Roll No.**24** Exam Seat No._____

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R. C. Marg,Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr./Miss **NARENDER KESWANI** of **FYMCA-1B** has satisfactorily completed a course of the necessary experiments in **MCAL13 - Advanced Database Management System Lab** under the supervision of **DR. MEENAKSHI GARG** in the academic year **2021- 2022.**

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony, Chembur, Mumbai

Department of M.C.A

INDEX

Sr no.	Contents	Date of Preparation	Date of Submission	Page Number	Marks	Sign
1	Introduction to SQL- To implement various DDL, DML commands and constraints.	08-01-2021	12-01-2021	1	10	
2	Implementation of Data partitioning through Range and List partitioning.	09-01-2022	13-01-2022	14	10	
3	Implementation of ORDBMS concepts like ADT (Abstract Data Types) and Reference	13-01-2022	15-01-2022	20	10	
4	Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Lead, Lag, Rank and Dense Rank	19-01-2022	24-01-2022	26	8	
5	Implementation of ETL Transformations using Pentaho.	15-02-2022	22-02-2022	47	10	
6	Introduction to R programming and Data acquisition.	17-02-2022	24-02-2022	68	10	
7	Implementation of Data preprocessing techniques.	19-02-2022	26-02-2022	92	10	
8	Implementation and analysis of Classification algorithms.	21-02-2022	28-02-2022	105	10	
9	Implementation and analysis of Apriori Algorithm using Market Basket Analysis.	22-02-2022	01-03-2022	123	10	
10	Implementation and analysis of Linear regression through graphical methods.	26-02-2022	04-03-2022	128	10	
11	Implementation and analysis of clustering algorithms like K-Means, Agglomerative.	28-02-2022	06-03-2022	136	10	

Aim:- To implement various DDL,DML commands and constraints.

THEORY:

DDL(Data Definition Language): DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database. Examples of DDL commands:

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database.

DML(Data Manipulation Language): The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements. Examples of DML:

- **SELECT** – is used to retrieve data from the a database.
- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

SQL constraints: It are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted. Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table. The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified

A) DDL(Data Definition Language):

1) CREATE:

SOURCE CODE:

```
CREATE TABLE student_narender(
id NUMBER(5) PRIMARY KEY,
Fname VARCHAR2(15),
Lname VARCHAR2(20),
Address VARCHAR2(50),
DOB DATE);
```

OUTPUT:

```
Table created.
```

2) DROP:

SOURCE CODE:

```
DROP TABLE student_narender;
```

OUTPUT:

```
Table dropped.
```

3) ALTER:

SOURCE CODE:

```
ALTER TABLE student_narender ADD Email varchar2(55);
desc student_narender;
```

OUTPUT:

```
Table altered.
```

TABLE STUDENT_NARENDER

Column	Null?	Type
ID	NOT NULL	NUMBER(5,0)
FNAME	-	VARCHAR2(15)
LNAME	-	VARCHAR2(20)
ADDRESS	-	VARCHAR2(50)
DOB	-	DATE
EMAIL	-	VARCHAR2(55)

Download CSV
6 rows selected.

4) **TRUNCATE:**

SOURCE CODE:

```
TRUNCATE TABLE student_narender;
```

OUTPUT:

```
Table truncated.
```

5) **RENAME:**

SOURCE CODE:

```
RENAME student_narender TO student_vesit_narender;
```

OUTPUT:

```
Statement processed.
```

B) DML(Data Manipulation Language):

1) SELECT:

SOURCE CODE:

```
select * from student_narender;
```

OUTPUT:

ID	FNAME	LNAME	ADDRESS	DOB
1	narender	keswani	ulhasnagar	10-NOV-99
2	neel	deshmukh	vasai	31-JAN-00
3	hassan	haque	mahalaxmi	26-AUG-00
4	ronak	karia	majiwada	15-OCT-00
5	wilson	rao	thane	01-JAN-72

Download CSV

5 rows selected.

2) INSERT:

SOURCE CODE:

```
insert into student_narender values(1,'narender','keswani','ulhasnagar','10-NOV-1999');
insert into student_narender values(2,'neel','deshmukh','vasai','31-JAN-2000');
insert into student_narender values(3,'hassan','haque','mahalaxmi','26-AUG-2000');
insert into student_narender values(4,'ronak','karia','majiwada','15-OCT-2000');
insert into student_narender values(5,'wilson','rao','thane','01-JAN-1972');
```

OUTPUT:

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

3) **UPDATE:**

SOURCE CODE:

```
UPDATE student_narender SET Lname = 'yadav' WHERE id = 4;
```

OUTPUT:

```
1 row(s) updated.
```

ID	FNAME	LNAME	ADDRESS	DOB
1	narender	keswani	ulhasnagar	10-NOV-99
2	neel	deshmukh	vasai	31-JAN-00
3	hassan	haque	mahalaxmi	26-AUG-00
4	ronak	yadav	majiwada	15-OCT-00
5	wilson	rao	thane	01-JAN-72

Download CSV

5 rows selected.

4) **DELETE:**

SOURCE CODE:

```
DELETE FROM student_narender WHERE id = 4;
```

OUTPUT:

```
1 row(s) deleted.
```

ID	FNAME	LNAME	ADDRESS	DOB
1	narender	keswani	ulhasnagar	10-NOV-99
2	neel	deshmukh	vasai	31-JAN-00
3	hassan	haque	mahalaxmi	26-AUG-00
5	wilson	rao	thane	01-JAN-72

Download CSV

4 rows selected.

C) SQL constraints:

1) PRIMARY KEY:

SOURCE CODE:

```
CREATE TABLE student_narender(
id NUMBER(5) PRIMARY KEY,
Fname VARCHAR2(15),
Lname VARCHAR2(20),
Address VARCHAR2(50),
DOB DATE);


desc student_narender;
insert into student_narender values(1,'narender','keswani','ulhasnagar','10-NOV-1999');
insert into student_narender values(2,'neel','deshmukh','vasai','31-JAN-2000');
insert into student_narender values(3,'hassan','haque','mahalaxmi','26-AUG-2000');
insert into student_narender values(4,'ronak','karia','majiwada','15-OCT-2000');
insert into student_narender values(5,'wilson','rao','thane','01-JAN-1972');

select * from student_narender;
```

OUTPUT:

```
Table created.
```

TABLE STUDENT_NARENDER

Column	Null?	Type
ID	NOT NULL	NUMBER(5,0)
FNAME	-	VARCHAR2(15)
LNAME	-	VARCHAR2(20)
ADDRESS	-	VARCHAR2(50)
DOB	-	DATE

Download CSV

```
5 rows selected.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

ID	FNAME	LNAME	ADDRESS	DOB
1	narender	keswani	ulhasnagar	10-NOV-99
2	neel	deshmukh	vasai	31-JAN-00
3	hassan	haque	mahalaxmi	26-AUG-00
4	ronak	karia	majiwada	15-OCT-00
5	wilson	rao	thane	01-JAN-72

Download CSV

```
5 rows selected.
```

2) FOREIGN KEY:

SOURCE CODE:

```
CREATE TABLE supplier_narender (
supplier_id numeric (10) not null,
supplier_name varchar2(50) not null,
contact_name varchar2(50),
CONSTRAINT supplier_narender_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products_narender (
product_id numeric(10) not null,
supplier_id numeric(10) not null,
CONSTRAINT fk_supplier
FOREIGN KEY (supplier_id)
REFERENCES supplier_narender (supplier_id)

);

desc supplier_narender;
desc products_narender;

insert into supplier_narender values(1,'datta supplier','narender');
insert into supplier_narender values(2,'sai supplier','neel');
insert into supplier_narender values(3,'shiv supplier','hassan');
insert into supplier_narender values(4,'swami supplier','ronak');
insert into supplier narender values(5,'vishnu supplier','wilson');
insert into products_narender values(1,2);
insert into products_narender values(2,5);
insert into products_narender values(3,1);
insert into products_narender values(4,3);
insert into products_narender values(5,4);

select * from supplier_narender;
select * from products_narender;
```

OUTPUT:

```
Table created.


Table created.
```

TABLE SUPPLIER_NARENDER

Column	Null?	Type
SUPPLIER_ID	NOT NULL	NUMBER(10,0)
SUPPLIER_NAME	NOT NULL	VARCHAR2(50)
CONTACT_NAME	-	VARCHAR2(50)

Download CSV

3 rows selected.

TABLE PRODUCTS_NARENDER

Column	Null?	Type
PRODUCT_ID	NOT NULL	NUMBER(10,0)
SUPPLIER_ID	NOT NULL	NUMBER(10,0)

Download CSV

2 rows selected.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

SUPPLIER_ID	SUPPLIER_NAME	CONTACT_NAME
1	datta supplier	narender
2	sai supplier	neel
3	shiv supplier	hassan
4	swami supplier	ronak
5	vishnu supplier	wilson

Download CSV

5 rows selected.

PRODUCT_ID	SUPPLIER_ID
1	2
2	5
3	1
4	3
5	4

Download CSV

5 rows selected.

3) UNIQUE CONSTRAINT:

SOURCE CODE:

```
55  CREATE TABLE employee_narender (
56  employee_id numeric (10) not null,
57  employee_name varchar2(50) not null,
58    CONSTRAINT employee_id_unique UNIQUE (employee_id)
59  );
60
61
62  desc employee_narender;
63  insert into employee_narender values(1,'narender');
64  insert into employee_narender values(1,'neel');
65
66  select * from employee_narender;
```

OUTPUT:

Unique Constraint is used to set unique value of the particular field
In this example, the e_id is set as unique value
If same value of e_id is inserted again then it will create an error.

Table created.

TABLE EMPLOYEE_NARENDER

Column	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(10,0)
EMPLOYEE_NAME	NOT NULL	VARCHAR2(50)

Download CSV
2 rows selected.

1 row(s) inserted.

ORA-00001: unique constraint (SQL_QAPUDNWFQHBCNSQDJHLIYTXCB.EMPLOYEE_ID_UNIQUE) violated ORA-06512: at "SYS.DBMS_SQL", line 1721

EMPLOYEE_ID	EMPLOYEE_NAME
1	narender

4) DEFAULT CONSTRAINT:

SOURCE CODE:

```
CREATE TABLE customers_narender (
  c_id numeric(10) not null,
  c_name varchar2(55) not null,
  c_country varchar2(55) DEFAULT 'INDIA'
);

desc customers_narender;

insert into customers_narender(c_id,c_name) values(1,'narender');
insert into customers_narender(c_id,c_name,c_country) values(2,'neel','germany');

select * from customers_narender;
```

OUTPUT:

Default constraint is used to set the default value for particular field.
In this example, the c_country is set to the default value of 'INDIA'
If we do not specify the value of the c_country then the default value will be 'INDIA'.

```
Table created.
```

TABLE CUSTOMERS_NARENDER

Column	Null?	Type
C_ID	NOT NULL	NUMBER(10,0)
C_NAME	NOT NULL	VARCHAR2(55)
C_COUNTRY	-	VARCHAR2(55)

Download CSV
3 rows selected.

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

C_ID	C_NAME	C_COUNTRY
1	narender	INDIA
2	neel	germany

Download CSV
2 rows selected.

5) CHECK CONSTRAINT:

SOURCE CODE:

```
CREATE TABLE elections_narender (
  e_id numeric(10) not null,
  e_name varchar2(55) not null,
  e_age numeric(10),
  CONSTRAINT check_age CHECK (e_age >= 18)
);

desc elections_narender;

insert into elections_narender values(1,'narender',22);
insert into elections_narender values(2,'ganesh',15);

select * from elections_narender;
```

OUTPUT:

The check constraint is used to check the condition
If the condition is true then only allow to insert the values, else it will throw error.
In this example, we have checked the age of person who are eligible for voting
So, if the person's age is less than 18 them it will not insert the value in the table.

```
Table created.
```

TABLE ELECTIONS_NARENDER

Column	Null?	Type
E_ID	NOT NULL	NUMBER(10,0)
E_NAME	NOT NULL	VARCHAR2(55)
E_AGE	-	NUMBER(10,0)

Download CSV
3 rows selected.

```
1 row(s) inserted.

ORA-02290: check constraint (SQL_IZGZXKBRMSEBAZVRAPTMQQIHV.CHECK_AGE) violated ORA-06512: at "SYS.DBMS_SQL", line 1721
```

E_ID	E_NAME	E_AGE
1	narender	22

Download CSV

6) NOT NULL:

SOURCE CODE:

```
CREATE TABLE student_narender(
id NUMBER(5) PRIMARY KEY,
Fname VARCHAR2(15) NOT NULL,
Lname VARCHAR2(20),
Address VARCHAR2(50),
DOB DATE);


desc student_narender;

insert into student_narender values(1,'narender','keswani','ulhasnagar','10-NOV-1999');
insert into student_narender values(2,'','deshmukh','vasai','31-JAN-2000');


select * from student_narender;
```

OUTPUT:

If we do not pass the value in the not null field, it will throw the error.

```
Table created.
```

TABLE STUDENT_NARENDER

Column	Null?	Type
ID	NOT NULL	NUMBER(5,0)
FNAME	NOT NULL	VARCHAR2(15)
LNAME	-	VARCHAR2(20)
ADDRESS	-	VARCHAR2(50)
DOB	-	DATE

Download CSV

```
5 rows selected.


1 row(s) inserted.


ORA-01400: cannot insert NULL into ("SQL_VZSRJRMVGKNFOUEAEIESNPRKR"."STUDENT_NARENDER"."FNAME") ORA-06512: at "SYS.DBMS_SQL", line 1721
```

ID	FNAME	LNAME	ADDRESS	DOB
1	narender	keswani	ulhasnagar	10-NOV-99

Download CSV

CONCLUSION:

I have learned the basics of DML, DDL, SQL Constraints from this assignment/tutorial.

Aim :- Implementation of Data partitioning through Range and List partitioning.

Distributed Databases: -

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sited that don't share physical components. This maybe required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:

1. Homogeneous Database: In a homogeneous database, all different sites store database identically. The operating system, database management system and the data structures used – all are same at all sites. Hence, they're easy to manage.

2. Heterogeneous Database: In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Data Partitioning: - Partitioning is the database process where very large tables are divided into multiple smaller parts. By splitting a large table into smaller, individual tables, queries that access only a fraction of the data can run faster because there is less data to scan. The main of goal of partitioning is to aid in maintenance of large tables and to reduce the overall response time to read and load data for particular SQL operations.

Range Partitioning: - Range partitioning is a type of relational database partitioning wherein the partition is based on a predefined range for a specific data field such as uniquely numbered IDs, dates or simple values like currency. A partitioning key column is assigned with a specific range, and when a data entry fits this range, it is assigned to this partition; otherwise it is placed in another partition where it fits.

List Partitioning: - Unlike range partitioning, with list partitioning, there is no apparent sense of order between partitions. You can also specify a default partition into which rows that do not map to any other partition are mapped.

A) Range Partitioning:

```
create table sales_range_narender(
salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE (sales_date)
(
PARTITION sales_jan2000 VALUES LESS THAN(TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION sales_feb2000 VALUES LESS THAN(TO_DATE('01/02/2000','DD/MM/YYYY')),
PARTITION sales_mar2000 VALUES LESS THAN(TO_DATE('01/03/2000','DD/MM/YYYY')),
PARTITION sales_apr2000 VALUES LESS THAN(TO_DATE('01/04/2000','DD/MM/YYYY')),
PARTITION sales_may2000 VALUES LESS THAN(TO_DATE('01/05/2000','DD/MM/YYYY'))
);

SELECT TABLE_NAME, PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE TABLESPACE_NAME='sales_range_narender';

insert into sales_range_narender values(1,'narender keswani',10000,TO_DATE('10/01/2000','DD/MM/YYYY'));
insert into sales_range_narender values(2,'neel deshmukh',20000,TO_DATE('10/02/2000','DD/MM/YYYY'));
insert into sales_range_narender values(3,'hassan haque',30000,TO_DATE('10/03/2000','DD/MM/YYYY'));
insert into sales_range_narender values(4,'ronak karia',40000,TO_DATE('10/04/2000','DD/MM/YYYY'));
insert into sales_range_narender values(5,'wilson rao',50000,TO_DATE('20/04/2000','DD/MM/YYYY'));

select * from sales_range_narender;

select * from sales_range_narender PARTITION(sales_jan2000);
```

```
Table created.


no data found


1 row(s) inserted.


1 row(s) inserted.


1 row(s) inserted.


1 row(s) inserted.


1 row(s) inserted.
```

SALESMAN_ID	SALESMAN_NAME	SALES_AMOUNT	SALES_DATE
1	narender keswani	10000	10-JAN-00
2	neel deshmukh	20000	10-FEB-00
3	hassan haque	30000	10-MAR-00
4	ronak karia	40000	10-APR-00
5	wilson rao	50000	20-APR-00

Download CSV

5 rows selected.

SALESMAN_ID	SALESMAN_NAME	SALES_AMOUNT	SALES_DATE
1	narender keswani	10000	10-JAN-00

Download CSV

B) List Partitioning:

```
create table sales_list_narender(
salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_state varchar2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY LIST (sales_state)
(
PARTITION sales_west VALUES ('Mumbai','Pune'),
PARTITION sales_east VALUES ('Kolkata'),
PARTITION sales_south VALUES ('Chennai'),
PARTITION sales_north VALUES ('Delhi'),
PARTITION sales_other VALUES (Default)
) enable row movement;

SELECT TABLE_NAME, PARTITION_NAME FROM USER_TAB_PARTITIONS;

insert into sales_list_narender values(1,'narender keswani','Mumbai',10000,TO_DATE('10/01/2000','DD/MM/YYYY'));
insert into sales_list_narender values(2,'neel deshmukh','Pune',20000,TO_DATE('10/02/2000','DD/MM/YYYY'));
insert into sales_list_narender values(3,'hassan haque','Delhi',30000,TO_DATE('10/03/2000','DD/MM/YYYY'));
insert into sales_list_narender values(4,'ronak karia','Kolkata',40000,TO_DATE('10/04/2000','DD/MM/YYYY'));
insert into sales_list_narender values(5,'wilson rao','Chennai',50000,TO_DATE('20/04/2000','DD/MM/YYYY'));
insert into sales_list_narender values(6,'ritesh yadav','Allahabad',60000,TO_DATE('10/05/2000','DD/MM/YYYY'));


select * from sales_list_narender;

select * from sales_list_narender PARTITION(sales_west);
```

Table created.

TABLE_NAME	PARTITION_NAME
SALES_LIST_NARENDER	SALES_EAST
SALES_LIST_NARENDER	SALES_NORTH
SALES_LIST_NARENDER	SALES_OTHER
SALES_LIST_NARENDER	SALES_SOUTH
SALES_LIST_NARENDER	SALES_WEST
SALES_RANGE_NARENDER	SALES_APR2000
SALES_RANGE_NARENDER	SALES_FEB2000
SALES_RANGE_NARENDER	SALES_JAN2000
SALES_RANGE_NARENDER	SALES_MAR2000
SALES_RANGE_NARENDER	SALES_MAY2000

Download CSV

10 rows selected.

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SALESMAN_ID	SALESMAN_NAME	SALES_STATE	SALES_AMOUNT	SALES_DATE
1	narender keswani	Mumbai	10000	10-JAN-00
2	neel deshmukh	Pune	20000	10-FEB-00
4	ronak karia	Kolkata	40000	10-APR-00
5	wilson rao	Chennai	50000	20-APR-00
3	hassan haque	Delhi	30000	10-MAR-00
6	ritesh yadav	Allahabad	60000	10-MAY-00

Download CSV

SALESMAN_ID	SALESMAN_NAME	SALES_STATE	SALES_AMOUNT	SALES_DATE
1	narender keswani	Mumbai	10000	10-JAN-00
2	neel deshmukh	Pune	20000	10-FEB-00

Download CSV

2 rows selected.

CONCLUSION:

I have learned the basics of distributed database management system with techniques such as: range and list in oracle.

SOURCE CODE:

```
create table sales_range_narender(
salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE (sales_date)
(
```

```
PARTITION sales_jan2000 VALUES LESS THAN(TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION sales_feb2000 VALUES LESS THAN(TO_DATE('01/02/2000','DD/MM/YYYY')),
PARTITION sales_mar2000 VALUES LESS THAN(TO_DATE('01/03/2000','DD/MM/YYYY')),
PARTITION sales_apr2000 VALUES LESS THAN(TO_DATE('01/04/2000','DD/MM/YYYY')),
PARTITION sales_may2000 VALUES LESS THAN(TO_DATE('01/05/2000','DD/MM/YYYY'))
);

SELECT TABLE_NAME, PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE
TABLESPACE_NAME='sales_range_narender';

insert into sales_range_narender values(1,'narender
keswani',10000,TO_DATE('10/01/2000','DD/MM/YYYY'));
insert into sales_range_narender values(2,'neel
deshmukh',20000,TO_DATE('10/02/2000','DD/MM/YYYY'));
insert into sales_range_narender values(3,'hassan
haque',30000,TO_DATE('10/03/2000','DD/MM/YYYY'));
insert into sales_range_narender values(4,'ronak
karia',40000,TO_DATE('10/04/2000','DD/MM/YYYY'));
insert into sales_range_narender values(5,'wilson
rao',50000,TO_DATE('20/04/2000','DD/MM/YYYY'));

select * from sales_range_narender;

select * from sales_range_narender PARTITION(sales_feb2000);
```

```
create table sales_list_narender(
salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_state varchar2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY LIST (sales_state)
(
PARTITION sales_west VALUES ('Mumbai','Pune'),
PARTITION sales_east VALUES ('Kolkata'),
PARTITION sales_south VALUES ('Chennai'),
PARTITION sales_north VALUES ('Delhi'),
PARTITION sales_other VALUES (Default)
) enable row movement;

SELECT TABLE_NAME, PARTITION_NAME FROM USER_TAB_PARTITIONS;

insert into sales_list_narender values(1,'narender
keswani','Mumbai',10000,TO_DATE('10/01/2000','DD/MM/YYYY'));
insert into sales_list_narender values(2,'neel
deshmukh','Pune',20000,TO_DATE('10/02/2000','DD/MM/YYYY'));
insert into sales_list_narender values(3,'hassan
haque','Delhi',30000,TO_DATE('10/03/2000','DD/MM/YYYY'));
insert into sales_list_narender values(4,'ronak
karia','Kolkata',40000,TO_DATE('10/04/2000','DD/MM/YYYY'));
```

```
insert into sales_list_narender values(5,'wilson
rao','Chennai',50000,TO_DATE('20/04/2000','DD/MM/YYYY'));
insert into sales_list_narender values(6,'ritesh
yadav','Allahabad',60000,TO_DATE('10/05/2000','DD/MM/YYYY'));


select * from sales_list_narender;

select * from sales_list_narender PARTITION(sales_west);
```

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

ORDBMS stands for Object-Relational Database Management System

It provides all the facilities of RDBMS with the additional support of object oriented concepts. The concept of classes, objects and inheritance are supported in this database. It is present in the ground level between the RDBMS and OODBMS. In this data can be manipulated by using any query language. It is complex because it has to take care of both Relational database concepts and as well as Object Oriented concepts

Examples of ORDBMSs include:

- Oracle Database by Oracle Corporation.
- Informix by IBM
- SQL Server by Microsoft
- Greenplum Database by Pivotal Software

Advantages of ORDBMS :-

Reuse and Sharing:- The main advantages of extending the Relational data model come from reuse and sharing. Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.

Increased Productivity:- ORDBMS provides increased productivity both for the developer and for the, end user

Use of experience in developing RDBMS:- Another obvious advantage is that .the extended relational approach preserves the significant body of knowledge and experience that has gone into developing relational applications. This is a significant advantage, as many organizations would find it prohibitively expensive to change. If the new functionality is designed appropriately, this approach should allow organizations to take advantage of the new extensions in an evolutionary way without losing the benefits of current database features and functions.

Disadvantages of ORDBMS:- The ORDBMS approach has the obvious disadvantages of complexity and associated increased costs. Further, there are the proponents of the relational approach that believe the· essential simplicity' and purity of the .relational model are lost with these types of extension.

Abstract Data Types (ADT):- By using abstract data types, which are user-defined types, together with various routines, you can uniquely define and use data with complex structures and perform operations on such data. When you define a column as having an abstract data type, you can conceptualize and model its data based on object-oriented concepts. In addition, by applying object-oriented software development techniques, you can reduce the workload for database design, UAP development, and maintenance

REF Function:-

In Oracle PL/SQL, REF data types are pointers that uniquely identify a piece of data as an object. A reference can be established between an existent valid object and a table or type attribute using the REF pointer data type. An attribute referring to a nonexistent object leads to "dangling" situation. Note that a NULL object reference is different from a Dangling Reference. To insert data into a ref column, the REF function is used to get an object instance reference

DEREF Function:-

DEREF returns the object reference of argument expr, where expr must return a REF to an object. If you do not use this function in a query, then Oracle Database returns the object ID of the REF instead.

A) <u>ABSTRACT DATA TYPE:</u>

Creating type : type_name_narender and type_address_narender

```
SQL> create type type_name_narender as object
  2  (
  3  fname varchar(20),
  4  mname varchar(20),
  5  lname varchar(20)
  6  );
  7  /

Type created.

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

Type created.
```

Creating table customer1_narender:

```
SQL> create table customer1_narender
  2  (
  3  c_id number(5) primary key,
  4  c_name type_name_narender,
  5  c_add type_address_narender,
  6  c_phone_number number(20)
  7  );

Table created.
```

Inserting the records in the customer1_narender table:

```
SQL> insert into customer1_narender values(1,type_name_narender('narender','rajesh','keswani'),type_address_narender('gol midi
an','ulhasnagar',421001),9320907041);

1 row created.
```

Displaying all records:

```
SQL> select * from customer1_narender;

      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
C_PHONE_NUMBER
--------------
         1
TYPE_NAME_NARENDER('narender', 'rajesh', 'keswani')
TYPE_ADDRESS_NARENDER('gol midian', 'ulhasnagar', 421001)
    9320907041
```

Checking the structure of the table:

```
SQL> desc customer1_narender;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(5)
 C_NAME                                             TYPE_NAME_NARENDER
 C_ADD                                              TYPE_ADDRESS_NARENDER
 C_PHONE_NUMBER                                     NUMBER(20)
```

Displaying only street of the customer of c_id=1:

```
SQL> select c.c_add.street from customer1_narender c where c_id=1;

C_ADD.STREET
--------------------
gol midian
```

Viewing in depth structure of the table:

```
SQL> set describe depth 2;
SQL> desc customer1_narender;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(5)
 C_NAME                                             TYPE_NAME_NARENDER
   FNAME                                            VARCHAR2(20)
   MNAME                                            VARCHAR2(20)
   LNAME                                            VARCHAR2(20)
 C_ADD                                              TYPE_ADDRESS_NARENDER
   STREET                                           VARCHAR2(20)
   CITY                                             VARCHAR2(20)
   PINCODE                                          NUMBER(10)
 C_PHONE_NUMBER                                     NUMBER(20)
```

Displaying first name , middle name , last name of the customer1_narender table:

```
SQL> select c.c_name.fname||' '||c.c_name.mname||' '||c.c_name.lname from customer1_narender c;

C.C_NAME.FNAME||''||C.C_NAME.MNAME||''||C.C_NAME.LNAME
------------------------------------------------------------
narender rajesh keswani
```

B) REF:

```
SQL> create or replace type animal_type as object
  2  (
  3  Breed varchar2(25),
  4  Name varchar2(25),
  5  BirthDate DATE
  6  );
  7  /

Type created.

SQL>
SQL> create table animal_narender of animal_type;

Table created.

SQL>
SQL> insert into animal_narender values (animal_type('Dog','Tucker','01-MAR-06'));

1 row created.

SQL> insert into animal_narender values (animal_type('Mule','France','02-MAY-07'));

1 row created.

SQL> insert into animal_narender values (animal_type('Dog','Benji','03-AUG-08'));

1 row created.

SQL> insert into animal_narender values (animal_type('Cat','Milo','04-DEC-09'));

1 row created.
```

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

REF(A)
--------------------------------------------------------------------------------
0000280209B26F2457BB574CBF850D052EE7E6D046D913B703427D47EF8509C6418F4BFF360041B6
E10000

0000280209257233 2F15534D62B3B4FB57CF0A46E9D913B703427D47EF8509C6418F4BFF360041B6
E10001

0000280209C711711E1AC14CF58B62D0DEA3BE240AD913B703427D47EF8509C6418F4BFF360041B6
E10002

00002802099A62C051BC044A73A87DC8E113AD8F0DD913B703427D47EF8509C6418F4BFF360041B6
E10003

REF(A)
--------------------------------------------------------------------------------
```

C) DEREF:

```
SQL> create table keeper_narender
  2  (
  3  keeper_name varchar2(30),
  4  animal_kept REF animal_type
  5  );

Table created.

SQL> insert into keeper_narender select 'narender', REF(A) from animal_narender A where name='Benji';

1 row created.

SQL> select * from keeper_narender;

KEEPER_NAME
------------------------------
ANIMAL_KEPT
--------------------------------------------------------------------------------
narender
0000220208C711711E1AC14CF58B62D0DEA3BE240AD913B703427D47EF8509C6418F4BFF36


SQL> select keeper_name, DEREF(kn.animal_kept) from keeper_narender kn;

KEEPER_NAME
------------------------------
DEREF(KN.ANIMAL_KEPT)(BREED, NAME, BIRTHDATE)
--------------------------------------------------------------------------------
narender
ANIMAL_TYPE('Dog', 'Benji', '03-AUG-08')
```

Aim :- Implementation of Analytical queries like RollUp, Cube, First, Last, Lead, Lag, Rank, Dense Rank, etc.

Analytical Queries :-

Analytic functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group. The group of rows is called a window and is defined by the analytic_clause. For each row, a sliding window of rows is defined. The window determines the range of rows used to perform the calculations for the current row. Window sizes can be based on either a physical number of rows or a logical interval such as time.

Analytic functions are the last set of operations performed in a query except for the final ORDER BY clause. All joins and all WHERE, GROUP BY, and HAVING clauses are completed before the analytic functions are processed. Therefore, analytic functions can appear only in the select list or ORDER BY clause.

Analytic functions are commonly used to compute cumulative, moving, centered, and reporting aggregates.

ROLLUP :-

ROLLUP enables a SELECT statement to calculate multiple levels of subtotals across a specified group of dimensions. It also calculates a grand total. ROLLUP is a simple extension to the GROUP BY clause, so its syntax is extremely easy to use. The ROLLUP extension is highly efficient, adding minimal overhead to a query.

Syntax :-

SELECT ... GROUP BY
ROLLUP(grouping_column_reference_list)

CUBE :-

CUBE enables a SELECT statement to calculate subtotals for all possible combinations of a group of dimensions. It also calculates a grand total. This is the set of information typically needed for all cross-tabular reports, so CUBE can calculate a cross-tabular report with a single SELECT statement. Like ROLLUP, CUBE is a

simple extension to the GROUP BY clause, and its syntax is also easy to learn.

Syntax :-

SELECT ... GROUP BY
CUBE (grouping_column_reference_list)

FIRST :-

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

Syntax :-

Function( ) KEEP (DENSE_RANK FIRST ORDER BY <expr>) OVER (<partitioning_clause>)

LAST :-

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

Syntax :-

Function( ) KEEP (DENSE_RANK LAST ORDER BY <expr>) OVER (<partitioning_clause>)

LEAD :-

The LEAD function is used to return data from rows further down result set.

Syntax :-

LEAD
{ (value_expr [, offset [, default]]) [{ RESPECT | IGNORE } NULLS] | (value_expr [{ RESPECT | IGNORE } NULLS] [, offset [, default]])
}
OVER ([query_partition_clause] order_by_clause)

LAG :-

The LAG function is used to access data from a previous row.

Syntax :-

LAG
{ (value_expr [, offset [, default]]) [{ RESPECT | IGNORE } NULLS] | (value_expr [{ RESPECT | IGNORE } NULLS] [, offset [, default]])
}
OVER ([query_partition_clause] order_by_clause)

RANK :-

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

The basic description for the RANK analytic function is shown below. The analytic clause is described in more detail here.

Syntax :-

RANK() OVER ([query_partition_clause] order_by_clause)

DENSE RANK :-

The DENSE_RANK function acts like the RANK function except that it assigns consecutive ranks, so this is not like olympic medaling. The basic description for the DENSE_RANK analytic function is shown below. The analytic clause is described in more detail here.

Syntax :-

DENSE_RANK() OVER([query_partition_clause] order_by_clause)

Creating table departments [dept_narender]:

```
create table dept_narender(
  deptno      number(2,0),
  dname       varchar2(14),
  loc         varchar2(13),
  constraint pk_dept primary key (deptno)
)
```

```
Table created.
```

Creating table employees [emp_narender]:

```
create table emp_narender(
  empno    number(4,0),
  ename    varchar2(50),
  job      varchar2(9),
  mgr      number(4,0),
  hiredate date,
  sal      number(7,2),
  comm     number(7,2),
  deptno   number(2,0),
  constraint pk_emp primary key (empno),
  constraint fk_deptno foreign key (deptno) references dept_narender (deptno)
)
```

```
Table created.
```

Inserting Records in department table:

```
/* Inserting records */
insert into dept_narender (DEPTNO, DNAME, LOC) values(10, 'ACCOUNTING', 'NEW YORK')
insert into dept_narender (DEPTNO, DNAME, LOC) values(20, 'RESEARCH', 'DALLAS')
insert into dept_narender (DEPTNO, DNAME, LOC) values(30, 'SALES', 'CHICAGO')
insert into dept_narender (DEPTNO, DNAME, LOC) values(40, 'OPERATIONS', 'BOSTON')
```

Checking Description of department table:

```
desc dept_narender;
```

TABLE DEPT_NARENDER

Column	Null?	Type
DEPTNO	NOT NULL	NUMBER(2,0)
DNAME	-	VARCHAR2(14)
LOC	-	VARCHAR2(13)

Download CSV

3 rows selected.

Viewing all the records of the department table:

```
select * from dept_narender;
```

DEPTNO	DNAME	LOC
20	RESEARCH	DALLAS
40	OPERATIONS	BOSTON
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO

Download CSV

4 rows selected.

Inserting records in employee table:

```
/* Inserting records */
insert into emp_narender values(7839, 'Narender Keswani', 'PRESIDENT', null, to_date('10-11-1999','dd-mm-yyyy'),5000, null, 10)
insert into emp_narender values(7698, 'Neel Deskhmukh', 'MANAGER', 7839, to_date('1-5-1981','dd-mm-yyyy'), 2850, null, 30)
insert into emp_narender values(7782, 'Hassan Haque', 'MANAGER', 7839, to_date('9-6-1981','dd-mm-yyyy'), 2450, null, 10)
insert into emp_narender values(7566, 'Wilson Rao', 'MANAGER', 7839, to_date('2-4-1981','dd-mm-yyyy'), 2975, null, 20)
insert into emp_narender values(7788, 'Ritesh Yadav', 'ANALYST', 7566, to_date('13-JUL-87','dd-mm-rr') - 85,  3000, null, 20)
insert into emp_narender values(7902, 'Ronak Karia', 'ANALYST', 7566, to_date('3-12-1981','dd-mm-yyyy'), 3000, null, 20)
insert into emp_narender values(7369, 'Ashok Gawade', 'CLERK', 7902, to_date('17-12-1980','dd-mm-yyyy'),  800, null, 20)
insert into emp_narender values(7499, 'Pranot Gawade', 'SALESMAN', 7698, to_date('20-2-1981','dd-mm-yyyy'), 1600, 300, 30)
insert into emp_narender values(7521, 'Rajaram Gawade', 'SALESMAN', 7698, to_date('22-2-1981','dd-mm-yyyy'), 1250, 500, 30)
insert into emp_narender values(7654, 'Ganesh Keswani', 'SALESMAN', 7698, to_date('28-9-1981','dd-mm-yyyy'), 1250, 1400, 30)
insert into emp_narender values(7844, 'Avnish Khandewal', 'SALESMAN', 7698, to_date('8-9-1981','dd-mm-yyyy'), 1500, 0, 30)
insert into emp_narender values(7876, 'Kalpesh Khandewal', 'CLERK', 7788, to_date('13-JUL-87', 'dd-mm-rr') - 51, 1100, null, 20)
insert into emp_narender values(7900, 'Shernik Jain', 'CLERK', 7698, to_date('3-12-1981','dd-mm-yyyy'), 950, null, 30)
insert into emp_narender values(7934, 'Ujjal Ray', 'CLERK', 7782, to_date('23-1-1982','dd-mm-yyyy'), 1300, null, 10)
```

Viewing description of employee table:

```
47  desc emp_narender;
```

TABLE EMP_NARENDER

Column	Null?	Type
EMPNO	NOT NULL	NUMBER(4,0)
ENAME	-	VARCHAR2(50)
JOB	-	VARCHAR2(9)
MGR	-	NUMBER(4,0)
HIREDATE	-	DATE
SAL	-	NUMBER(7,2)
COMM	-	NUMBER(7,2)
DEPTNO	-	NUMBER(2,0)

Download CSV

8 rows selected.

Viewing all the records of the employee table:

```
select * from emp_narender;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	Narender Keswani	PRESIDENT	-	10-NOV-99	5000	-	10
7698	Neel Deskhmukh	MANAGER	7839	01-MAY-81	2850	-	30
7788	Ritesh Yadav	ANALYST	7566	19-APR-87	3000	-	20
7902	Ronak Karia	ANALYST	7566	03-DEC-81	3000	-	20
7369	Ashok Gawade	CLERK	7902	17-DEC-80	800	-	20
7521	Rajaram Gawade	SALESMAN	7698	22-FEB-81	1250	500	30
7900	Shernik Jain	CLERK	7698	03-DEC-81	950	-	30
7782	Hassan Haque	MANAGER	7839	09-JUN-81	2450	-	10
7876	Kalpesh Khandewal	CLERK	7788	23-MAY-87	1100	-	20
7934	Ujjal Ray	CLERK	7782	23-JAN-82	1300	-	10
7566	Wilson Rao	MANAGER	7839	02-APR-81	2975	-	20
7654	Ganesh Keswani	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	Pranot Gawade	SALESMAN	7698	20-FEB-81	1600	300	30
7844	Avnish Khandewal	SALESMAN	7698	08-SEP-81	1500	0	30

Download CSV

14 rows selected.

Viewing combining records of the employee and department table based on the joins:

```
51 /* Viewing all Records */
52 select ename, dname, job, empno, hiredate, loc from emp_narender, dept_narender where emp_narender.deptno = dept_narender.deptno order by ename
```

ENAME	DNAME	JOB	EMPNO	HIREDATE	LOC
Ashok Gawade	RESEARCH	CLERK	7369	17-DEC-80	DALLAS
Avnish Khandewal	SALES	SALESMAN	7844	08-SEP-81	CHICAGO
Ganesh Keswani	SALES	SALESMAN	7654	28-SEP-81	CHICAGO
Hassan Haque	ACCOUNTING	MANAGER	7782	09-JUN-81	NEW YORK
Kalpesh Khandewal	RESEARCH	CLERK	7876	23-MAY-87	DALLAS
Narender Keswani	ACCOUNTING	PRESIDENT	7839	10-NOV-99	NEW YORK
Neel Deskhmukh	SALES	MANAGER	7698	01-MAY-81	CHICAGO
Pranot Gawade	SALES	SALESMAN	7499	20-FEB-81	CHICAGO
Rajaram Gawade	SALES	SALESMAN	7521	22-FEB-81	CHICAGO
Ritesh Yadav	RESEARCH	ANALYST	7788	19-APR-87	DALLAS
Ronak Karia	RESEARCH	ANALYST	7902	03-DEC-81	DALLAS
Shernik Jain	SALES	CLERK	7900	03-DEC-81	CHICAGO
Ujjal Ray	ACCOUNTING	CLERK	7934	23-JAN-82	NEW YORK
Wilson Rao	RESEARCH	MANAGER	7566	02-APR-81	DALLAS

Download CSV

14 rows selected.

A) ROLL -UP:

```
55  /* ROLL-UP */
56  select deptno, job, count(*), sum(sal) from emp_narender group by rollup(deptno, job);
57
58  select deptno, job, count(*), avg(sal) from emp_narender group by rollup(deptno, job);
59
60  select deptno, job, count(*), min(sal) from emp_narender group by rollup(deptno, job);
61
62  select deptno, job, count(*), max(sal) from emp_narender group by rollup(deptno, job);
```

EXAMPLE OF SUM:

DEPTNO	JOB	COUNT(*)	SUM(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	8750
20	CLERK	2	1900
20	ANALYST	2	6000
20	MANAGER	1	2975
20	-	5	10875
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	5600
30	-	6	9400
-	-	14	29025

Download CSV

13 rows selected.

EXAMPLE OF AVG:

DEPTNO	JOB	COUNT(*)	AVG(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	2916.66666666666666666666666666666666666667
20	CLERK	2	950
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	2175
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1400
30	-	6	1566.66666666666666666666666666666666666667
-	-	14	2073.21428571428571428571428571428571428571

Download CSV

13 rows selected.

EXAMPLE OF MIN:

DEPTNO	JOB	COUNT(*)	MIN(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	1300
20	CLERK	2	800
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	800
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1250
30	-	6	950
-	-	14	800

Download CSV

13 rows selected.

EXAMPLE OF MAX:

DEPTNO	JOB	COUNT(*)	MAX(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	5000
20	CLERK	2	1100
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	3000
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1600
30	-	6	2850
-	-	14	5000

Download CSV

13 rows selected.

B) CUBE:

```
64  /* CUBE */
65  select deptno, job, count(*), sum(sal) from emp_narender group by cube(deptno, job);
66
67  select deptno, job, count(*), avg(sal) from emp_narender group by cube(deptno, job);
68
69  select deptno, job, count(*), min(sal) from emp_narender group by cube(deptno, job);
70
71  select deptno, job, count(*), max(sal) from emp_narender group by cube(deptno, job);
```

DEPTNO	JOB	COUNT(*)	SUM(SAL)
-	-	14	29025
-	CLERK	4	4150
-	ANALYST	2	6000
-	MANAGER	3	8275
-	SALESMAN	4	5600
-	PRESIDENT	1	5000
10	-	3	8750
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	10875
20	CLERK	2	1900
20	ANALYST	2	6000
20	MANAGER	1	2975
30	-	6	9400
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	5600

Download CSV

18 rows selected.

DEPTNO	JOB	COUNT(*)	AVG(SAL)
-	-	14	2073.2142857142857142857142857142857142857
-	CLERK	4	1037.5
-	ANALYST	2	3000
-	MANAGER	3	2758.3333333333333333333333333333333333333
-	SALESMAN	4	1400
-	PRESIDENT	1	5000
10	-	3	2916.6666666666666666666666666666666666667
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	2175
20	CLERK	2	950
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	1566.6666666666666666666666666666666666667
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1400

Download CSV

18 rows selected.

DEPTNO	JOB	COUNT(*)	MIN(SAL)
-	-	14	800
-	CLERK	4	800
-	ANALYST	2	3000
-	MANAGER	3	2450
-	SALESMAN	4	1250
-	PRESIDENT	1	5000
10	-	3	1300
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	800
20	CLERK	2	800
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	950
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1250

Download CSV

18 rows selected.

DEPTNO	JOB	COUNT(*)	MAX(SAL)
-	-	14	5000
-	CLERK	4	1300
-	ANALYST	2	3000
-	MANAGER	3	2975
-	SALESMAN	4	1600
-	PRESIDENT	1	5000
10	-	3	5000
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	3000
20	CLERK	2	1100
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	2850
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1600

Download CSV

18 rows selected.

C) **RANK:**

```
/* RANK */
select empno, deptno, sal, comm, rank() over(partition by deptno order by sal)as Rank from emp_narender;
```

EMPNO	DEPTNO	SAL	COMM	RANK
7934	10	1300	-	1
7782	10	2450	-	2
7839	10	5000	-	3
7369	20	800	-	1
7876	20	1100	-	2
7566	20	2975	-	3
7902	20	3000	-	4
7788	20	3000	-	4
7900	30	950	-	1
7654	30	1250	1400	2
7521	30	1250	500	2
7844	30	1500	0	4
7499	30	1600	300	5
7698	30	2850	-	6

Download CSV

14 rows selected.

D) **DENSE-RANK:**

```
/* DENSE-RANK */
select empno, deptno, sal, comm, dense_rank() over(partition by deptno order by sal)as Rank from emp_narender;
```

EMPNO	DEPTNO	SAL	COMM	RANK
7934	10	1300	-	1
7782	10	2450	-	2
7839	10	5000	-	3
7369	20	800	-	1
7876	20	1100	-	2
7566	20	2975	-	3
7902	20	3000	-	4
7788	20	3000	-	4
7900	30	950	-	1
7654	30	1250	1400	2
7521	30	1250	500	2
7844	30	1500	0	3
7499	30	1600	300	4
7698	30	2850	-	5

Download CSV

14 rows selected.

E) LEAD:

```
/* LEAD */
select empno, hiredate, lead(hiredate,1) over(order by hiredate) as "next" from emp_narender;

select empno, hiredate, lead(hiredate,1) over(order by hiredate) as "next" from emp_narender where deptno=10;
```

EMPNO	HIREDATE	next
7369	17-DEC-80	20-FEB-81
7499	20-FEB-81	22-FEB-81
7521	22-FEB-81	02-APR-81
7566	02-APR-81	01-MAY-81
7698	01-MAY-81	09-JUN-81
7782	09-JUN-81	08-SEP-81
7844	08-SEP-81	28-SEP-81
7654	28-SEP-81	03-DEC-81
7900	03-DEC-81	03-DEC-81
7902	03-DEC-81	23-JAN-82
7934	23-JAN-82	19-APR-87
7788	19-APR-87	23-MAY-87
7876	23-MAY-87	10-NOV-99
7839	10-NOV-99	-

Download CSV

14 rows selected.

EMPNO	HIREDATE	next
7782	09-JUN-81	23-JAN-82
7934	23-JAN-82	10-NOV-99
7839	10-NOV-99	-

Download CSV

3 rows selected.

F) LAG:

```
/* LAG */
select empno, hiredate, lag(hiredate,1) over(order by hiredate) as "previous" from emp_narender;

select empno, hiredate, lag(hiredate,1) over(order by hiredate) as "previous" from emp_narender where deptno=10;
```

EMPNO	HIREDATE	previous
7369	17-DEC-80	-
7499	20-FEB-81	17-DEC-80
7521	22-FEB-81	20-FEB-81
7566	02-APR-81	22-FEB-81
7698	01-MAY-81	02-APR-81
7782	09-JUN-81	01-MAY-81
7844	08-SEP-81	09-JUN-81
7654	28-SEP-81	08-SEP-81
7900	03-DEC-81	28-SEP-81
7902	03-DEC-81	03-DEC-81
7934	23-JAN-82	03-DEC-81
7788	19-APR-87	23-JAN-82
7876	23-MAY-87	19-APR-87
7839	10-NOV-99	23-MAY-87

Download CSV

14 rows selected.

EMPNO	HIREDATE	previous
7782	09-JUN-81	-
7934	23-JAN-82	09-JUN-81
7839	10-NOV-99	23-JAN-82

Download CSV

3 rows selected.

G) **FIRST:**

```
/* FIRST */
select deptno, sal, max(sal) keep(DENSE_RANK FIRST ORDER BY sal desc) over(PARTITION BY deptno)"max" from emp_narender;
```

DEPTNO	SAL	max
10	2450	5000
10	1300	5000
10	5000	5000
20	2975	3000
20	1100	3000
20	800	3000
20	3000	3000
20	3000	3000
30	1250	2850
30	2850	2850
30	950	2850
30	1250	2850
30	1500	2850
30	1600	2850

Download CSV

14 rows selected.

H) LAST:

```
/* LAST */
select deptno, sal, min(sal) keep(DENSE_RANK LAST ORDER BY sal desc) over(PARTITION BY deptno)"min" from emp_narender;
```

DEPTNO	SAL	min
10	2450	1300
10	1300	1300
10	5000	1300
20	2975	800
20	1100	800
20	800	800
20	3000	800
20	3000	800
30	1250	950
30	2850	950
30	950	950
30	1250	950
30	1500	950
30	1600	950

Download CSV

14 rows selected.

CONCLUSION:

I have learned the implementation of Analytical queries like RollUp, Cube, First, Last, Lead, Lag, Rank, Dense Rank, etc.

AIM: Extract and load data using Pentaho Data integration tool

THEORY:

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

- ✓ Data extraction from source databases
- ✓ Transport of the data
- ✓ Data transformation
- ✓ Loading of data into a data warehouse Kettle is a set of tools and applications which allows data manipulations across multiple sources.

The main components of Pentaho Data Integration are:

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

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

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

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

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

ETL Process Transformation:

As the name suggested ETL stands for Extract Transform and Load.Just like the name applies ETL tool Extracts data from the source.Transforms the data while in transit and then it loads the data in to Specified database.

- **Sort Transformation:**
  The Sort transformation sorts input data in ascending or descending order and copies the sorted data to the transformation output. You can apply multiple sorts to an input; each sort is identified by a numeral that determines the sort order. The column with the lowest number is sorted first, the sort column with the second lowest number is sorted next, and so on.

- **Add Sequence:**
  The Add sequence step adds a sequence to the stream. A sequence is an everchanging integer value with a specific start and increment value. You can either use a database sequence to determine the value of the sequence, or have it generated by Kettle.

- **Concat:**
  The Concat Fields step is used to concatenate multiple fields into one target field. The fields can be separated by a separator and the enclosure logic is completely compatible with the Text File Output step.

- **Unique Rows:**
  The Unique rows step removes duplicate rows from the input stream(s).

- **Split:**
  The Split Fields step allows you to split fields based on delimiter information

Features of Pentaho:

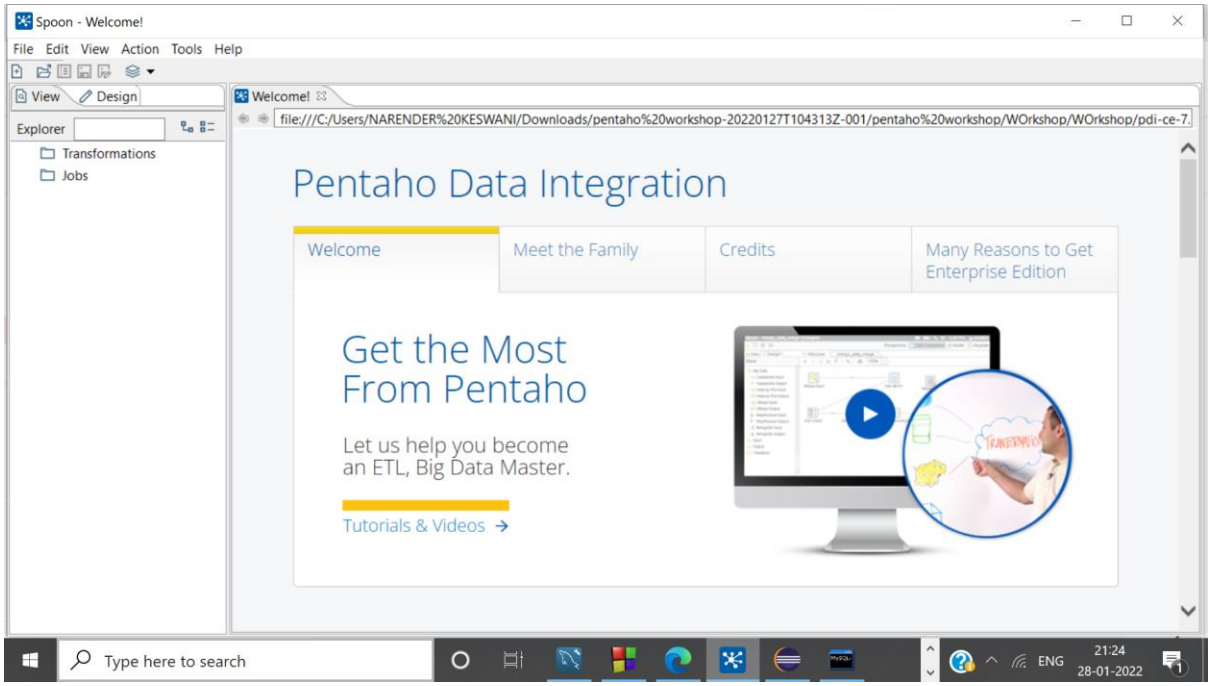
Pentaho Reporting primarily includes a Reporting Engine, a Report Designer, a

Business Intelligence (BI) Server. It comes loaded with the following features − • Report Designer − Used for creating pixel perfect report.

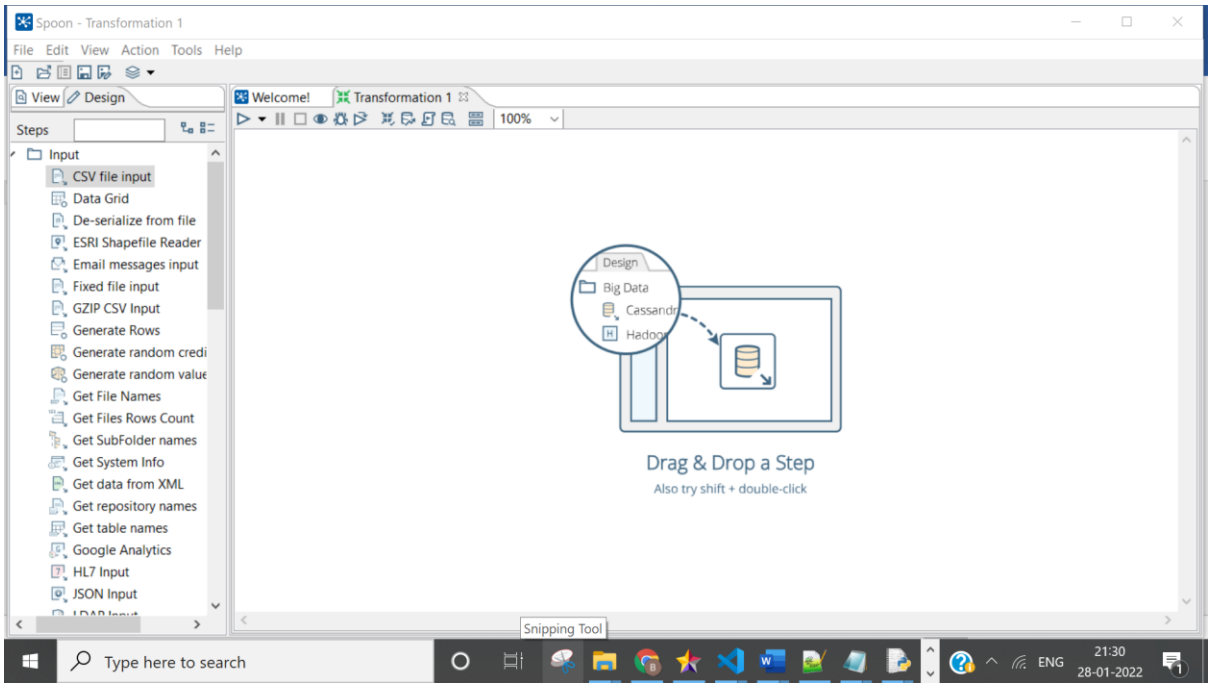
- Metadata Editor − Allows to add user-friendly metadata domain to a data source. • Report Designer and Design Studio − Used for fine-tuning of reports and ad-hoc
- reporting.
- Pentaho user console web interface − Used for easily managing reports and
- analyzing views.
- Ad-Hoc reporting interface − Offers a step-by-step wizard for designing simple
- reports. Output formats include PDF, RTF, HTML, and XLS.
- A complex scheduling sub-system − Allows users to execute reports at given
- intervals.
- Mailing − Users can email a published report to other users.

A) Demo of reading and writing csv file.

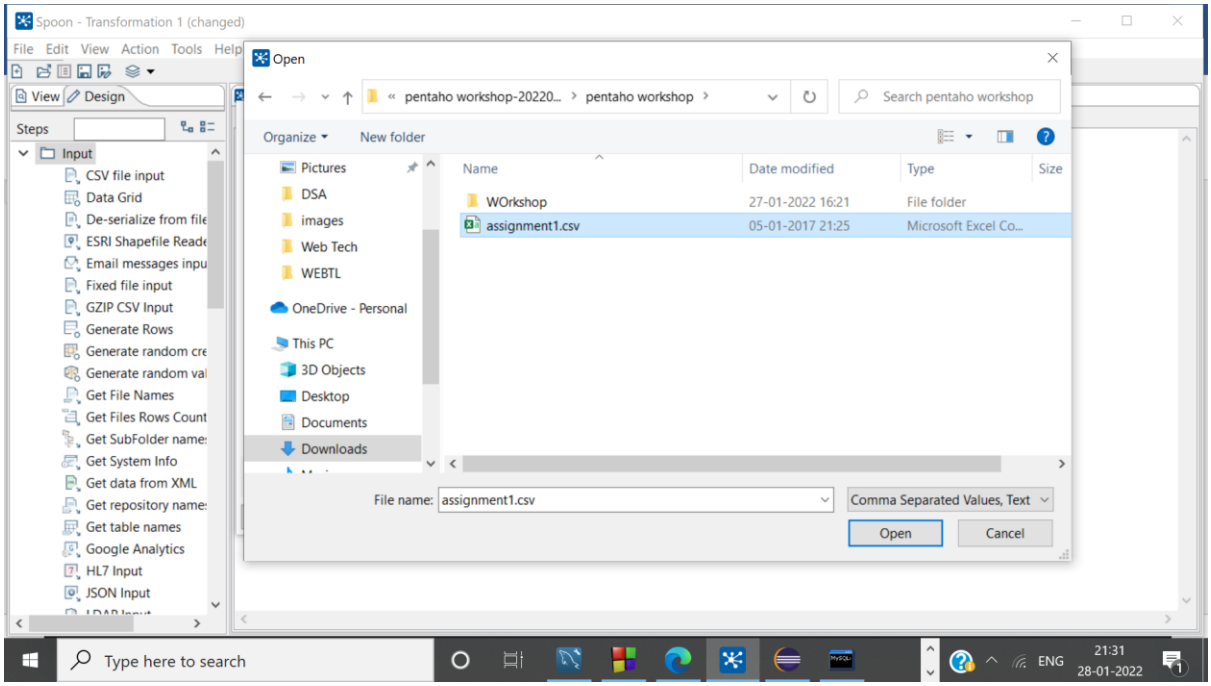
Step 1. Open Data Integration folder
Step 2. Double click on spoon(Windows Batch file)
Step 3. A welcome! window appears with some useful links for you to see.



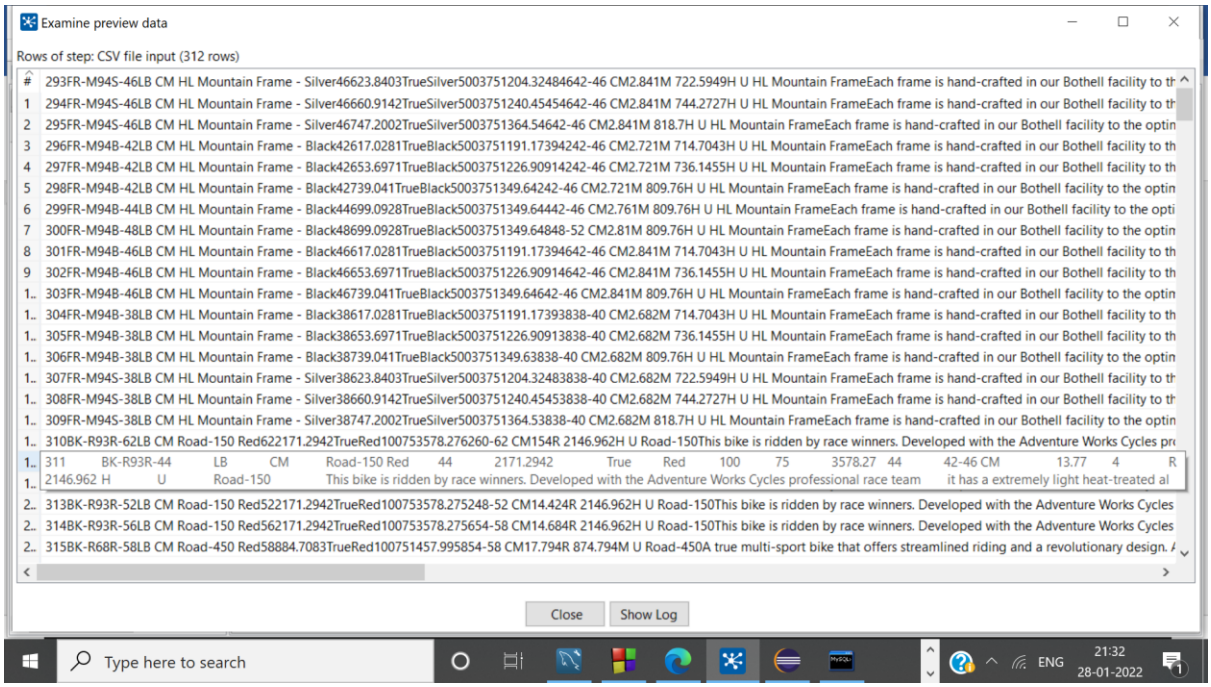
Step 4: Create new transformation, in input drag & drop the csv file.



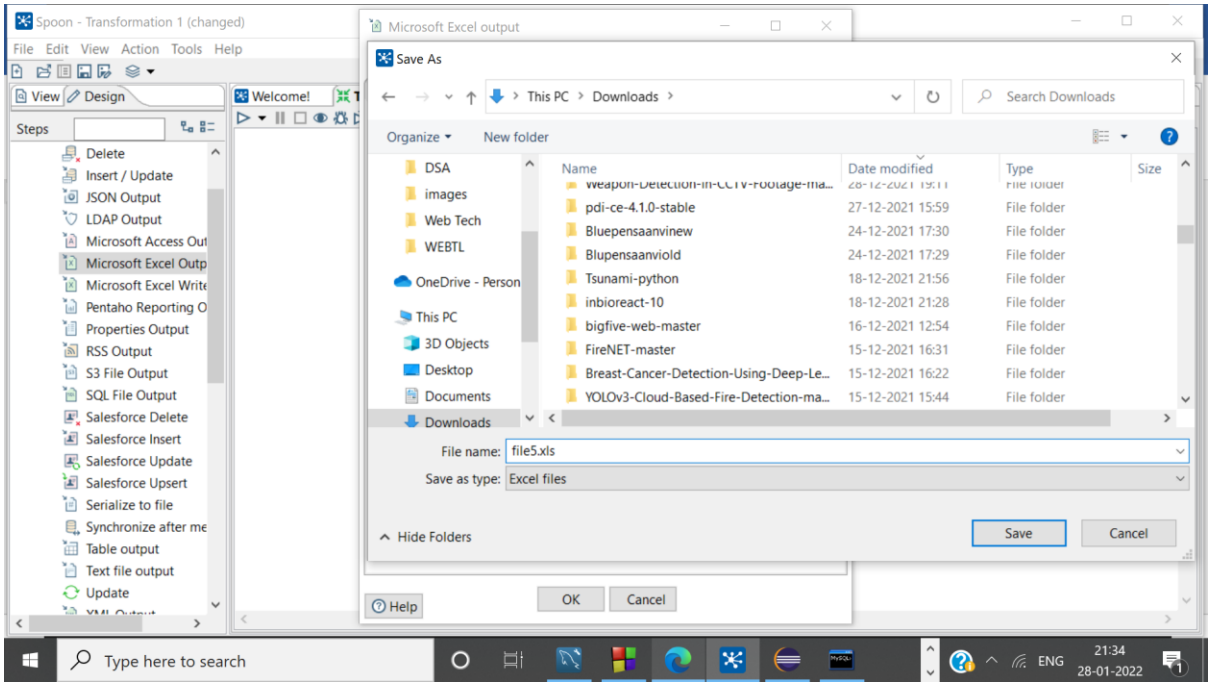
Step 5: Specify the path of input file



Step 6: Check Preview Data

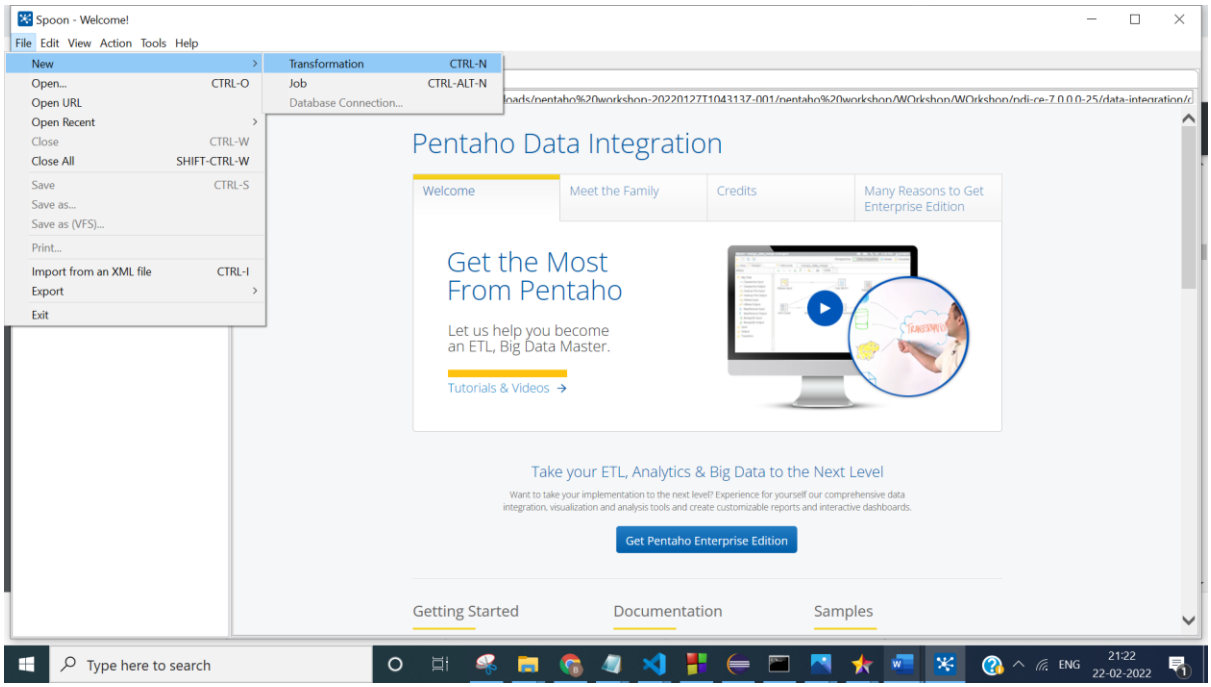


Step 7: Now, drag & drop csv output file from output section

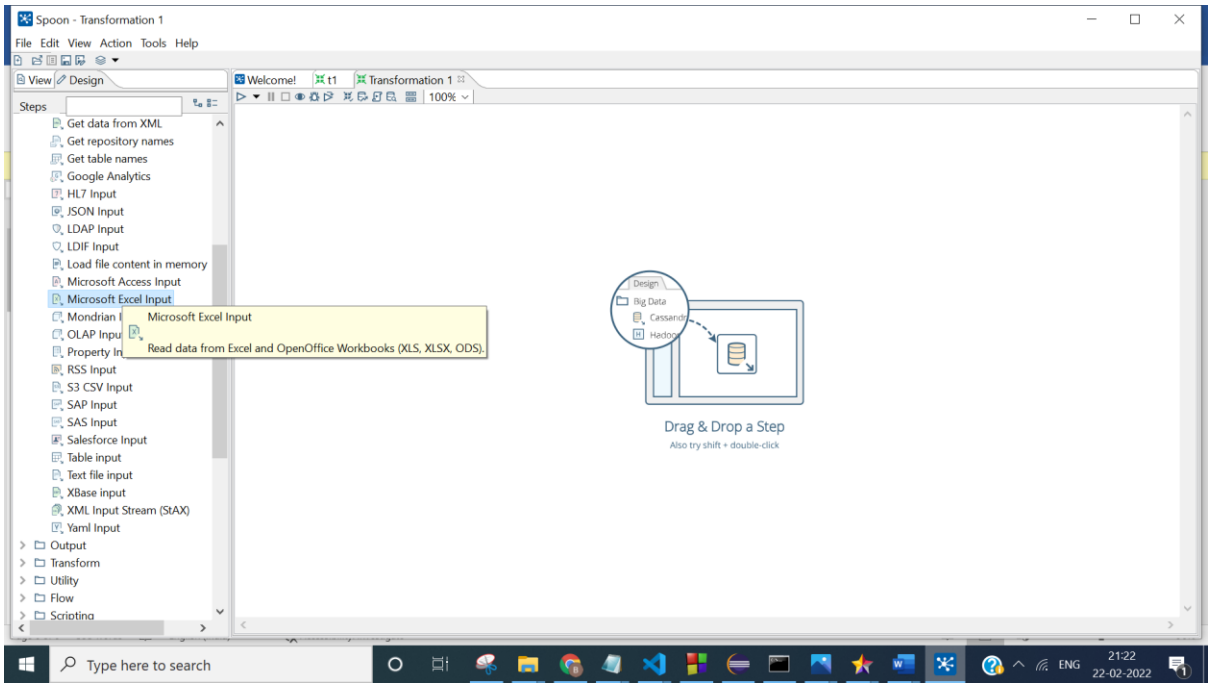


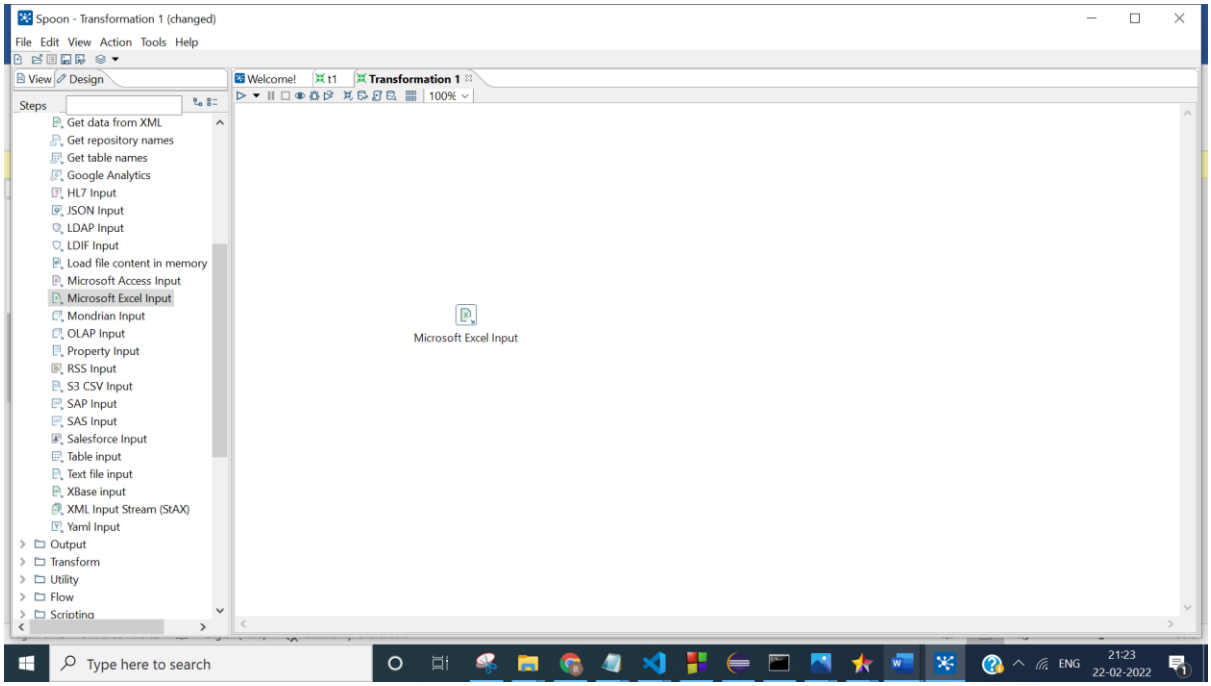
B) SORT ROWS TRANSFORMATIOON:

Step 1. In the Pentaho Data Integration tools (Spoon), Pull down the File menu and select the New menu item followed by Transformation.

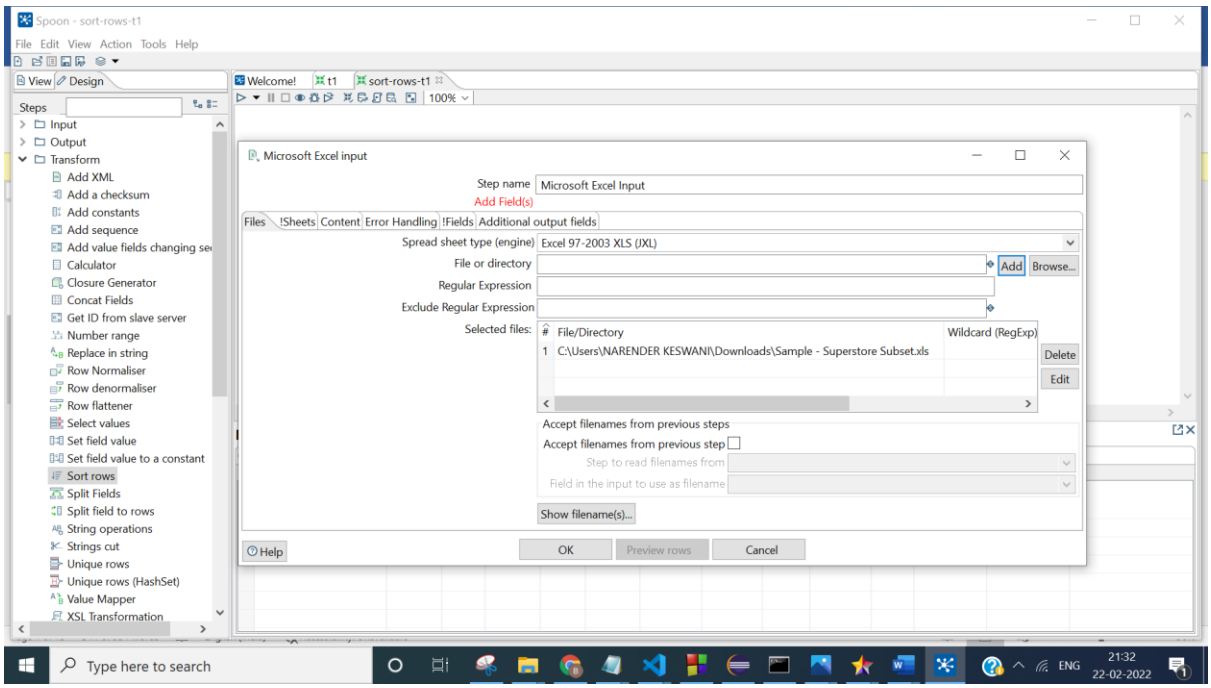


Step 2. Open up the Input folder and drag and drop the Microsoft excel input step on to the transformation window.

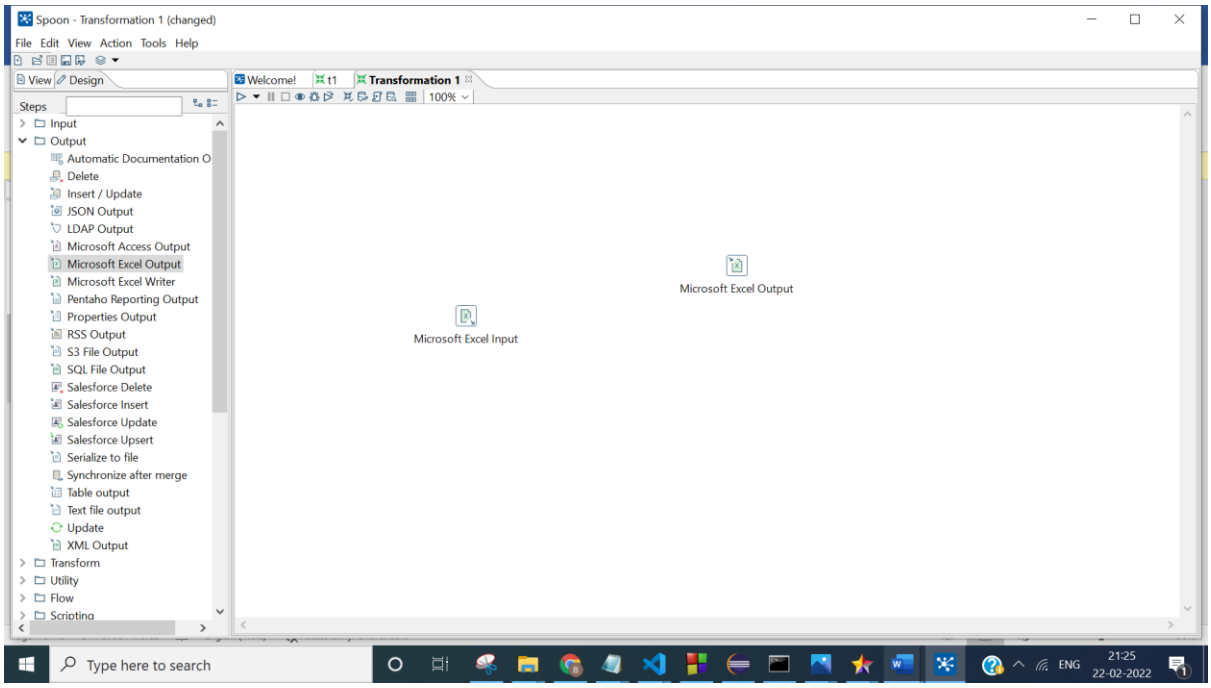


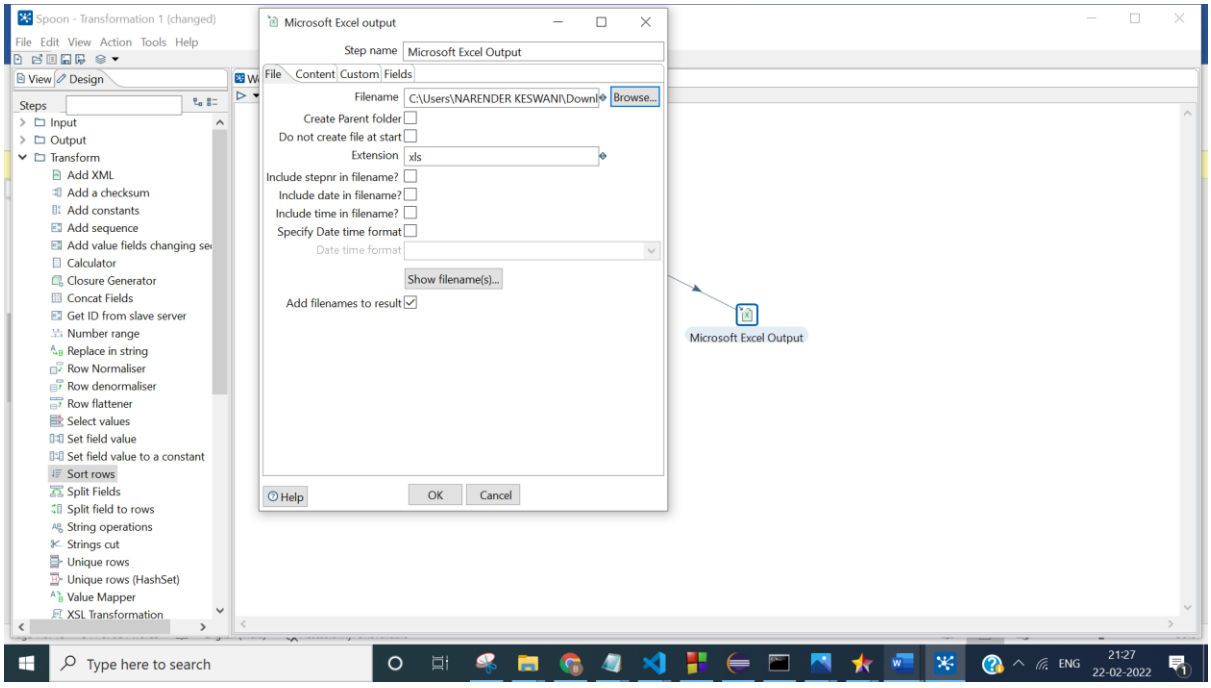


Step 3. Double-click on the Microsoft Excel Input step to view its properties Click on the Browse button next to Filename field and navigate to the folder with the Excel files. Select the Excel file as shown below and then click the Open button. After that click on ADD button to add excel file .

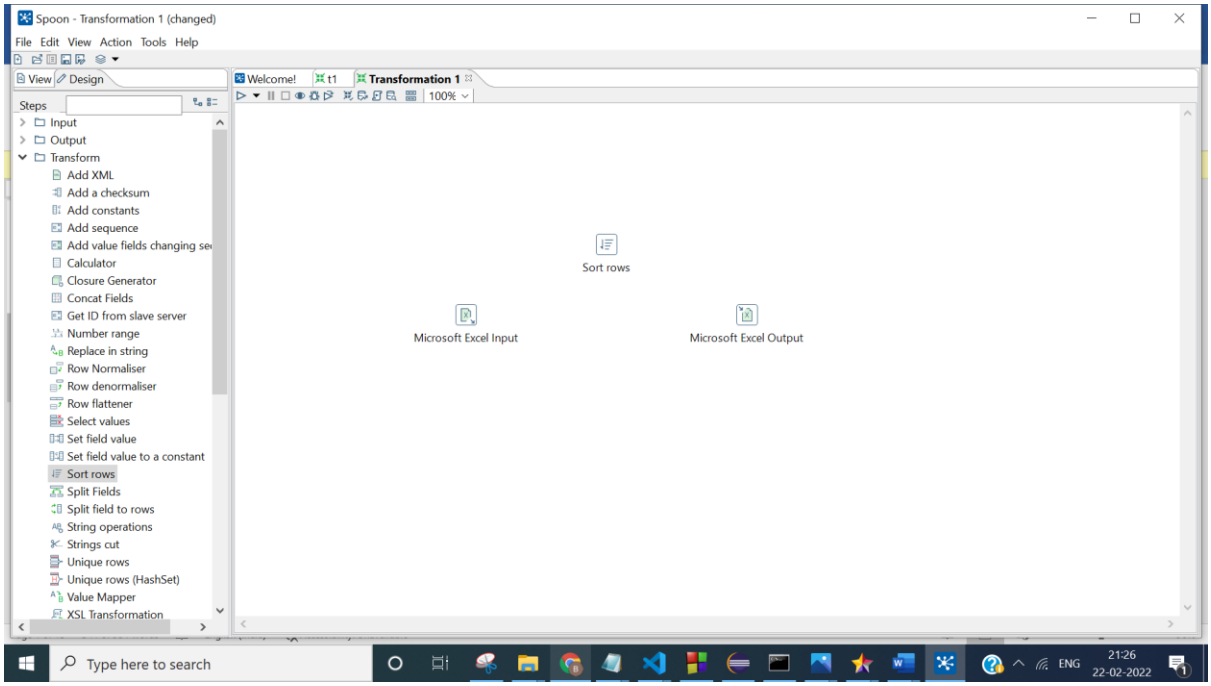


Step 4: Double-click on the Microsoft Excel output step to view its properties Click on the Browse button next to Filename field and navigate to the folder to store the output.

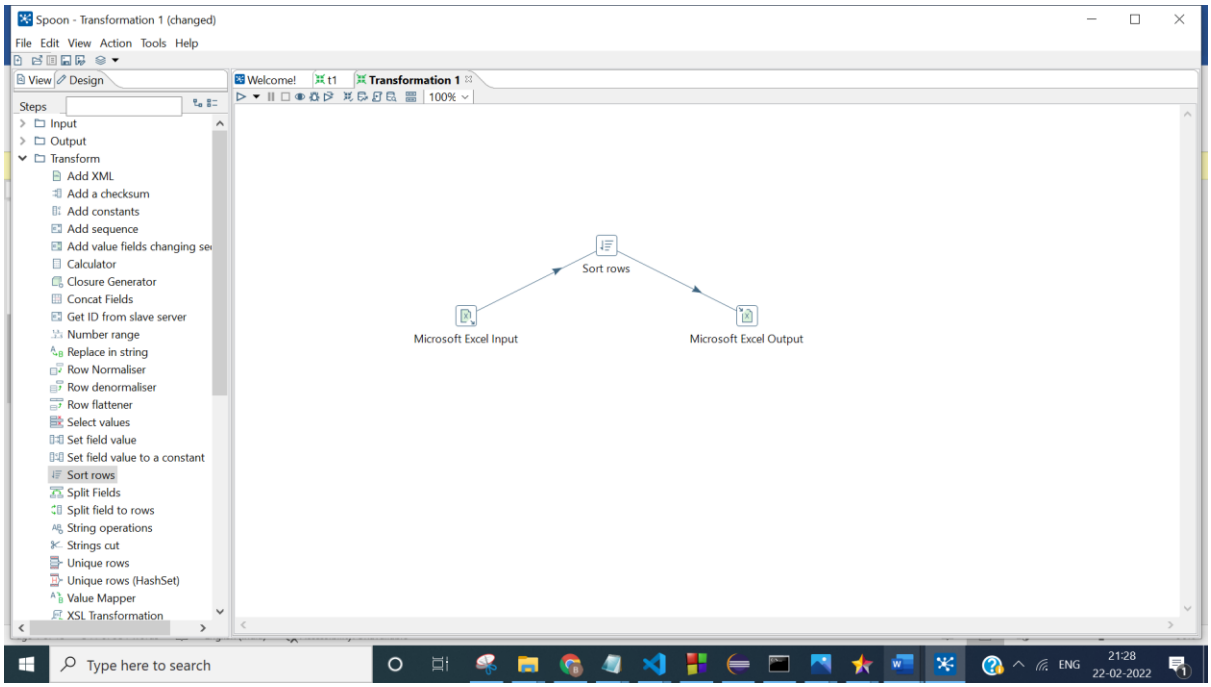




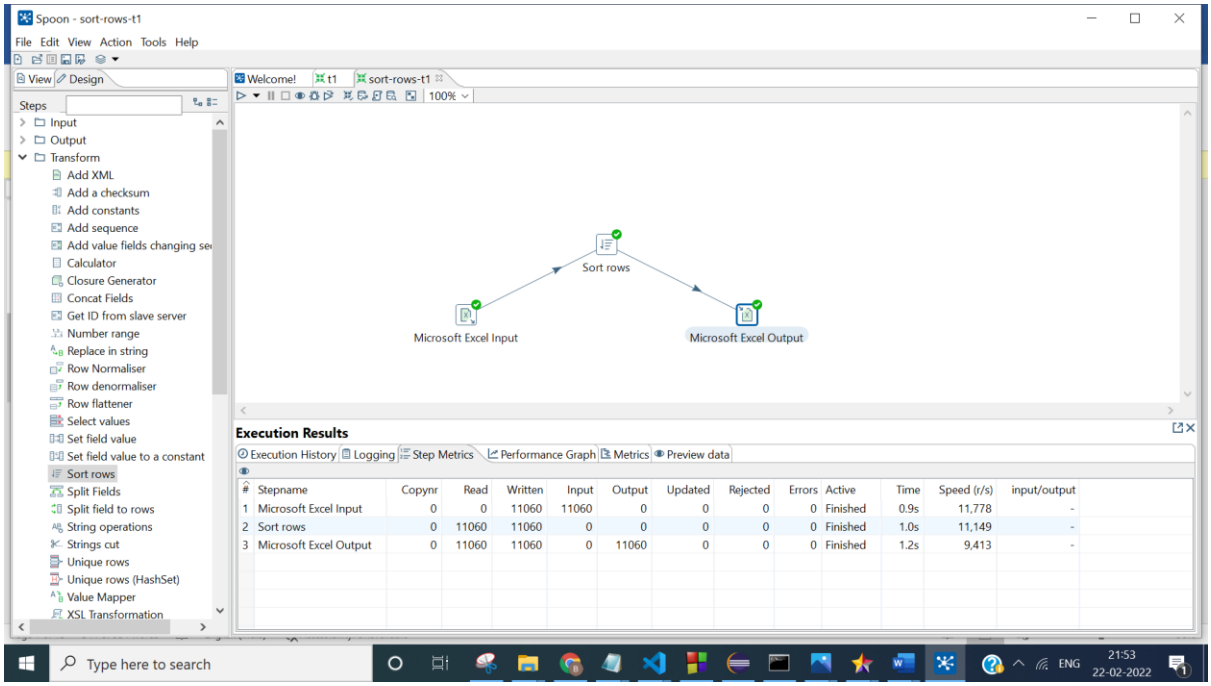
Step 5: Drag & Drop sort rows from the transformation sections.



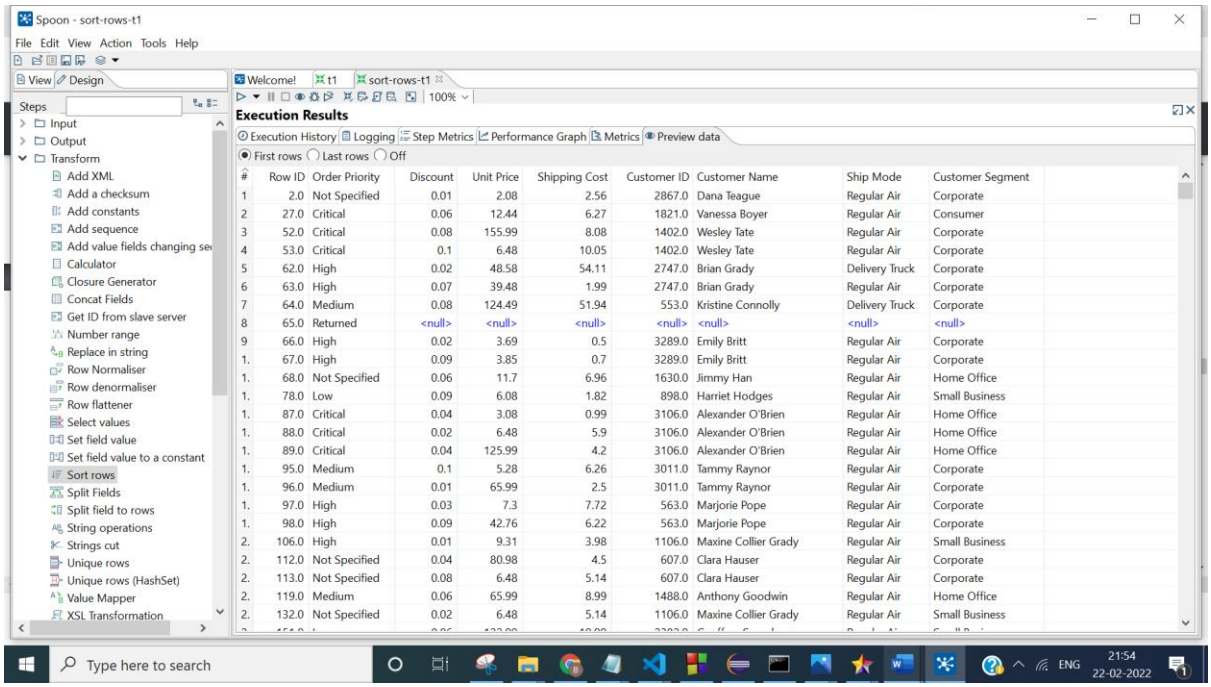
Step 6: Create connections among input, sort & output using hops.

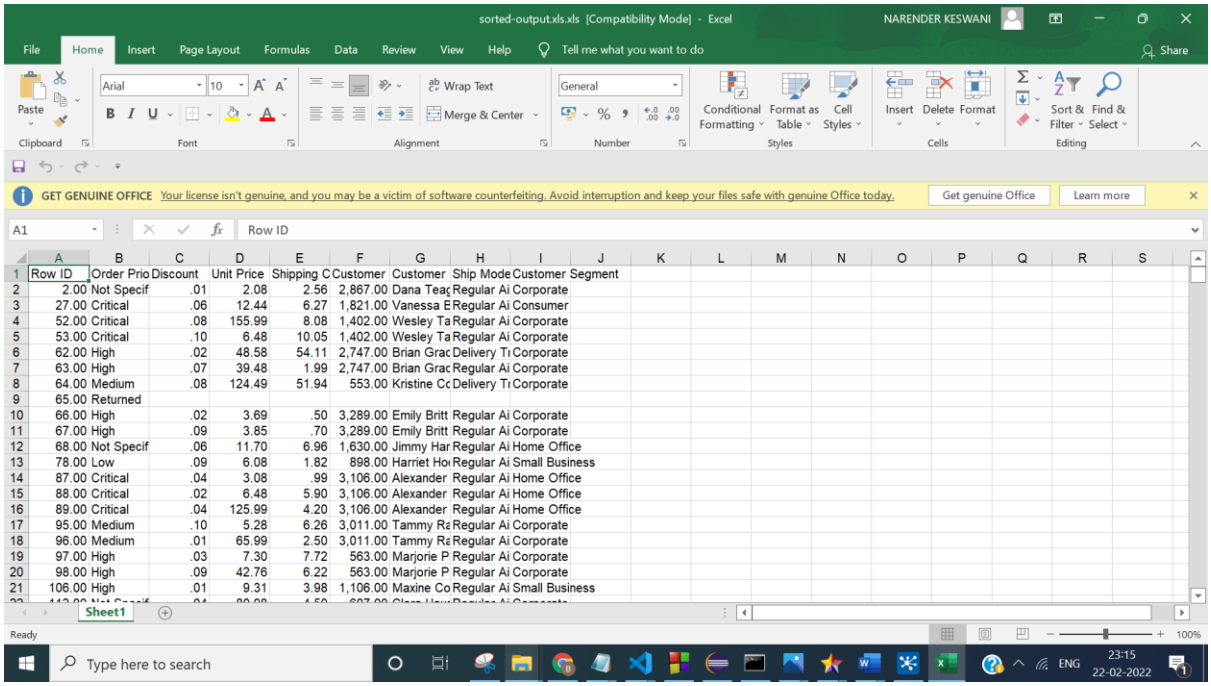


Step 7: Execute the program:



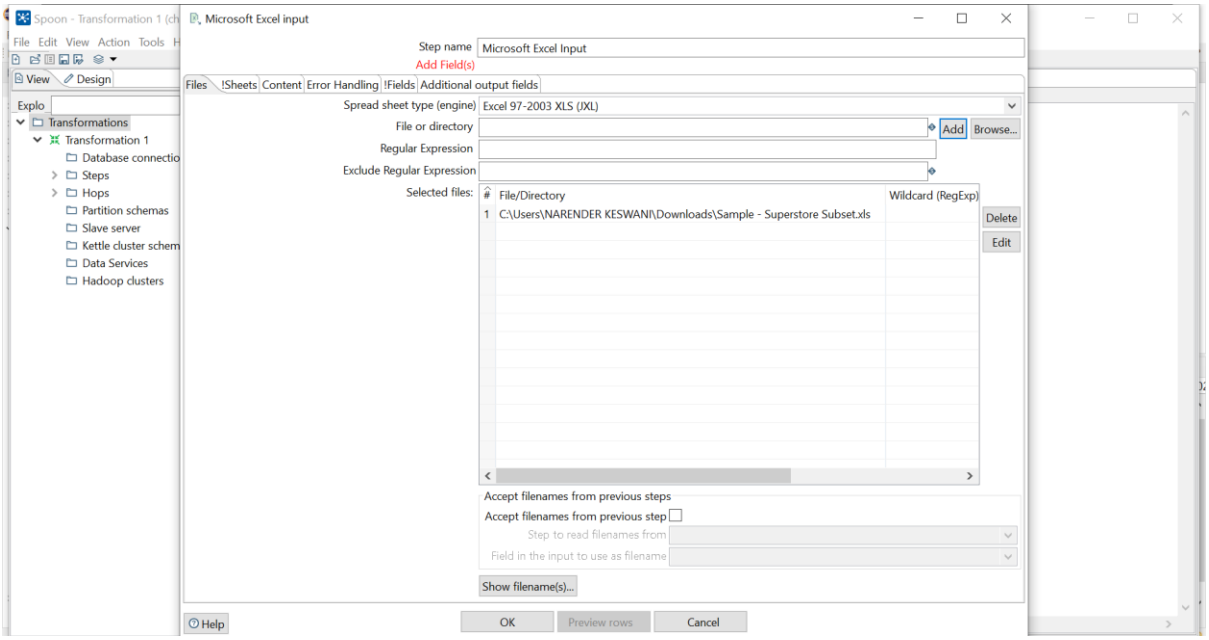
OUTPUT:


Spoon - sort-rows-t1
File Edit View Action Tools Help
Execution Results
Execution History Logging Step Metrics Performance Graph Metrics Preview data
First rows Last rows Off
Row ID Order Priority Discount Unit Price Shipping Cost Customer ID Customer Name Ship Mode Customer Segment
1 2.0 Not Specified 0.01 2.08 2.56 2867.0 Dana Teague Regular Air Corporate
2 27.0 Critical 0.06 12.44 6.27 1821.0 Vanessa Boyer Regular Air Consumer
3 52.0 Critical 0.08 155.99 8.08 1402.0 Wesley Tate Regular Air Corporate
4 53.0 Critical 0.1 6.48 10.05 1402.0 Wesley Tate Regular Air Corporate
5 62.0 High 0.02 48.58 54.11 2747.0 Brian Grady Delivery Truck Corporate
6 63.0 High 0.07 39.48 1.99 2747.0 Brian Grady Regular Air Corporate
7 64.0 Medium 0.08 124.49 51.94 553.0 Kristine Connolly Delivery Truck Corporate
8 65.0 Returned <null> <null> <null> <null> <null> <null> <null>
9 66.0 High 0.02 3.69 0.5 3289.0 Emily Britt Regular Air Corporate
1. 67.0 High 0.09 3.85 0.7 3289.0 Emily Britt Regular Air Corporate
1. 68.0 Not Specified 0.06 11.7 6.96 1630.0 Jimmy Han Regular Air Home Office
1. 78.0 Low 0.09 6.08 1.82 898.0 Harriet Hodges Regular Air Small Business
1. 87.0 Critical 0.04 3.08 0.99 3106.0 Alexander O'Brien Regular Air Home Office
1. 88.0 Critical 0.02 6.48 5.9 3106.0 Alexander O'Brien Regular Air Home Office
1. 89.0 Critical 0.04 125.99 4.2 3106.0 Alexander O'Brien Regular Air Home Office
1. 95.0 Medium 0.1 5.28 6.26 3011.0 Tammy Raynor Regular Air Corporate
1. 96.0 Medium 0.01 65.99 2.5 3011.0 Tammy Raynor Regular Air Corporate
1. 97.0 High 0.03 7.3 7.72 563.0 Marjorie Pope Regular Air Corporate
1. 98.0 High 0.09 42.76 6.22 563.0 Marjorie Pope Regular Air Corporate
2. 106.0 High 0.01 9.31 3.98 1106.0 Maxine Collier Grady Regular Air Small Business
2. 112.0 Not Specified 0.04 80.98 4.5 607.0 Clara Hauser Regular Air Corporate
2. 113.0 Not Specified 0.08 6.48 5.14 607.0 Clara Hauser Regular Air Corporate
2. 119.0 Medium 0.06 65.99 8.99 1488.0 Anthony Goodwin Regular Air Home Office
2. 132.0 Not Specified 0.02 6.48 5.14 1106.0 Maxine Collier Grady Regular Air Small Business
Type here to search
21:54
22-02-2022

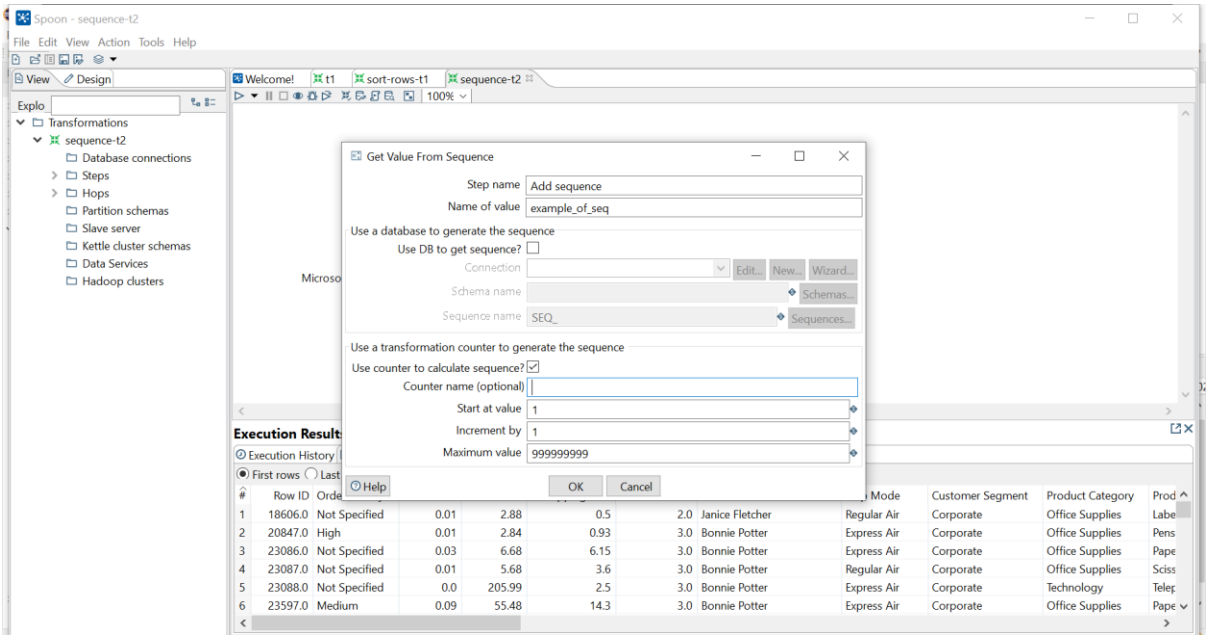

sorted-output.xls.xls [Compatibility Mode] - Excel
NARENDER KESWANI
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today.
Get genuine Office
Learn more
Row ID Order Prio Discount Unit Price Shipping C Customer Customer Ship Mode Customer Segment
2 2.00 Not Specif .01 2.08 2.56 2,867.00 Dana Teag Regular Ai Corporate
3 27.00 Critical .06 12.44 6.27 1,821.00 Vanessa B Regular Ai Consumer
4 52.00 Critical .08 155.99 8.08 1,402.00 Wesley Ta Regular Ai Corporate
5 53.00 Critical .10 6.48 10.05 1,402.00 Wesley Ta Regular Ai Corporate
6 62.00 High .02 48.58 54.11 2,747.00 Brian Grac Delivery Tr Corporate
7 63.00 High .07 39.48 1.99 2,747.00 Brian Grac Regular Ai Corporate
8 64.00 Medium .08 124.49 51.94 553.00 Kristine Cc Delivery Tr Corporate
9 65.00 Returned
10 66.00 High .02 3.69 .50 3,289.00 Emily Britt Regular Ai Corporate
11 67.00 High .09 3.85 .70 3,289.00 Emily Britt Regular Ai Corporate
12 68.00 Not Specif .06 11.70 6.96 1,630.00 Jimmy Har Regular Ai Home Office
13 78.00 Low .09 6.08 1.82 898.00 Harriet Hoi Regular Ai Small Business
14 87.00 Critical .04 3.08 .99 3,106.00 Alexander Regular Ai Home Office
15 88.00 Critical .02 6.48 5.90 3,106.00 Alexander Regular Ai Home Office
16 89.00 Critical .04 125.99 4.20 3,106.00 Alexander Regular Ai Home Office
17 95.00 Medium .10 5.28 6.26 3,011.00 Tammy Ra Regular Ai Corporate
18 96.00 Medium .01 65.99 2.50 3,011.00 Tammy Ra Regular Ai Corporate
19 97.00 High .03 7.30 7.72 563.00 Marjorie P Regular Ai Corporate
20 98.00 High .09 42.76 6.22 563.00 Marjorie P Regular Ai Corporate
21 106.00 High .01 9.31 3.98 1,106.00 Maxine Co Regular Ai Small Business
Sheet1
Ready
Type here to search
23:15
22-02-2022

C) SEQUENCE TRANSFORMATION:

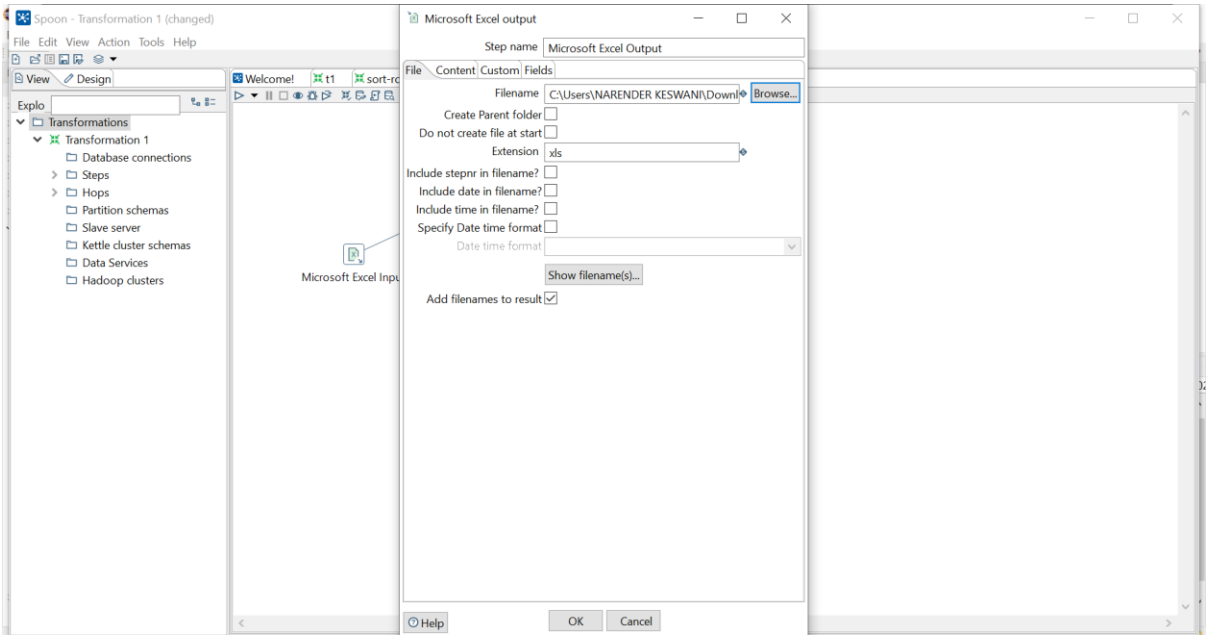
Step 1. Open up the Input folder and drag and drop the Microsoft excel input step on to the transformation window.



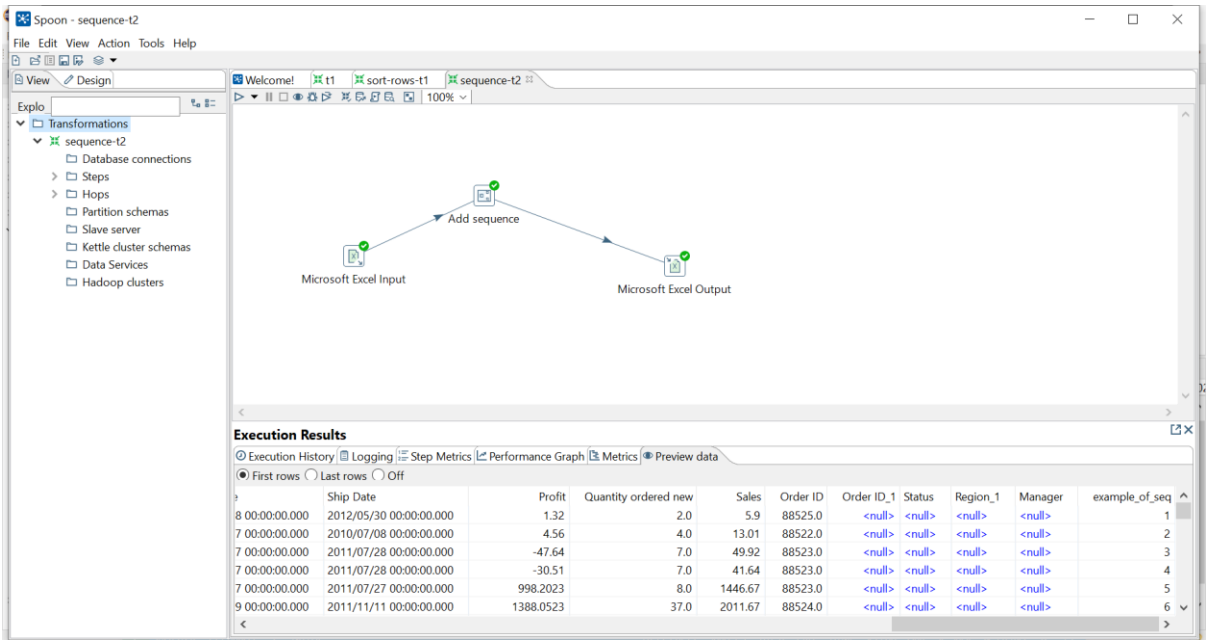
Step 2. Open up the Transformation folder and drag and drop the Add Sequence and config according to requirements



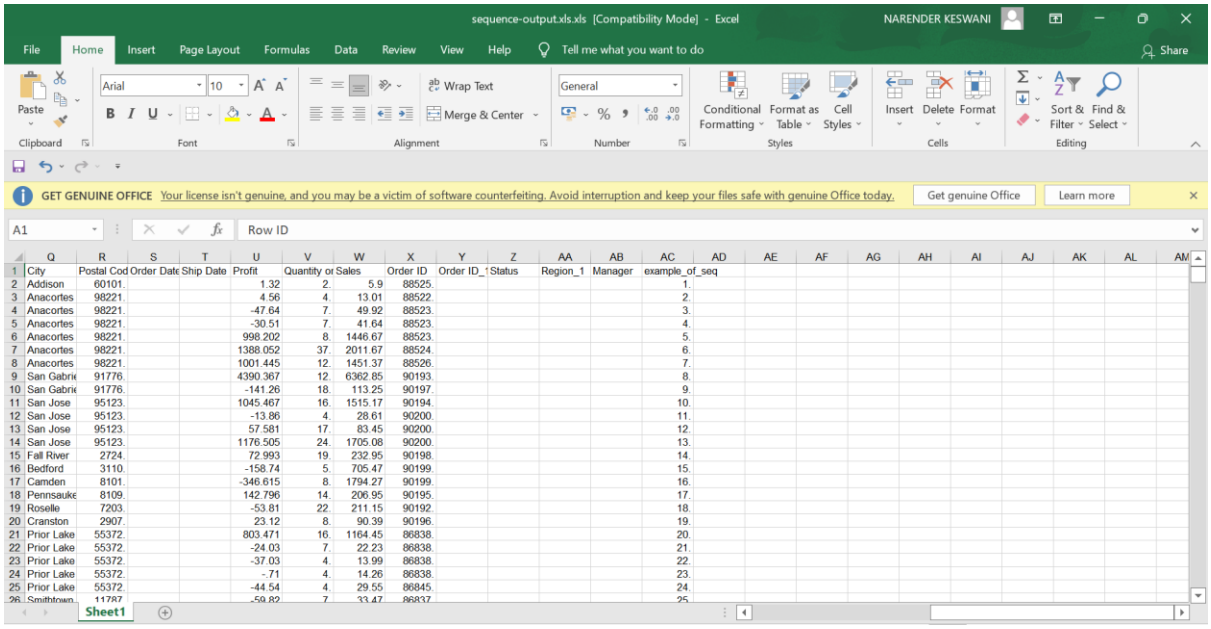
Step 3. Open up the Output folder and drag and drop the Microsoft excel output and config it.



Step 4. Execute the transformation.



OUTPUT:

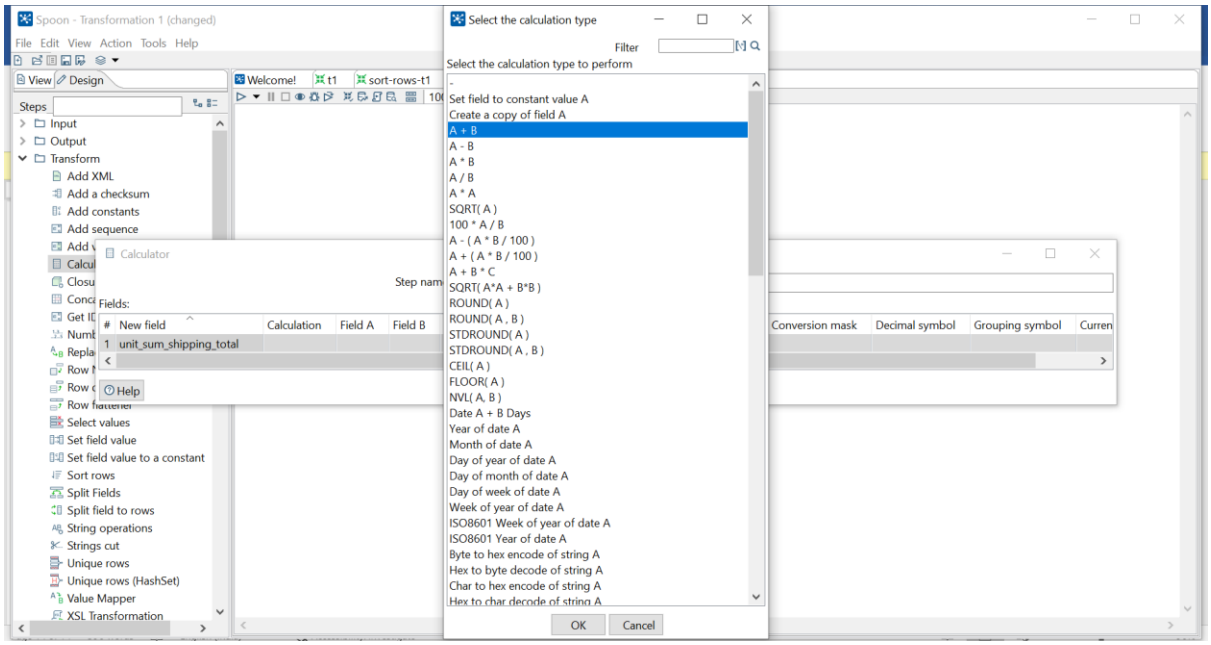


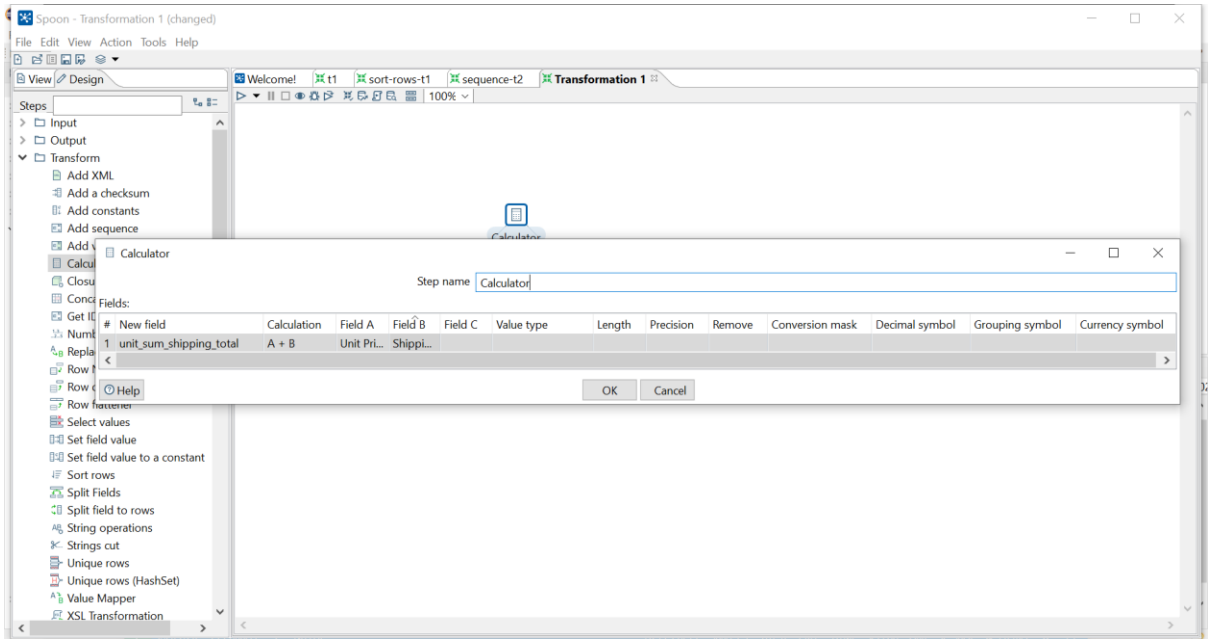
	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1	City	Postal Cod	Order Date	Ship Date	Profit	Quantity or	Sales	Order ID	Order ID_1	Status	Region_1	Manager	example_of_seq										
2	Addison	60101.			1.32	2.	5.9	88525.					1.										
3	Anacortes	98221.			4.56	4.	13.01	88522.					2.										
4	Anacortes	98221.			-47.64	7.	49.92	88523.					3.										
5	Anacortes	98221.			-30.51	7.	41.64	88523.					4.										
6	Anacortes	98221.			998.202	8.	1446.67	88523.					5.										
7	Anacortes	98221.			1388.052	37.	2011.67	88524.					6.										
8	Anacortes	98221.			1001.445	12.	1451.37	88526.					7.										
9	San Gabrie	91776.			4390.367	12.	6362.85	90193.					8.										
10	San Gabrie	91776.			-141.26	18.	113.25	90197.					9.										
11	San Jose	95123.			1045.467	16.	1515.17	90194.					10.										
12	San Jose	95123.			-13.86	4.	28.61	90200.					11.										
13	San Jose	95123.			57.581	17.	83.45	90200.					12.										
14	San Jose	95123.			1176.505	24.	1705.08	90200.					13.										
15	Fall River	2724.			72.993	19.	232.95	90198.					14.										
16	Bedford	3110.			-158.74	5.	705.47	90199.					15.										
17	Camden	8101.			-346.615	8.	1794.27	90199.					16.										
18	Pennsauke	8109.			142.796	14.	206.95	90195.					17.										
19	Roselle	7203.			-53.81	22.	211.15	90192.					18.										
20	Cranston	2907.			23.12	8.	90.39	90196.					19.										
21	Prior Lake	55372.			803.471	16.	1164.45	86838.					20.										
22	Prior Lake	55372.			-24.03	7.	22.23	86838.					21.										
23	Prior Lake	55372.			-37.03	4.	13.99	86838.					22.										
24	Prior Lake	55372.			-.71	4.	14.26	86838.					23.										
25	Prior Lake	55372.			-44.54	4.	29.55	86845.					24.										
26	Smithtown	11787.			-59.82	7.	33.47	86837.					25.										

D) CALCULATOR (SUM):

Step 1. Open up the Input folder and drag and drop the Microsoft excel input step on to the transformation window.

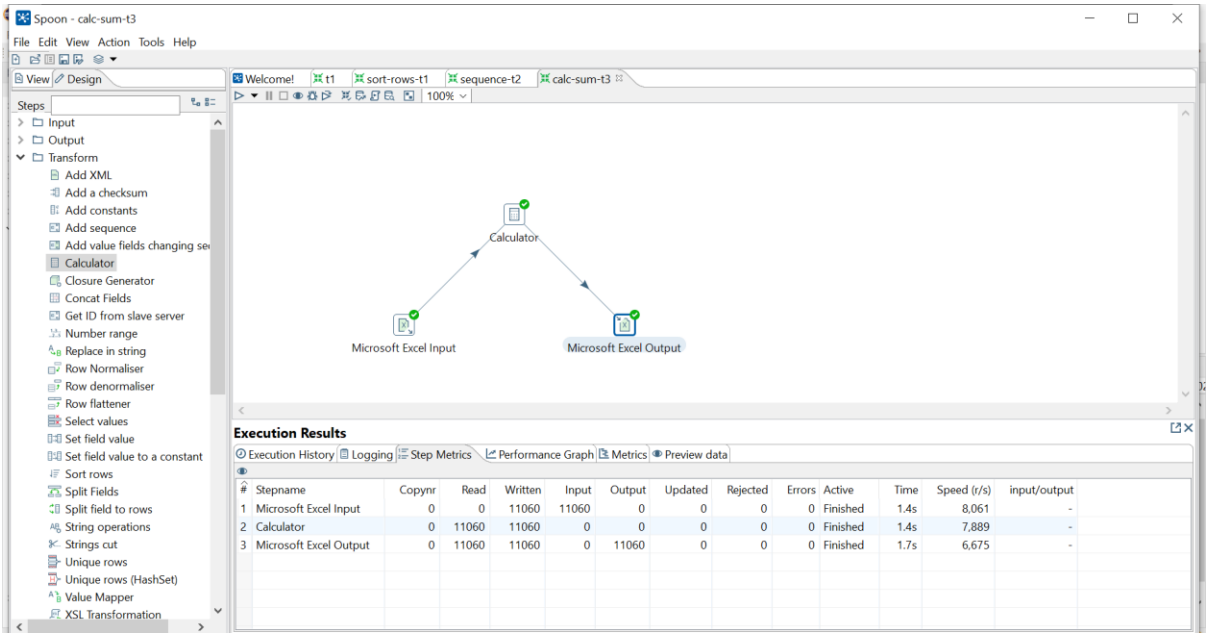
Step 2. Open up the Transformation folder and drag and drop the calculator and config it according to requirements.



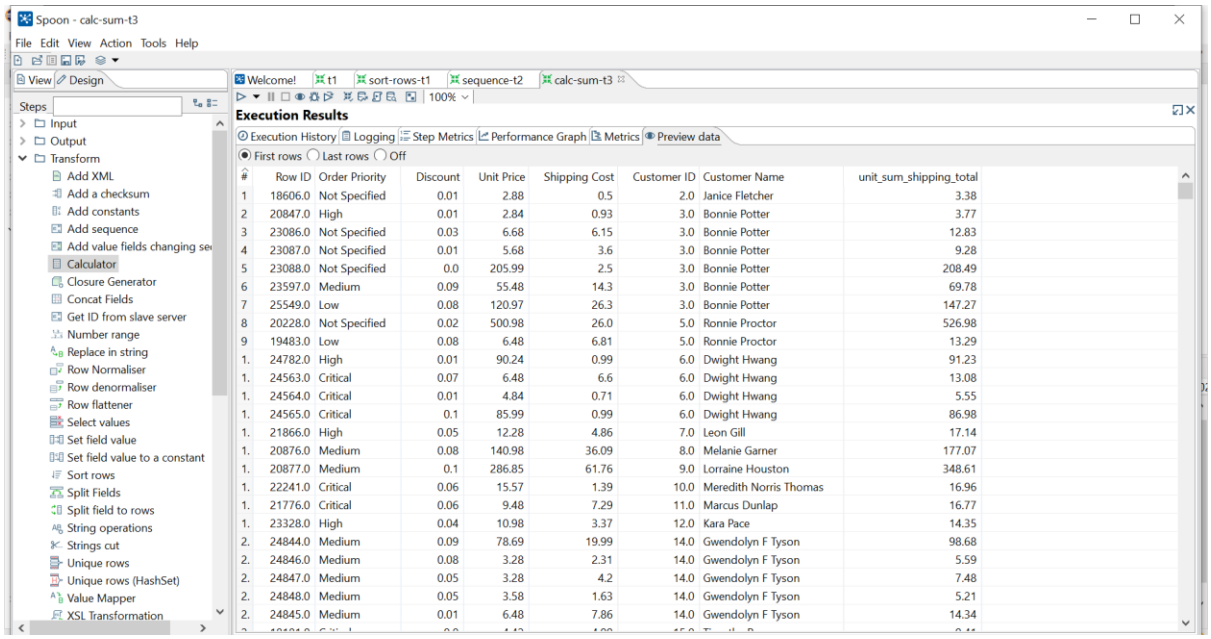


Step 3. Open up the output folder and drag and drop the Microsoft excel output step on to the transformation window.

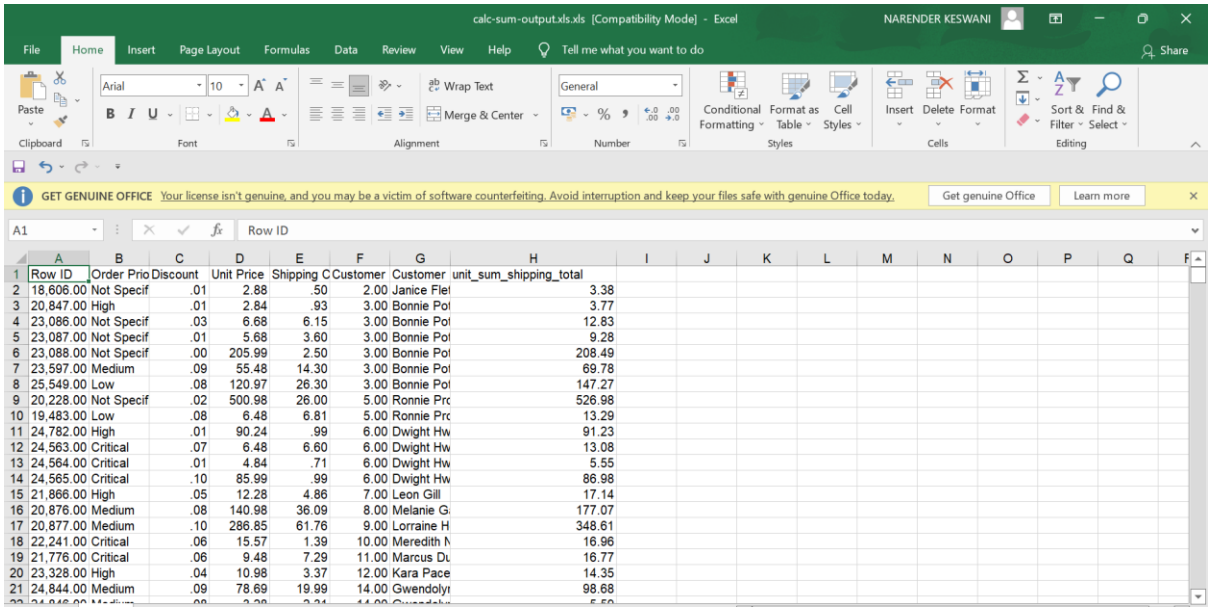
Step 4. Execute the transformation



OUTPUT:



#	Row ID	Order Priority	Discount	Unit Price	Shipping Cost	Customer ID	Customer Name	unit_sum_shipping_total
1	18606.0	Not Specified	0.01	2.88	0.5	2.0	Janice Fletcher	3.38
2	20847.0	High	0.01	2.84	0.93	3.0	Bonnie Potter	3.77
3	23086.0	Not Specified	0.03	6.68	6.15	3.0	Bonnie Potter	12.83
4	23087.0	Not Specified	0.01	5.68	3.6	3.0	Bonnie Potter	9.28
5	23088.0	Not Specified	0.0	205.99	2.5	3.0	Bonnie Potter	208.49
6	23597.0	Medium	0.09	55.48	14.3	3.0	Bonnie Potter	69.78
7	25549.0	Low	0.08	120.97	26.3	3.0	Bonnie Potter	147.27
8	20228.0	Not Specified	0.02	500.98	26.0	5.0	Ronnie Proctor	526.98
9	19483.0	Low	0.08	6.48	6.81	5.0	Ronnie Proctor	13.29
1.	24782.0	High	0.01	90.24	0.99	6.0	Dwight Hwang	91.23
1.	24563.0	Critical	0.07	6.48	6.6	6.0	Dwight Hwang	13.08
1.	24564.0	Critical	0.01	4.84	0.71	6.0	Dwight Hwang	5.55
1.	24565.0	Critical	0.1	85.99	0.99	6.0	Dwight Hwang	86.98
1.	21866.0	High	0.05	12.28	4.86	7.0	Leon Gill	17.14
1.	20876.0	Medium	0.08	140.98	36.09	8.0	Melanie Garner	177.07
1.	20877.0	Medium	0.1	286.85	61.76	9.0	Lorraine Houston	348.61
1.	22241.0	Critical	0.06	15.57	1.39	10.0	Meredith Norris Thomas	16.96
1.	21776.0	Critical	0.06	9.48	7.29	11.0	Marcus Dunlap	16.77
1.	23328.0	High	0.04	10.98	3.37	12.0	Kara Pace	14.35
2.	24844.0	Medium	0.09	78.69	19.99	14.0	Gwendolyn F Tyson	98.68
2.	24846.0	Medium	0.08	3.28	2.31	14.0	Gwendolyn F Tyson	5.59
2.	24847.0	Medium	0.05	3.28	4.2	14.0	Gwendolyn F Tyson	7.48
2.	24848.0	Medium	0.05	3.58	1.63	14.0	Gwendolyn F Tyson	5.21
2.	24845.0	Medium	0.01	6.48	7.86	14.0	Gwendolyn F Tyson	14.34



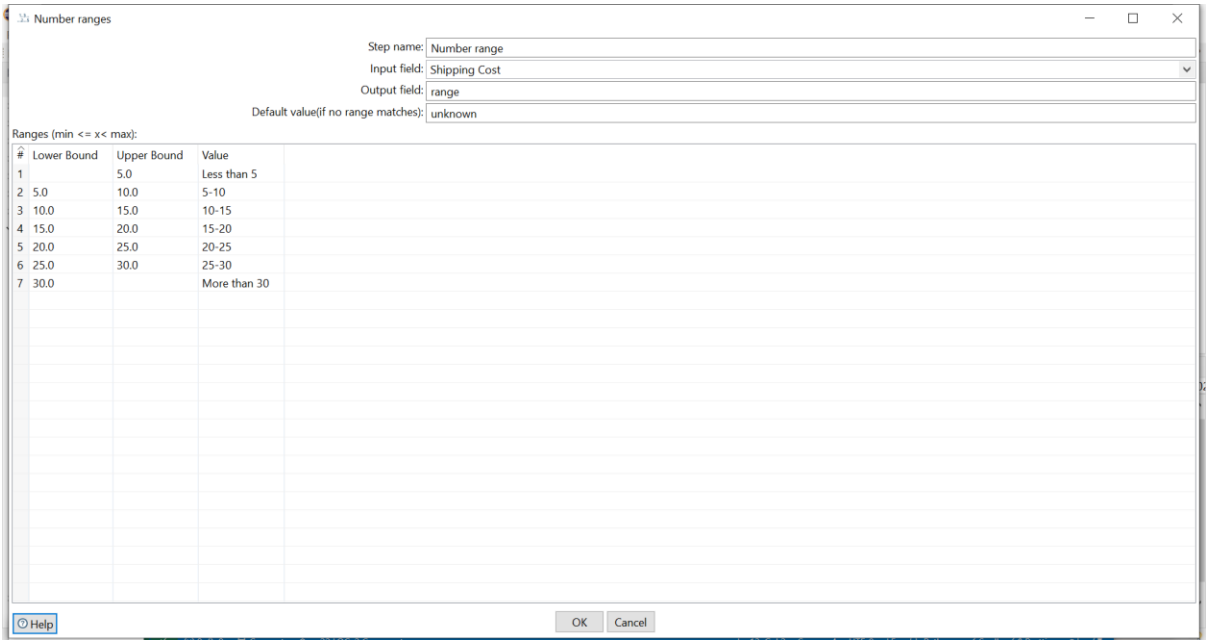
	A	B	C	D	E	F	G	H
1	Row ID	Order Prio	Discount	Unit Price	Shipping C	Customer	Customer	unit_sum_shipping_total
2	18,606.00	Not Specif	.01	2.88	.50	2.00	Janice Flet	3.38
3	20,847.00	High	.01	2.84	.93	3.00	Bonnie Pot	3.77
4	23,086.00	Not Specif	.03	6.68	6.15	3.00	Bonnie Pot	12.83
5	23,087.00	Not Specif	.01	5.68	3.60	3.00	Bonnie Pot	9.28
6	23,088.00	Not Specif	.00	205.99	2.50	3.00	Bonnie Pot	208.49
7	23,597.00	Medium	.09	55.48	14.30	3.00	Bonnie Pot	69.78
8	25,549.00	Low	.08	120.97	26.30	3.00	Bonnie Pot	147.27
9	20,228.00	Not Specif	.02	500.98	26.00	5.00	Ronnie Prc	526.98
10	19,483.00	Low	.08	6.48	6.81	5.00	Ronnie Prc	13.29
11	24,782.00	High	.01	90.24	.99	6.00	Dwight Hw	91.23
12	24,563.00	Critical	.07	6.48	6.60	6.00	Dwight Hw	13.08
13	24,564.00	Critical	.01	4.84	.71	6.00	Dwight Hw	5.55
14	24,565.00	Critical	.10	85.99	.99	6.00	Dwight Hw	86.98
15	21,866.00	High	.05	12.28	4.86	7.00	Leon Gill	17.14
16	20,876.00	Medium	.08	140.98	36.09	8.00	Melanie G	177.07
17	20,877.00	Medium	.10	286.85	61.76	9.00	Lorraine H	348.61
18	22,241.00	Critical	.06	15.57	1.39	10.00	Meredith N	16.96
19	21,776.00	Critical	.06	9.48	7.29	11.00	Marcus Du	16.77
20	23,328.00	High	.04	10.98	3.37	12.00	Kara Pace	14.35
21	24,844.00	Medium	.09	78.69	19.99	14.00	Gwendolyr	98.68

E) Number Range:

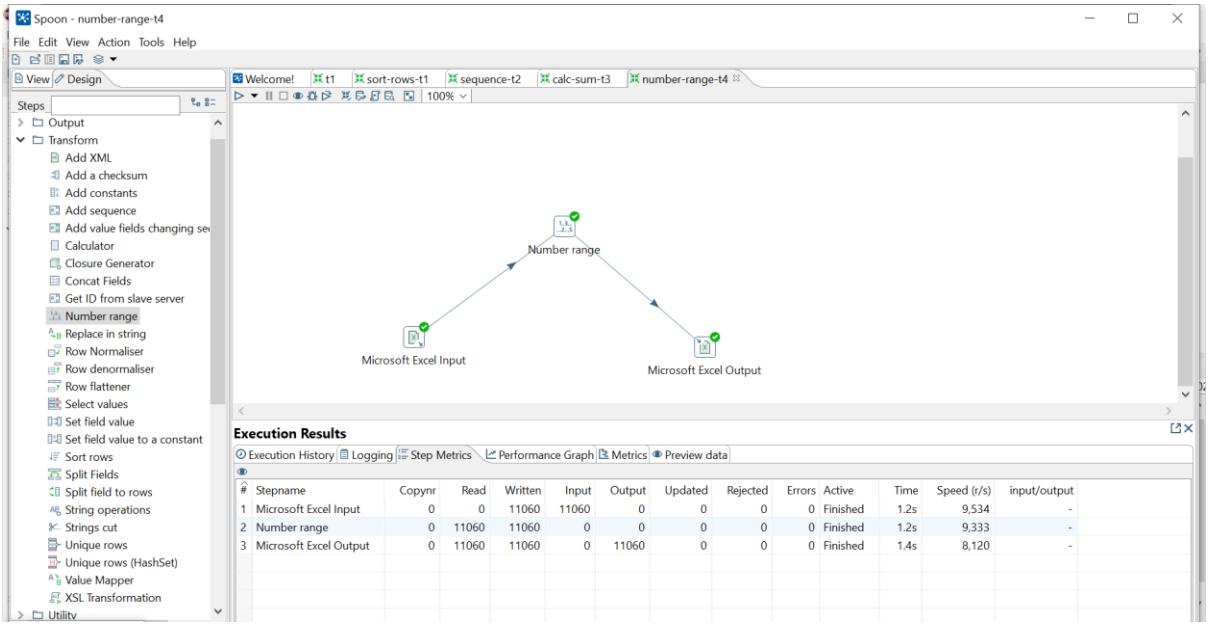
Step 1. Open up the Input folder and drag and drop the Microsoft excel input step on to the transformation window.

Step 2. Open up the Transformation folder and drag and drop the number range and config it according to requirements.

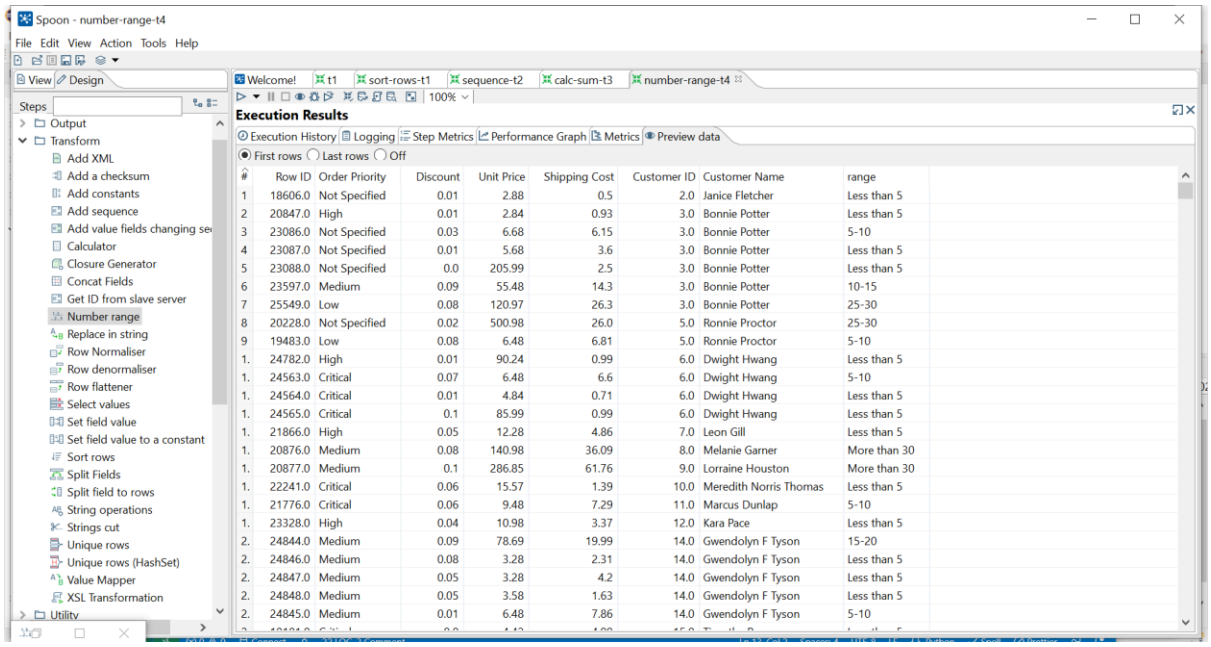
Step 3. Open up the output folder and drag and drop the Microsoft excel output.

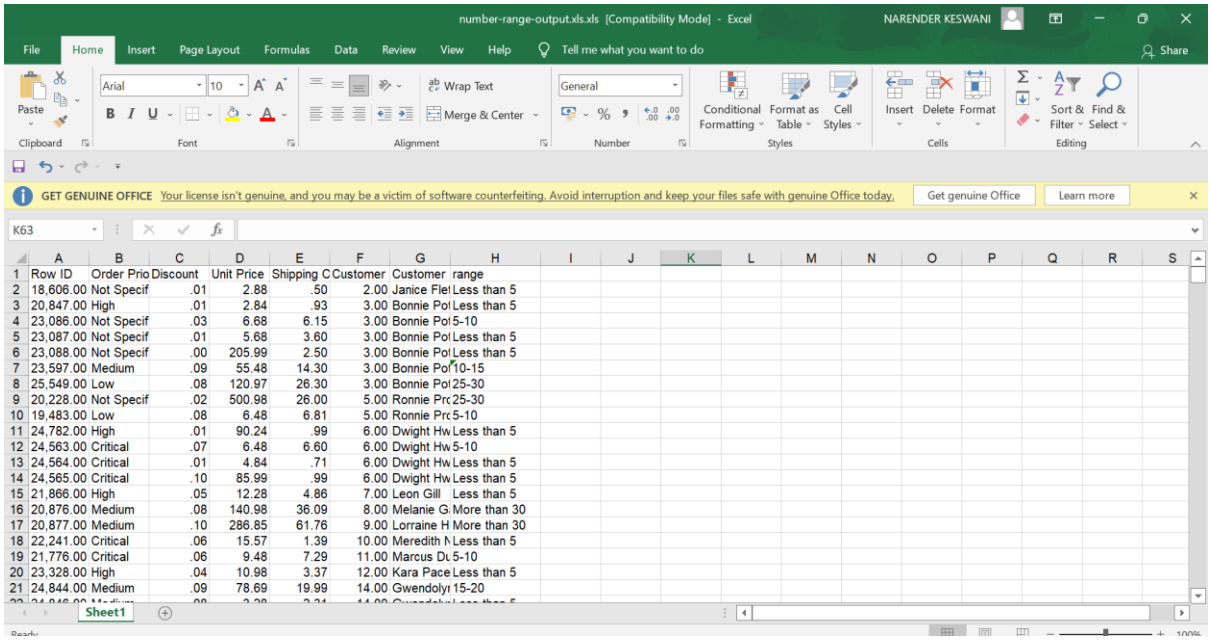


Step 4. Execute the transformation



OUTPUT:


Spoon - number-range-t4
File Edit View Action Tools Help
Welcome! t1 sort-rows-t1 sequence-t2 calc-sum-t3 number-range-t4
Execution Results
Execution History Logging Step Metrics Performance Graph Metrics Preview data
First rows Last rows Off
Steps
Output
Transform
Add XML
Add a checksum
Add constants
Add sequence
Add value fields changing se
Calculator
Closure Generator
Concat Fields
Get ID from slave server
Number range
Replace in string
Row Normaliser
Row denormaliser
Row flattener
Select values
Set field value
Set field value to a constant
Sort rows
Split Fields
Split field to rows
String operations
Strings cut
Unique rows
Unique rows (HashSet)
Value Mapper
XSL Transformation
Utility
Row ID Order Priority Discount Unit Price Shipping Cost Customer ID Customer Name range
1 18606.0 Not Specified 0.01 2.88 0.5 2.0 Janice Fletcher Less than 5
2 20847.0 High 0.01 2.84 0.93 3.0 Bonnie Potter Less than 5
3 23086.0 Not Specified 0.03 6.68 6.15 3.0 Bonnie Potter 5-10
4 23087.0 Not Specified 0.01 5.68 3.6 3.0 Bonnie Potter Less than 5
5 23088.0 Not Specified 0.0 205.99 2.5 3.0 Bonnie Potter Less than 5
6 23597.0 Medium 0.09 55.48 14.3 3.0 Bonnie Potter 10-15
7 25549.0 Low 0.08 120.97 26.3 3.0 Bonnie Potter 25-30
8 20228.0 Not Specified 0.02 500.98 26.0 5.0 Ronnie Proctor 25-30
9 19483.0 Low 0.08 6.48 6.81 5.0 Ronnie Proctor 5-10
1. 24782.0 High 0.01 90.24 0.99 6.0 Dwight Hwang Less than 5
1. 24563.0 Critical 0.07 6.48 6.6 6.0 Dwight Hwang 5-10
1. 24564.0 Critical 0.01 4.84 0.71 6.0 Dwight Hwang Less than 5
1. 24565.0 Critical 0.1 85.99 0.99 6.0 Dwight Hwang Less than 5
1. 21866.0 High 0.05 12.28 4.86 7.0 Leon Gill Less than 5
1. 20876.0 Medium 0.08 140.98 36.09 8.0 Melanie Garner More than 30
1. 20877.0 Medium 0.1 286.85 61.76 9.0 Lorraine Houston More than 30
1. 22241.0 Critical 0.06 15.57 1.39 10.0 Meredith Norris Thomas Less than 5
1. 21776.0 Critical 0.06 9.48 7.29 11.0 Marcus Dunlap 5-10
1. 23328.0 High 0.04 10.98 3.37 12.0 Kara Pace Less than 5
2. 24844.0 Medium 0.09 78.69 19.99 14.0 Gwendolyn F Tyson 15-20
2. 24846.0 Medium 0.08 3.28 2.31 14.0 Gwendolyn F Tyson Less than 5
2. 24847.0 Medium 0.05 3.28 4.2 14.0 Gwendolyn F Tyson Less than 5
2. 24848.0 Medium 0.05 3.58 1.63 14.0 Gwendolyn F Tyson Less than 5
2. 24845.0 Medium 0.01 6.48 7.86 14.0 Gwendolyn F Tyson 5-10

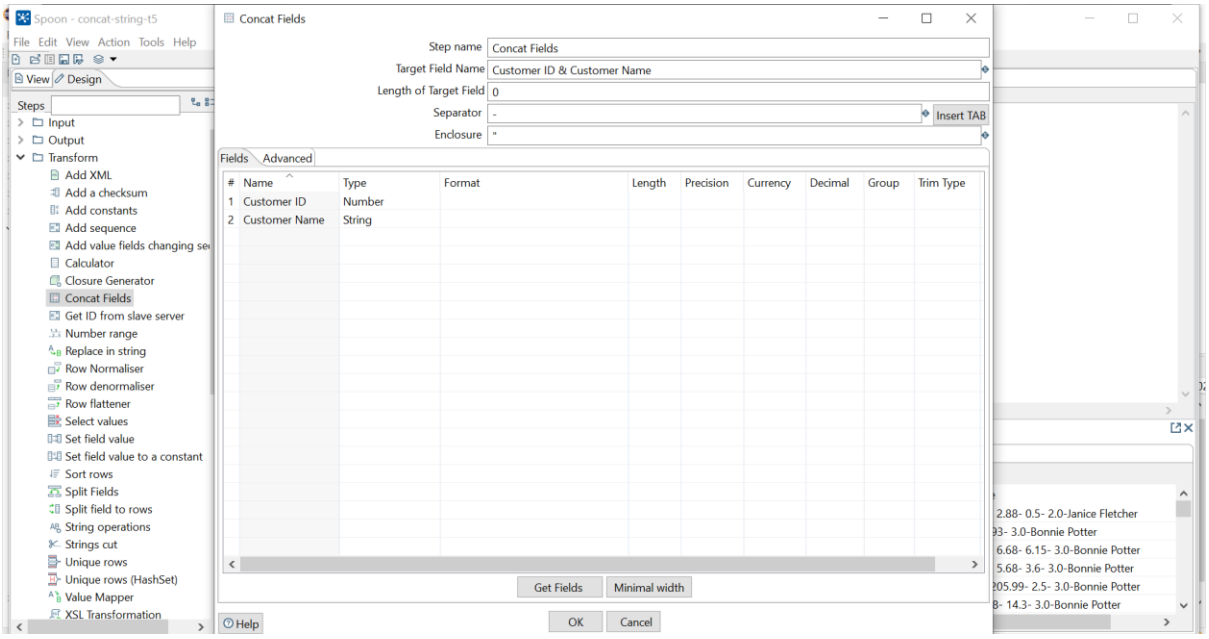

number-range-output.xls.xls [Compatibility Mode] - Excel
NARENDER KESWANI
File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do
Share
Paste Clipboard Arial 10 Font Wrap Text Merge & Center Alignment General Number
Conditional Formatting Format as Table Cell Styles Styles
Insert Delete Format Cells
Sort & Filter Find & Select Editing
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today.
Get genuine Office Learn more
K63
Row ID Order Prio Discount Unit Price Shipping C Customer Customer range
18,606.00 Not Specif .01 2.88 .50 2.00 Janice Fle Less than 5
20,847.00 High .01 2.84 .93 3.00 Bonnie Pol Less than 5
23,086.00 Not Specif .03 6.68 6.15 3.00 Bonnie Pol 5-10
23,087.00 Not Specif .01 5.68 3.60 3.00 Bonnie Pol Less than 5
23,088.00 Not Specif .00 205.99 2.50 3.00 Bonnie Pol Less than 5
23,597.00 Medium .09 55.48 14.30 3.00 Bonnie Pol 10-15
25,549.00 Low .08 120.97 26.30 3.00 Bonnie Pol 25-30
20,228.00 Not Specif .02 500.98 26.00 5.00 Ronnie Prc 25-30
19,483.00 Low .08 6.48 6.81 5.00 Ronnie Prc 5-10
24,782.00 High .01 90.24 .99 6.00 Dwight Hw Less than 5
24,563.00 Critical .07 6.48 6.60 6.00 Dwight Hw 5-10
24,564.00 Critical .01 4.84 .71 6.00 Dwight Hw Less than 5
24,565.00 Critical .10 85.99 .99 6.00 Dwight Hw Less than 5
21,866.00 High .05 12.28 4.86 7.00 Leon Gill Less than 5
20,876.00 Medium .08 140.98 36.09 8.00 Melanie G More than 30
20,877.00 Medium .10 286.85 61.76 9.00 Lorraine H More than 30
22,241.00 Critical .06 15.57 1.39 10.00 Meredith N Less than 5
21,776.00 Critical .06 9.48 7.29 11.00 Marcus Du 5-10
23,328.00 High .04 10.98 3.37 12.00 Kara Pace Less than 5
24,844.00 Medium .09 78.69 19.99 14.00 Gwendolyn 15-20
Sheet1

F) CONCAT TRANSFORMATION:

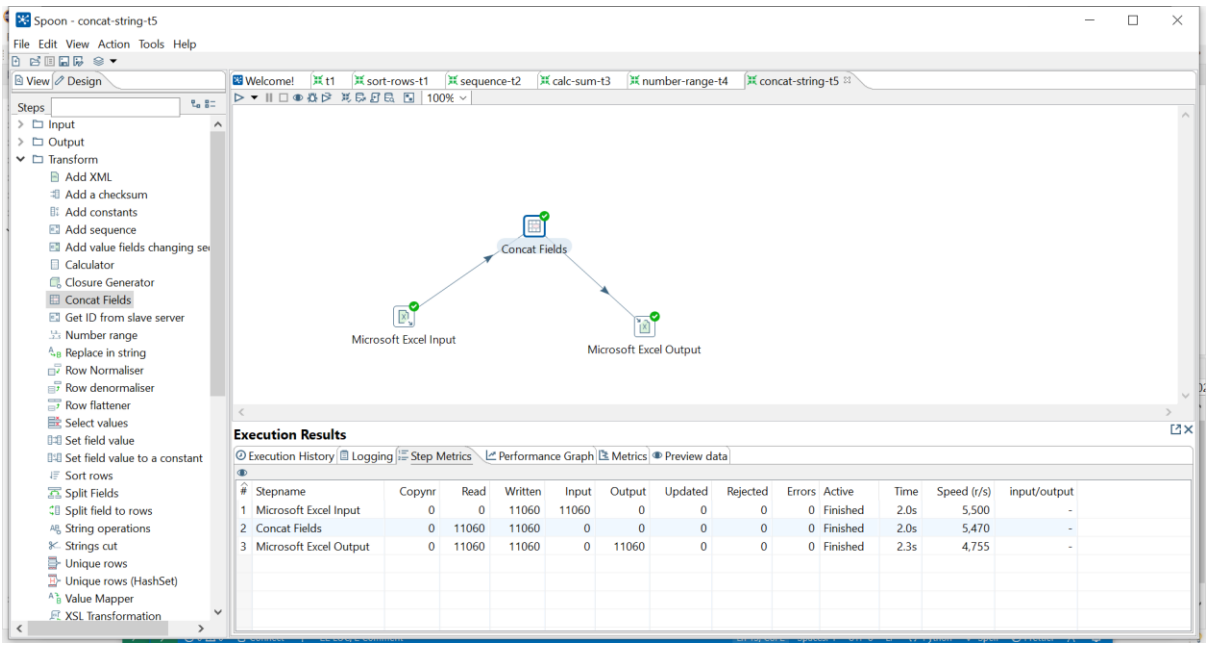
Step 1. Open up the Input folder and drag and drop the Microsoft excel input step on to the transformation window.

Step 2. Open up the Transformation folder and drag and drop the concat fields and config it according to requirements.

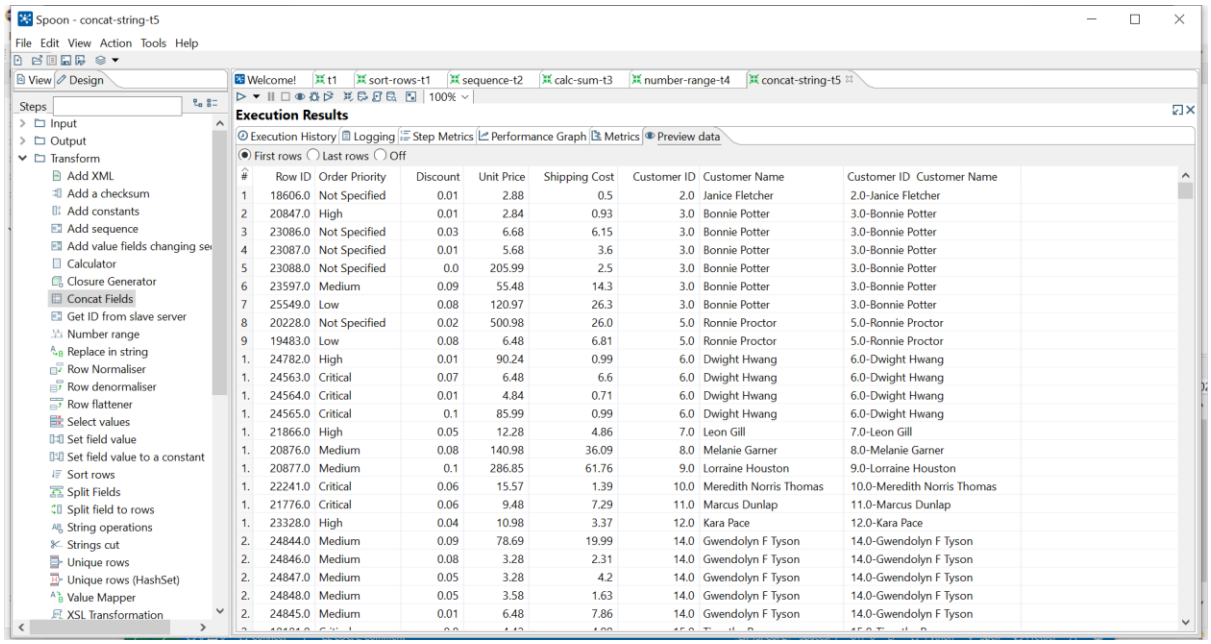
Step 3. Open up the output folder and drag and drop the Microsoft excel output.



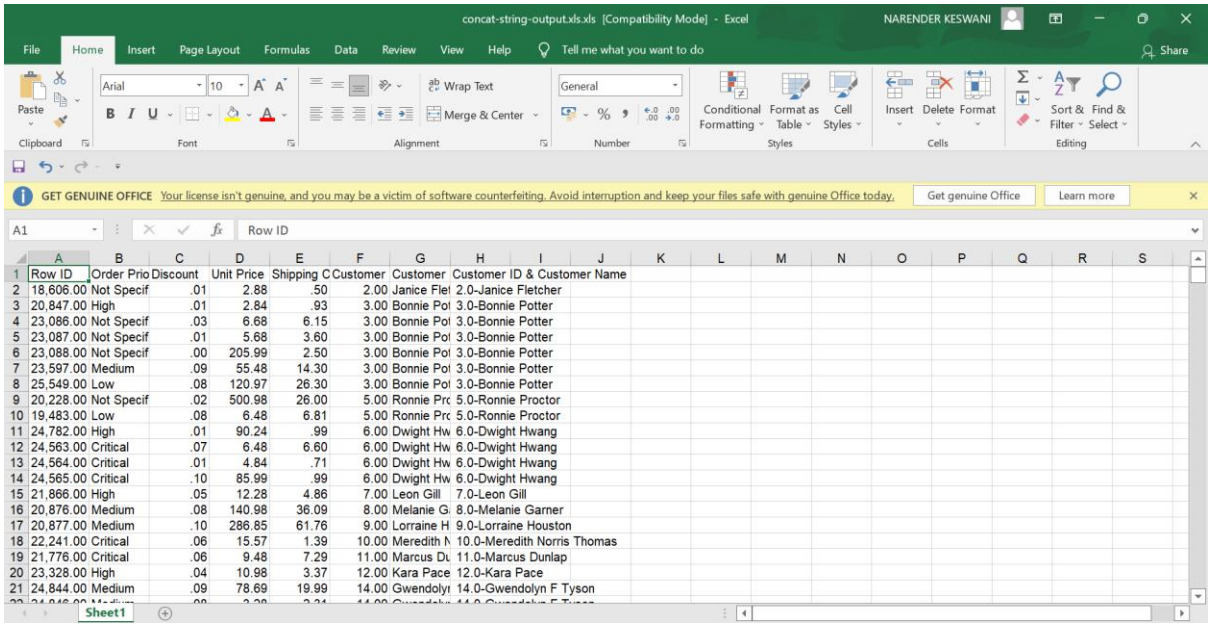
Step 4. Execute the transformation



OUTPUT:



#	Row ID	Order Priority	Discount	Unit Price	Shipping Cost	Customer ID	Customer Name	Customer ID Customer Name
1	18606.0	Not Specified	0.01	2.88	0.5	2.0	Janice Fletcher	2.0-Janice Fletcher
2	20847.0	High	0.01	2.84	0.93	3.0	Bonnie Potter	3.0-Bonnie Potter
3	23086.0	Not Specified	0.03	6.68	6.15	3.0	Bonnie Potter	3.0-Bonnie Potter
4	23087.0	Not Specified	0.01	5.68	3.6	3.0	Bonnie Potter	3.0-Bonnie Potter
5	23088.0	Not Specified	0.0	205.99	2.5	3.0	Bonnie Potter	3.0-Bonnie Potter
6	23597.0	Medium	0.09	55.48	14.3	3.0	Bonnie Potter	3.0-Bonnie Potter
7	25549.0	Low	0.08	120.97	26.3	3.0	Bonnie Potter	3.0-Bonnie Potter
8	20228.0	Not Specified	0.02	500.98	26.0	5.0	Ronnie Proctor	5.0-Ronnie Proctor
9	19483.0	Low	0.08	6.48	6.81	5.0	Ronnie Proctor	5.0-Ronnie Proctor
1.	24782.0	High	0.01	90.24	0.99	6.0	Dwight Hwang	6.0-Dwight Hwang
1.	24563.0	Critical	0.07	6.48	6.6	6.0	Dwight Hwang	6.0-Dwight Hwang
1.	24564.0	Critical	0.01	4.84	0.71	6.0	Dwight Hwang	6.0-Dwight Hwang
1.	24565.0	Critical	0.1	85.99	0.99	6.0	Dwight Hwang	6.0-Dwight Hwang
1.	21866.0	High	0.05	12.28	4.86	7.0	Leon Gill	7.0-Leon Gill
1.	20876.0	Medium	0.08	140.98	36.09	8.0	Melanie Garner	8.0-Melanie Garner
1.	20877.0	Medium	0.1	286.85	61.76	9.0	Lorraine Houston	9.0-Lorraine Houston
1.	22241.0	Critical	0.06	15.57	1.39	10.0	Meredith Norris Thomas	10.0-Meredith Norris Thomas
1.	21776.0	Critical	0.06	9.48	7.29	11.0	Marcus Dunlap	11.0-Marcus Dunlap
1.	23328.0	High	0.04	10.98	3.37	12.0	Kara Pace	12.0-Kara Pace
2.	24844.0	Medium	0.09	78.69	19.99	14.0	Gwendolyn F Tyson	14.0-Gwendolyn F Tyson
2.	24846.0	Medium	0.08	3.28	2.31	14.0	Gwendolyn F Tyson	14.0-Gwendolyn F Tyson
2.	24847.0	Medium	0.05	3.28	4.2	14.0	Gwendolyn F Tyson	14.0-Gwendolyn F Tyson
2.	24848.0	Medium	0.05	3.58	1.63	14.0	Gwendolyn F Tyson	14.0-Gwendolyn F Tyson
2.	24845.0	Medium	0.01	6.48	7.86	14.0	Gwendolyn F Tyson	14.0-Gwendolyn F Tyson



Row ID	Order Prio	Discount	Unit Price	Shipping C	Customer	Customer	Customer ID & Customer Name
18,606.00	Not Specif	.01	2.88	.50	2.00	Janice Fle	2.0-Janice Fletcher
20,847.00	High	.01	2.84	.93	3.00	Bonnie Pot	3.0-Bonnie Potter
23,086.00	Not Specif	.03	6.68	6.15	3.00	Bonnie Pot	3.0-Bonnie Potter
23,087.00	Not Specif	.01	5.68	3.60	3.00	Bonnie Pot	3.0-Bonnie Potter
23,088.00	Not Specif	.00	205.99	2.50	3.00	Bonnie Pot	3.0-Bonnie Potter
23,597.00	Medium	.09	55.48	14.30	3.00	Bonnie Pot	3.0-Bonnie Potter
25,549.00	Low	.08	120.97	26.30	3.00	Bonnie Pot	3.0-Bonnie Potter
20,228.00	Not Specif	.02	500.98	26.00	5.00	Ronnie Prc	5.0-Ronnie Proctor
19,483.00	Low	.08	6.48	6.81	5.00	Ronnie Prc	5.0-Ronnie Proctor
24,782.00	High	.01	90.24	.99	6.00	Dwight Hw	6.0-Dwight Hwang
24,563.00	Critical	.07	6.48	6.60	6.00	Dwight Hw	6.0-Dwight Hwang
24,564.00	Critical	.01	4.84	.71	6.00	Dwight Hw	6.0-Dwight Hwang
24,565.00	Critical	.10	85.99	.99	6.00	Dwight Hw	6.0-Dwight Hwang
21,866.00	High	.05	12.28	4.86	7.00	Leon Gill	7.0-Leon Gill
20,876.00	Medium	.08	140.98	36.09	8.00	Melanie G	8.0-Melanie Garner
20,877.00	Medium	.10	286.85	61.76	9.00	Lorraine H	9.0-Lorraine Houston
22,241.00	Critical	.06	15.57	1.39	10.00	Meredith N	10.0-Meredith Norris Thomas
21,776.00	Critical	.06	9.48	7.29	11.00	Marcus Du	11.0-Marcus Dunlap
23,328.00	High	.04	10.98	3.37	12.00	Kara Pace	12.0-Kara Pace
24,844.00	Medium	.09	78.69	19.99	14.00	Gwendolyr	14.0-Gwendolyn F Tyson

CONCLUSION:

From this practical, I have learned how to do ETL (Extract, Transform, Load) in Pentaho.

AIM: BASIC R COMMANDS

THEORY:

Introduction to R:

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's

GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

R environment:

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- ✓ an effective data handling and storage facility,
- ✓ a suite of operators for calculations on arrays, in particular matrices,
- ✓ a large, coherent, integrated collection of intermediate tools for data analysis,
- ✓ graphical facilities for data analysis and display either on-screen or on hardcopy, and
- ✓ a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

A) PRINTING STRING:

SOURCE CODE:

```
myString<-"Narender Keswani"
print(myString)
```

OUTPUT:

```
> #Basic R commands
>
> myString<-"Narender Keswani"
> print(myString)
[1] "Narender Keswani"
```

B) GET CURRENT WORKING DIRECTORY:

SOURCE CODE:

```
## getwd() - get current working directory.
getwd()
```

OUTPUT:

```
> ## getwd() - get current working directory.
> getwd()
[1] "C:/Users/NARENDER KESWANI/Documents"
```

C) GET LIST OF DIRECTORIES:

SOURCE CODE:

```
## dir() - lists the contents of current working directory.
dir()
```

OUTPUT:

```
 [3] "AAA"
 [4] "Adobe"
 [5] "AGREEMENT OF MAARULA CLASSES.docx"
 [6] "AGREEMENT OF MAARULA CLASSES.pdf"
 [7] "andritfbnot.txt.txt"
 [8] "anjali-developerFolio-master"
 [9] "apache"
[10] "Apex Travel Paradise Narender Keswani Internship - Copy.docx"
[11] "Apex Travel Paradise Narender Keswani Internship.docx"
[12] "Apex Travel Paradise Neel Deshmukh Internship - Copy.docx"
[13] "Apex Travel Paradise Neel Deshmukh Internship.docx"
[14] "Apowersoft"
[15] "Arduino"
[16] "Arduino-20200827T161625Z-001.zip"
[17] "Backup_of_narender google logo.cdr"
[18] "Backup_of_Narender Portfoilo Resume.cdr"
[19] "Backup_of_narender visiting card.cdr"
[20] "beta-orionis-functions-master"
[21] "cache"
[22] "Cerebranium"
[23] "certificate 1-3.pdf"
[24] "Certificate for Narender Keswani for CYBER FORENSICIS.pdf"
[25] "Certificate for NARENDER KESWANI for DIGITAL FORENSICS.pdf"
[26] "Corel"
[27] "Custom Office Templates"
[28] "desktop.ini"
[29] "dumps"
[30] "E-SUMMIT CERTIFICATE"
[31] "e1.rda"
[32] "eclipse"
```

D) GET LIST NAMES OF OBJECTS IN R ENVIRONMENT:

SOURCE CODE:

```
##ls() - lists names of objects in R environment
ls()
```

OUTPUT:

```
> ##ls() - lists names of objects in R environment
> ls()
[1] "myString"
```

E) CHECKING TYPE OF OBJECT:

SOURCE CODE:

```
x<-1
## Checking the type of variable:
class(x)
```

OUTPUT:

```
> x<-1
> ## Checking the type of variable:
> class(x)
[1] "numeric"
```

F) EXAMPLE OF AUTO-PRINTING:

SOURCE CODE:

```
#Printing a variable:
#auto-printing
X
```

OUTPUT:

```
> #Printing a variable:
> #auto-printing
> x
[1] 1
```

G) EXAMPLE OF EXPLICIT PRINTING:

SOURCE CODE:

```
#explicit printing
print(x)
```

OUTPUT:

```
> #explicit printing
> print(x)
[1] 1
```

H) CHECK DATATYPE

1) CHARACTER:

SOURCE CODE:

```
## is., as. functions: R has is.* and as.* family of functions that can be used to check
whether a varix<-'c'
#check if character
is.character(x)
```

OUTPUT:

```
> ## is., as. functions: R has is.* and as.* family of functions that can be used to check whether
 a varix<-'c'
> #check if character
> is.character(x)
[1] FALSE
```

2) INTEGER:

SOURCE CODE:

```
#check if integer
is.integer(x)
```

OUTPUT:

```
> #check if integer
> is.integer(x)
[1] FALSE
```

I) CONVERT TO INTEGER:

SOURCE CODE:

```
y<-'2.14'
as.integer(y)
```

OUTPUT:

```
> y<-'2.14'
> as.integer(y)
[1] 2
```

J) CREATE VECTOR:

1) USING c() FUNCTION:

SOURCE CODE:

```
###Creating Vector: contains objects of same class.
#using c() function
x<-c(11.3,27.5,33.8)
x
```

OUTPUT:

```
> ###Creating Vector: contains objects of same class.
> #using c() function
> x<-c(11.3,27.5,33.8)
> x
[1] 11.3 27.5 33.8
```

2) USING VECTOR() FUNCTION:

SOURCE CODE:

```
#using vector() function
y<-vector("logical", length=10)
y
y<-c(4,5,6)
y
```

OUTPUT:

```
> #using vector() function
> y<-vector("logical", length=10)
> y
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> y
[1] 4 5 6
```

K) FIND LENGTH OF VECTOR:

SOURCE CODE:

```
#length of vector x
length(x)
```

OUTPUT:

```
> #length of vector x
> length(x)
[1] 3
```

L) ARTHIMETIC OPERATIONS:

1) MULTIPLICATION OF SCALAR:

SOURCE CODE:

```
#multiplication by a scalar
5*x
```

OUTPUT:

```
> #multiplication by a scalar
> 5*x
[1]  56.5 137.5 169.0
```

2) ADDITION OF VECTORS:

SOURCE CODE:

```
#addition of two vectors x+y
x+y
```

OUTPUT:

```
> #addition of two vectors x+y
> x+y
[1] 15.3 32.5 39.8
```

3) **MULTIPLICATION OF VECTORS:**

SOURCE CODE:

```
#multiplication of two vectors
x*y
```

OUTPUT:

```
> #multiplication of two vectors
> x*y
[1]  45.2 137.5 202.8
```

4) **SUBTRACTION OF VECTORS:**

SOURCE CODE:

```
#subtraction of two vectors
x-y
```

OUTPUT:

```
> #subtraction of two vectors
> x-y
[1]  7.3 22.5 27.8
```

5) **DIVISION OF VECTORS:**

SOURCE CODE:

```
#divison of two vectors
x/y
```

OUTPUT:

```
> #divison of two vectors
> x/y
[1] 2.825000 5.500000 5.633333
```

6) **POWER:**

SOURCE CODE:

```
#x to the power y
x^y
```

OUTPUT:

```
> #x to the power y
> x^y
[1] 1.630474e+04 1.572764e+07 1.491077e+09
```

M) **CREATION OF MATRIX:**

SOURCE CODE:

```
###Creating Matrix: Two-dimensional array having elements of same class.
#using matrix() function
m<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3)
m

#By default, elements in matrix are filled by column. "byrow" attribute of matrix() can be
used to
fillm<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)
fillm
```

OUTPUT:

```
> ###Creating Matrix: Two-dimensional array having elements of same class.
> #using matrix() function
> m<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3)
> m
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
> #By default, elements in matrix are filled by column. "byrow" attribute of matrix() can be used
 to
> fillm<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)
> fillm
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
```

N) FIND DIMENSION & ATTRIBUTE OF A MATRIX:

SOURCE CODE:

```
#dimensions of matrix m
dim(m)

#attributes of matrix m
attributes(m)
```

OUTPUT:

```
> #dimensions of matrix m
> dim(m)
[1] 3 3
>
> #attributes of matrix m
> attributes(m)
$dim
[1] 3 3
```

O) CBIND() & RBIND():

SOURCE CODE:

```
#cbinding and rbinding:
#By using cbind() and rbind() functions
x<-c(1,2,3)
y<-c(11,12,13)

#cbind
cbind(x,y)

#rbind
rbind(x,y)
```

OUTPUT:

```
> #cbinding and rbinding:
> #By using cbind() and rbind() functions
> x<-c(1,2,3)
> y<-c(11,12,13)
>
> #cbind
> cbind(x,y)
     x  y
[1,] 1 11
[2,] 2 12
[3,] 3 13
>
> #rbind
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y   11   12   13
```

P) OPERATIONS ON MATRIX:

1) MULTIPLICATION BY A SCALAR:

SOURCE CODE:

```
##Matrix operations/functions:
#Multiplication by a scalar.
print(x*5)

p<-3*m
p
```

OUTPUT:

```
> ##Matrix operations/functions:
> #Multiplication by a scalar.
> print(x*5)
[1]  5 10 15
```

```
> p<-3*m
> p
     [,1] [,2] [,3]
[1,]   33  165  198
[2,]   36  180  216
[3,]   39  195  234
```

2) **ADDITION OF MATRICES:**

SOURCE CODE:

```
print(x+y)

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

OUTPUT:

```
> print(x+y)
[1] 12 14 16

> n<-matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)
> #addition of two matrices
> q<-m+n
> q
     [,1] [,2] [,3]
[1,]   15   69   90
[2,]   17   75   97
[3,]   19   81  104
```

3) **SUBTRACTION OF MATRICES:**

SOURCE CODE:

```
print(x-y)
```

OUTPUT:

```
> print(x-y)
[1] -10 -10 -10
```

4) **MULTIPLICATION OF MATRICES:**

SOURCE CODE:

```
print(x*y)
```

OUTPUT:

```
> print(x*y)
[1] 11 24 39
```

5) **DIVISION OF MATRICES:**

SOURCE CODE:

```
print(x/y)
```

OUTPUT:

```
> print(x/y)
[1] 0.09090909 0.16666667 0.23076923
```

6) **MATRIX MULTIPLICATION BY USING %*%**

SOURCE CODE:

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

#matrix multiplication by using %*%
r<-m %*% o
r
```

OUTPUT:

```
> o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)
> o
     [,1] [,2]
[1,]    4   14
[2,]    5   15
[3,]    6   16
>
> #matrix multiplication by using %*%
> r<-m %*% o
> r
     [,1] [,2]
[1,]  715 2035
[2,]  780 2220
[3,]  845 2405
```

7) TRANSPOSE OF MATRIX:

SOURCE CODE:

```
#transpose of matrix
mdash<-t(m)
mdash
```

OUTPUT:

```
> #transpose of matrix
> mdash<-t(m)
> mdash
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
```

8) FIND DETERMINANT FROM MATRIX:

SOURCE CODE:

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

OUTPUT:

```
> s<-matrix(c(4,5,6,14,15,16,24,25,26), nrow=3,ncol=3,byrow=TRUE)
> #determinant of s
> s_det<-det(s)
> s_det
[1] 1.110223e-14
```

Q) EXAMPLE OF LIST:

SOURCE CODE:

```
#using list() function
x<-list(1,"p",TRUE,2+4i)
x
```

OUTPUT:

```
> #using list() function
> x<-list(1,"p",TRUE,2+4i)
> x
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

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

R) EXAMPLE OF FACTOR & LEVELS:

SOURCE CODE:

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

##levels' argument is used to set the order of levels.
```

```
#First level forms the baseline level.
# Without any order, levels are called nominal. Ex. - Type1, Type2, .
# With order, levels are called ordinal. Ex. - low, medium, .
```

OUTPUT:

```
> ###Factor: Represents categorical data. Can be ordered or unordered.
> status<-c("low","high","medium","high","low")
> #using factor() function
> x<-factor(status, ordered=TRUE,levels=c("low","medium","high"))
> x
[1] low    high   medium high   low
Levels: low < medium < high
>
> ##levels' argument is used to set the order of levels.
> #First level forms the baseline level.
> # Without any order, levels are called nominal. Ex. - Type1, Type2, .
> # With order, levels are called ordinal. Ex. - low, medium, .
```

S) EXAMPLE OF DATAFRAME:

SOURCE CODE:

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

OUTPUT:

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

T) FUNCTIONS OF DATAFRAME:

1) ACCESSING A PARTICULAR COLUMN:

SOURCE CODE:

#accessing a particular column
data$student_id

OUTPUT:

```
> #accessing a particular column
> data$student_id
[1] 1 2 3
```

2) **NUMBER OF ROWS IN DATAFRAME:**

SOURCE CODE:

#no. of rows in data
nrow(data)

OUTPUT:

```
> #no. of rows in data
> nrow(data)
[1] 3
```

3) **NUMBER OF COLUMNS IN DATAFRAME:**

SOURCE CODE:

#no. of columns in data
ncol(data)

OUTPUT:

```
> #no. of columns in data
> ncol(data)
[1] 3
```

4) **GET COLUMN NAMES OF A DATAFRAME:**

SOURCE CODE:

#column names of data. for a dataframe, colnames() can also be used.
names(data)

OUTPUT:

```
> #column names of data. for a dataframe, colnames() can also be used.
> names(data)
[1] "student_id"    "student_names" "position"
```

U) CREATE 2-DIMENSIONAL TABLE:

SOURCE CODE:

```
###Table command is used to create a 2dimensional table in R
smoke <- matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
colnames(smoke) <- c("High","Low","Middle")
rownames(smoke) <- c("current","former","never")
smoke <- as.table(smoke)
smoke
```

OUTPUT:

```
> ###Table command is used to create a 2dimensional table in R
> smoke <- matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
> colnames(smoke) <- c("High","Low","Middle")
> rownames(smoke) <- c("current","former","never")
> smoke <- as.table(smoke)
> smoke
        High Low Middle
current   51  43     22
former    92  28     21
never     68  22      9
```

V) INSTALL LIBARIES:

SOURCE CODE:

```
#install.packages("package_name")
library(XLConnect)
install.packages("readxl")
library(readxl)
install.packages("writexl")
library(writexl)
```

OUTPUT:

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

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/readxl_1.3.1.zip'
Content type 'application/zip' length 1716858 bytes (1.6 MB)
downloaded 1.6 MB

package 'readxl' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\NARENDER KESWANI\AppData\Local\Temp\RtmpCUXSNN\downloaded_packages
> library(readxl)
Warning message:
package 'readxl' was built under R version 4.0.5
> install.packages("writexl")
WARNING: Rtools is required to build R packages but is not currently installed. Please download an
d install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/writexl_1.4.0.zip'
Content type 'application/zip' length 351289 bytes (343 KB)
downloaded 343 KB
```

W) EXAMPLES OF CSV:

1) READING DATA FROM CSV:

SOURCE CODE:

```
dataT <- read.table("C:\\Users\\NARENDER KESWANI\\Desktop\\r-prac-check.csv", sep
=",", header = T)
dataT
```

OUTPUT:

```
> dataT <- read.table("C:\\Users\\NARENDER KESWANI\\Desktop\\r-prac-check.csv", sep =",", header =
 T)
Warning message:
In scan(file = file, what = what, sep = sep, quote = quote, dec = dec,  :
  number of items read is not a multiple of the number of columns
> dataT
  id             name  dept  X
1  1 narender keswani   mca NA
2  2    neel deshmukh  bvoc NA
3  3     hassan haque   mba NA
4  4      ronak karia    ba NA
5  5     ritesh yadav  bcom NA
```

2) GET DIMENSIONS OF CSV FILE:

SOURCE CODE:

```
# dimension
dim(dataT)
```

OUTPUT:

```
> # dimension
> dim(dataT)
[1] 5 4
```

3) **HEAD & TAIL:**

SOURCE CODE:

```
# Load just few lines at the top or bottom
head(dataT, 2)

tail(dataT, 2)
```

OUTPUT:

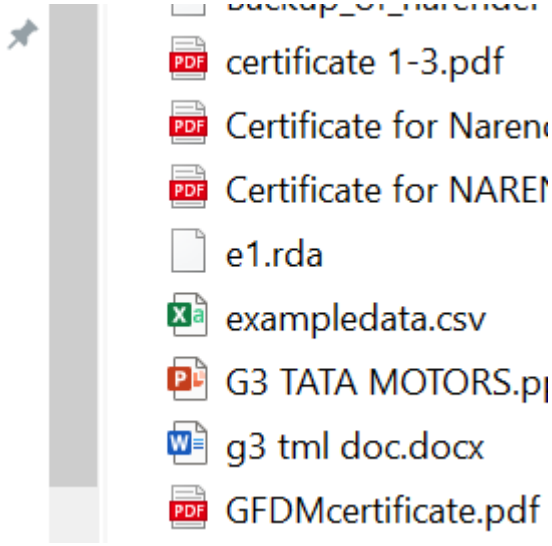
```
> # Load just few lines at the top or bottom
> head(dataT, 2)
  id             name  dept  X
1  1 narender keswani   mca NA
2  2     neel deshmukh  bvoc NA
>
> tail(dataT, 2)
  id          name  dept  X
4  4  ronak karia    ba NA
5  5 ritesh yadav  bcom NA
```

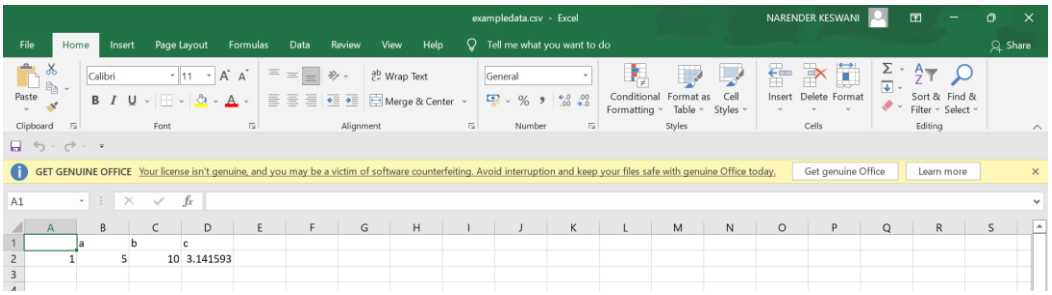
4) **WRITING DATA TO CSV FILE:**

SOURCE CODE:

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

OUTPUT:





X) READING AND WRITING DATA FROM EXCEL USING XLCONNECT:

SOURCE CODE:

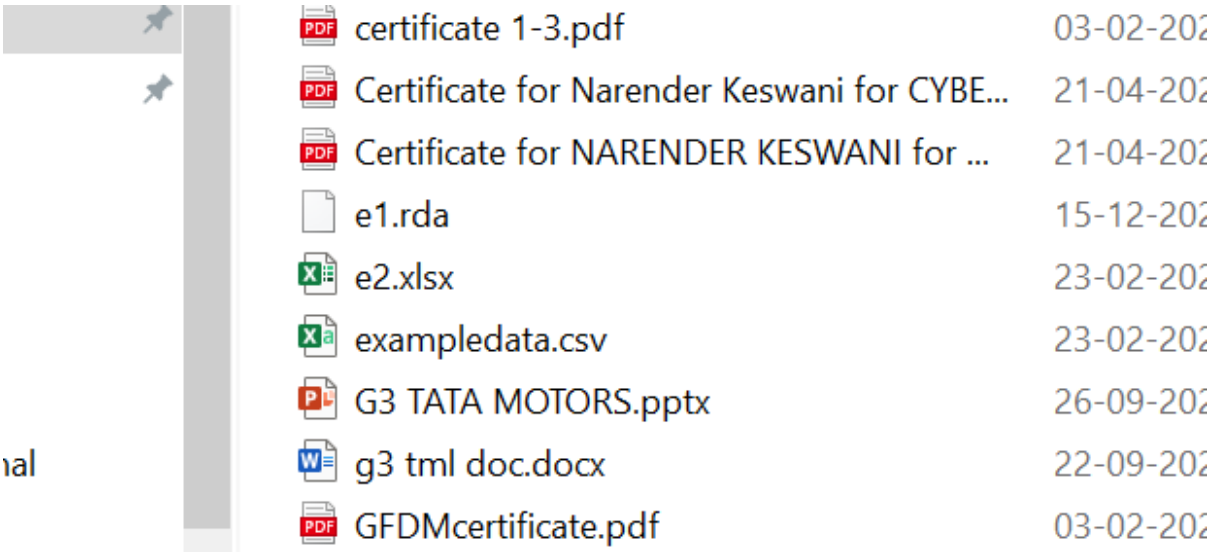
```
install.packages('Rcpp')
library(Rcpp)

#Reading and writing data from Excel using XLConnect
dataX <- XLConnect:: readWorksheetFromFile("C:\\Users\\NARENDER
KESWANI\\Downloads\\01 Contoso Employee Info.xlsx",sheet=1)
dataX

# Following is called Subsetting - It will print rows from 1 to 2 and all columns
dataY<- dataX[1:2,]
dataY

#Reading and writing data from Excel using readXL and writeXL
data2 <- read_excel("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee
Info.xlsx", sheet = "1")
data2
data3<- data2[1:5,]
write_xlsx(data3, "e2.xlsx")

# create an empty data frame
```

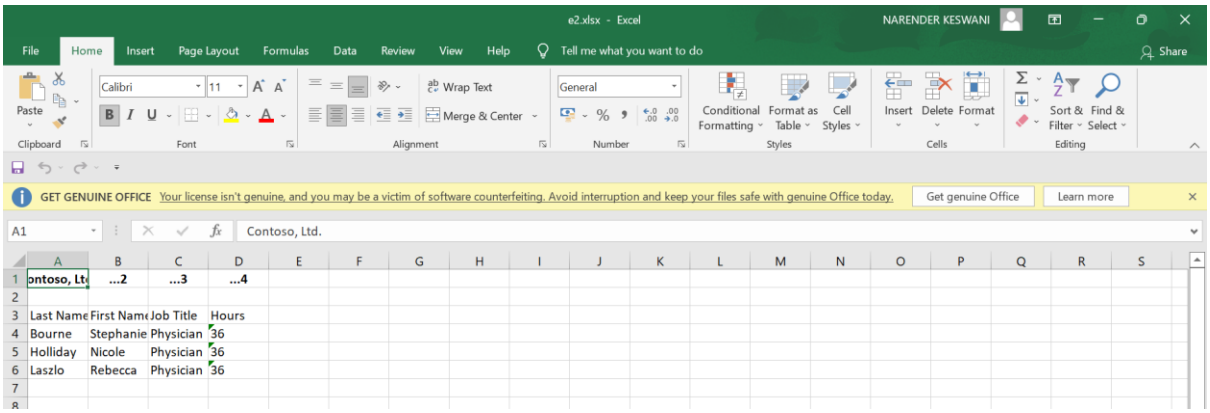
```
data <- data.frame(Name=character(), Age=numeric())
# load interface and assign edited values to data back - uncomment following
data <- edit(data)
#print those values
data
```

OUTPUT:

```
> #Reading and writing data from Excel using XLConnect
> dataX <- XLConnect:: readWorksheetFromFile("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee
 Info.xlsx",sheet=1)
> dataX
   Contoso..Ltd.       Col2                Col3  Col4
1           <NA>       <NA>                <NA>  <NA>
2      Last Name First Name           Job Title Hours
3         Bourne  Stephanie           Physician    36
4       Holliday     Nicole           Physician    36
5         Laszlo    Rebecca           Physician    36
6       Barnhill       Josh       Billing Clerk    36
7           Kane       John    Registered Nurse    30
8        Trenary       Jean    Registered Nurse    30
9       Da Silva     Sergio Physician Assistant    36
10          Wang       Jian Referral Specialist    36
11        Wilson        Dan           Physician    36
12        Valdez     Rachel        Receptionist    30
13         Giest      Jenny      Office Manager    40
14     Gottfried        Jim        Receptionist    30
15       Delaney      Aidan        Receptionist    20
16     Dellamore       Luca   Medical Assistant    36
17      Hamilton      David   Medical Assistant    36
18        Hoeing      Helge   Medical Assistant    36
19        Munson     Stuart Referral Specialist    36
20        Murray  Billie Jo   Medical Assistant    36
21       Kenneth      Kevin          File Clerk    15
22       Hensien       Kari          File Clerk    20
23         Moore      Bobby          File Clerk    15
24      Moreland    Barbara       Billing Clerk    20
25       Metters      Susan       Billing Clerk    25
26        Poland     Carole  Nurse Practitioner    25
> # Following is called Subsetting - It will print rows from 1 to 2 and all columns
> dataY<- dataX[1:2,]
> dataY
  Contoso..Ltd.       Col2      Col3  Col4
1           <NA>       <NA>      <NA>  <NA>
2      Last Name First Name Job Title Hours
>
```

```
> #Reading and writing data from Excel using readXL and writeXL
> data2 <- read_excel("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee Info.xlsx")
New names:
* `` -> ...2
* `` -> ...3
* `` -> ...4
> data2
# A tibble: 26 x 4
   `Contoso, Ltd.` ...2       ...3              ...4
   <chr>           <chr>      <chr>             <chr>
 1 NA              NA         NA                NA
 2 Last Name       First Name Job Title         Hours
 3 Bourne          Stephanie  Physician         36
 4 Holliday        Nicole     Physician         36
 5 Laszlo          Rebecca    Physician         36
 6 Barnhill        Josh       Billing Clerk     36
 7 Kane            John       Registered Nurse  30
 8 Trenary         Jean       Registered Nurse  30
 9 Da Silva        Sergio     Physician Assistant 36
10 Wang            Jian       Referral Specialist 36
# ... with 16 more rows
> data3<- data2[1:5,]
> write_xlsx(data3, "e2.xlsx")
```





Data Editor

File Edit Help

	Name	Age	var3	var4	var5	var6	var7
1	naren>	21	10	10	9	9	10
2	neel	20	5	6	8	9	10
3	hassan	19	10	9	7	8	10
4	wilson	42	10	10	10	10	10
5	ronak	21	8	9	10	0	0
6	ritesh	20	7	8	9	10	10
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

```
> # load interface and assign edited values to data back - uncomment following
> data <- edit(data)
> #print those values
> data
      Name Age var3 var4 var5 var6 var7
1 narender  21   10   10    9    9   10
2     neel  20    5    6    8    9   10
3   hassan  19   10    9    7    8   10
4   wilson  42   10   10   10   10   10
5    ronak  21    8    9   10    0    0
6   ritesh  20    7    8    9   10   10
```

CONCLUSION:

From this practical, I have learned the basics of R programming.

AIM: DATA PREPROCESSING IN R

THEORY:

Data Preprocessing Techniques

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data Preprocessing:

1. Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a). Missing Data:

This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

1. **Ignore the tuples:** This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
2. **Fill the Missing values:** There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

(b). Noisy Data:

Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

1. **Binning Method:** This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
2. **Regression:** Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
3. **Clustering:** This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization :-** It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)
2. **Attribute Selection :-** In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
3. **Discretization :-** This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
4. **Concept Hierarchy Generation :-** Here attributes are converted from level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

3. Data Reduction:

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data

reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation :-** Aggregation operation is applied to data for the construction of the data cube.
2. **Attribute Subset Selection :-** The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded.
3. **Numerosity Reduction :-** This enable to store the model of data instead of whole data, for example: Regression Models.
4. **Dimensionality Reduction :-** This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Componenet Analysis).

A) PRINT HEAD OF MTCARS DATASET [PREDEFINED IN R]:

SOURCE CODE:

```
my_data<-mtcars
head(my_data,5)
```

OUTPUT:

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

B) READING DATA OF SPECFIC ROWS & COLUMNS:

SOURCE CODE:

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

OUTPUT:

```
> #my_data
> my_data1 <- my_data[1:6,1:5]
> my_data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
```

C) RENAME COLUMN NAME USING DPLYR:

SOURCE CODE:

```
install.packages("dplyr")
library(dplyr, warn.conflicts = FALSE)
my_data1 = dplyr::rename(my_data1, "horse_power" = "hp")
my_data1
```

OUTPUT:

```
WARNING: Rtools is required to build R packages but is not currently installed. Plea
ll the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/dplyr_1.0.8.zip'
Content type 'application/zip' length 1381575 bytes (1.3 MB)
downloaded 1.3 MB

package 'dplyr' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\NARENDER KESWANI\AppData\Local\Temp\RtmpMTGJjs\downloaded_packages
> library(dplyr, warn.conflicts = FALSE)
Registered S3 methods overwritten by 'tibble':
  method     from
  format.tbl pillar
  print.tbl  pillar
Warning message:
package 'dplyr' was built under R version 4.0.5
> my_data1 = dplyr::rename(my_data1, "horse_power" = "hp")
> my_data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
```

D) ADDING NEW COLUMN:

SOURCE CODE:

```
## Adding new variable
my_data1$new_hp1 <- my_data1$horse_power * 0.5
colnames(my_data1)

my_data1
```

OUTPUT:

```
> ## Adding new variable
> my_data1$new_hp1 <- my_data1$horse_power * 0.5
> colnames(my_data1)
[1] "mpg"         "cyl"         "disp"        "horse_power" "drat"        "new_hp1"
>
> my_data1
                   mpg cyl disp horse_power drat new_hp1
Mazda RX4         21.0   6  160         110 3.90    55.0
Mazda RX4 Wag     21.0   6  160         110 3.90    55.0
Datsun 710        22.8   4  108          93 3.85    46.5
Hornet 4 Drive    21.4   6  258         110 3.08    55.0
Hornet Sportabout 18.7   8  360         175 3.15    87.5
Valiant           18.1   6  225         105 2.76    52.5
```

E) READING DATA FROM CSV FILE:

SOURCE CODE:

```
#Reading with read.table() assumes no headers by default. First few lines :
data2 = read.table(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv",
sep = ",")
data2
```

OUTPUT:

```
> #Reading with read.table() assumes no headers by default. First few lines :
> data2 = read.table(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv", sep = ",")
> data2
   V1       V2      V3         V4         V5
1   1     Rick  623.30 01/01/2012         IT
2   2      Dan  515.20 23/09/2013 Operations
3   3 Michelle  611.00 15/11/2014         IT
4   4     Ryan  729.00 11/05/2014         HR
5  NA     Gary  843.25 27/03/2015    Finance
6   6     Nina      NA 21/05/2013         IT
7   7    Simon  632.80 30/07/2013 Operations
8   8     Guru  722.50 17/06/2014    Finance
9   9     John      NA 21/05/2012
10 10     Rock  600.80 30/07/2013         HR
11 11     Brad 1032.80 30/07/2013 Operations
12 12     Ryan  729.00 11/05/2014         HR
```

F) READING DATA FROM SPECFIC COLUMNS OF CSV FILE:

SOURCE CODE:

```
#V1, V2, V3.. are given as default names (titles) by R
data2 = read.csv(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv",
col.names=c("Sno", "NAME","SALARY","DateOfJodata2"))
```

OUTPUT:

```
> #V1, V2, V3.. are given as default names (titles) by R
> data2 = read.csv(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv", col.names=c("Sno", "N
AME","SALARY","DateOfJodata2"))
Warning message:
In read.table(file = file, header = header, sep = sep, quote = quote,  :
  header and 'col.names' are of different lengths
> data2
        Sno    NAME     SALARY DateOfJodata2
2       Dan  515.20 23/09/2013    Operations
3  Michelle  611.00 15/11/2014            IT
4      Ryan  729.00 11/05/2014            HR
       Gary  843.25 27/03/2015       Finance
6      Nina      NA 21/05/2013            IT
7     Simon  632.80 30/07/2013    Operations
8      Guru  722.50 17/06/2014       Finance
9      John      NA 21/05/2012
10     Rock  600.80 30/07/2013            HR
11     Brad 1032.80 30/07/2013    Operations
12     Ryan  729.00 11/05/2014            HR
```

G) OPERATION WITH NA:

SOURCE CODE:

```
#Operation with NA
NA + 4
```

OUTPUT:

```
> #Operation with NA
> NA + 4
[1] NA
```

H) CREATE A VECTOR V WITH 1 NA VALUE:

SOURCE CODE:

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

OUTPUT:

```
> #Create a vector V with 1 NA value
> V <- c(1,2,NA,3)
> V
[1]  1  2 NA  3
```

I) FIND MEDIAN:

1) WITH NA:

SOURCE CODE:

```
#Median with NA
median(V)
```

OUTPUT:

```
> #Median with and without NA (remove NA)
> #with NA
> median(V)
[1] NA
```

2) WITHOUT NA:

SOURCE CODE:

```
#On removing NAs
median(V, na.rm = T)
```

OUTPUT:

```
> #without NA
> #On removing NAs
> median(V, na.rm = T)
[1] 2
```

J) CHECK WHETHER IT IS NA OR NOT:

SOURCE CODE:

```
#Apply is.na() to vector
is.na(V)
```

OUTPUT:

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

K) REMOVING THE NA VALUES BY USING LOGICAL INDEXING:

SOURCE CODE:

```
#Removing the NA values by using logical indexing
naVals <- is.na(V)
```

OUTPUT:

```
> #Removing the NA values by using logical indexing
> naVals <- is.na(V)
> naVals
[1] FALSE FALSE  TRUE FALSE
```

L) Get values that are not NA

SOURCE CODE:

```
V[!naVals]
```

OUTPUT:

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

M) SUBSETTING WITH COMPLETE CASES - VALUES THAT ARE NOT NA:

SOURCE CODE:

```
#Subsetting with complete cases - values that are not NA
V[complete.cases(V)]
```

OUTPUT:

```
> #Subsetting with complete cases - values that are not NA
> V[complete.cases(V)]
[1] 1 2 3
```

N) SUBSETTING A DATA FRAME WITH COMPLETE CASES:

SOURCE CODE:

```
#Subsetting a data frame with complete cases
#Complete Data of Prime Ministers. Notice NAs
dataC <- read.csv(file ="C:\\Users\\NARENDER KESWANI\\Downloads\\na_data.csv",
na.strings = "")
dataC

# Subset only the rows without NA
dataCompleteCases <- dataC[complete.cases(dataC),]
dataCompleteCases
```

OUTPUT:

```
> #Subsetting a data frame with complete cases
> #Complete Data of Prime Ministers. Notice NAs
> dataC <- read.csv(file ="C:\\Users\\NARENDER KESWANI\\Downloads\\na_data.csv", na.strings = "")
> dataC
   X1     Rick  X623.3 X01.01.2012         IT
1   2      Dan  515.20  23/09/2013 Operations
2   3 Michelle  611.00  15/11/2014         IT
3   4     Ryan  729.00  11/05/2014         HR
4  NA     Gary  843.25  27/03/2015    Finance
5   6     Nina      NA  21/05/2013         IT
6   7    Simon  632.80  30/07/2013 Operations
7   8     Guru  722.50  17/06/2014    Finance
8   9     John      NA  21/05/2012       <NA>
9  10     Rock  600.80  30/07/2013         HR
10 11     Brad 1032.80  30/07/2013 Operations
11 12     Ryan  729.00  11/05/2014         HR
```

```
> # Subset only the rows without NA
> dataCompleteCases <- dataC[complete.cases(dataC),]
> dataCompleteCases
   X1     Rick X623.3 X01.01.2012         IT
1   2      Dan  515.2  23/09/2013 Operations
2   3 Michelle  611.0  15/11/2014         IT
3   4     Ryan  729.0  11/05/2014         HR
6   7    Simon  632.8  30/07/2013 Operations
7   8     Guru  722.5  17/06/2014    Finance
9  10     Rock  600.8  30/07/2013         HR
10 11     Brad 1032.8  30/07/2013 Operations
11 12     Ryan  729.0  11/05/2014         HR
```

O) MEAN IMPUTATION & MEDIAN IMPUTATION

SOURCE CODE:

```
install.packages('Hmisc')
library(Hmisc)

## create a vector
x = c(1,2,3,NA,4,4,NA)
x

# mean imputation - from package, mention name of function to be used
x <- impute(x, fun = mean)
x

#median imputation
x <- impute(x, fun = median)
x
```

OUTPUT:

```
> ## create a vector
> x = c(1,2,3,NA,4,4,NA)
> # mean imputation - from package, mention name of function to be used
> x <- impute(x, fun = mean)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
>
> #median imputation
> x <- impute(x, fun = median)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
```

P) CONVERT:

1) Convert Character into Factor(categorical data):

SOURCE CODE:

```
#Convert Character into Factor(categorical data)
#Create gender vector
gender_vector <- c("Male", "Female", "Female", "Male", "Male")
class(gender_vector)

#Convert gender_vector to a factor
factor_gender_vector <-factor(gender_vector)
class(factor_gender_vector)
```

OUTPUT:

```
> #Convert Character into Factor(categorical data)
> #Create gender vector
> gender_vector <- c("Male", "Female", "Female", "Male", "Male")
> class(gender_vector)
[1] "character"
>
> #Convert gender_vector to a factor
> factor_gender_vector <-factor(gender_vector)
> class(factor_gender_vector)
[1] "factor"
```

Q) CREATE ORDINAL CATEGORICAL VECTOR:

SOURCE CODE:

```
#Create Ordinal categorical vector
day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
day_vector
```

OUTPUT:

```
> #Create Ordinal categorical vector
> day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
> day_vector
[1] "evening"   "morning"   "afternoon" "midday"    "midnight"  "evening"
```

R) CONVERT VECTOR INTO A FACTOR WITH ORDERED LEVEL:

SOURCE CODE:

```
#Convert `day_vector` to a factor with ordered level
factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon',
'evening', 'midnight'))

#Print the new variable
factor_day
```

OUTPUT:

```
> #Convert `day_vector` to a factor with ordered level
> factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'evening',
 'midnight'))
> #Print the new variable
> factor_day
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
```

S) CREATE DATAFRAME FROM VECTOR:

SOURCE CODE:

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

# Creating data frame named employee
employee<- data.frame(age, salary, gender)
employee
```

OUTPUT:

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

T) CERATE FACTOR WITH LABELS:

SOURCE CODE:

```
# Creating a factor corresponding to age with labels
wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))
table(wfact)
```

OUTPUT:

```
> # Creating a factor corresponding to age with labels
> wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))
> table(wfact)
wfact
 Young Medium   Aged
     4      2      1
```

CONCLUSION:

From this practical, I have learned data preprocessing techniques in R.

AIM: CLASSIFICATION ALGORITHMS - NAIVE BAYES, ID3, C 4.5, K NEAREST NEIGHBOUR

THEORY:

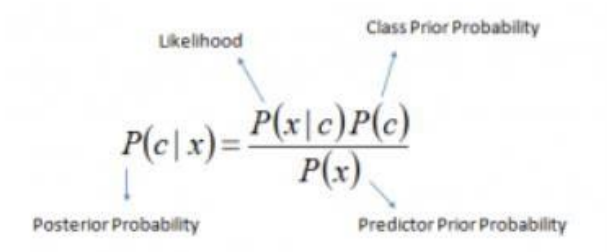
Naive Bayesian:

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor. **Pros:**

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

K nearest Neighbors:

KNN (K — Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it's a classifier algorithm where the learning is based "how similar" is a data (a vector) from other .

The KNN's steps are:

1. Receive an unclassified data;
2. Measure the distance (Euclidian, Manhattan, Minkowski or Weighted) from the new data to all others data that is already classified;
3. Gets the K(K is a parameter that you difine) smaller distances;
4. Check the list of classes had the shortest distance and count the amount of each class that appears;
5. Takes as correct class the class that appeared the most times; 6. Classifies the new data with the class that you took in step 5;

Choosing the right value for K:

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages:

1. The algorithm is simple and easy to implement.
2. There's no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages:

1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

R Packages:

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

2) **ggplot2:**
   R provides the ggplot package for creating graphics declaratively. This package is famous for its elegant and quality graphs which sets it apart from other visualization packages.

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

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

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

A) IMPLEMENT NAIVE BAYES

INSTALLING LOADING LIBS:

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

# Loading library e1071
library(e1071)

# Loading library k1aR
library(k1aR)

## Loading required package: caret
library(caret)

## Loading required package: lattice
library(lattice)

## Loading required package: ggplot2
library(ggplot2)
```

```
> # Loading library e1071
> library(e1071)
Warning message:
package 'e1071' was built under R version 4.0.5
>
> # Loading library klaR
> library(klaR)
Loading required package: MASS
Warning message:
package 'klaR' was built under R version 4.0.5
>
> ## Loading required package: caret
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
Warning messages:
1: package 'caret' was built under R version 4.0.5
2: package 'ggplot2' was built under R version 4.0.3
>
> ## Loading required package: lattice
> library(lattice)
>
> ## Loading required package: ggplot2
> library(ggplot2)
```

PRINTING DATASET:

```
# iris dataset
data(iris)
head(iris)
```

```
> # iris dataset
> 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
```

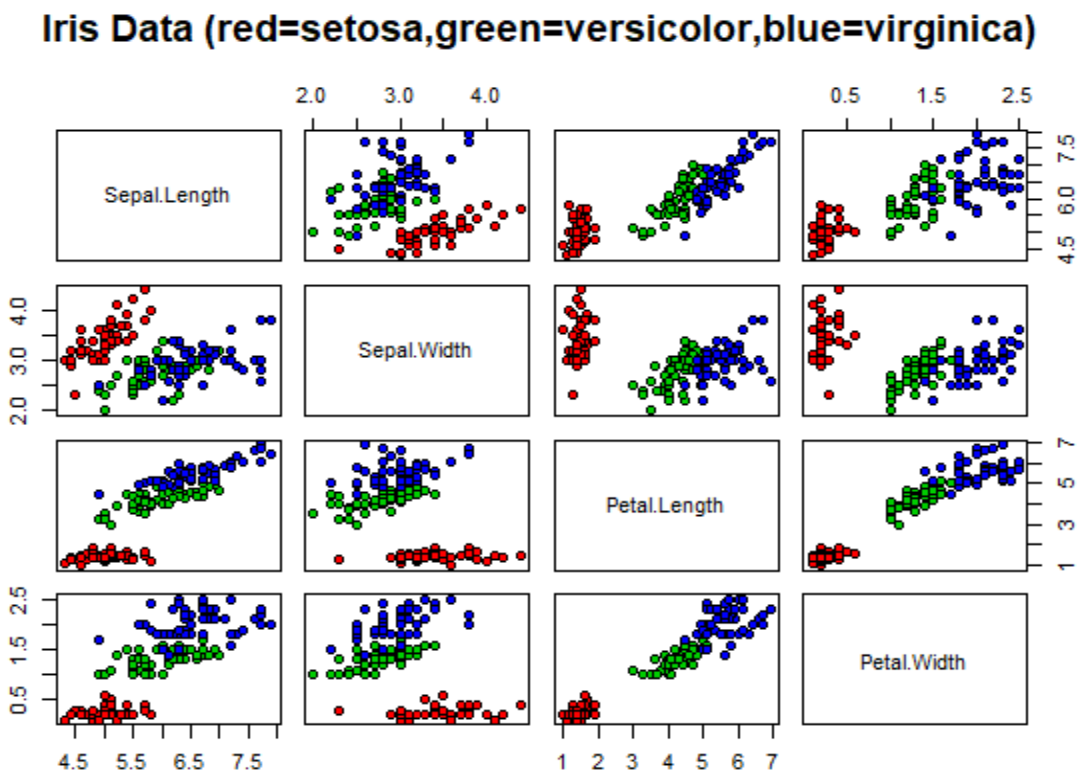
PRINTING UNIQUE ELEMENTS:

#finding unique values
unique(iris$Species)

```
> #finding unique values
> unique(iris$Species)
[1] setosa     versicolor virginica
Levels: setosa versicolor virginica
```

PLOTTING GRAPH:

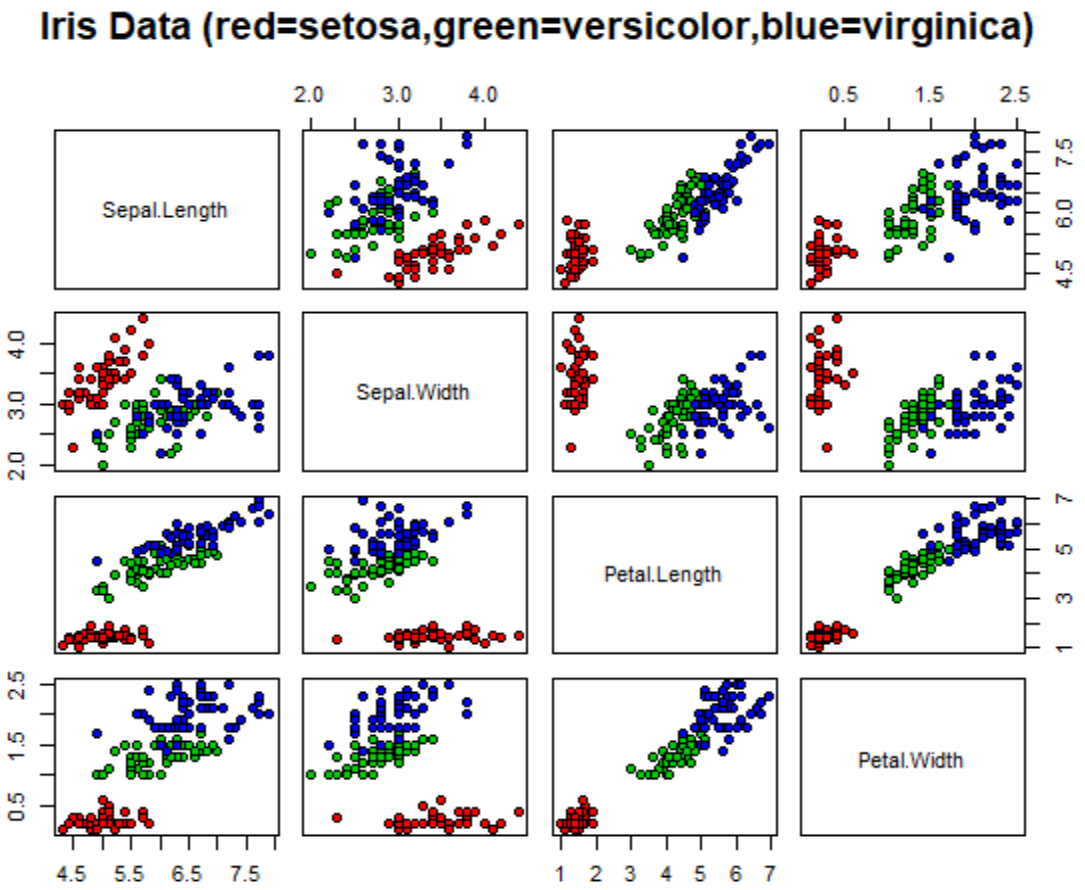
#Plot graph
pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)",
pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])



TRAINING NAIVE BAYES:

```
# training a naive Bayes model
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$Species # only y.
xTest = test[,-5]
yTest = test$Species

# nb - tells to use naive bayes
# cv - cross validation
model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))
model

## table() gives frequency table, prop.table() gives freq% table.
prop.table(table(predict(model$finalModel,xTest)$class,yTest))
```

```
> # training a naive Bayes model
> 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$Species # only y.
> xTest = test[,-5]
> yTest = test$Species
>
> # nb - tells to use naive bayes
> # cv - cross validation
> model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))
> model
Naive Bayes

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

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

  usekernel  Accuracy   Kappa
  FALSE      0.9507071  0.9257564
   TRUE      0.9523737  0.9281818

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was
 held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
>
> ## table() gives frequency table, prop.table() gives freq% table.
> prop.table(table(predict(model$finalModel,xTest)$class,yTest))
            yTest
                 setosa versicolor  virginica
  setosa     0.33333333 0.00000000 0.00000000
  versicolor 0.00000000 0.31111111 0.04444444
  virginica  0.00000000 0.02222222 0.28888889
```


Iris Data (red=setosa,green=versicolor,blue=virginica)
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0
3.0
4.0
0.5
1.5
2.5
4.5
6.0
7.5
1
3
5
7
4.5
5.5
6.5
7.5
1
2
3
4
5
6
7

B) IMPLEMENT K NEAREST NEIGHBOUR

LOADING DATASET:

```
#K nearest Neighbour
df <- data(iris) ##load data
head(iris) ## see the structure

##Generate a random number that is 90% of the total number of rows in dataset.
ran <- sample(1:nrow(iris), 0.9 * nrow(iris))
ran
```

```
> #K nearest Neighbour
> df <- data(iris) ##load data
> head(iris) ## see the structure
  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
>
> ##Generate a random number that is 90% of the total number of rows in dataset.
> ran <- sample(1:nrow(iris), 0.9 * nrow(iris))
> ran
  [1]  13 145 149  63  27 119 112  87  94  51 107 114 136  19  65  74  41 100  25   9  99   4
 [23]  64 113  37  15 140  92  50 111  72 134 108  43  49   1  69  54  59 105  36 104 118 121
 [45] 110  39  42 138  40  24 141  30  86 103  21 123  81   6  48  38   2 101  58  61  78 128
 [67]   5  47 130  60  28  84 124  14  75  93  62  70  45  11  53 144  17  56  22  55 146  68
 [89] 150  32  33 127  98   3 148  66 132 117  77  90  85 133  44  95  34  35  83   7  67  12
[111]  71  16 125  79  82  76  57  18  97 135 102  31 120  26  20  88 143 122  29 129  91  52
[133] 139  73 115
```

NORMILIZATION:

```
##the normalization function is created
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }
##Run nomalization on first 4 coulumns of dataset because they are the predictors
iris_norm <- as.data.frame(lapply(iris[,c(1,2,3,4)], nor))
iris_norm
```

```
> ##the normalization function is created
> nor <-function(x) { (x -min(x))/(max(x)-min(x)) }
> ##Run nomalization on first 4 coulumns of dataset because they are the predictors
> iris_norm <- as.data.frame(lapply(iris[,c(1,2,3,4)], nor))
> iris_norm
    Sepal.Length Sepal.Width Petal.Length Petal.Width
1     0.22222222  0.62500000   0.06779661  0.04166667
2     0.16666667  0.41666667   0.06779661  0.04166667
3     0.11111111  0.50000000   0.05084746  0.04166667
4     0.08333333  0.45833333   0.08474576  0.04166667
5     0.19444444  0.66666667   0.06779661  0.04166667
6     0.30555556  0.79166667   0.11864407  0.12500000
7     0.08333333  0.58333333   0.06779661  0.08333333
8     0.19444444  0.58333333   0.08474576  0.04166667
9     0.02777778  0.37500000   0.06779661  0.04166667
10    0.16666667  0.45833333   0.08474576  0.00000000
11    0.30555556  0.70833333   0.08474576  0.04166667
12    0.13888889  0.58333333   0.10169492  0.04166667
13    0.13888889  0.41666667   0.06779661  0.00000000
14    0.00000000  0.41666667   0.01694915  0.00000000
15    0.41666667  0.83333333   0.03389831  0.04166667
16    0.38888889  1.00000000   0.08474576  0.12500000
17    0.30555556  0.79166667   0.05084746  0.12500000
18    0.22222222  0.62500000   0.06779661  0.08333333
19    0.38888889  0.75000000   0.11864407  0.08333333
20    0.22222222  0.75000000   0.08474576  0.08333333
21    0.30555556  0.58333333   0.11864407  0.04166667
22    0.22222222  0.70833333   0.08474576  0.12500000
23    0.08333333  0.66666667   0.00000000  0.04166667
24    0.22222222  0.54166667   0.11864407  0.16666667
25    0.13888889  0.58333333   0.15254237  0.04166667
26    0.19444444  0.41666667   0.10169492  0.04166667
27    0.19444444  0.58333333   0.10169492  0.12500000
28    0.25000000  0.62500000   0.08474576  0.04166667
29    0.25000000  0.58333333   0.06779661  0.04166667
30    0.11111111  0.50000000   0.10169492  0.04166667
31    0.13888889  0.45833333   0.10169492  0.04166667
32    0.30555556  0.58333333   0.08474576  0.12500000
33    0.25000000  0.87500000   0.08474576  0.00000000
34    0.33333333  0.91666667   0.06779661  0.04166667
35    0.16666667  0.45833333   0.08474576  0.04166667
36    0.19444444  0.50000000   0.03389831  0.04166667
37    0.33333333  0.62500000   0.05084746  0.04166667
38    0.16666667  0.66666667   0.06779661  0.00000000
39    0.02777778  0.41666667   0.05084746  0.04166667
40    0.22222222  0.58333333   0.08474576  0.04166667
41    0.19444444  0.62500000   0.05084746  0.08333333
42    0.05555556  0.12500000   0.05084746  0.08333333
43    0.02777778  0.50000000   0.05084746  0.04166667
44    0.19444444  0.62500000   0.10169492  0.20833333
45    0.22222222  0.75000000   0.15254237  0.12500000
46    0.13888889  0.41666667   0.06779661  0.08333333
47    0.22222222  0.75000000   0.10169492  0.04166667
48    0.08333333  0.50000000   0.06779661  0.04166667
49    0.27777778  0.70833333   0.08474576  0.04166667
```

50	0.19444444	0.54166667	0.06779661	0.04166667
51	0.75000000	0.50000000	0.62711864	0.54166667
52	0.58333333	0.50000000	0.59322034	0.58333333
53	0.72222222	0.45833333	0.66101695	0.58333333
54	0.33333333	0.12500000	0.50847458	0.50000000
55	0.61111111	0.33333333	0.61016949	0.58333333
56	0.38888889	0.33333333	0.59322034	0.50000000
57	0.55555556	0.54166667	0.62711864	0.62500000
58	0.16666667	0.16666667	0.38983051	0.37500000
59	0.63888889	0.37500000	0.61016949	0.50000000
60	0.25000000	0.29166667	0.49152542	0.54166667
61	0.19444444	0.00000000	0.42372881	0.37500000
62	0.44444444	0.41666667	0.54237288	0.58333333
63	0.47222222	0.08333333	0.50847458	0.37500000
64	0.50000000	0.37500000	0.62711864	0.54166667
65	0.36111111	0.37500000	0.44067797	0.50000000
66	0.66666667	0.45833333	0.57627119	0.54166667
67	0.36111111	0.41666667	0.59322034	0.58333333
68	0.41666667	0.29166667	0.52542373	0.37500000
69	0.52777778	0.08333333	0.59322034	0.58333333
70	0.36111111	0.20833333	0.49152542	0.41666667
71	0.44444444	0.50000000	0.64406780	0.70833333
72	0.50000000	0.33333333	0.50847458	0.50000000
73	0.55555556	0.20833333	0.66101695	0.58333333
74	0.50000000	0.33333333	0.62711864	0.45833333
75	0.58333333	0.37500000	0.55932203	0.50000000
76	0.63888889	0.41666667	0.57627119	0.54166667
77	0.69444444	0.33333333	0.64406780	0.54166667
78	0.66666667	0.41666667	0.67796610	0.66666667
79	0.47222222	0.37500000	0.59322034	0.58333333
80	0.38888889	0.25000000	0.42372881	0.37500000
81	0.33333333	0.16666667	0.47457627	0.41666667
82	0.33333333	0.16666667	0.45762712	0.37500000
83	0.41666667	0.29166667	0.49152542	0.45833333
84	0.47222222	0.29166667	0.69491525	0.62500000
85	0.30555556	0.41666667	0.59322034	0.58333333
86	0.47222222	0.58333333	0.59322034	0.62500000
87	0.66666667	0.45833333	0.62711864	0.58333333
88	0.55555556	0.12500000	0.57627119	0.50000000
89	0.36111111	0.41666667	0.52542373	0.50000000
90	0.33333333	0.20833333	0.50847458	0.50000000
91	0.33333333	0.25000000	0.57627119	0.45833333
92	0.50000000	0.41666667	0.61016949	0.54166667
93	0.41666667	0.25000000	0.50847458	0.45833333
94	0.19444444	0.12500000	0.38983051	0.37500000
95	0.36111111	0.29166667	0.54237288	0.50000000
96	0.38888889	0.41666667	0.54237288	0.45833333
97	0.38888889	0.37500000	0.54237288	0.50000000
98	0.52777778	0.37500000	0.55932203	0.50000000
99	0.22222222	0.20833333	0.33898305	0.41666667
100	0.38888889	0.33333333	0.52542373	0.50000000
101	0.55555556	0.54166667	0.84745763	1.00000000
102	0.41666667	0.29166667	0.69491525	0.75000000
103	0.77777778	0.41666667	0.83050847	0.83333333

104	0.55555556	0.37500000	0.77966102	0.70833333
105	0.61111111	0.41666667	0.81355932	0.87500000
106	0.91666667	0.41666667	0.94915254	0.83333333
107	0.16666667	0.20833333	0.59322034	0.66666667
108	0.83333333	0.37500000	0.89830508	0.70833333
109	0.66666667	0.20833333	0.81355932	0.70833333
110	0.80555556	0.66666667	0.86440678	1.00000000
111	0.61111111	0.50000000	0.69491525	0.79166667
112	0.58333333	0.29166667	0.72881356	0.75000000
113	0.69444444	0.41666667	0.76271186	0.83333333
114	0.38888889	0.20833333	0.67796610	0.79166667
115	0.41666667	0.33333333	0.69491525	0.95833333
116	0.58333333	0.50000000	0.72881356	0.91666667
117	0.61111111	0.41666667	0.76271186	0.70833333
118	0.94444444	0.75000000	0.96610169	0.87500000
119	0.94444444	0.25000000	1.00000000	0.91666667
120	0.47222222	0.08333333	0.67796610	0.58333333
121	0.72222222	0.50000000	0.79661017	0.91666667
122	0.36111111	0.33333333	0.66101695	0.79166667
123	0.94444444	0.33333333	0.96610169	0.79166667
124	0.55555556	0.29166667	0.66101695	0.70833333
125	0.66666667	0.54166667	0.79661017	0.83333333
126	0.80555556	0.50000000	0.84745763	0.70833333
127	0.52777778	0.33333333	0.64406780	0.70833333
128	0.50000000	0.41666667	0.66101695	0.70833333
129	0.58333333	0.33333333	0.77966102	0.83333333
130	0.80555556	0.41666667	0.81355932	0.62500000
131	0.86111111	0.33333333	0.86440678	0.75000000
132	1.00000000	0.75000000	0.91525424	0.79166667
133	0.58333333	0.33333333	0.77966102	0.87500000
134	0.55555556	0.33333333	0.69491525	0.58333333
135	0.50000000	0.25000000	0.77966102	0.54166667
136	0.94444444	0.41666667	0.86440678	0.91666667
137	0.55555556	0.58333333	0.77966102	0.95833333
138	0.58333333	0.45833333	0.76271186	0.70833333
139	0.47222222	0.41666667	0.64406780	0.70833333
140	0.72222222	0.45833333	0.74576271	0.83333333
141	0.66666667	0.45833333	0.77966102	0.95833333
142	0.72222222	0.45833333	0.69491525	0.91666667
143	0.41666667	0.29166667	0.69491525	0.75000000
144	0.69444444	0.50000000	0.83050847	0.91666667
145	0.66666667	0.54166667	0.79661017	1.00000000
146	0.66666667	0.41666667	0.71186441	0.91666667
147	0.55555556	0.20833333	0.67796610	0.75000000
148	0.61111111	0.41666667	0.71186441	0.79166667
149	0.52777778	0.58333333	0.74576271	0.91666667
150	0.44444444	0.41666667	0.69491525	0.70833333

EXTRACTING TRAINING SET:

```
##extract training set
iris_train <- iris_norm[ran,]
iris_train
```

```
> ##extract training set
> iris_train <- iris_norm[ran,]
> iris_train
    Sepal.Length Sepal.Width Petal.Length Petal.Width
13     0.13888889  0.41666667   0.06779661  0.00000000
145    0.66666667  0.54166667   0.79661017  1.00000000
149    0.52777778  0.58333333   0.74576271  0.91666667
63     0.47222222  0.08333333   0.50847458  0.37500000
27     0.19444444  0.58333333   0.10169492  0.12500000
119    0.94444444  0.25000000   1.00000000  0.91666667
112    0.58333333  0.29166667   0.72881356  0.75000000
87     0.66666667  0.45833333   0.62711864  0.58333333
94     0.19444444  0.12500000   0.38983051  0.37500000
51     0.75000000  0.50000000   0.62711864  0.54166667
107    0.16666667  0.20833333   0.59322034  0.66666667
114    0.38888889  0.20833333   0.67796610  0.79166667
136    0.94444444  0.41666667   0.86440678  0.91666667
19     0.38888889  0.75000000   0.11864407  0.08333333
65     0.36111111  0.37500000   0.44067797  0.50000000
74     0.50000000  0.33333333   0.62711864  0.45833333
41     0.19444444  0.62500000   0.05084746  0.08333333
100    0.38888889  0.33333333   0.52542373  0.50000000
25     0.13888889  0.58333333   0.15254237  0.04166667
9      0.02777778  0.37500000   0.06779661  0.04166667
99     0.22222222  0.20833333   0.33898305  0.41666667
4      0.08333333  0.45833333   0.08474576  0.04166667
64     0.50000000  0.37500000   0.62711864  0.54166667
113    0.69444444  0.41666667   0.76271186  0.83333333
37     0.33333333  0.62500000   0.05084746  0.04166667
15     0.41666667  0.83333333   0.03389831  0.04166667
140    0.72222222  0.45833333   0.74576271  0.83333333
92     0.50000000  0.41666667   0.61016949  0.54166667
50     0.19444444  0.54166667   0.06779661  0.04166667
111    0.61111111  0.50000000   0.69491525  0.79166667
72     0.50000000  0.33333333   0.50847458  0.50000000
134    0.55555556  0.33333333   0.69491525  0.58333333
108    0.83333333  0.37500000   0.89830508  0.70833333
43     0.02777778  0.50000000   0.05084746  0.04166667
49     0.27777778  0.70833333   0.08474576  0.04166667
1      0.22222222  0.62500000   0.06779661  0.04166667
69     0.52777778  0.08333333   0.59322034  0.58333333
54     0.33333333  0.12500000   0.50847458  0.50000000
59     0.63888889  0.37500000   0.61016949  0.50000000
105    0.61111111  0.41666667   0.81355932  0.87500000
36     0.19444444  0.50000000   0.03389831  0.04166667
104    0.55555556  0.37500000   0.77966102  0.70833333
118    0.94444444  0.75000000   0.96610169  0.87500000
121    0.72222222  0.50000000   0.79661017  0.91666667
110    0.80555556  0.66666667   0.86440678  1.00000000
39     0.02777778  0.41666667   0.05084746  0.04166667
42     0.05555556  0.12500000   0.05084746  0.08333333
138    0.58333333  0.45833333   0.76271186  0.70833333
40     0.22222222  0.58333333   0.08474576  0.04166667
24     0.22222222  0.54166667   0.11864407  0.16666667
141    0.66666667  0.45833333   0.77966102  0.95833333
```

EXTRACT TESTING DATASET:

##extract testing set
iris_test <- iris_norm[-ran,]
iris_test

```
> ##extract testing set
> iris_test <- iris_norm[-ran,]
> iris_test
    Sepal.Length Sepal.Width Petal.Length Petal.Width
8     0.19444444   0.5833333   0.08474576  0.04166667
10    0.16666667   0.4583333   0.08474576  0.00000000
23    0.08333333   0.6666667   0.00000000  0.04166667
46    0.13888889   0.4166667   0.06779661  0.08333333
80    0.38888889   0.2500000   0.42372881  0.37500000
89    0.36111111   0.4166667   0.52542373  0.50000000
96    0.38888889   0.4166667   0.54237288  0.45833333
106   0.91666667   0.4166667   0.94915254  0.83333333
109   0.66666667   0.2083333   0.81355932  0.70833333
116   0.58333333   0.5000000   0.72881356  0.91666667
126   0.80555556   0.5000000   0.84745763  0.70833333
131   0.86111111   0.3333333   0.86440678  0.75000000
137   0.55555556   0.5833333   0.77966102  0.95833333
142   0.72222222   0.4583333   0.69491525  0.91666667
147   0.55555556   0.2083333   0.67796610  0.75000000
```

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

```
> ##extract 5th column of train dataset because it will be used as 'cl' argument in knn function.
> iris_target_category <- iris[ran,5]
> iris_target_category
  [1] setosa     virginica  virginica  versicolor setosa     virginica  virginica  versicolor
  [9] versicolor versicolor virginica  virginica  virginica  setosa     versicolor versicolor
 [17] setosa     versicolor setosa     setosa     versicolor setosa     versicolor virginica
 [25] setosa     setosa     virginica  versicolor setosa     virginica  versicolor virginica
 [33] virginica  setosa     setosa     setosa     versicolor versicolor versicolor virginica
 [41] setosa     virginica  virginica  virginica  virginica  setosa     setosa     virginica
 [49] setosa     setosa     virginica  setosa     versicolor virginica  setosa     virginica
 [57] versicolor setosa     setosa     setosa     setosa     virginica  versicolor versicolor
 [65] versicolor virginica  setosa     setosa     virginica  versicolor setosa     versicolor
 [73] virginica  setosa     versicolor versicolor versicolor versicolor setosa     setosa
 [81] versicolor virginica  setosa     versicolor setosa     versicolor virginica  versicolor
 [89] virginica  setosa     setosa     virginica  versicolor setosa     virginica  versicolor
 [97] virginica  virginica  versicolor versicolor versicolor virginica  setosa     versicolor
[105] setosa     setosa     versicolor setosa     versicolor setosa     versicolor setosa
[113] virginica  versicolor versicolor versicolor versicolor setosa     versicolor virginica
[121] virginica  setosa     virginica  setosa     setosa     versicolor virginica  virginica
[129] setosa     virginica  versicolor versicolor virginica  versicolor virginica
Levels: setosa versicolor virginica
```

##extract 5th column if test dataset to measure the accuracy

iris_test_category <- iris[-ran,5]

iris_test_category

```
> ##extract 5th column if test dataset to measure the accuracy
> iris_test_category <- iris[-ran,5]
> iris_test_category
 [1] setosa     setosa     setosa     setosa     versicolor versicolor versicolor virginica
 [9] virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

EXECUTE KNN FUNCTION:

##load the package class
library(class)

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

```
> ##load the package class
> library(class)
>
> ##run knn function
> pr <- knn(iris_train,iris_test,cl=iris_target_category,k=13)
> pr
 [1] setosa     setosa     setosa     setosa     versicolor versicolor versicolor virginica
 [9] virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

CREATE CONFUSION MATRIX:

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

```
> ##create confusion matrix
> tab <- table(pr,iris_test_category)
> tab
            iris_test_category
pr           setosa versicolor virginica
  setosa          4          0         0
  versicolor      0          3         0
  virginica       0          0         8
```

CHECKING ACCURACY:

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

```
> ##this function divides the correct predictions by total number of predictions that tell us how
 it is accurate
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(tab)
[1] 100
```

C) IMPLEMENT K-MEANS CLUSTERING

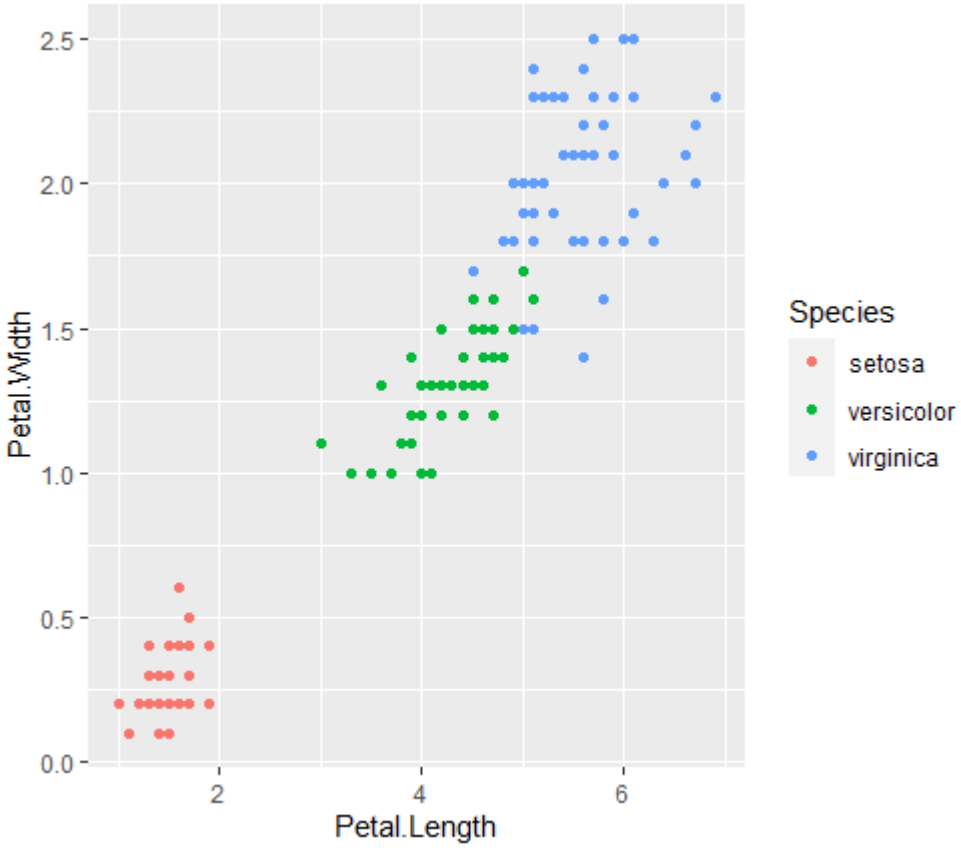
LOADING DATASET:

#K-Means clustering
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
```

LOADING LIBRARY FOR PLOTING GRAPH:

library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()



EXECUTING K-MENAS AND PLOTTING GRAPH:

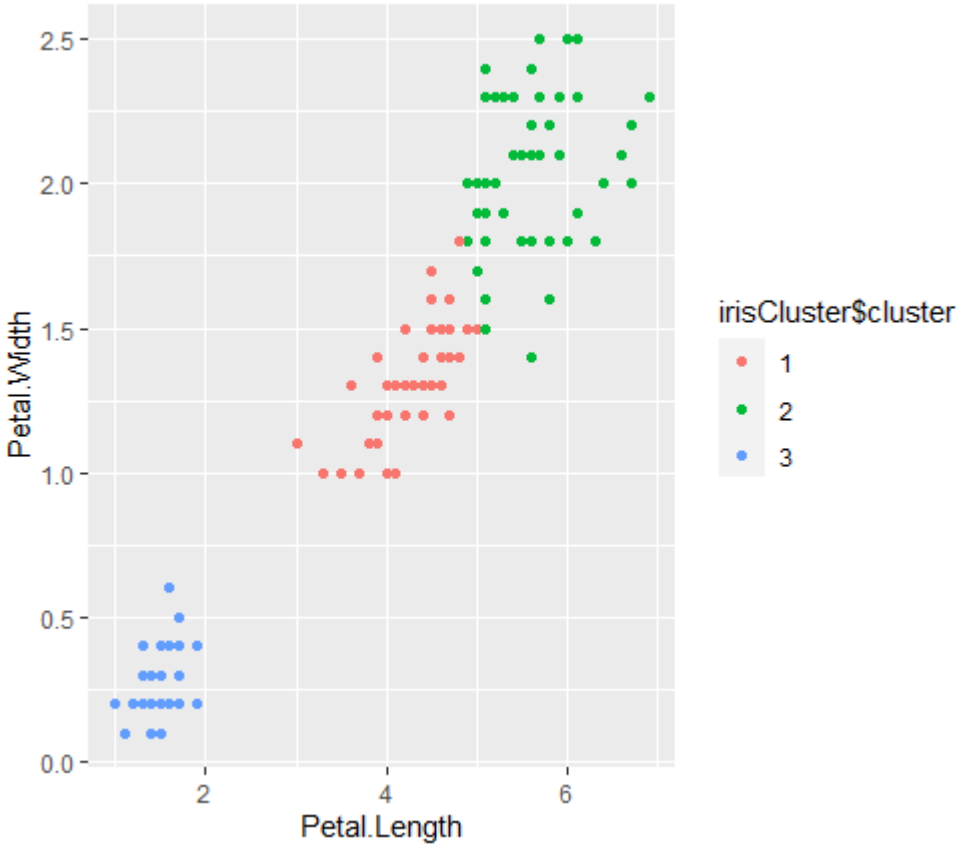
#Setting a seed in R means to initialize a pseudorandom number generator.
set.seed(20)

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

table(irisCluster$cluster, iris$Species)

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

```
> #Setting a seed in R means to initialize a pseudorandom number generator.
> set.seed(20)
>
> #Executing Kmeans
> 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
 [46] 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1
 [91] 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 2 2 2 1 2 2 2 2 2 2 1 2 2
[136] 2 2 2 1 2 2 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"
[7] "size"         "iter"         "ifault"
`
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
>
> irisCluster$cluster <- as.factor(irisCluster$cluster)
> ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
`
```



CONCLUSION:

From this practical, I have learned how to implement naive bayes, k – nn, k – means clustering in R.

AIM: IMPLEMENT APRIORI ALGOEITHM

THEORY:

Apriori Algorithm:

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

Apriori says:

The probability that item I is not frequent is if:

- P(I) < minimum support threshold, then I is not frequent.
- P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to itemset.
- If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property.

The steps followed in the Apriori Algorithm of data mining are:

1. **Join Step** :- This step generates (K+1) itemset from K-itemsets by joining each item with itself.
2. **Prune Step** :- This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

Support: Support is the rate of the frequency of an item appears in the total number of items. Like the frequency of burger among all the transactions. Mathematically, for an item A will be given as –

$$\text{Support}\ (A) = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

Confidence: Confidence is the conditional probability of occurrence of a consequent (then) providing the occurrence of an antecedent (if). It's kind of testing a rule. Like if a customer buys a burger(antecedent), he is supposed to buy french fries(consequent). Mathematically, the confidence of **B** given **A** will be given as-

$$\text{Confidence}\ (A \longrightarrow B) = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

Advantages:

- Easy to understand algorithm.
- Join and Prune steps are easy to implement on large itemsets in large databases

Disadvantages:

- It requires high computation if the itemsets are very large and the minimum support is kept very low.
- The entire database needs to be scanned.

Applications Of Apriori Algorithm :-

Some fields where Apriori is used:

- **In Education Field:** Extracting association rules in data mining of admitted students through characteristics and specialties.
- **In the Medical field:** For example Analysis of the patient's database.
- **In Forestry:** Analysis of probability and intensity of forest fire with the forest fire data.
- Apriori is used by many companies like Amazon in the Recommender System and by Google for the auto-complete feature.

1) **INSTALL & LOAD LIBRARY:**

install.packages("arules")

library(arules)

```
> library(arules)
Loading required package: Matrix

Attaching package: ‘arules’

The following objects are masked from ‘package:base’:

    abbreviate, write

Warning message:
package ‘arules’ was built under R version 4.0.5
```

2) **LOAD DATASET & PRINT SOME RECORDS:**

market_data<-read.csv("C:\\Users\\NARENDER KESWANI\\Downloads\\data_apriori.csv")
head(market_data)

```
> market_data<-read.csv("C:\\Users\\NARENDER KESWANI\\Downloads\\data_apriori.csv")
> head(market_data)
  Customer_Id Products
1           1    bread
2           1   butter
3           1     eggs
4           1     milk
5           4    bread
6           4   butter
```

3) DIVIDING DATA INTO GROUPS:

trans<- split(market_data$Products, market_data$Customer_Id,"trasnactions")
head(trans)

```
> trans<- split(market_data$Products, market_data$Customer_Id,"trasnactions")
> head(trans)
$`1`
[1] "bread"  "butter" "eggs"   "milk"

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

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

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

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

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

4) APPLYING RULES ON DATASET

rules<- apriori(trans,parameter=list(support=0.5, confidence=0.9, maxlen=3, minlen=2))
inspect(rules)

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

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

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

Absolute minimum support count: 7

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[15 item(s), 15 transaction(s)] done [0.00s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [11 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning messages:
1: In asMethod(object) : removing duplicated items in transactions
2: In apriori(trans, parameter = list(support = 0.5, confidence = 0.9,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
> inspect(rules)
     lhs               rhs      support   confidence coverage  lift     count
[1]  {eggs}         => {milk}   0.6000000 1          0.6000000 1.666667 9
[2]  {milk}         => {eggs}   0.6000000 1          0.6000000 1.666667 9
[3]  {butter}       => {bread}  0.6000000 1          0.6000000 1.250000 9
[4]  {butter, eggs} => {milk}   0.5333333 1          0.5333333 1.666667 8
[5]  {butter, milk} => {eggs}   0.5333333 1          0.5333333 1.666667 8
[6]  {bread, eggs}  => {milk}   0.5333333 1          0.5333333 1.666667 8
[7]  {bread, milk}  => {eggs}   0.5333333 1          0.5333333 1.666667 8
[8]  {butter, eggs} => {bread}  0.5333333 1          0.5333333 1.250000 8
[9]  {bread, eggs}  => {butter} 0.5333333 1          0.5333333 1.666667 8
[10] {butter, milk} => {bread}  0.5333333 1          0.5333333 1.250000 8
[11] {bread, milk}  => {butter} 0.5333333 1          0.5333333 1.666667 8
```

CONCLUSION:

From this practical, I have learned how to implement the apriori algorithm in R.

AIM: IMPLEMENTATION AND ANALYSIS OF LINEAR REGRESSION THROUGH GRAPHICAL METHODS.

THEORY:

Linear regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression- Simple and Multiple.

Linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other. For example, using temperature in degree Celsius it is possible to accurately predict Fahrenheit. Statistical relationship is not accurate in determining relationship between two variables. For example, relationship between height and weight.

The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.

Y(pred) = b0 + b1*x

The values b0 and b1 must be chosen so that they minimize the error. If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.

Error Calculation

$$\text{Error} = \sum_{i=1}^{n} (actual\_output - predicted\_output) ** 2$$

If we don't square the error, then positive and negative point will cancel out each other.
For model with one predictor,

Intercept Calculation:

$$b_0 = \bar{y} - b_1 \bar{x}$$

Co-efficient Formula:

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

A) SIMPLE LINEAR REGRESSION:

SOURCE CODE:

LOADING LIBRARY & DATASET:

```
#load library
library(ggplot2)


#load cars dataset
my_data <-mtcars


#printing names of columns
names(my_data)
```

```
> #load library
> library(ggplot2)
>
> #load cars dataset
> my_data <-mtcars
> #load library
> library(ggplot2)
>
> #load cars dataset
> my_data <-mtcars
>
> #printing names of columns
> names(my_data)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear" "carb"
```

PRINTING DIMENSIONS OF DATASET:

#printing dimensions of dataset

dim(my_data)

```
> #printing dimensions of dataset
> dim(my_data)
[1] 32 11
```

CREATING RANDOM SAMPLE:

#randomize

my_data <- my_data[sample(nrow(my_data),),]

head(my_data)

```
> #randomize
> my_data <- my_data[sample(nrow(my_data), ), ]
> head(my_data)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

CREATING TRAINING & TESTING DATASET:

#Creating Training & Testing Dataset
TrainData <- my_data[1:20,]
TestData <- my_data[21:32,]

TrainData
TestData

```
> #Creating Training & Testing Dataset
> TrainData <- my_data[1:20,]
> TestData <- my_data[21:32,]
>
> TrainData
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> TestData
                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Camaro Z28       13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Ford Pantera L   15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Merc 240D        24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Hornet 4 Drive   21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
AMC Javelin      15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Merc 230         22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Datsun 710       22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Honda Civic      30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
```

CREATING LINEAR REGRESSION MODEL AND PRINTING RESULTS:

```
## Linear Model
fit = lm(mpg ~ hp, data=mtcars)
summary(fit)

preds <- predict(fit, newdata = TestData)
df1 <- data.frame(preds,TestData$mpg)
head(df1)
```

```
> ## Linear Model
> fit = lm(mpg ~ hp, data=mtcars)
> summary(fit)

Call:
lm(formula = mpg ~ hp, data = mtcars)

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

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

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,    Adjusted R-squared:  0.5892
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07

>
> preds <- predict(fit, newdata = TestData)
> df1 <- data.frame(preds,TestData$mpg)
> head(df1)
                   preds TestData.mpg
Camaro Z28       13.38293         13.3
Ford Pantera L   12.08660         15.8
Lotus Europa     22.38907         30.4
Pontiac Firebird 18.15891         19.2
Merc 240D        25.86871         24.4
Hornet 4 Drive   22.59375         21.4
```

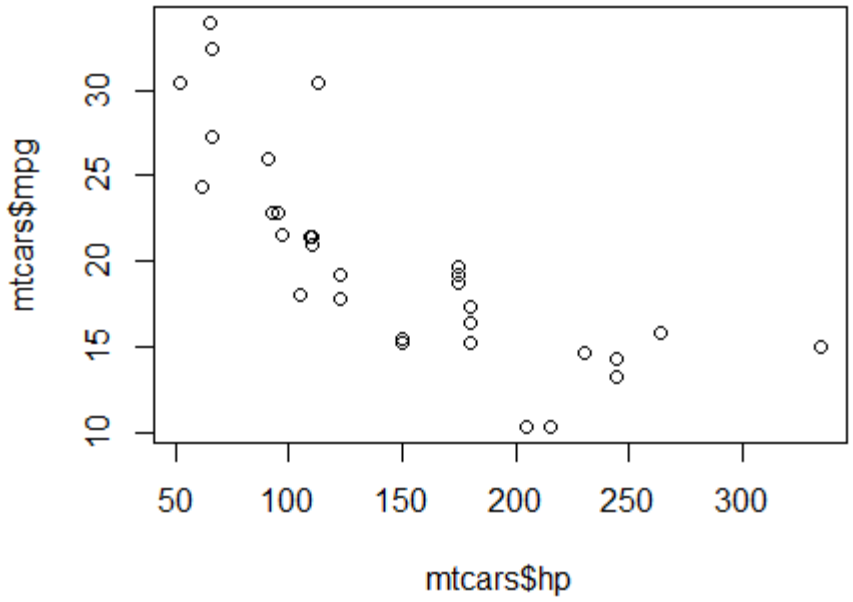
CALCULATE CORRELATION:

```
#correlation
cor(preds,TestData$mpg)
```

```
> cor(preds,TestData$mpg)
[1] 0.8115641
```

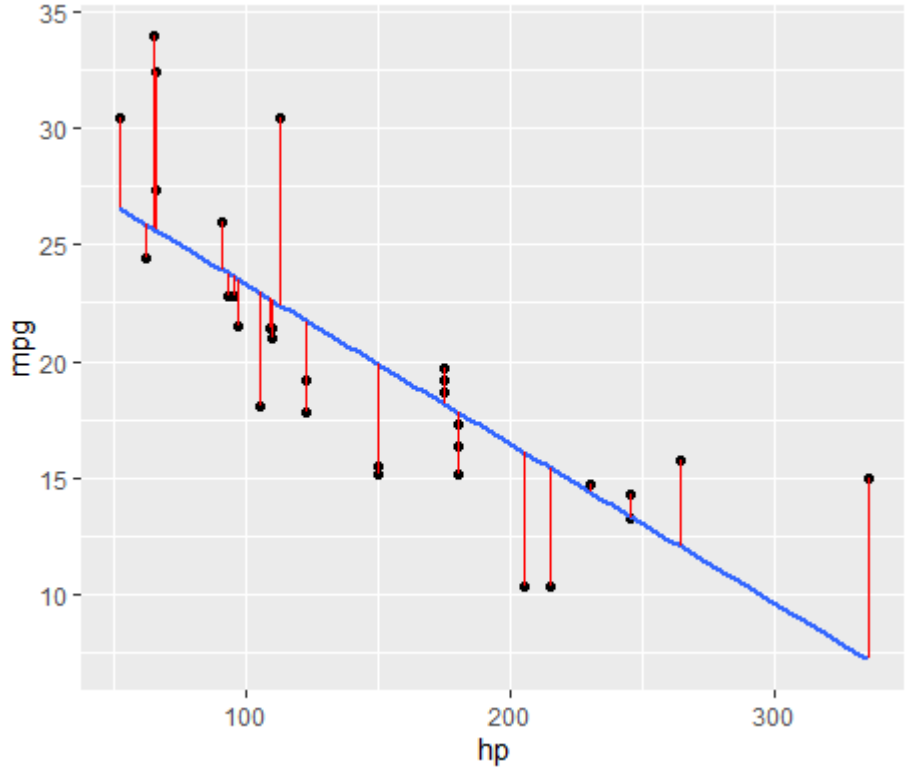
PLOTTING POINTS:

plot(mtcars$hp, mtcars$mpg)



PLOTTING LINEAR REGRESSION GRAPH:

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



B) MULTI LINEAR REGRESSION:

SOURCE CODE:

CREATING MULTI LINEAR REGRESSION MODEL & PRINTING SUMMARY:

```
fit = lm(mpg ~ hp, data=mtcars)
summary(fit)

preds <- predict(fit, newdata = TestData)
df1 <- data.frame(preds,TestData$mpg)
head(df1)
```

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

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

Residuals:
    Min      1Q  Median      3Q     Max
-3.5841 -1.6948 -0.7332  0.8613  6.0984

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.62113    8.27833   4.424 0.000493 ***
hp          -0.01719    0.02116  -0.813 0.429114
cyl         -1.12428    0.97720  -1.151 0.267946
gear         0.44956    1.46444   0.307 0.763078
wt          -2.72647    1.27234  -2.143 0.048931 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.997 on 15 degrees of freedom
Multiple R-squared:  0.8164,    Adjusted R-squared:  0.7675
F-statistic: 16.68 on 4 and 15 DF,  p-value: 2.145e-05

>
> preds_new <- predict(lmmodel1, newdata = TestData)
> df2 <- data.frame(preds_new,TestData$mpg)
> head(df2)
                 preds_new TestData.mpg
Camaro Z28        14.29345         13.3
Ford Pantera L    16.69262         15.8
Lotus Europa      28.30375         30.4
Pontiac Firebird  15.48339         19.2
Merc 240D         24.15879         24.4
Hornet 4 Drive    20.56722         21.4
```

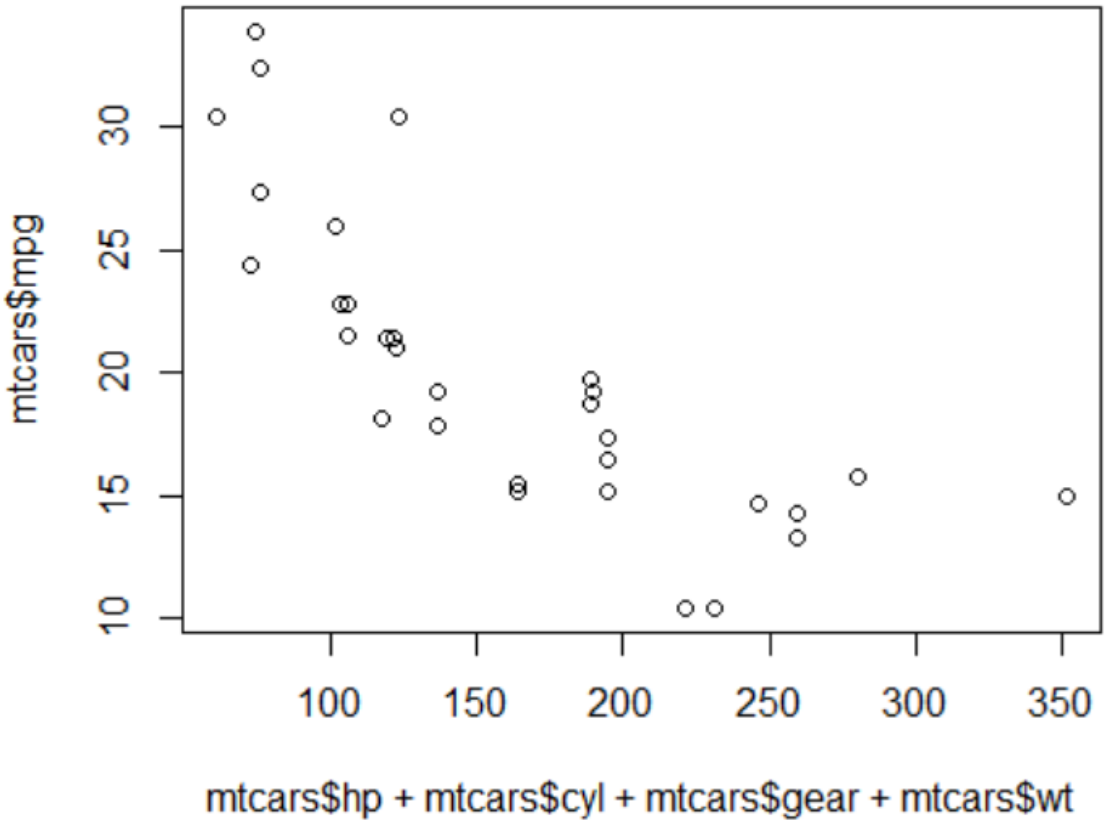
CALCULATING CORREALATION:

#correlation
cor(preds_new,TestData$mpg)

```
> cor(preds_new,TestData$mpg)
[1] 0.934887
```

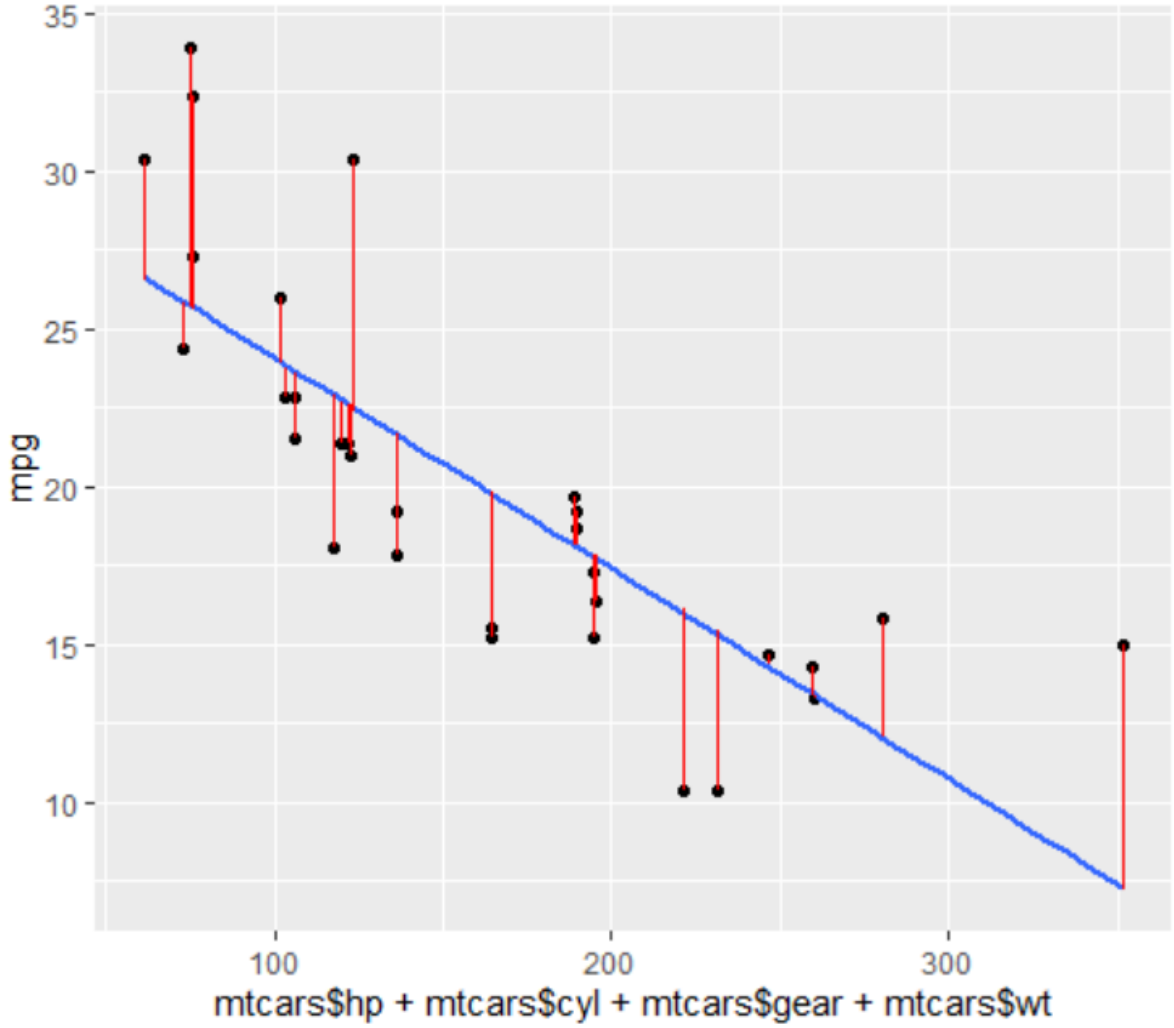
PLOTTING POINTS:

plot(mtcars$hp+mtcars$cyl+mtcars$gear+mtcars$wt, mtcars$mpg)



PLOTTING MULTI LINEAR REGRESSION GRAPH:

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



CONCLUSION:

From this practical, I have learned how to implement linear regression in r programming.

AIM: IMPLEMENTATION AND ANALYSIS OF CLUSTERING ALGORITHMS LIKE K MEANS AGGLOMERATIVE

THEORY:

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

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

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

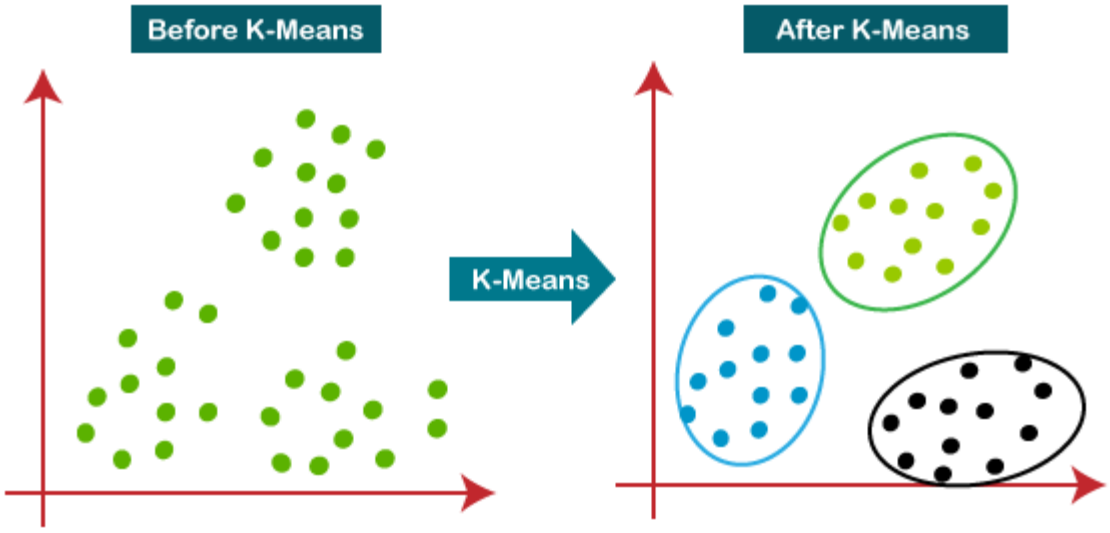
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

FUNCTIONS USED IN R:

`hclust:` Performs hierarchical clustering on a distance or similarity structure.

`cutree:` Returns a vector of group numbers for the observations that were clustered. Specify either the number of groups desired or a clustering height.

A) IMPLEMENT K-MEANS CLUSTERING

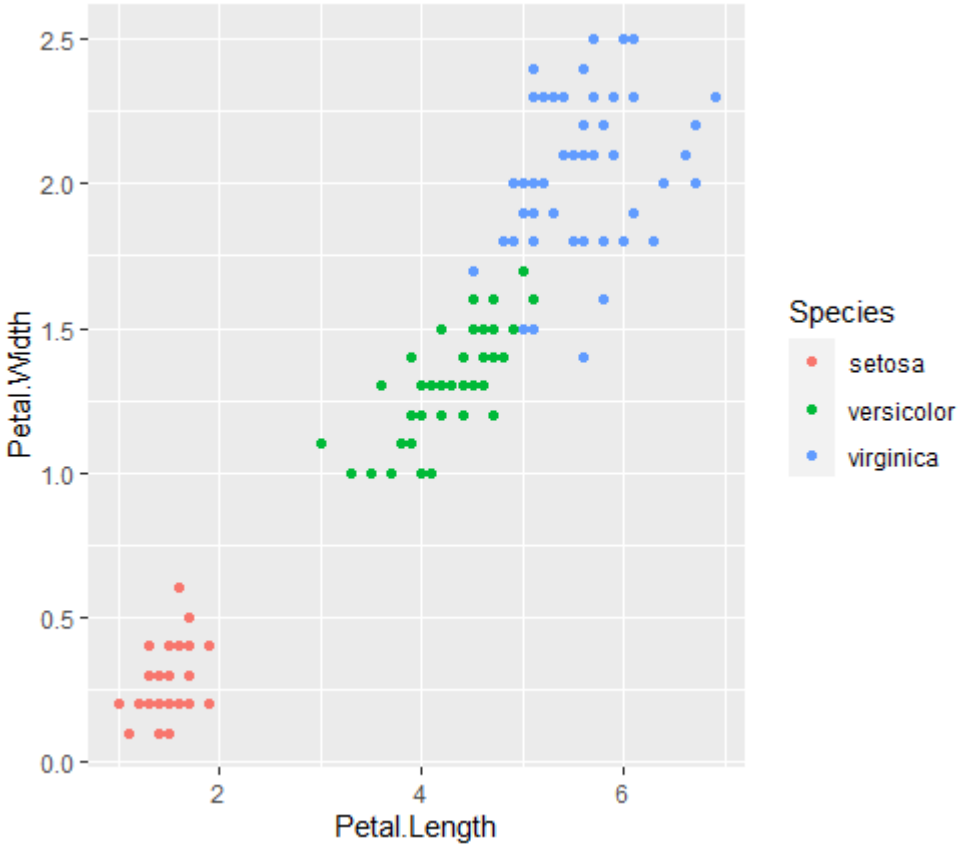
LOADING DATASET:

#K-Means clustering
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
```

LOADING LIBRARY FOR PLOTING GRAPH:

library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()



EXECUTING K-MENAS AND PLOTTING GRAPH:

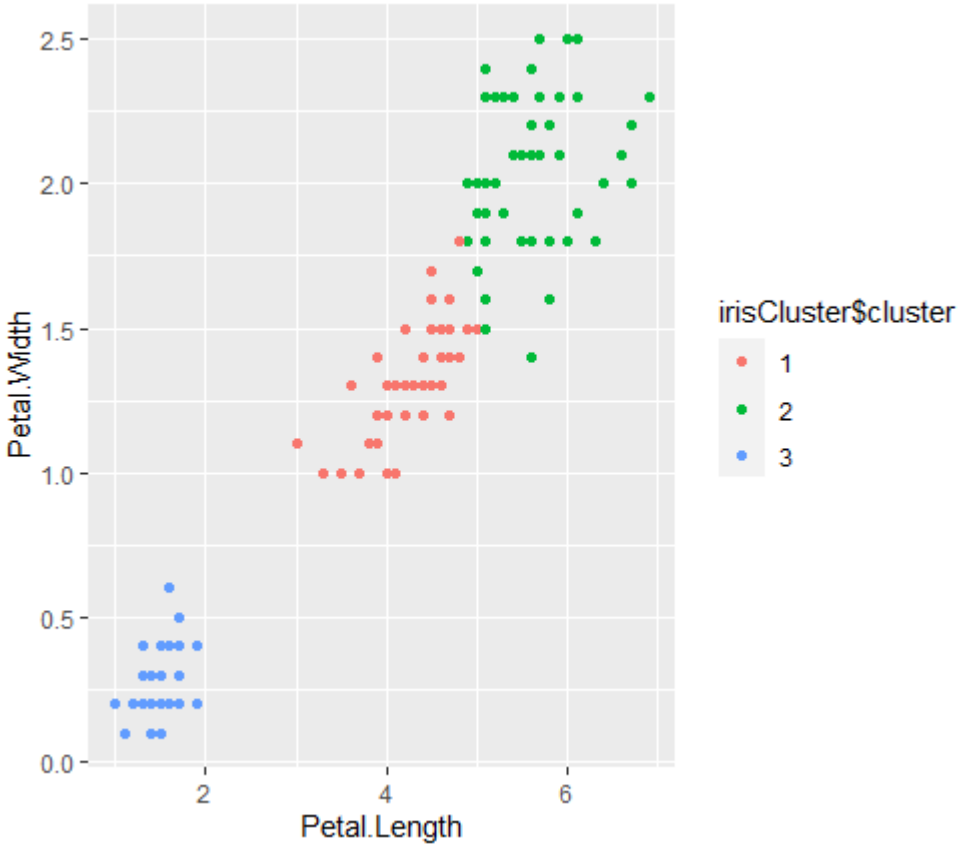
#Setting a seed in R means to initialize a pseudorandom number generator.
set.seed(20)

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

table(irisCluster$cluster, iris$Species)

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

```
> #Setting a seed in R means to initialize a pseudorandom number generator.
> set.seed(20)
>
> #Executing Kmeans
> 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
 [46] 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1
 [91] 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 2 2 2 1 2 2 2 2 2 2 1 2 2
[136] 2 2 2 1 2 2 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"
[7] "size"         "iter"         "ifault"
`
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
>
> irisCluster$cluster <- as.factor(irisCluster$cluster)
> ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
`
```


2.5
2.0
1.5
1.0
0.5
0.0
Petal.Width
2
4
6
Petal.Length
irisCluster$cluster
1
2
3

B) K-MEANS AGGOLOMERATIVE

LOADING DATASET:

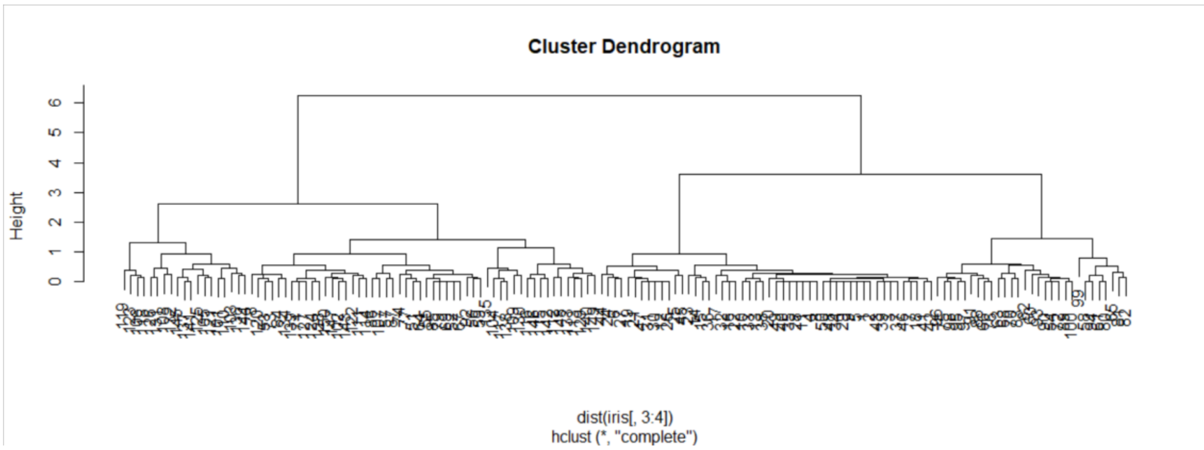
#K-Means Agglomerative
head(iris)

```
> #K-Means Agglomerative
> 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
```

CREATING CLUSTERS:

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

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

Call:
hclust(d = dist(iris[, 3:4]))

Cluster method   : complete
Distance         : euclidean
Number of objects: 150

> plot(clusters)
```



CREATING GROUP OF CLUSTERS:

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

```
> clusterCut<- cutree(clusters, 3)
> clusterCut
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [46] 1 1 1 1 1 2 2 2 3 2 2 2 3 2 3 3 3 3 2 3 3 2 3 2 3 2 3 2 2 3 3 2 2 2 3 3 3 3 2 2 2 2 3 3 3
 [91] 3 2 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[136] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> table(clusterCut, iris$Species)

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

CREATING AVERAGE LINKAGE CLUSTERS:

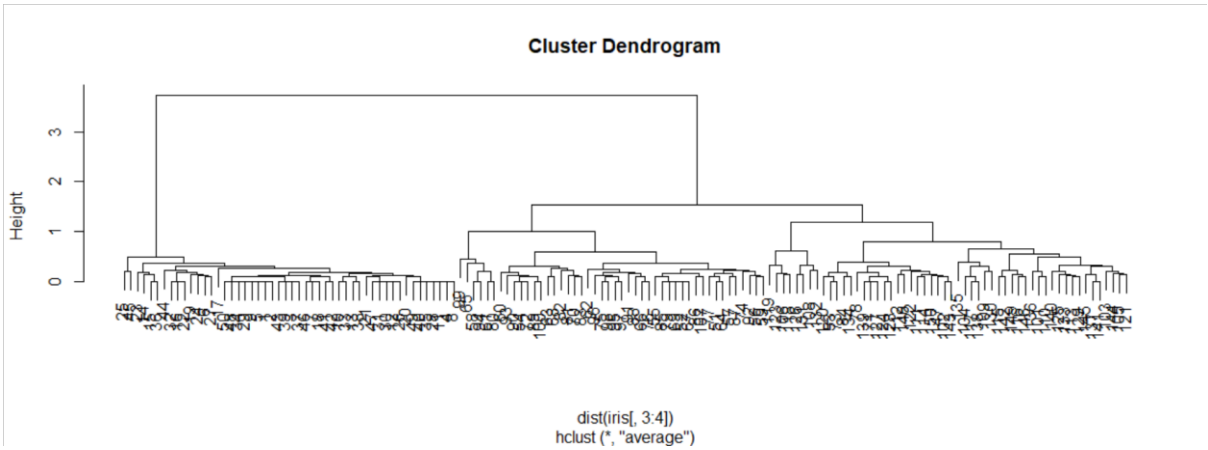
clusters <- hclust(dist(iris[, 3:4]), method = 'average')
clusters
plot(clusters)

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

Call:
hclust(d = dist(iris[, 3:4]), method = "average")

Cluster method   : average
Distance         : euclidean
Number of objects: 150

> plot(clusters)
```

CREATING GROUP CLUSTER FOR AVERGE LINKAGE

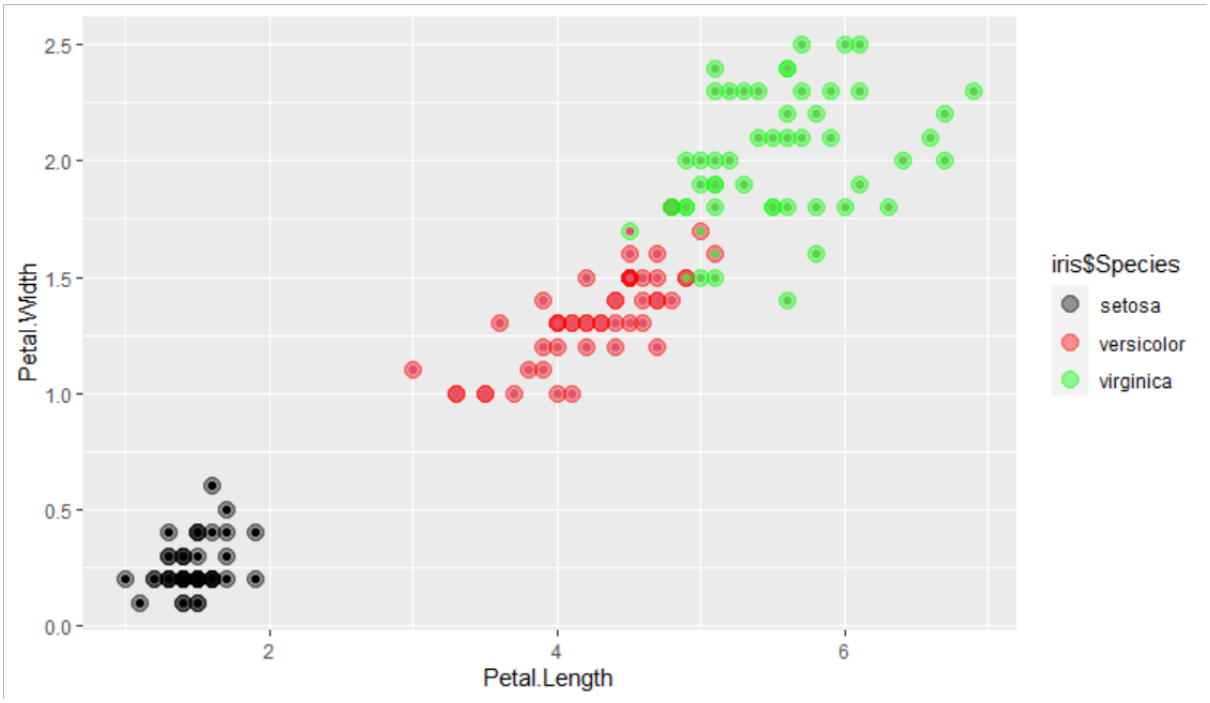
clusterCut <- cutree(clusters, 3)
clusterCut
table(clusterCut,iris$Species)

```
> clusterCut
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [46] 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2
 [91] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 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
[136] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
> table(clusterCut,iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         45         1
         3      0          5        49
```



PLOTTING CLUSTERD GRAPH:

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'))



CONCLUSION:

From this practical, I have learned how to implement k – means & aggolomerative clustering in R.