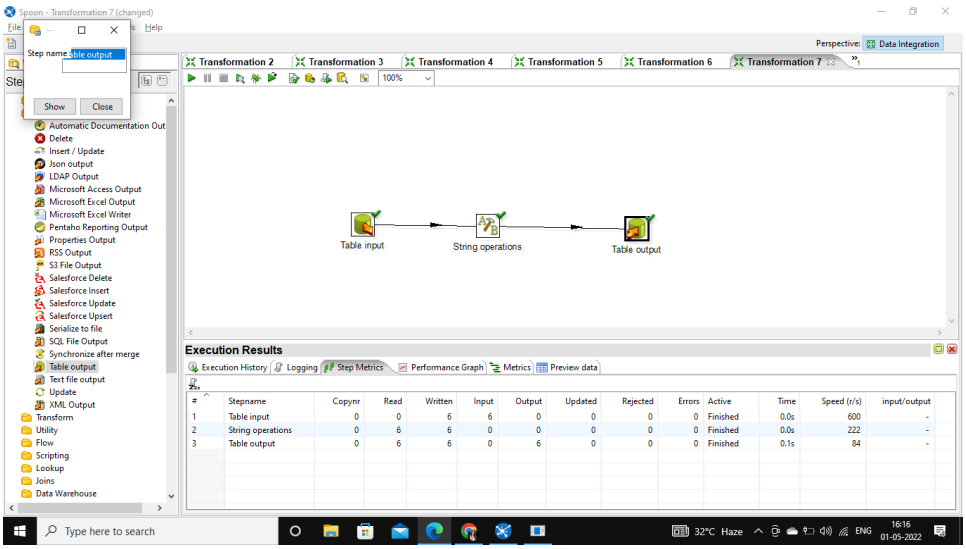




OUTPUT:

#	EMP_NO	FNAME	LNAME	SALARAY	COMM
1	106	Skandha	SHETTY	20000	1000
2	105	Nidhi	SHETTY	30000	1500
3	104	Gopal	SHETTY	35000	1500
4	103	Laxmi	SHETTY	45000	1500
5	102	Harshal	SHETTY	40000	1500
6	101	Diksha	SHETTY	50000	1500

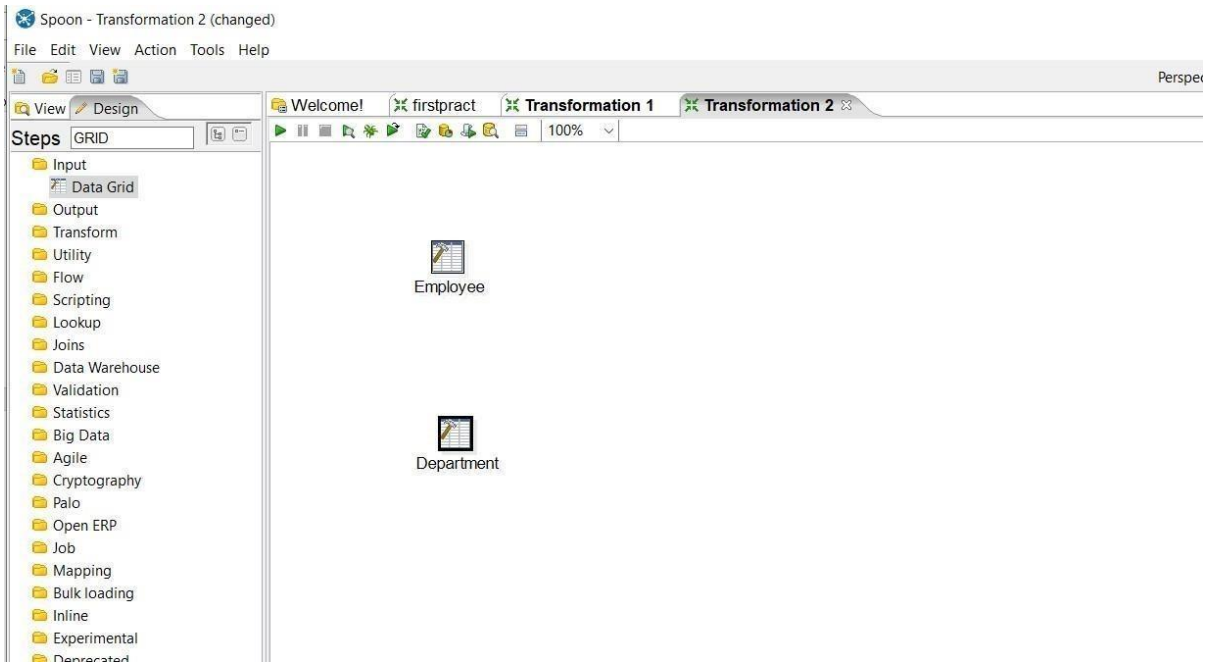
SUCCESS



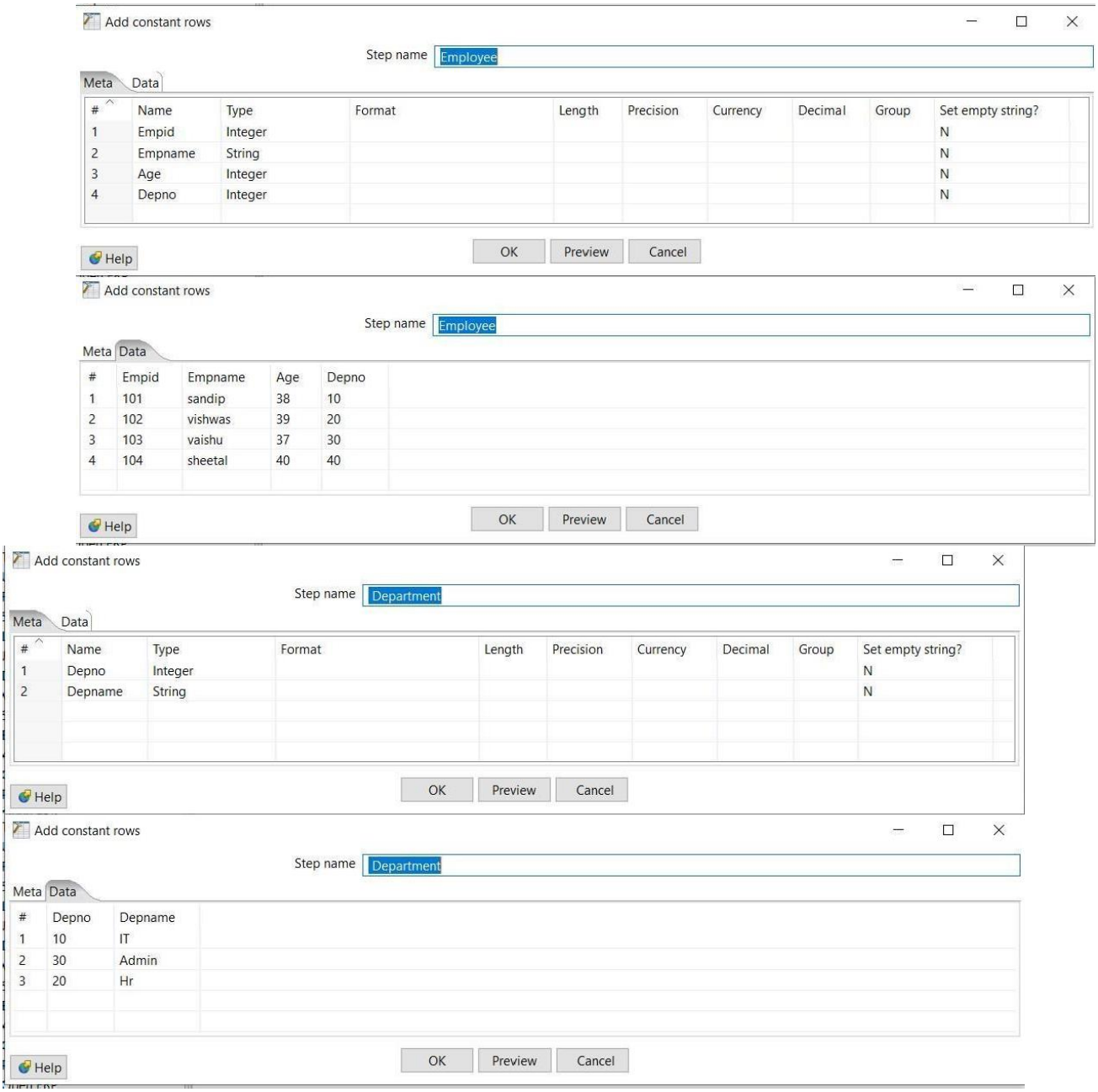
```
SQL> select * from target07;

    EMP_NO FNAME      LNAME          SALARAY       COMM
---------- ---------- ---------- ---------- ----------
       101 Diksha     SHETTY          50000       1500
       102 Harshal    SHETTY          40000       1500
       103 Laxmi      SHETTY          45000       1500
       104 Gopal      SHETTY          35000       1500
       105 Nidhi      SHETTY          30000       1500
       106 Skandha    SHETTY          20000       1000

6 rows selected.
```

- **Implement the merge join transformation on tables**

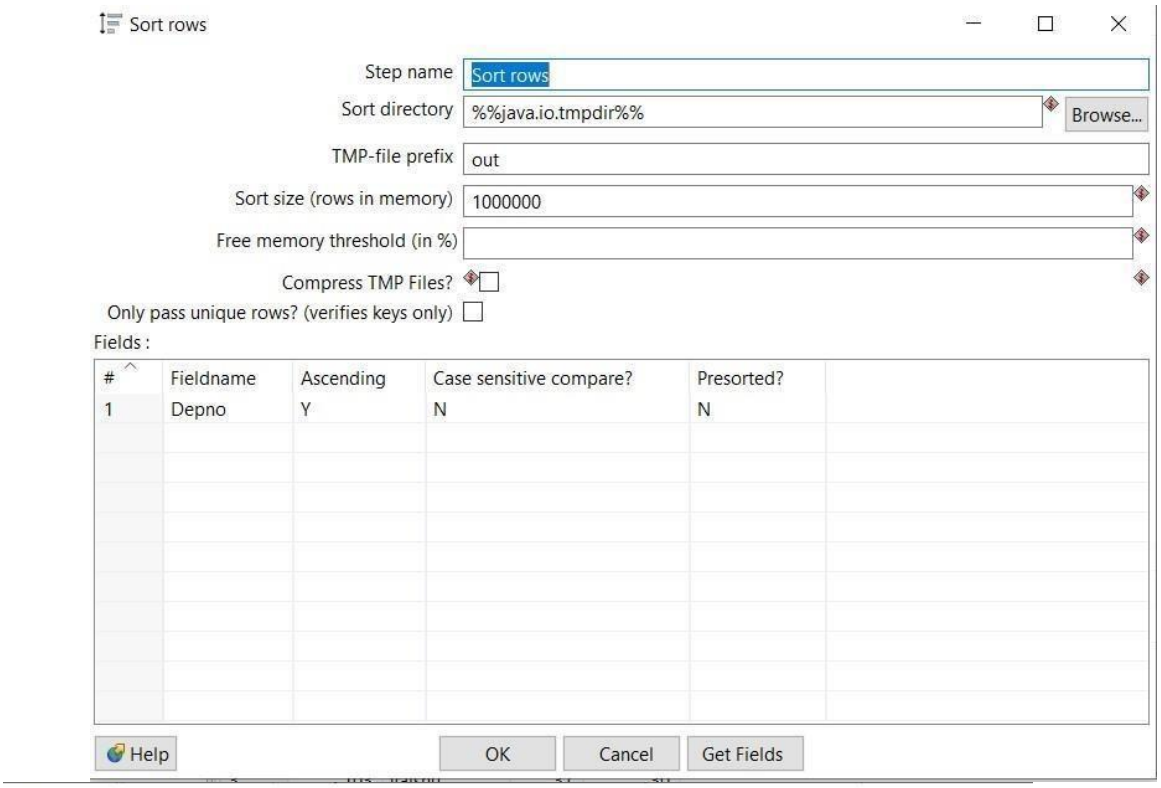
Step1: Drag two Data grid on panel rename them with Employee and Department respectively.

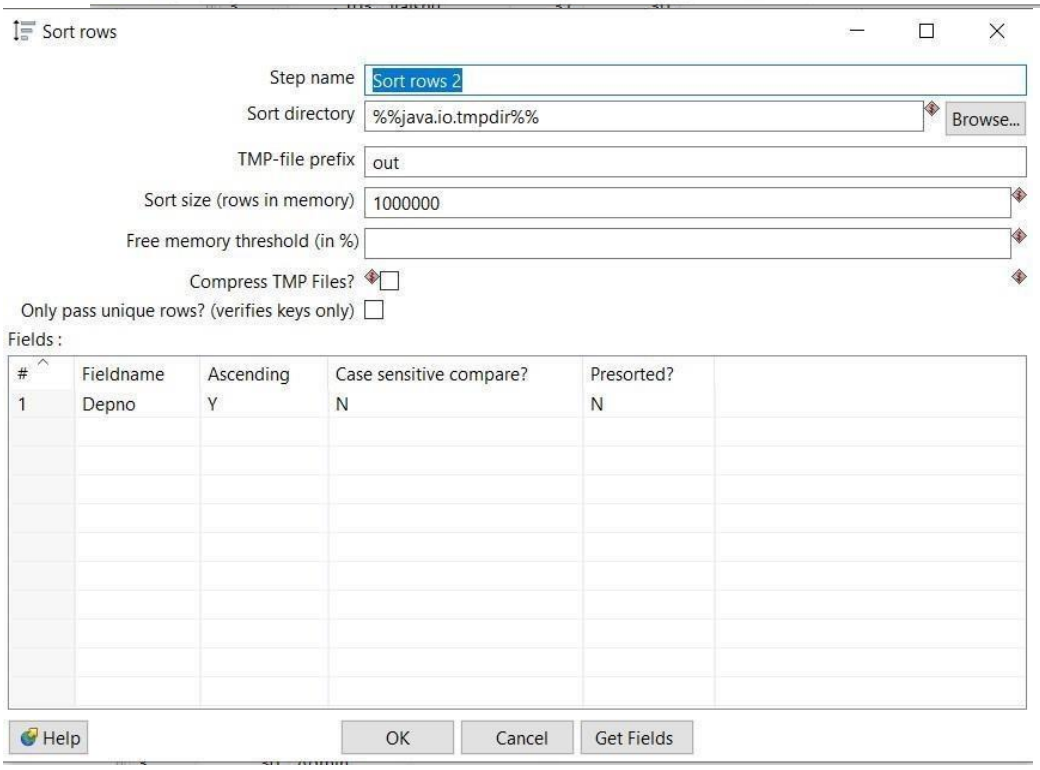


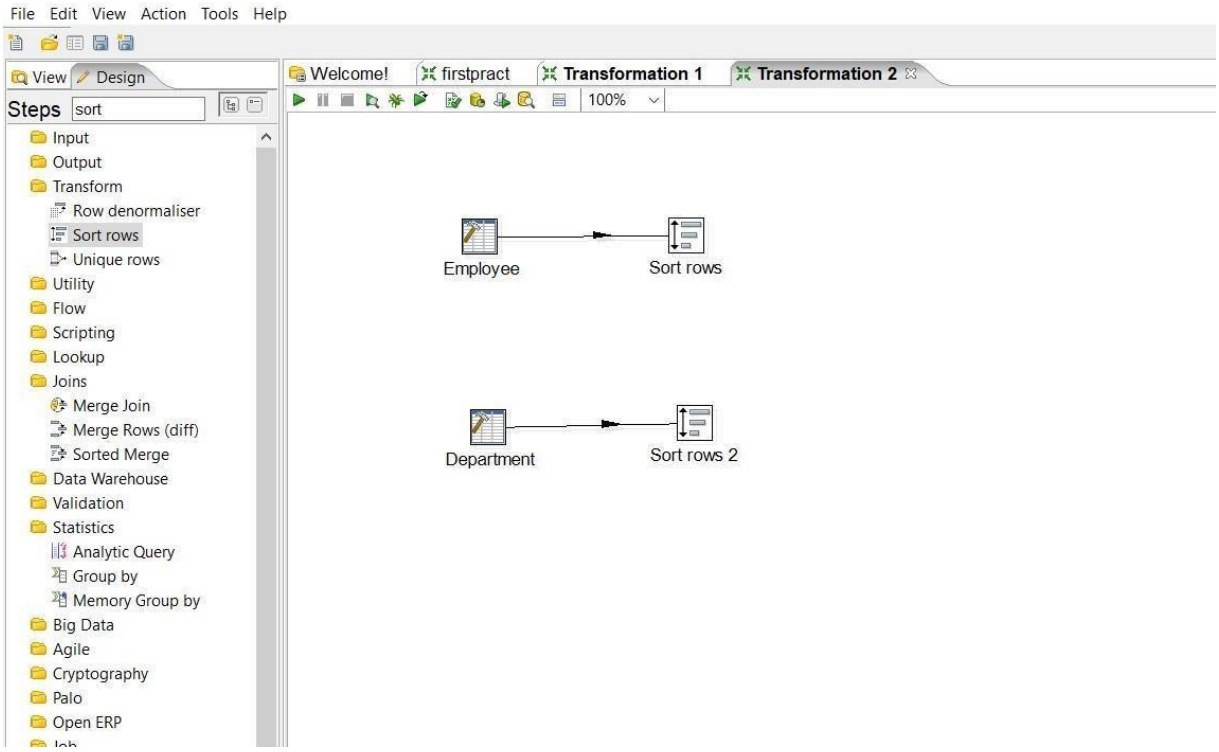
Step2: Insert Records into respective grids.



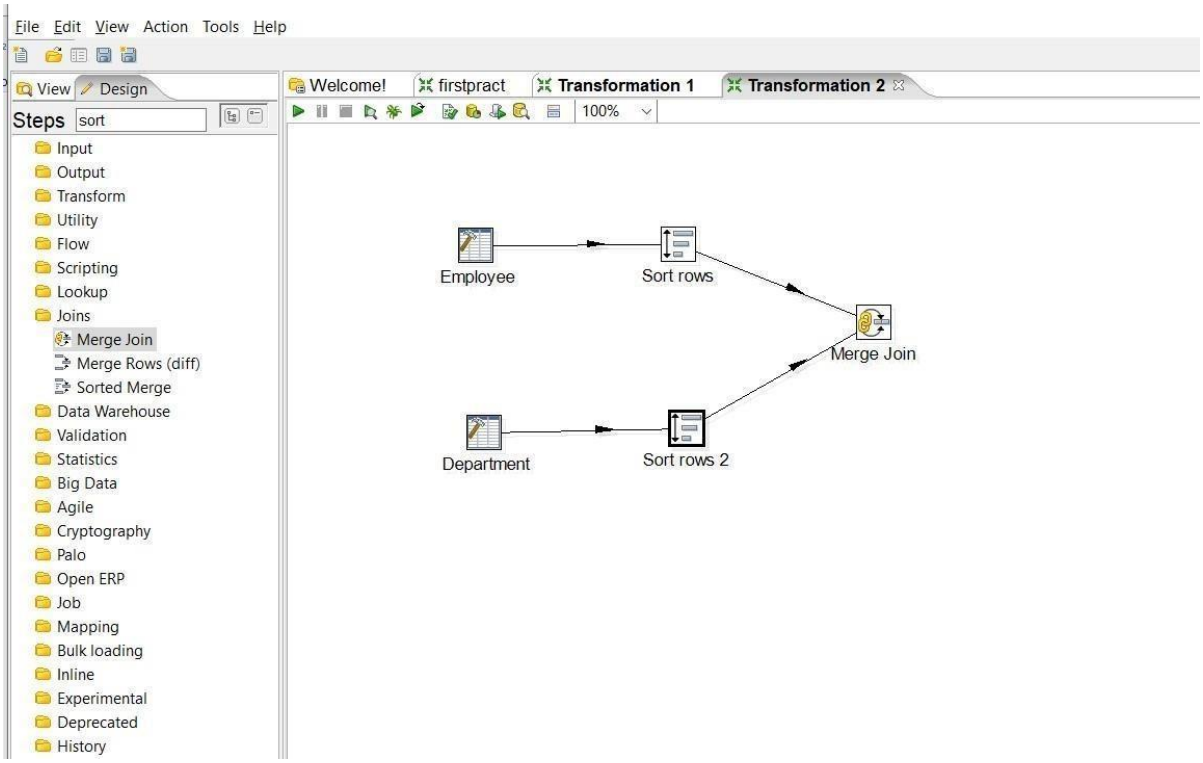
Step3: Insert sort rows transformation(on Empid) for both Employee and Department.


Sort rows
Step name
Sort rows
Sort directory
%%java.io.tmpdir%%
Browse...
TMP-file prefix
out
Sort size (rows in memory)
1000000
Free memory threshold (in %)
Compress TMP Files?
Only pass unique rows? (verifies keys only)
Fields :
Fieldname Ascending Case sensitive compare? Presorted?
1 Depno Y N N
Help
OK
Cancel
Get Fields


Sort rows
Step name
Sort rows 2
Sort directory
%%java.io.tmpdir%%
Browse...
TMP-file prefix
out
Sort size (rows in memory)
1000000
Free memory threshold (in %)
Compress TMP Files?
Only pass unique rows? (verifies keys only)
Fields :
Fieldname Ascending Case sensitive compare? Presorted?
1 Depno Y N N
Help
OK
Cancel
Get Fields

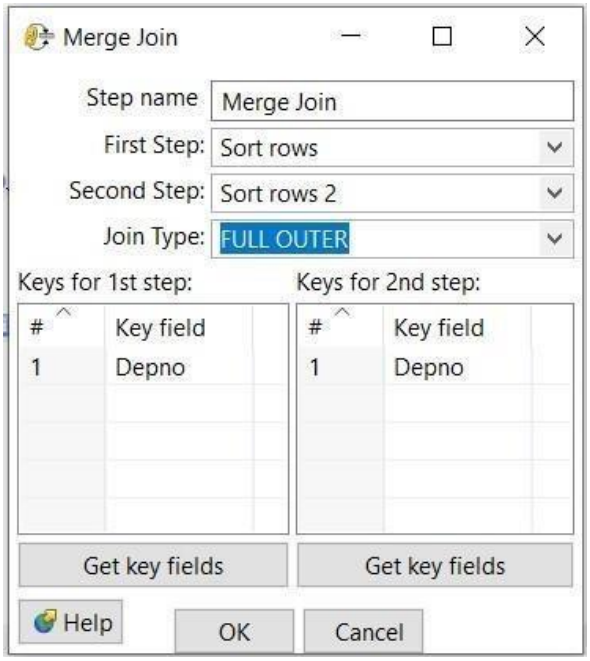


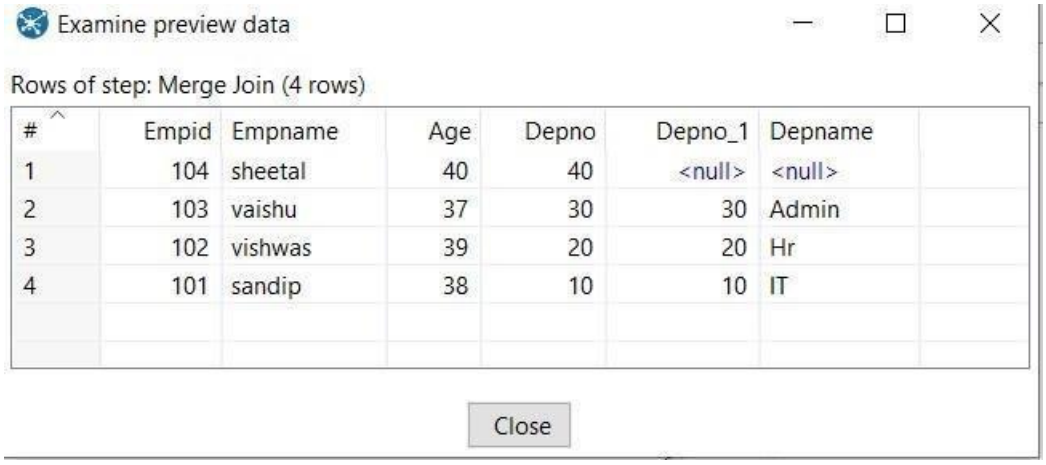
Step4: Add Merge join From Joins and connect it with sort rows.



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.

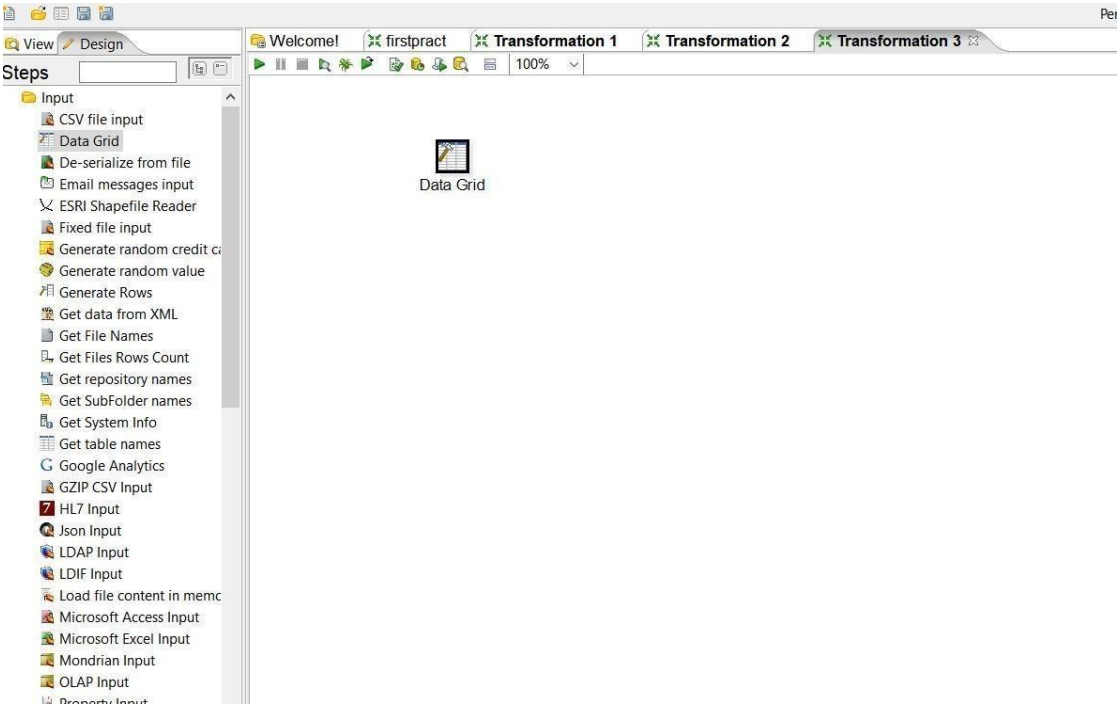
Step 6: Select Join Type Left Outer, then Right Outer and finally Full Outer join



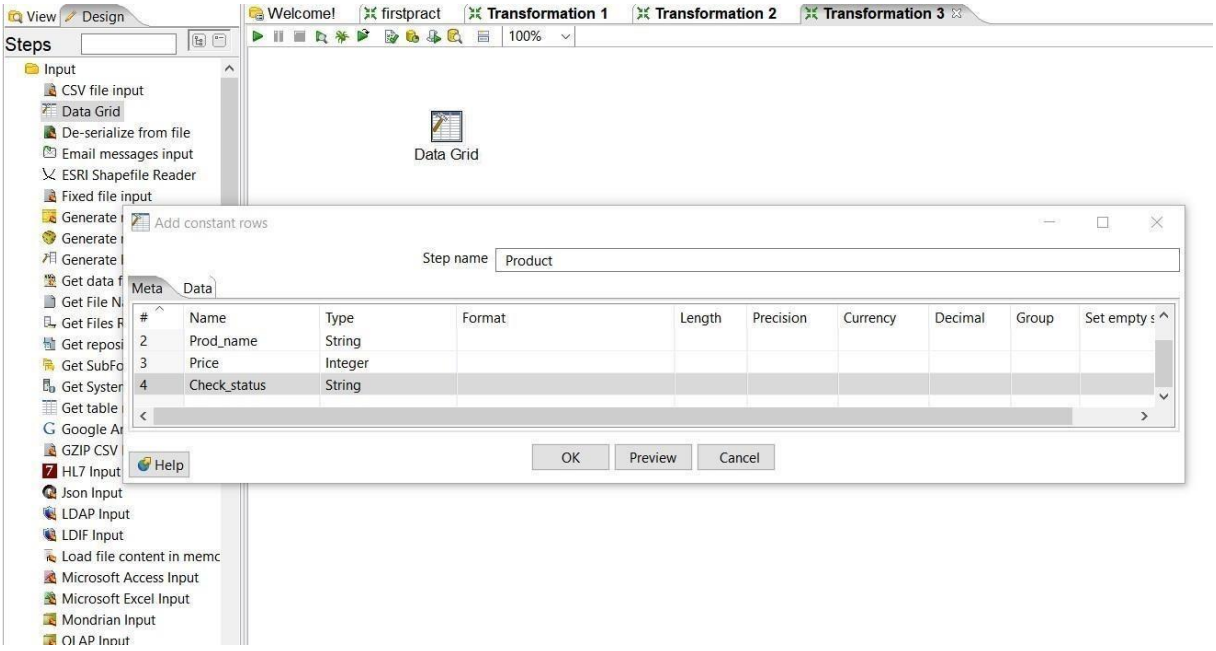


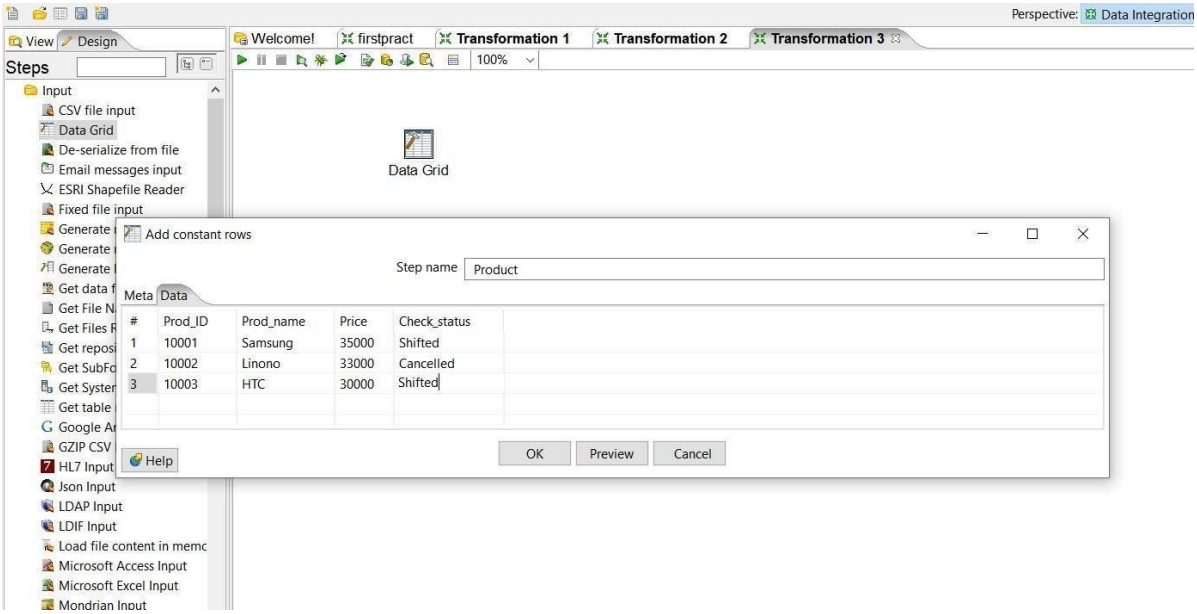
- **Implement different data validations on table**

Step1: Drag and drop Data grid on panel.

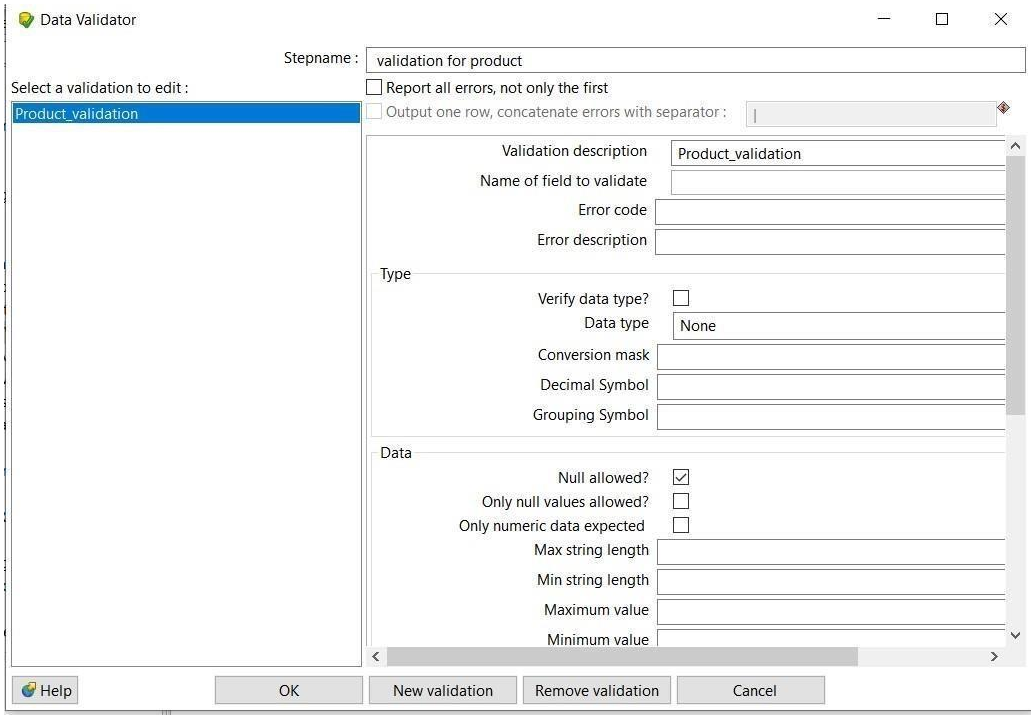


Step2: Double click on Data grid and create product table in Meta tab and enter records into it using Data tab.

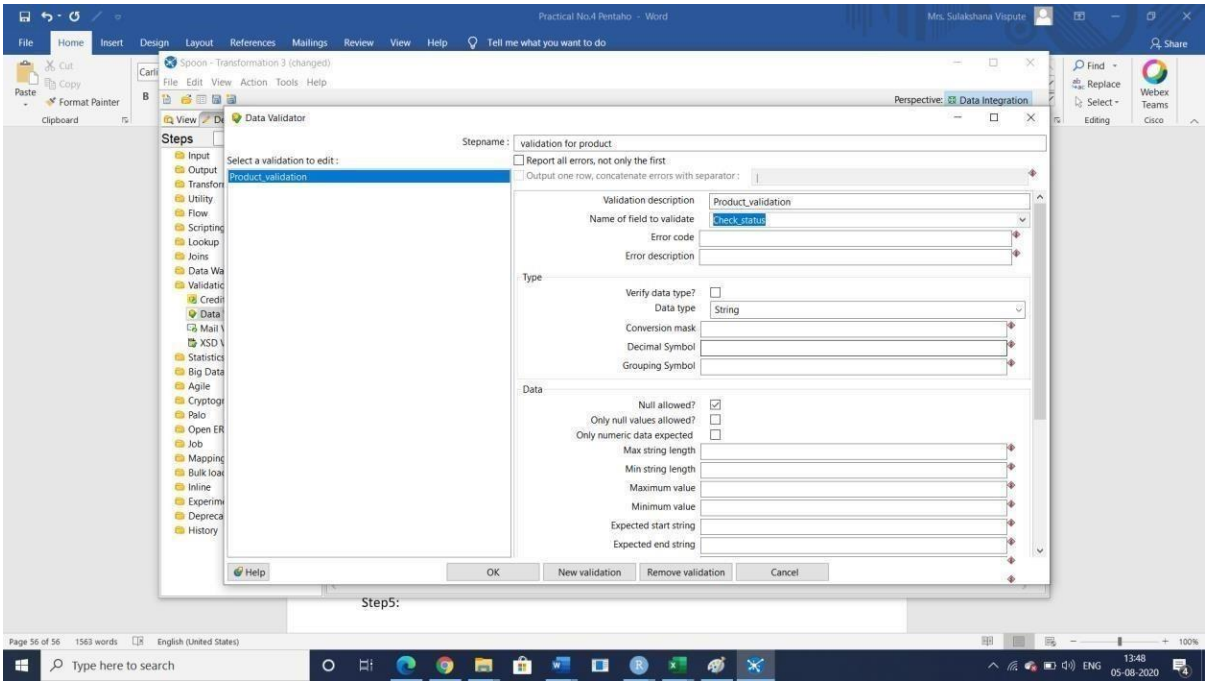




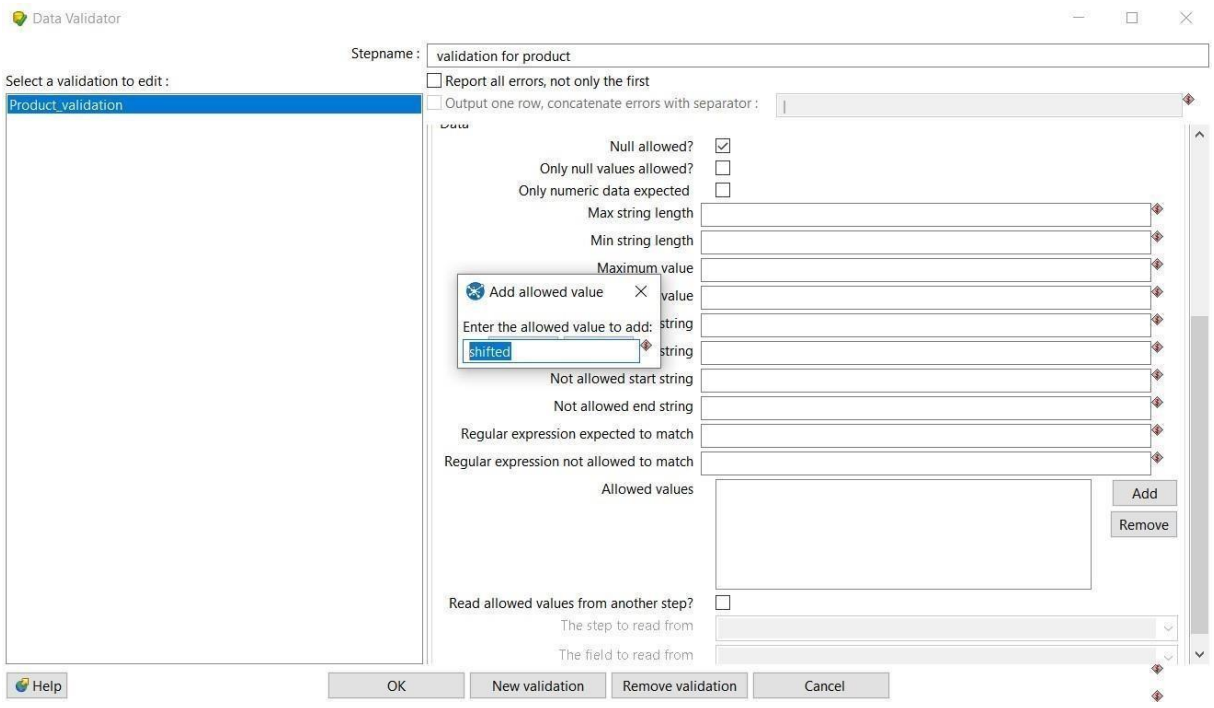
Step3: Go to Validation steps and select, Drag and Drop Data validator on panel and double click on it.



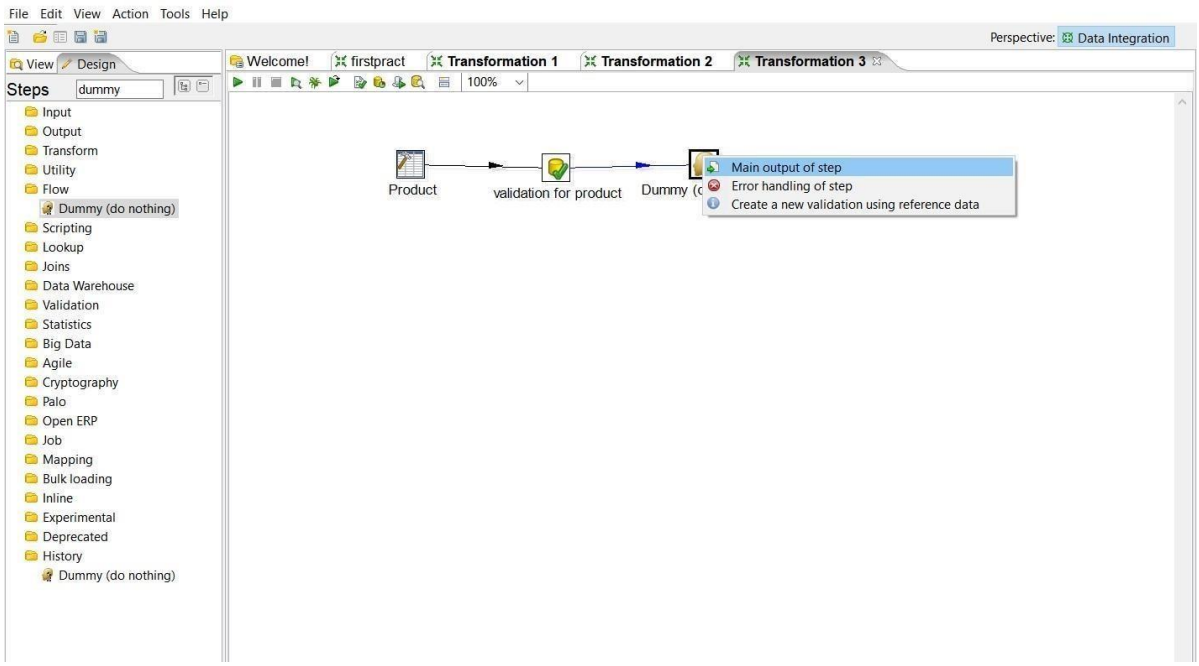
Step5: Set values for Name of field to validate=check status, and Data type =string



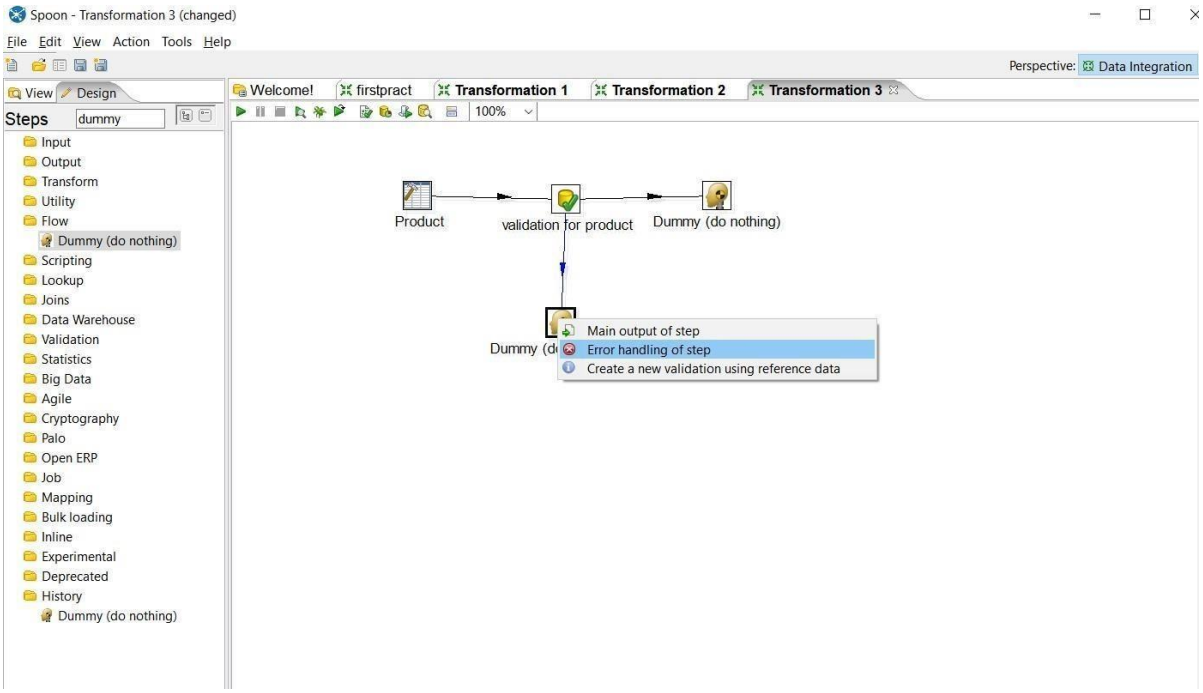
Step6: Click on add to set validation, set it 65 to Shifted and press Enter and click on ok.



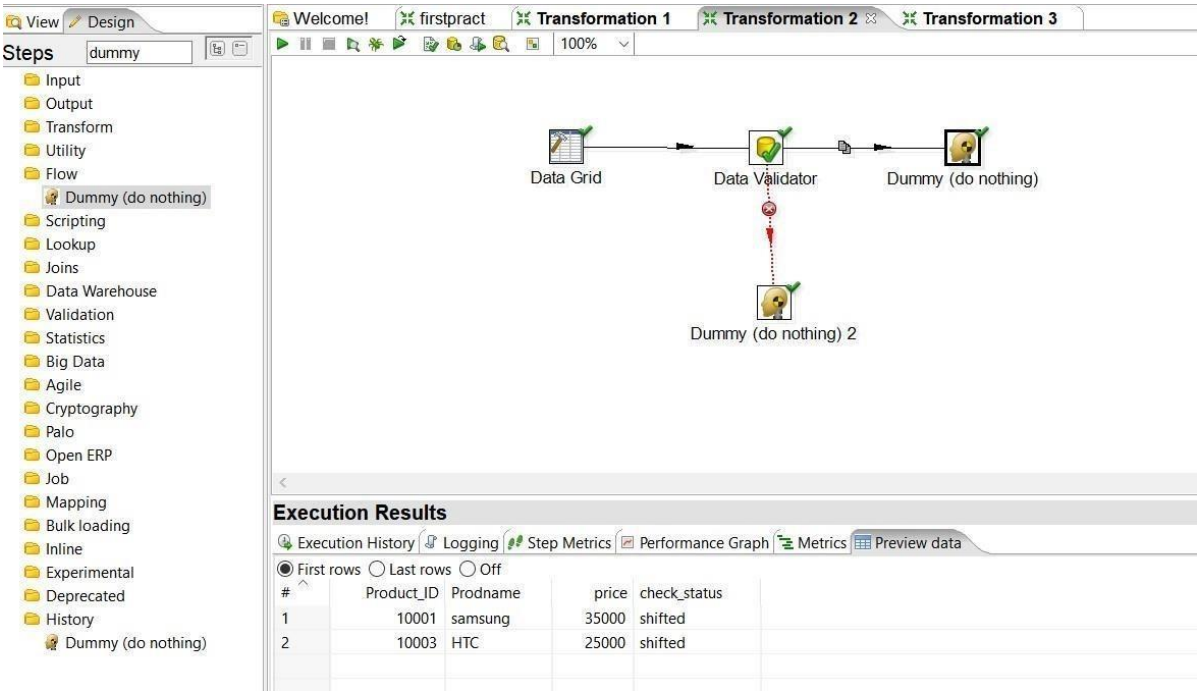
Step7: Select Dummy file from Flow to hold output and connect it with Data validation, while connecting select option " Main Output of step"

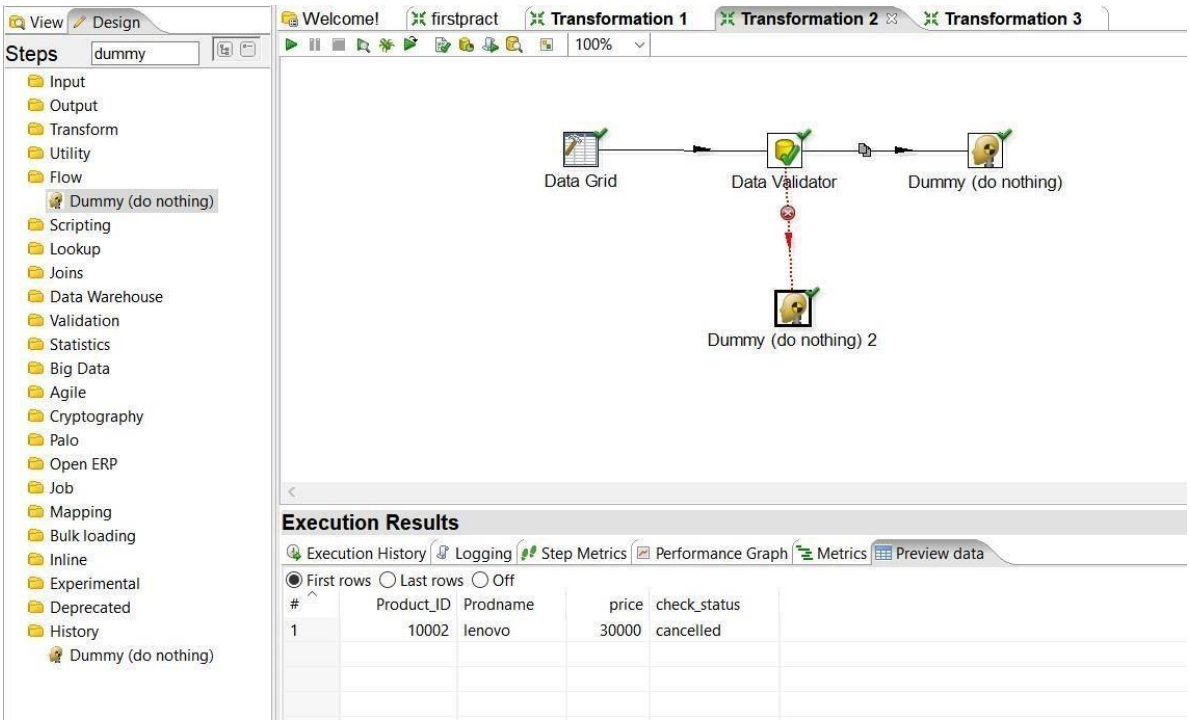


Step8: Select another dummy and connect it to 'Validation for product' and while selecting select 'Error handling of step and in next window click on copy



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





5. Introduction to R programming and Data acquisition

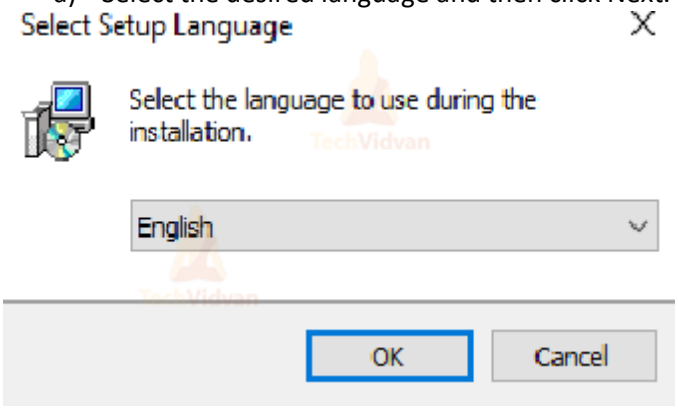
- **Install packages , Loading packages**

How To Download R in Windows

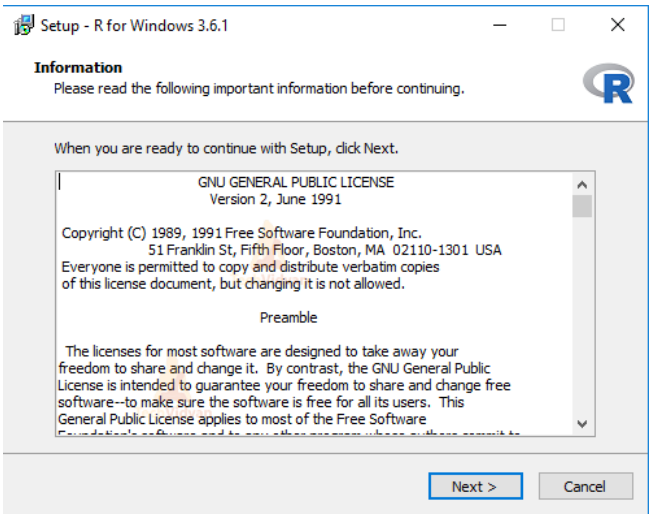
1. Go to CRAN R project website.
1. Click on the Download R for Windows link.
2. Click on the base subdirectory link or install R for the first time link.
3. Click Download R X.X.X for Windows (X.X.X stand for the latest version of R. eg: 3.6.1) and save the executable .exe file.

5. Run the .exe file and follow the installation instructions.

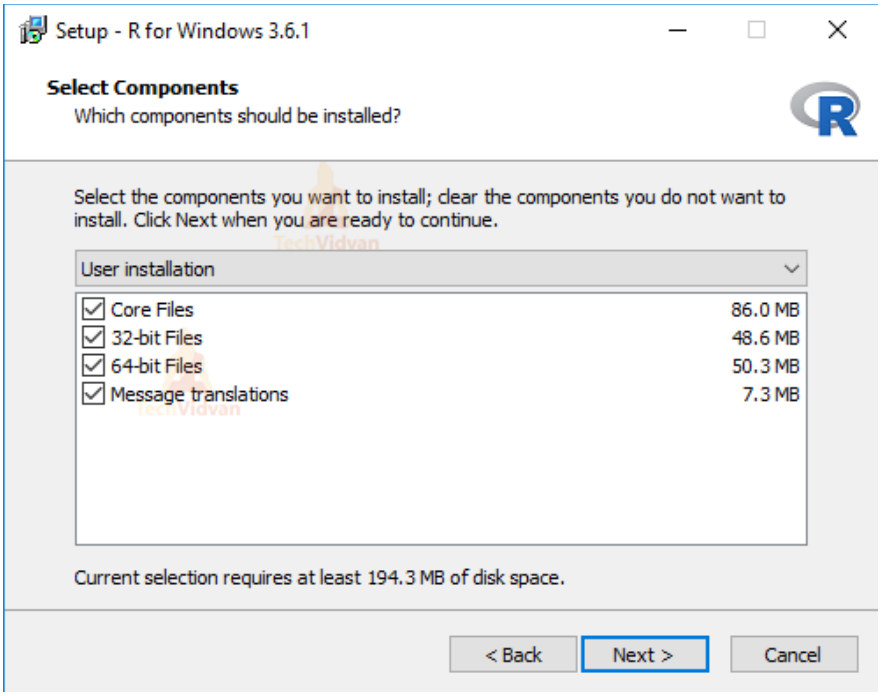
a) Select the desired language and then click Next.



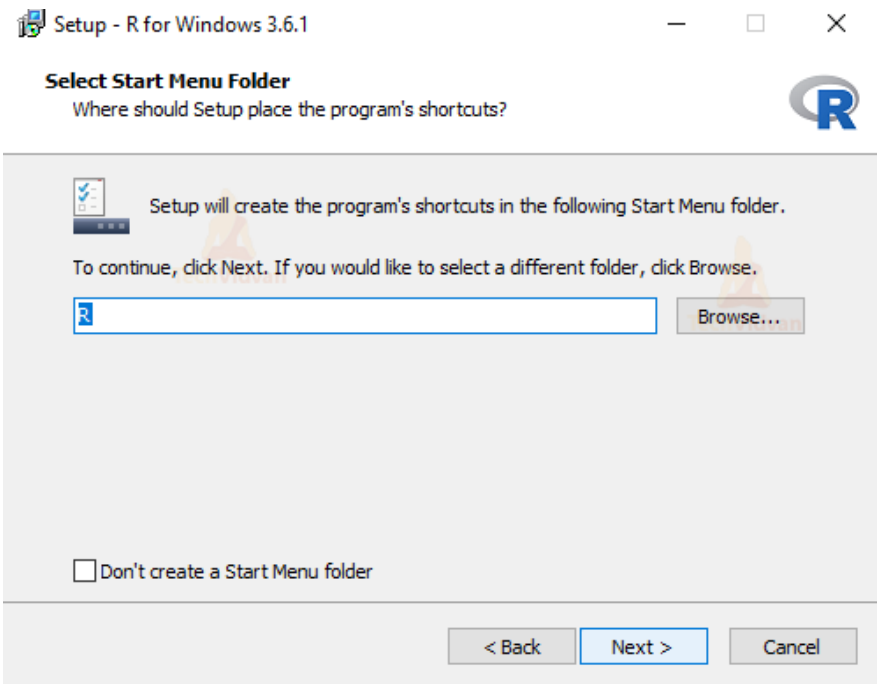
b) Read the license agreement and click Next.



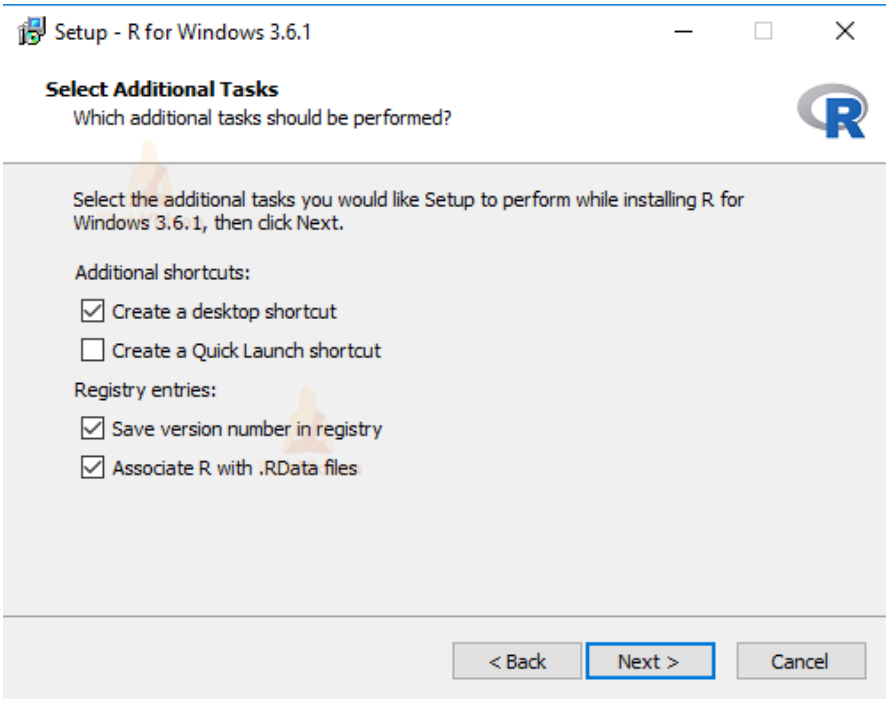
a) Select the components you wish to install (it is recommended to install all the components). Click Next.



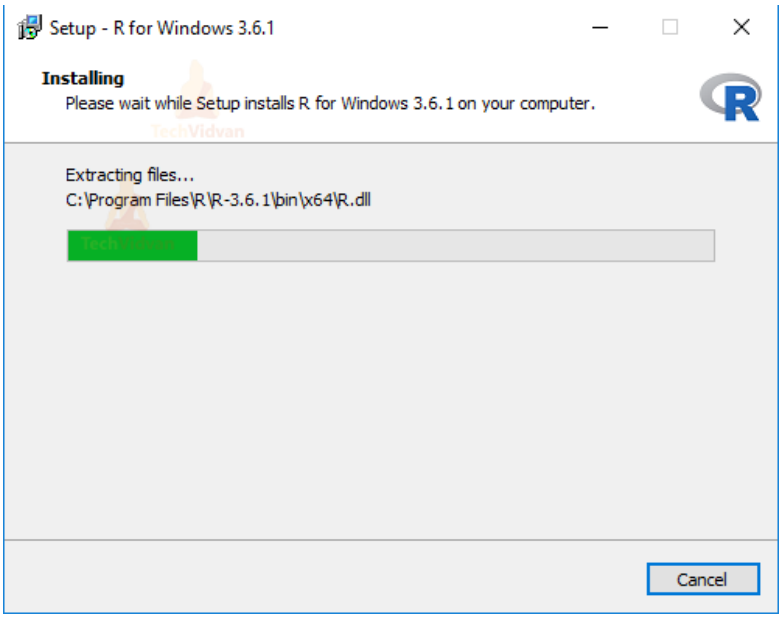
b) Enter/browse the folder/path you wish to install R into and then confirm by clicking Next.



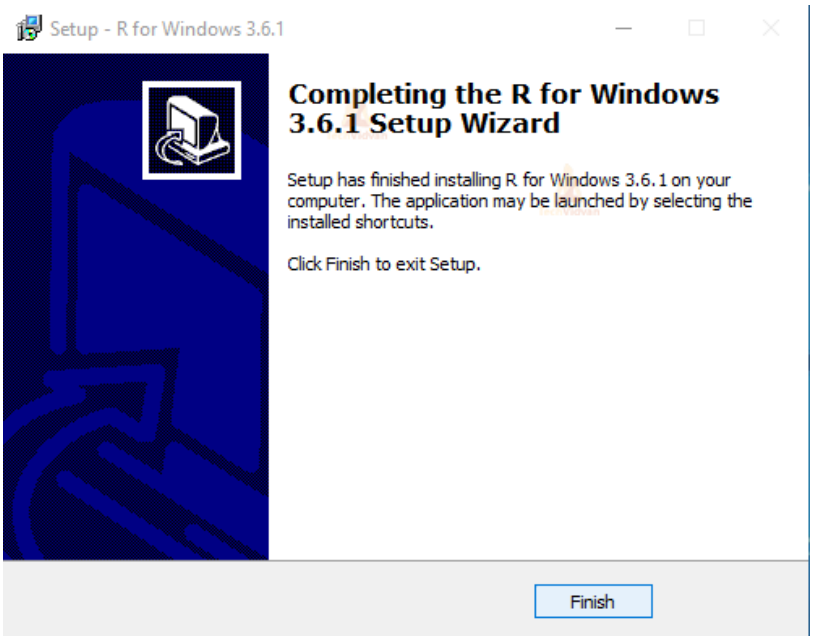
c) Select additional tasks like creating desktop shortcuts etc. then click Next.



a) Wait for the installation process to complete.



b) Click on Finish to complete the installation.



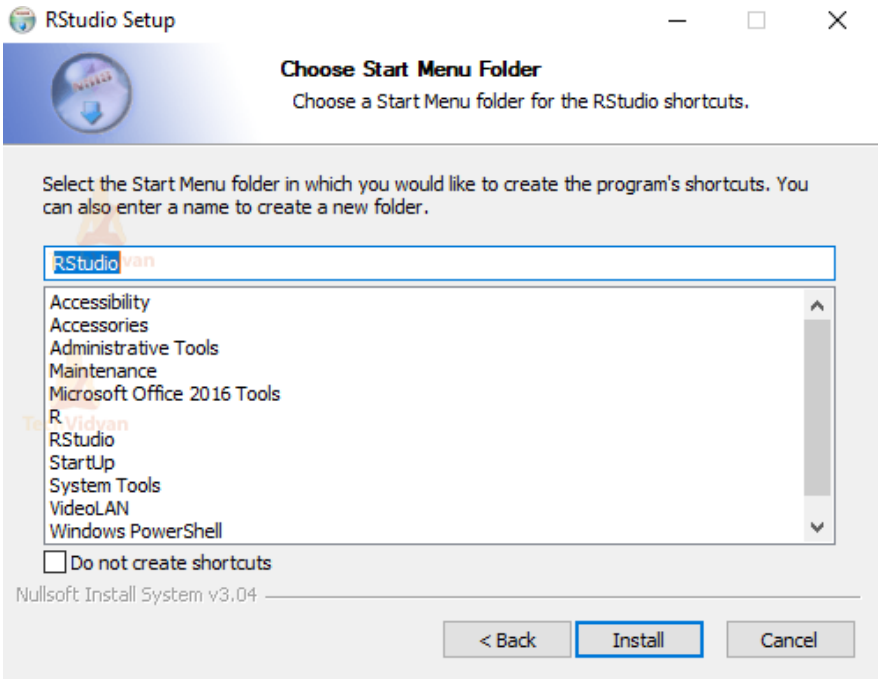
How to download R studio in windows

I. With R-base installed, now installing RStudio editor. To begin, go to download RStudio and click on the download button for RStudio desktop.

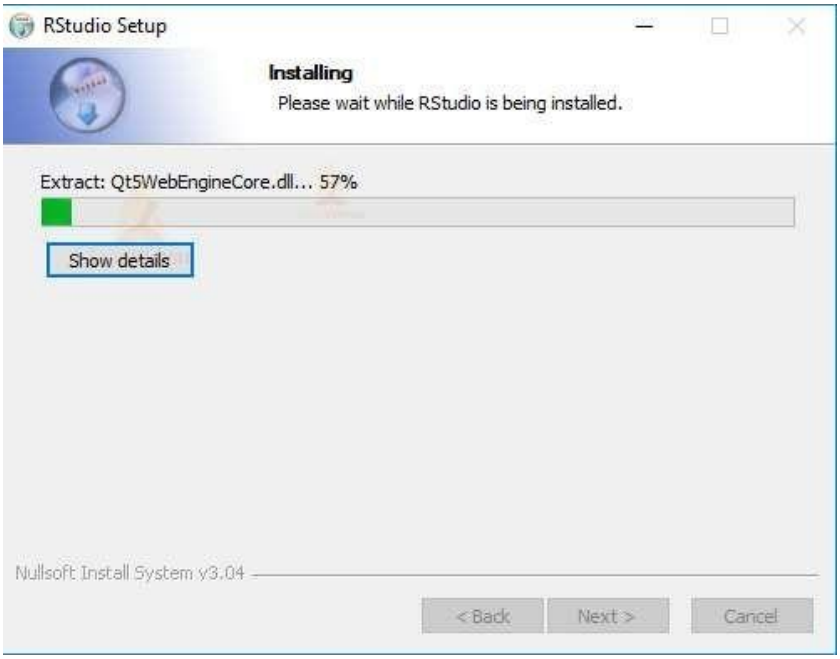
II. Click on the link for the windows version of RStudio and save the .exe file.

III. Run the .exe and follow the installation instructions.

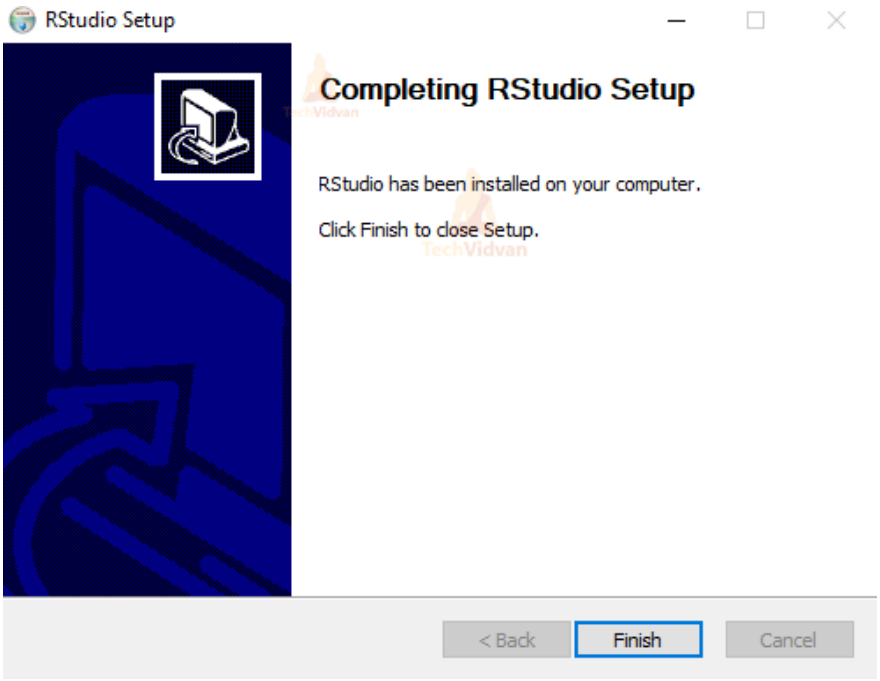
- Click Next on the welcome window.



- Wait for the installation process to complete.



- Click Finish to end the installation.



- **Data types, checking type of variable, printing variable and objects**

```
> setwd("D:/r programming/RFiles")
> x=1
> print(x)
[1] 1
> class(x)
[1] "numeric"
> y = "c"
> print(y)
```

```
[1] "c"
> is.character(y)
[1] TRUE
> is.integer(y)
[1] FALSE
```

- **Vector, Matrix, List, Factor, Data frame, Table**

```
> # Here is vector containing three numeric values 2,5 and 9:
> c(2,5,9)
[1] 2 5 9
> # No of members in a vector is given by length Function
> length(c("aa","bb","cc","dd","ee"))
[1] 5
> length(c("aa","bb","cc","dd"))
[1] 4
> # Vector Operations
> x = c(11.3,12.5,31.2)
> y=c(9,1,4)
> x*y
[1] 101.7  12.5 124.8
> x+y
[1] 20.3 13.5 35.2
> x-y
[1]  2.3 11.5 27.2
> x/y
[1]  1.255556 12.500000  7.800000
> # using matrix() function
> m = matrix(c(21,22,74,96,54,36,27,63,41),nrow=3,ncol=3)
```

```
> m
     [,1] [,2] [,3]
[1,]   21   96   27
[2,]   22   54   63
[3,]   74   36   41
> dim(m)
[1] 3 3
> m = matrix(c(21,22,74,96,54,36,27,63,41),nrow=3,ncol=3 ,byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]   21   22   74
[2,]   96   54   36
[3,]   27   63   41
> # cbind(column bind) and rbind(row bind)
> x = c(10,21,32)
> y = c(18,12,23)
> cbind(x,y)
      x  y
[1,] 10 18
[2,] 21 12
[3,] 32 23
> rbind(x,y)
  [,1] [,2] [,3]
x   10   21   32
y   18   12   23
> # matrix multiplication
> p = 3*m
> p
```

```
     [,1] [,2] [,3]
[1,]   63   66  222
[2,]  288  162  108
[3,]   81  189  123
> n = matrix(c(74,55,98,84,36,2,15,20,94),nrow = 3,ncol = 3)
> q = m+n
> q
     [,1] [,2] [,3]
[1,]   95  106   89
[2,]  151   90   56
[3,]  125   65  135
> mdash = t(m)
> mdash
     [,1] [,2] [,3]
[1,]   21   96   27
[2,]   22   54   63
[3,]   74   36   41
>  #DETERMINANT
> s = matrix(c(12,25,65,74,98,36,88,54,35),nrow=3,ncol=3,byrow = TRUE)
> s_det = det(s)
> s_det
[1] -268538
> r_det = det(m)
> r_det
[1] 273318
> # Dataframe used to store tabular data. Can contain different classes
> student_id = c(41,42,43)
> student_names = c("Gaurav","Kundan","Mayuresh")
```

```
> position = c("First","Second","Third")
> data = data.frame(student_id,student_names,position)
> data
  student_id student_names position
1         41        Gaurav    First
2         42        Kundan   Second
3         43      Mayuresh    Third
> data$student_id
[1] 41 42 43
> names(data)
[1] "student_id"    "student_names" "position"
> # Table command is used to create a 2dimensional table in R
> smoke = matrix(c(9,8,7,5,6,4,2,1,3),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    9   8      7
former     5   6      4
never      2   1      3
```

G. **Data Frames in R: It is used to store tabular data. Can contain different classes.**

```
> emp_no=c(1,2,3)
> emp_name=c('Diksha','Rani','Danish')
> emp_sal=c(2300,7800,8900)
>    data_emp=data.frame(emp_no,emp_name,emp_sal)//    using
     data.frame function
> print(data_emp)
```

```
 emp_no
emp_name
emp_sal 1   1
      Diksha 2300
2     2    Rani    7800
3     3 Danish   8900

// accessing the specific element using $
> data_emp$emp_name
[1] "Diksha" "Rani" "Danish"
> nrow(data_emp) // total number of data present in row
[1] 3
> ncol(data_emp)// total number of data present in column
[1] 3
> names(data_emp) // displaying the headings.
[1] "emp_no" "emp_name" "emp_sal"
>   smoke=matrix(c(34,45,78,89,67,70,13,45,67),nrow=3,ncol=3,byrow
=TRUE) // Creating 3x3 matrix called smoke
> colnames(smoke)=c('High','Medium','Low') // Using colnames giving
the column name.
> rownames(smoke)=c('Summer','Rainy','Winter')// Giving the row names.
> print(smoke)
           High
Medium Low
Summer   34      45       78
Rainy       89    67       70


Winter     13    45        67
```

H. Installing the packages and Read / Write the file.

```
install.packages(ConnectXL)
```

install.packag

es(readxl)

install.packag

es(writexl)

To check whether the packages installed

properly or not. library(ConnectXL)

libr

ary(

rea

dxl)

libr

ary(

writ

exl)

How to read the file from

directory? Answer:

> data_E=read.table("Example1.csv",sep=",",header=T)
> print(data_E)

```
 Emp_no
Emp_name
Salary 1 101
Diksha                  10000
2   102   soham        30000
3   103   Tanmay       40000
4   104   Gauri         67000
5   105   Netra         45000
> d
im(
dat
a_
E)
[1]
5 3
> head(data_E,3)
 Emp_no
Emp_name
Salary 1   101
Diksha                10000
2   102 soham       30000
3   103   Tanmay    40000
> tail(data_E,2)
 Emp_no
Emp_name
Salary 4   104          Gauri   67000
5   105       Netra          45000
```

How to write file in givendirectory? Answer:

```
>z=data.frame(a=12,b=45,pi=3.14
2)print(z)

  a  b   pi
1 12 45 3.142
>write.csv(z,file="data.csv")
```

The file is present in given directory. When we open the file called as data.csv we get our data.

- **Using ConnectXL**

Reading and writing data from Excel using XLConnect

```
>data_S=XLConnect:: readWorksheetFromFile("studentexcel.xlsx",sheet=1)
> print(data_S)
  Roll_no StudentName Marks
1     101      Danish     90
2     102      Sameer     56
3     103      Diksha     98
4     104     Florina     55
5     105     Kamlesh     67
```

6. Implementation of Data preprocessing techniques

- **Naming and Renaming variables, adding a new variable.**

Dealing with missing data.

- **Dealing with categorical data.**

Data reduction using subsetting

Ans:

Step 1: creating csv file (dataset .csv)

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

Step 2: Importing the dataset.

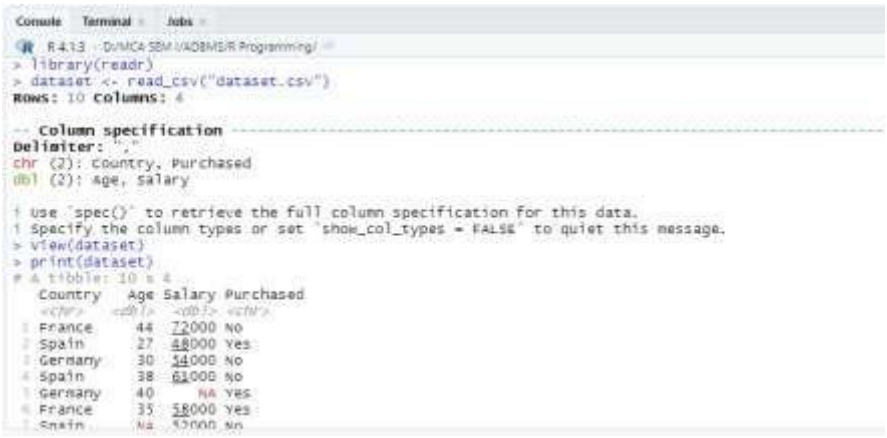
dataset <- read_csv("dataset.csv")

#This code imports our data stored

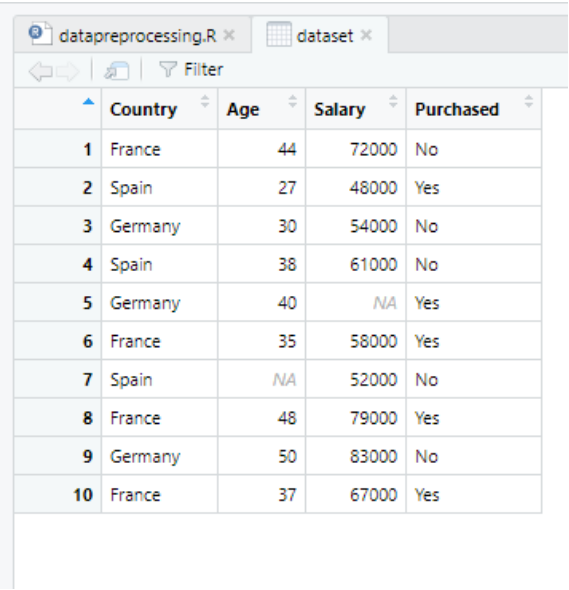
in CSV format. View(dataset)

Looking at our data using the 'view()' function. Upon executing we

obtain our dataset. print(dataset)



Output:



datapreprocessing.R × dataset ×

Filter

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

Our dataset has four columns and ten observations, it shows how customers from three different countries with different ages and salaries responded to the purchase of a certain product.

Step 2: Handling the missing data.

From the dataset, the Age and Salary column report missing data. Before implementing our machine learning models, this problem needs to be solved, otherwise it will cause a serious problem to our machine learning models. Therefore, it's our responsibility to ensure this missing data is eliminated from our dataset using the most appropriate technique.

The code below carries out such a task.

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

What does the code above really do?

Dataset$Age: Simply take the Age column from our dataset.

In the Age column, we've just taken that from our data set, we need to replace the missing data, and at the same time keep the data that is not missing.

Our condition is is.na(Dataset$Age). This will tell us if a value in the Dataset$Age is missing or not. It returns a logical output, YES if a value is missing and NO if a value is not missing. The second parameter, the 'ave()' function, finds the mean of the Age column.

Because this column reports NA values, we need to exclude the null data in the calculation of the mean, otherwise we shall obtain the mean as NA. This is the reason we pass na.rm = TRUE in our mean function just as to declare those values that should be used and those should be excluded when calculating the mean of the vector Age.

The third condition is the value that will be returned if the value in the Age column of the dataset is not missing.

Executing the code we obtain:

```
print(dataset$Age)
 [1] 44.00000 27.00000 30.00000 38.00000 40.00000 35.00000 38.77778 48.00000
50.00000 37.00000
```

The missing value that was in the Age column of our data set has successfully been replaced with the mean of the same column.

We do the same for the Salary column by executing the code below:

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

print(dataset$Salary)
 [1] 72000.00 48000.00 54000.00 61000.00 63777.78 58000.00 52000.00 79000.00
```

83000.00 67000.00

The missing value that was in the Salary column was successfully replaced with the mean of the same column.

Step 3: Encoding categorical data

Encoding refers to transforming text data into numeric data. Encoding Categorical data simply means we are transforming data that fall into categories into numeric data.

In our dataset, the Country column is Categorical data with 3 levels i.e. France, Spain, and Germany. The purchased column is Categorical data as well with 2 categories, i.e. YES and NO.

The machine models we built on our dataset are based on mathematical equations and it's only take numbers in those equations.

Keeping texts of a categorical variable in the equation can cause some troubles to the machine learning models and this why we encode those variables. To transform a categorical variable into numeric, we use the factor() function.

Let start by encoding the Country column.

```
dataset$Country = factor(dataset$Country,
                         levels =
                         c('France','Spain','German
                         y'), labels = c(1.0, 2.0 , 3.0
                         ))
print(dataset$Country)
```

Output: Country names were successfully replaced

with numbers. print(dataset$Country)

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

Levels: 1 2 3

Same for

purchased column

as.factor(dataset$

Purchased) [1] 0 1

0 0 1 1 0 1 0 1

Levels: 0 1

> print(dataset$

Purchased) [1]

0 1 0 0 1 1 0 1 0

1

Levels: 0 1

> prin

t(data

set) #

A

tibble

: 10 x

4

Country Age Salary Purchased

<fc <dbl> <dbl>
t> <fct>

1 44 72000 0

1

2 27 48000 1

2

3 30 54000 0

3

4 38 61000 0

2

5 3 40 63778. 1

6 1 35 58000 1

7 2 38.8 52000 0

8 1 48 79000 1

9 3 50 83000 0

10 1 37 67000 1

Purchased column was successfully encoded into 0,s, and 1,s.

Step 4: Splitting the dataset into the training and test set.

Training set: The part of the data that we implement our machine learning model on. Test set: The part of the data that we evaluate the performance of our machine learning model on.

The reason we split this data is to ensure that our machine learning model does not overlearn the correlation of data it's trained on. If we let it learn too much on the data, it may perform poorly when tested on a new dataset with a different correlation.

Therefore, whenever we are building a machine learning model, the idea is to implement it on the training set and evaluate it on the test set. We expect the performance in the training set and test set to be different and if this is the case the model can adapt to new datasets.

print(dataset)

library(caTools)# required library for

data splition set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.8)# returns true if observation goes to the Training set and false if observation goes to the test set.

```
#Creating the training set and test set
separately training_set =
subset(dataset, split == TRUE)
test_set = subset(dataset, split ==
FALSE) training_set
test_set
```

Executing the code yields:

```
> training_set


# A tibble: 8 x 4
  Country Age Salary Purchased
  <fct> <dbl> <dbl> <fct>
1 1     44    72000 0
2 2     27    48000 1
3 3     30    54000 0
4 2     38    61000 0
5 3     40    63778 1
6 2     38.8  52000 0
7 1     48    79000 1
8 1     37    67000 1
```

From the results it clear that eight observations, 0.8 of our dataset observations, were split into the training set.

```
> test_set
# A tibble: 2 x 4
```

Country Age Salary Purchased

<fct> <dbl> <dbl> <fct>

1 1 35 58000
1
2 3 50 8300
0 0

From the output it clear that two observations went to the test set.

Step 5: Feature Scaling

Scale both the training set and test set of our dataset

separately. training_set[, 2:3] = scale(training_set[,

2:3])

test_set[, 2:3] =

scale(test_set[, 2:3])

training_set

test_set

Executing the code:

> tra

inin

g_se

t # A

tibbl

e: 8

x 4

Country Age Salary Purchased

<fct> <dbl> <dbl> <fct>

1 0.901 0.939 0
1
2 -1.59 -1.34 :
2

```
3       -1.15 -0.768        (
3
4        0.0224 -           (
2        0.104
5        0.315 0.159        1
3
6        0.136 -0.958       (
2
7        1.49 1.60          1
1
8       -0.124 0.465        1
1
```

```
> test_set
# A tibble: 2 x 4
  Country Age Salary Purchased
  <fct>   <dbl> <dbl> <fct>
1 -0.707 -            1
1 0.707
2  0.707 0.707        (
3
```

Data Preprocessing Code:

```
library(readr)
dataset <-
read_csv("dataset.csv")
View(dataset)
print(dataset)


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


dataset$Salary = ifelse(is.na(dataset$Salary),
```

```
                ave(dataset$Salary, FUN = function (x)mean(x,
                na.rm = TRUE)), dataset$Salary)
print(dataset$Salary)
dataset$Country = factor(dataset$Country,
                levels = c('France','Spain','Germany'),


                labels = c(1.0, 2.0 , 3.0 ))

print(dataset$Country)
dataset$Purchased = factor(dataset$Purchased,
                levels =
                c('No',
                'Yes'),
                labels =
                c(0, 1))
dataset$Purchased[is.na(dataset$Purch
ased)] <- 0
as.factor(dataset$Purchased)
print(dataset$Purchased)
print(dataset)
library(caTools)# required library for
data splition set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.8)# returns true if
observation goes to the Training set and false if observation goes to the test set.

#Creating the training set and test set
separately training_set =
subset(dataset, split == TRUE)
```

```
test_set = subset(dataset, split ==
FALSE) training_set
test_set
training_set[, 2:3] = scale(training_set[, 2:3])
test_set[, 2:3] =
scale(test_set[, 2:3])
training_set
test_set
#training_set = scale(training_set)
```

7. Implementation and analysis of Classification algorithms like Naive Bayesian, K- Nearest Neighbor, ID3 , C4.5

Ans:

Naïve Bayes :

Naive Bayes is a Supervised Non-linear classification algorithm in R Programming. Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with strong(Naive) independence assumptions between the features or variables. The Naive Bayes algorithm is called "Naive" because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features.

Dataset: Iris dataset consists of 50 samples from each of 3 species of Iris(Iris setosa, Iris virginica, Iris versicolor) and a multivariate dataset introduced by British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems. Four features were measured from each sample i.e length and width of the sepals and petals and based on the

combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Performing Naive Bayes on Dataset:

Using Naive Bayes algorithm on the dataset which includes 11 persons and 6 variables or attributes.

```
> library(e1071)
> library("klaR")
```

Loading required package: MASS

```
> library("caret")
```

Loading required

package: ggplot2

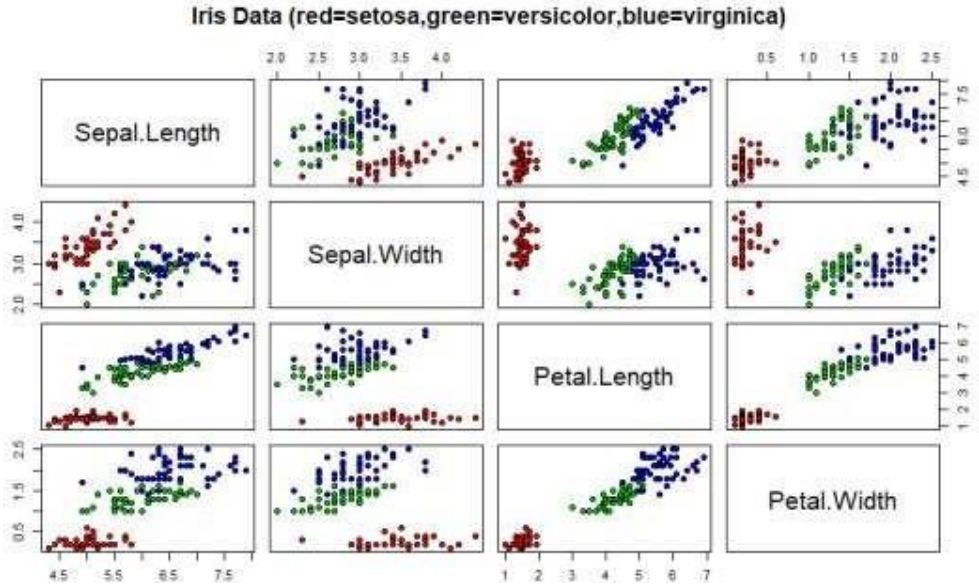
Loading required

package: lattice

```
> library(ggplot2)
> data(iris)
> head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

unique(iris$Species)



[1] setosa versicolor

virginica Levels:

setosa versicolor

virginica

Output:

The Conditional probability for each feature or variable is created by model separately. The apriori probabilities are also calculated which indicates the distribution of our data.

Naiv

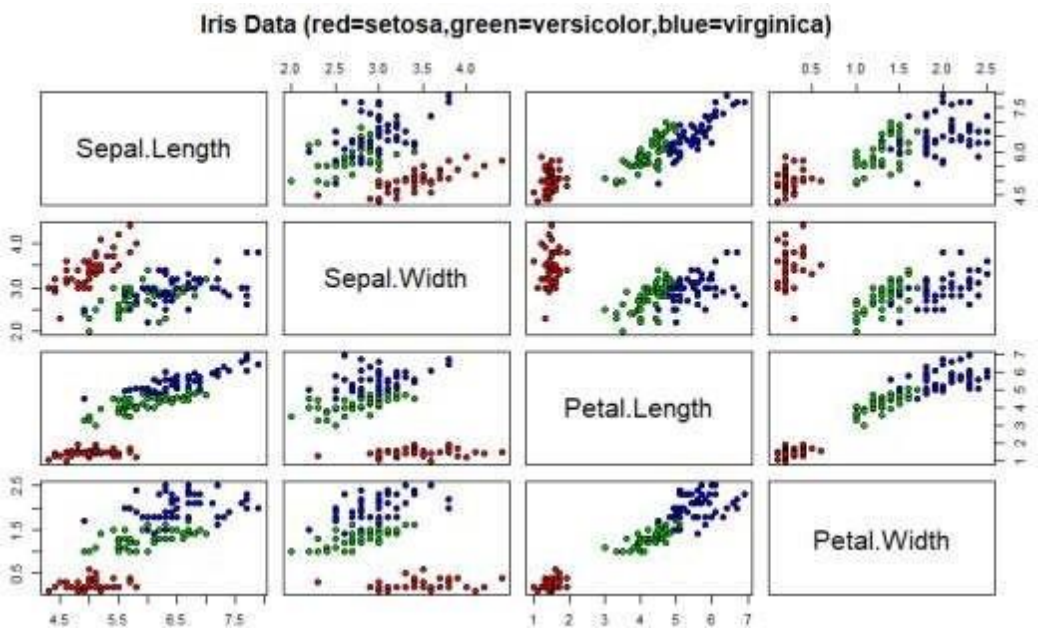
e

Baye

s

105

```
sam
ples
 4 predictor
 3 classes: 'setosa',
'versicolor', 'virginica' No pre-
processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 95, 94, 95, 96, 94, 95,
... Resampling results across tuning parameters:


 usekernel Accuracy
 Kappa FALSE
             0.9500000
 0.9262187
  TRUE     0.9309091 0.8977131

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 = FALSE and adjust = 1.
> prop.table(table(predict(model$finalModel,xTest)$c
        lass,yTest)) yTest
            setosa versicolor virginica
 setosa     0.37777778 0.00000000 0.00000000
 versicolor 0.00000000 0.28888889 0.06666667
 virginica 0.00000000 0.00000000 0.26666667
```



Confusion Matrix: So, 20 Setosa are correctly classified as Setosa. Out of 16 Versicolor, 15 Versicolor are correctly classified as Versicolor, and 1 are classified as virginica. Out of 24 virginica, 19 virginica are correctly classified as virginica and 5 are classified as Versicolor. **Model Evaluation:** The model achieved 90% accuracy with a p-value of less than 1. With Sensitivity, Specificity, and Balanced accuracy, the model build is good.

Code of naïve bayes:

```
libra
ry(e
1071
)
libra
ry("k
laR")
libra
ry("c
aret
```

```
")
libra
ry(gg
plot
2)
data
(iris)
head
(iris)
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))
#70/30 split. train = iris[index,]
test = iris[-index,]
xTrain = train[,-5] # removing y-
outcome variable. yTrain =
train$Species # only y.
xTest =
test[,-5]
yTest =
test$Spe
cies
model =
train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))
model
prop.table(table(predict(model$finalModel,xTest)$class,yTest))
```

K-Means Algorithm:

Ans: 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)

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


##extract
training set
iris_train <-
iris_norm[ran,]
##extract
testing set
iris_test <-
iris_norm[-ran,]
##extract 5th column of train dataset because it will be used as 'cl'
argument in knn function.


iris_target_category <- iris[ran,5]


##extract 5th column if test dataset to measure
the accuracy iris_test_category <- iris[-ran,5]
##load the package class library(class)
##run knn function
pr <- knn(iris_train,iris_test,cl=iris_target_category,k=13)


##create confusion matrix
tab <- table(pr,iris_test_category)
```

```
##this function divides the correct predictions by  total number of predictions
that tell us how accuracy.

accuracy <-
function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
[1] 100
```

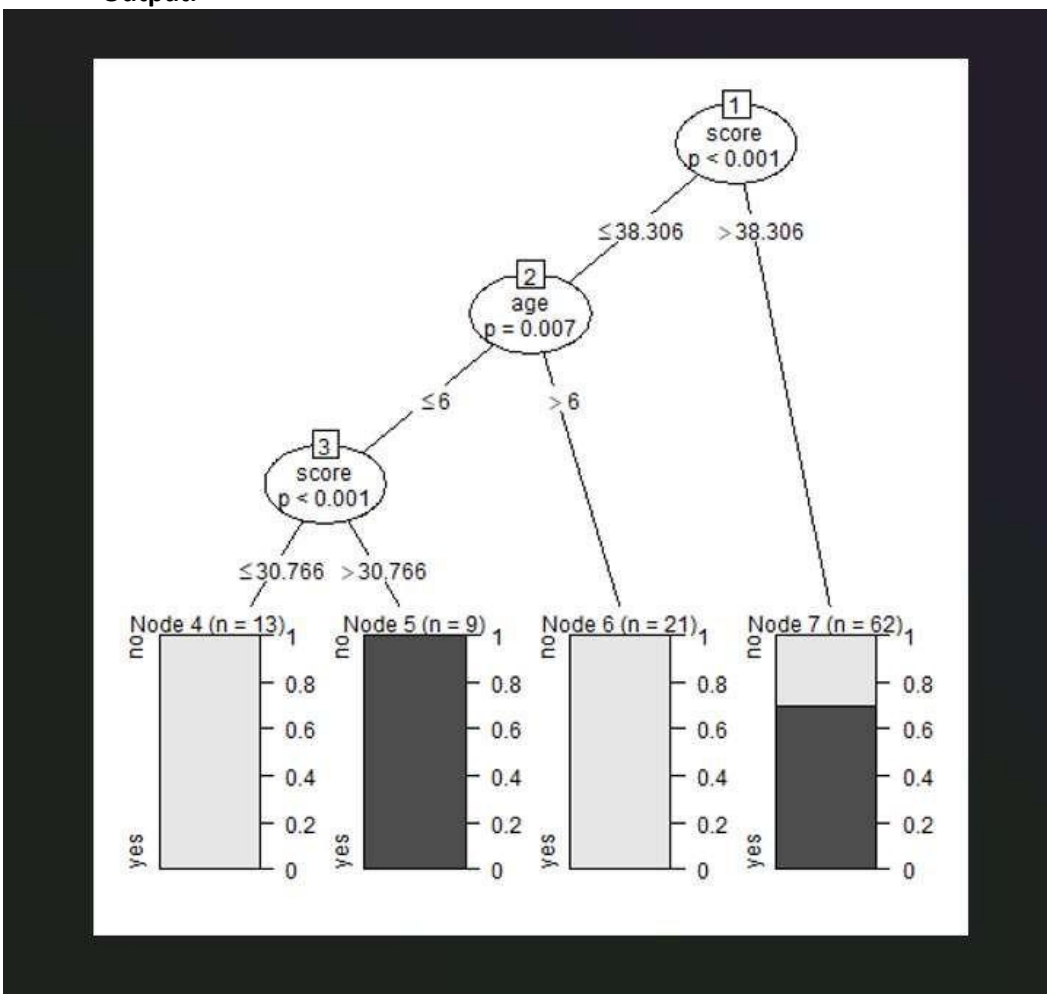
Decision Tree:

We will use the ctree() function to create the decision tree and see its graph. This code will create jpg image of decision tree in working directory in our example file decision_tree.png is created.

```
library(party)
# Create the input data
frame. input.dat <-
readingSkills[c(1:105),] #
Give the chart file a
name. png(file =
"decision_tree.png")
# Create the
tree.
output.tree <-
ctree(
  nativeSpeaker ~ age + shoeSize + score, data = input.dat)


# Plot
the
tree.
```

```
plot(o
utput.
tree)
# Save
the
file.
dev.of
f()
```

Output:



8. Implementation and analysis of Apriori Algorithm using

Market Basket Analysis.

Ans: apriorialgo.r

Step 1: Load required library.

'arules' package provides the infrastructure for representing, manipulating, and analyzing transaction data and patterns.

library(arules)

'arulesviz' package is used for visualizing Association Rules and Frequent Itemsets. It extends the package 'arules' with various visualization techniques for association rules and itemsets. The package also includes several interactive visualizations for rule exploration. library(arulesViz)

'RColorBrewer' is a ColorBrewer Palette which provides color schemes for maps and other graphics.

library(RColorBrewer)

Step 2: Import the dataset

'Groceries' dataset is predefined in the R package. It is a set of 9835 records/ transactions, each having 'n' number of items, which were bought together from the grocery store.

Data("Groceries")

Step 3: Applying apriori() function

'apriori()' function is in-built in R to mine frequent itemsets and association rules using the Apriori algorithm. Here, 'Groceries' is the transaction data. 'parameter' is a named list that specifies the minimum support and confidence for finding

the association rules. The default behavior is to mine the rules with minimum support of 0.1 and 0.8 as the minimum confidence. Here, we have specified the minimum support to be 0.01 and the minimum confidence to be 0.2.

```
# using
apriori()
function rules
<-
apriori(Groceri
es,
        parameter = list(supp = 0.01, conf = 0.2))
```

Step 4: Applying inspect() function

inspect() function prints the internal representation of an R object or the result of an expression. Here, it displays the first 10 strong association rules.

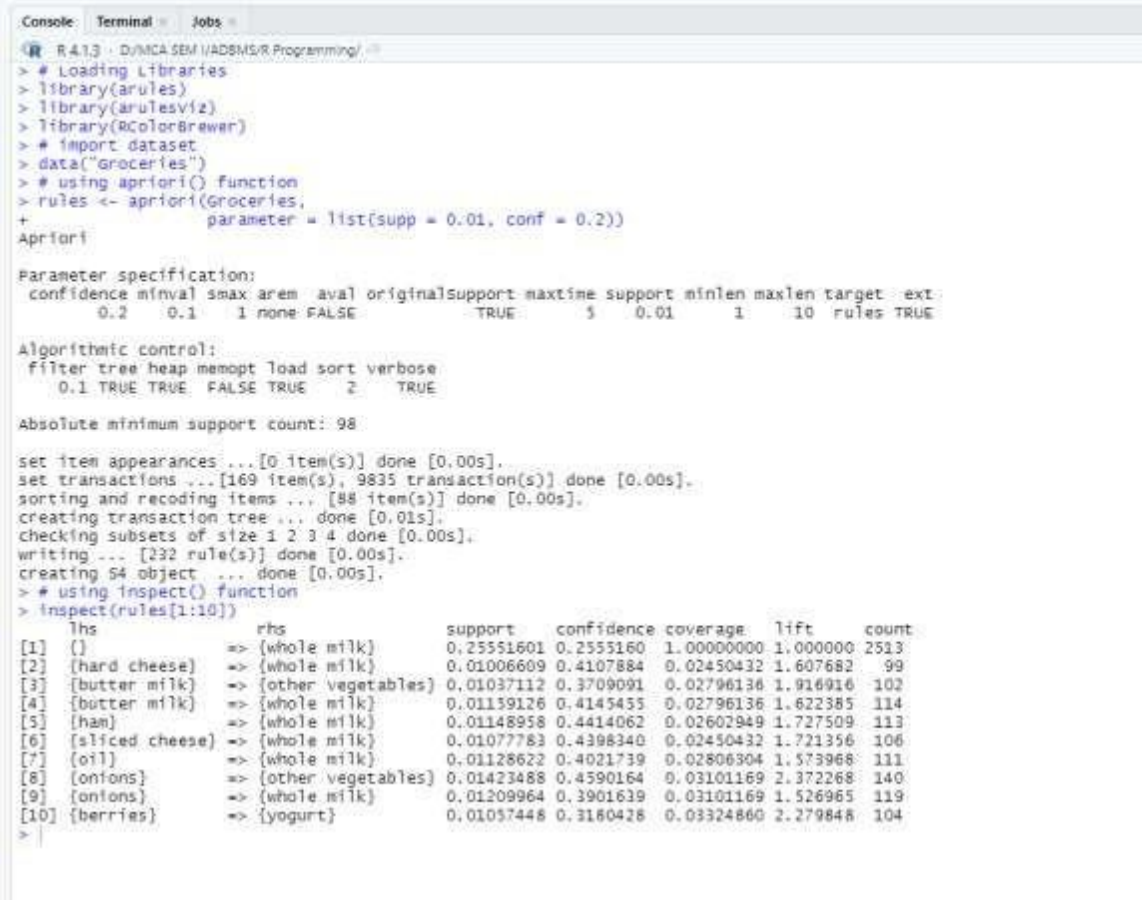
```
# using
inspect()
function
inspect(rules[
1:10]) Apriori
```

```
Parameter specification:
 confidence minval smax arem aval originalSupport maxtime support minlen
maxlen target ext
      0.2    0.1   1 none FALSE           TRUE       5    0.01      1     10 rules TRUE

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

```
Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)]
done [0.00s]. sorting and recoding items ... [88 item(s)]
done [0.00s].
creating transaction tree ... done
[0.01s]. checking subsets of size 1
2 3 4 done [0.00s]. writing ... [232
rule(s)] done [0.00s]. creating S4
object ... done [0.00s].

> # using inspect() function
> inspect(rules[1:10])
    lhs              rhs              support   confidence coverage lift       count
[1] {}            => {whole milk}      0.25551601 0.2555160 1.00000000 1.000000 2513
[2] {hard cheese} => {whole milk}      0.01006609 0.4107884 0.02450432 1.607682 99
[3] {butter milk} => {other vegetables} 0.01037112 0.3709091 0.02796136 1.916916 102

[4] {butter milk} => {whole milk}      0.01159126 0.4145455 0.02796136 1.622385 114
[5] {ham}         => {whole milk}      0.01148958 0.4414062 0.02602949 1.727509 113
[6] {sliced cheese} => {whole milk}    0.01077783 0.4398340 0.02450432 1.721356 106
[7] {oil}         => {whole milk}      0.01128622 0.4021739 0.02806304 1.573968 111
[8] {onions}      => {other vegetables} 0.01423488 0.4590164 0.03101169 2.372268 140
[9] {onions}      => {whole milk}      0.01209964 0.3901639 0.03101169 1.526965 119
[10] {berries}    => {yogurt}          0.01057448 0.3180428 0.03324860 2.279848 104
```

After running the above code for the Apriori algorithm, we can see the following output, specifying the first 10 strongest Association rules, based on the support (minimum support of 0.01), confidence (minimum confidence of 0.2), and lift,

along with mentioning the count of times the products occur together in the transactions.



```
Console  Terminal  Jobs
R 4.1.3 · D:/MCA SEM I/ADBMS/R Programming/
> # Loading Libraries
> library(arules)
> library(arulesViz)
> library(RColorBrewer)
> # import dataset
> data("Groceries")
> # using apriori() function
> rules <- apriori(Groceries,
+                  parameter = list(supp = 0.01, conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules TRUE

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

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [232 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> # using inspect() function
> inspect(rules[1:10])
     lhs                rhs                support    confidence coverage   lift     count
[1]  {}              => {whole milk}       0.25551601 0.2555160  1.00000000 1.000000 2513
[2]  {hard cheese}   => {whole milk}       0.01006609 0.4107884  0.02450432 1.607682   99
[3]  {butter milk}   => {other vegetables} 0.01037112 0.3709091  0.02796136 1.916916  102
[4]  {butter milk}   => {whole milk}       0.01159126 0.4145455  0.02796136 1.622385  114
[5]  {ham}           => {whole milk}       0.01148958 0.4414062  0.02602949 1.727509  113
[6]  {sliced cheese} => {whole milk}       0.01077783 0.4398340  0.02450432 1.721356  106
[7]  {oil}           => {whole milk}       0.01128622 0.4021739  0.02806304 1.573968  111
[8]  {onions}        => {other vegetables} 0.01423488 0.4590164  0.03101169 2.372268  140
[9]  {onions}        => {whole milk}       0.01209964 0.3901639  0.03101169 1.526965  119
[10] {berries}       => {yogurt}           0.01057448 0.3160428  0.03324860 2.279848  104
>
```

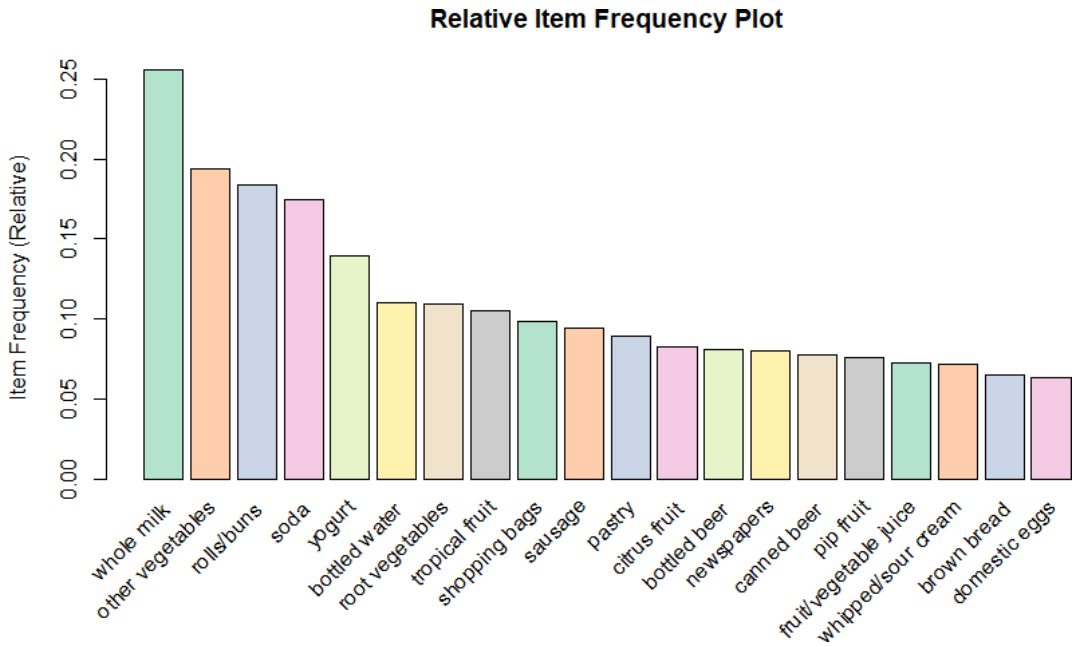
Step 5: Applying itemFrequencyPlot() function

itemFrequencyPlot() creates a bar plot for item frequencies/ support. It creates an item frequency bar plot for inspecting the distribution of objects based on the transactions. The

items are plotted ordered by descending support. Here, 'topN=20' means that 20 items with the highest item frequency/ lift will be plotted.

```
# using itemFrequencyPlot() function
arules::itemFrequencyPlot(Groceries, topN = 20,
                          col = brewer.pal(8, 'Pastel2'),
```

```
main = 'Relative Item
Frequency Plot', type =
"relative",
ylab = "Item Frequency (Relative)")
```



Box Plot of the Top 20 Items having the Highest Item Frequency (Relative) using Lift as a Parameter. On running the Apriori algorithm over the dataset with a minimum support value of 0.01 and minimum confidence of 0.2, we have filtered out the strong association rules in the transaction. We have listed the first 10 transactions above, along with the box plot of the top 20 items having the highest relative item frequency.

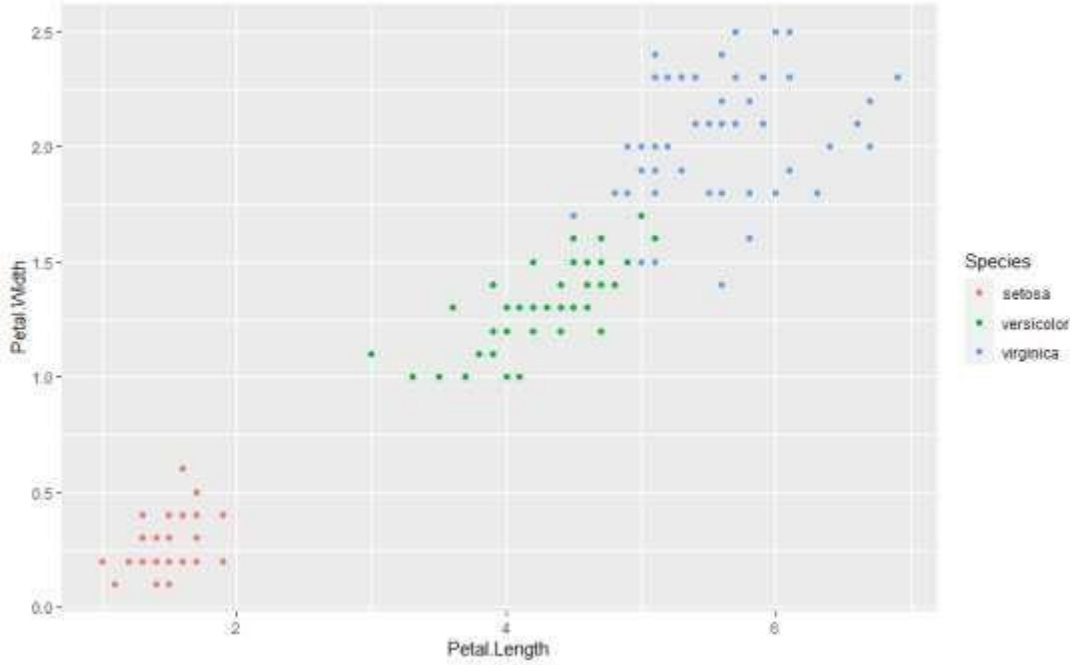
9. Implementation and analysis of clustering algorithms like K-Means , Agglomerative Ans: knncluster.R(k means clustering)

```
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

```
library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```



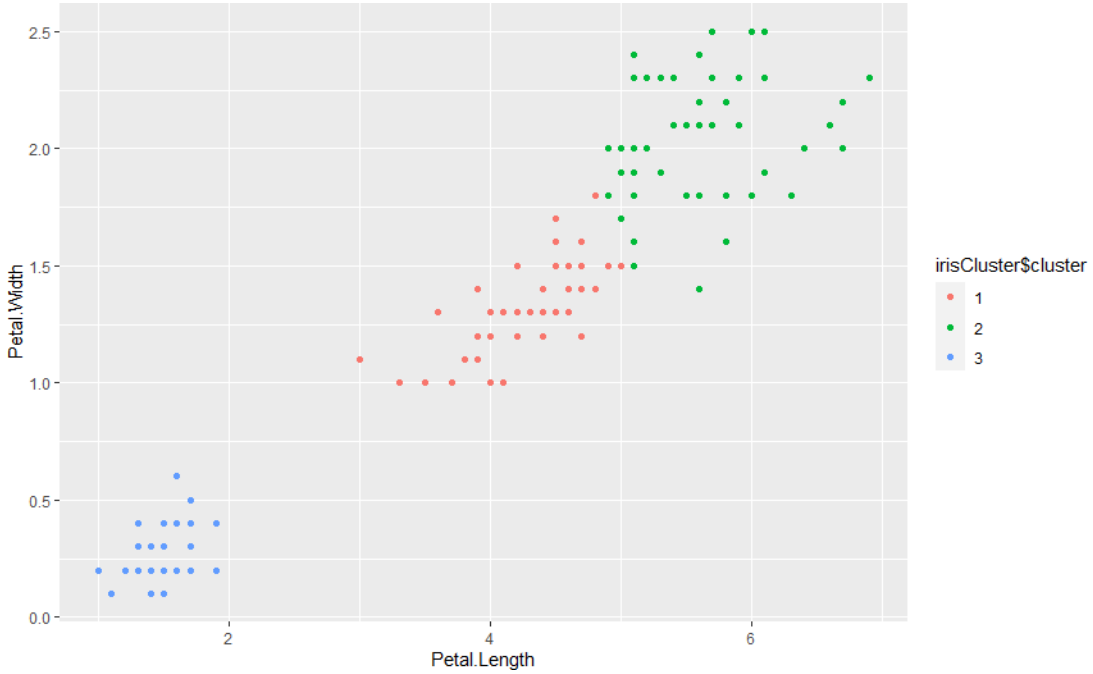

```
set.seed(20)

irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
```

```
irisCluster
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 3 3 3 3
3 1 1 1 1 1 1 1
 [59] 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 1 2 2 2 2 2 2
1 2 2 2 2 2 2 2 2
[117] 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 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"   "size"
[8] "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()



Agglometric Algorithm:

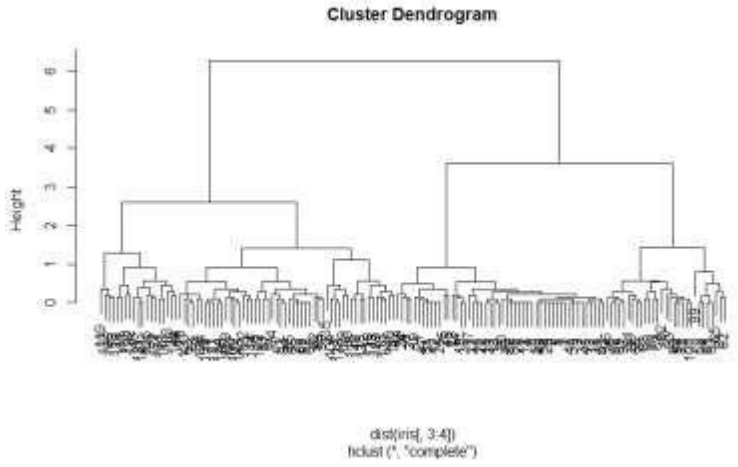
head(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

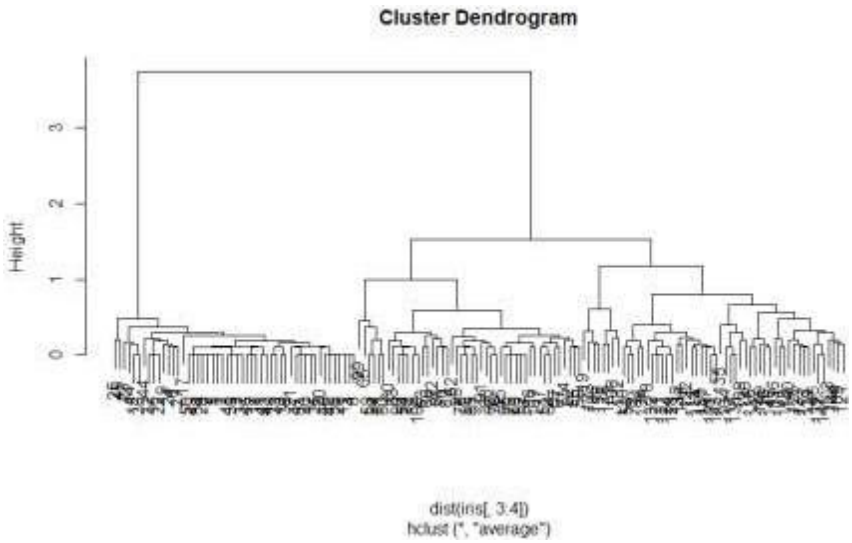
1	5.1	3.5	1.4	0.2 setosa
2	4.9	3.0	1.4	0.2 setosa

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

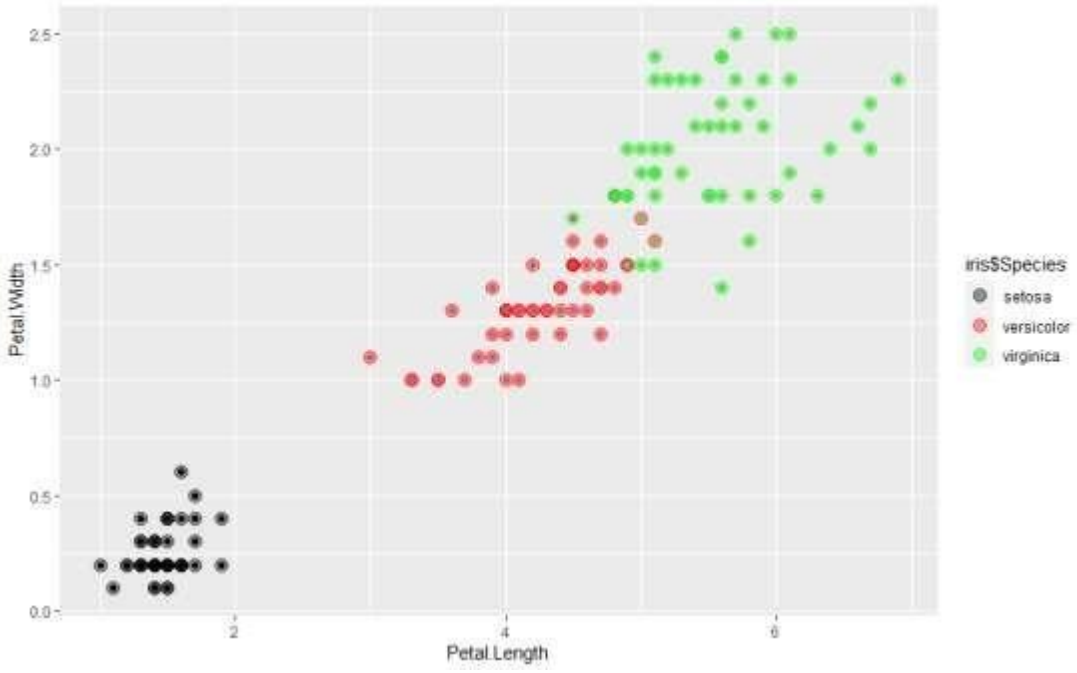
```
clusters <-
hclust(dist(iris[, 3:4]))
plot(clusters)
```




```
clusterCut<-
cutree(clusters, 3)
table(clusterCut,
iris$Species)
clusters <- hclust(dist(iris[,3:4]), method
= 'average') plot(clusters)
```




```
clusterCut <-
cutree(clusters, 3)
table(clusterCut,
iris$Species) clusterCut
setosa versicolor virginica
      1    50         0         0
      2     0        45         1
      3     0         5        49
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$Species)) +geom_point(alpha
      = 0.4, size = 3.5) + geom_point(col = clusterCut) +        scale_color_manual(values
= c('black', 'red', 'green'))
```


Petal.Width
Petal.Length
iris$Species
setosa
versicolor
virginica