Shiromani Gurdwara Prabandhak Committee's (S.G.P.C.)
GURU NANAK INSTITUTE OF MANAGEMENT STUDIES
(Management Institute of G N Khalsa College),
Matunga, Mumbai – 400 019
Affiliated to University of Mumbai

Name : Khan Tabish Mujeeb

Roll No : 25MCA-33

Subject code : MCAL13

Subject Name : Advanced Database Management System Lab

Shiromani Gurdwara Prabandhak Committee's (S.G.P.C.)
GURU NANAK INSTITUTE OF MANAGEMENT STUDIES
(Management Institute of G N Khalsa College),
Matunga, Mumbai – 400 019
Affiliated to University of Mumbai

Advanced Database Management System Lab

Index

PRACTICAL 1

Aim: Implementation of Data partitioning through Hash, Range and List Partitioning.

Writeup: Data partitioning is a technique used to divide a large set of data into smaller, more manageable parts, or "partitions." This makes it easier to distribute the data across multiple servers, ensuring faster access, better performance, and easier management. There are several ways to partition data, each with its own method and use case. Three common types of data partitioning are Hash Partitioning, Range Partitioning, and List Partitioning.

1.Hash Partitioning

Hash partitioning splits data based on a hash function. A hash function takes the data (like a key or a column value) and computes a fixed-size output, called a hash value. The data is then placed in a partition based on the hash value.

•The hash function assigns a numerical value to each data element.

•This value determines which partition the data should go into.

•For example, if you have 3 partitions, the hash function might divide the data into partitions 1, 2, or 3 based on the hash value.

Example: Suppose you're partitioning data based on the "user_id" column and you have 4 partitions.

•A hash function will calculate the hash value of the "user_id" and assign it to one of the four partitions.

•If hash(user_id) % 4 == 0, the data goes to Partition 0; if hash(user_id) % 4 == 1, it goes to Partition 1, and so on.

Advantages:

•The distribution is usually even, which helps avoid skewed partitions.

•It’s good for situations where you don’t need to maintain a specific order of data. Disadvantages:

•Hash partitioning can be difficult to reverse. You cannot easily predict in which partition a given record will land just by looking at the data.

•Adding or removing partitions can be complex, as it may require rehashing existing data.

2.Range Partitioning

Range partitioning divides data into partitions based on a specified range of values. For example, you might partition data by date or by age range.

•You define specific ranges for a column, and the data is grouped into those ranges.

For example, if you’re partitioning by the "age" of people, you might have partitions for:

•Ages 0-20 in Partition 1,

•Ages 21-40 in Partition 2,

•Ages 41-60 in Partition 3, and so on.

Example: If you have a "sales_date" column and you're partitioning by the year:

•Partition 1 could hold data from 2020,

•Partition 2 could hold data from 2021,

•Partition 3 could hold data from 2022, etc. Advantages:

•Useful for time-based data, like partitioning by year, quarter, or month.

•Easy to understand and manage, as data is organized by a natural range. Disadvantages:

•If the data isn't evenly distributed, some partitions may have much more data than others (this is known as "skew").

•Adding new ranges may require reorganizing data if the range definitions change

3.List Partitioning

List partitioning divides data into partitions based on specific, predefined lists of values. Instead of dividing by ranges (like in Range Partitioning), the data is grouped based on certain categories.

•You define a list of values for each partition.

•Each data item is assigned to a partition based on whether its value matches one of the values in the predefined list.

Example: If you're partitioning by "region," you might define:

•Partition 1 for North America,

•Partition 2 for Europe,

•Partition 3 for Asia,

•Partition 4 for Africa, etc. Advantages:

•Easy to understand and manage, especially when the categories are well-defined.

•Great for cases where you have clear categories, such as geographical locations or product types.

Disadvantages:

•If the categories are too broad or don't have a natural division, partitions may be uneven.

•If new categories are added or existing ones change, you might have to reorganize data

Code:

1. Create the student Table with Range Partitioning:

Design a table named student with the following columns:

s_id (integer) – student ID

s_name (varchar of 20 characters) – student name

dob (date) – date of birth

Partition this table based on the year of birth (dob) using range partitions:

p0: Includes students born before 1990.

p1: Includes students born between 1990 and 1994.

p2: Includes students born between 1995 and 1999.

create table student(s_id int, s_name varchar(20), dob date)partition by range(year(dob))(partition p0 values less than (1990),partition p1 values less than(1995),partition p2 values less than (2000));

insert into student values(2,'ajay','1998-10-20'),(3,'sumit','1989-9-14'),(4,'vijay','1988-04-21'),(5,'joshna','1999-04-24'),(6,'ankita','1994-04-23');

Table Creation:

Query:

```
CREATE TABLE student (
   s_id   INT,
   s_name VARCHAR2(20),
   dob    DATE
)
PARTITION BY RANGE (dob) (
   PARTITION p0 VALUES LESS THAN (TO_DATE('01-Jan-1990','DD-MON-YYYY')),
   PARTITION p1 VALUES LESS THAN (TO_DATE('01-Jan-1995','DD-MON-YYYY')),
   PARTITION p2 VALUES LESS THAN (TO_DATE('01-Jan-2000','DD-MON-YYYY'))
);
```

Output:

```
SQL> CREATE TABLE student (
  2      s_id   INT,
  3      s_name VARCHAR2(20),
  4      dob    DATE
  5  )
  6  PARTITION BY RANGE (dob) (
  7      PARTITION p0 VALUES LESS THAN (TO_DATE('01-Jan-1990','DD-MON-YYYY')),
  8      PARTITION p1 VALUES LESS THAN (TO_DATE('01-Jan-1995','DD-MON-YYYY')),
  9      PARTITION p2 VALUES LESS THAN (TO_DATE('01-Jan-2000','DD-MON-YYYY'))
 10  );

Table created.
```

Inserting values into table:

```
SQL> INSERT INTO student values(1, 'Atharva','12-May-1997');

1 row created.

SQL> INSERT INTO student values(2, 'Ajay', '20-Oct-1998');

1 row created.

SQL> INSERT INTO student values(3, 'Sumit', '14-Sep-1989');

1 row created.

SQL> INSERT INTO student values(4, 'Vijay', '21-Apr-1988');

1 row created.

SQL> INSERT INTO student values(5, 'Joshna', '24-Apr-1999');

1 row created.

SQL> INSERT INTO student values(6, 'Ankita', '23-Apr-1994');

1 row created.
```

Display Table:

```
SQL> select * from student;

      S_ID S_NAME               DOB
---------- -------------------- ---------
         3 Sumit                14-SEP-89
         4 Vijay                21-APR-88
         6 Ankita               23-APR-94
         1 Atharva              12-MAY-97
         2 Ajay                 20-OCT-98
         5 Joshna               24-APR-99

6 rows selected.
```

2. Design a table named stores with the following columns:

shop_id (integer) – store shop ID

s_name (varchar of 20 characters) – store name

st_number (integer) – store contact number

email_id (varchar of 40 characters) – store email

store_id (integer) – unique store identifier

Partition this table based on the store_id using list partitions:

peast: Includes store IDs 104, 106, and 109.

pwest: Includes store IDs 103, 105, and 107.

pnorth: Includes store IDs 101, 102, and 108.

psouth: Includes store IDs 110, 111, and 112.

Table Creation:

```
SQL> create table stores(shop_id int, s_name varchar(20),st_number int,email_id varchar(40),store_id int) partition
 by list(store_id)
  2  (partition peast values (104,106,109),
  3  partition pwest values (103,105,107),
  4  partition pnorth values (101,102,108),
  5  partition psouth values (110,111,112));

Table created.
```

Inserting values in table:

```
SQL> insert into stores values(1,'pune_stores',2937363,'pn@gmail.com',104);

1 row created.

SQL> insert into stores values(2,'mumbai_stores',2937364,'mu@gmail.com',105);

1 row created.

SQL> insert into stores values(3,'hyderabad_stores',2937365,'hy@gmail.com',104);

1 row created.
```

Display Table:

```
SQL> select * from stores;

   SHOP_ID S_NAME               ST_NUMBER EMAIL_ID                                STORE_ID
---------- -------------------- ---------- ---------------------------------------- ----------
         1 pune_stores             2937363 pn@gmail.com                                104
         3 hyderabad_stores        2937365 hy@gmail.com                                106
         2 mumbai_stores           2937364 mu@gmail.com                                105
```


3. Fire the following Queries:

Solution:

1. Retrieve all records from the student table that belong to partition p1.

select * from student partition(p1);

2. Write a query to count the number of rows in each partition of the student table. The result should show partition_name and table_rows for each partition.

select partition_name,table_rows from information_schema.partitions where table_name='student';

Output:

```
SQL> select * from student partition(p1);

      S_ID S_NAME               DOB
---------- -------------------- ---------
         6 Ankita               23-APR-94
```

Practical 2

Aim: Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Rank AND Dense Rank using oracle.

Writeup: Analytical Queries in Oracle: ROLLUP, CUBE, FIRST, LAST, RANK, and DENSE_RANK
Oracle provides a suite of powerful analytical functions to perform advanced data analysis in SQL.

Among them, ROLLUP, CUBE, FIRST, LAST, RANK, and DENSE_RANK are commonly used to generate summaries, rankings, and provide insights into datasets. Here's an overview and implementation of these functions:

1.ROLLUP

The ROLLUP operator is used for generating summary data by grouping the result set in a hierarchical manner. It performs aggregation at multiple levels of a hierarchy, rolling up the intermediate summaries to a grand total.

2.CUBE:

The CUBE operator extends ROLLUP by generating subtotals for all combinations of the specified grouping columns. It is useful for performing multidimensional aggregation and allows the calculation of all possible groupings.

3.FIRST and LAST:

The FIRST and LAST functions return the first and last values in a specific ordering of a column within a group.

4.RANK:

The RANK function assigns a rank to each row in the result set, with ties receiving the same rank. However, it leaves gaps in the ranking if there are ties (e.g., if two rows are ranked 1, the next row will be ranked 3).

5.DENSE_RANK:

The DENSE_RANK function is similar to RANK, but it does not leave gaps in the ranking when there are ties. For example, if two rows have the same rank, the next row will be ranked immediately after (i.e., no gaps).

Conclusion

These analytical functions in Oracle are essential tools for business intelligence and data analysis. They allow for flexible aggregation, ranking, and windowing operations, providing deep insights into data across multiple dimensions.

--for formatting purpose only

-- set lines 256

set trim out on set tab off

Query:

Create a Partitioned Table (for ROLLUP and CUBE):

Query:

CREATE TABLE partitioned_employees (

```
    employee_id NUMBER,
    ename VARCHAR2(50),
    department VARCHAR2(50),
    job VARCHAR2(50),
    salary NUMBER,
    hire_date DATE
)
PARTITION BY RANGE (hire_date) (
    PARTITION emp_hiredate_jan2020 VALUES LESS THAN (TO_DATE('2020-02-01', 'YYYY-MM-DD')),
    PARTITION emp_hiredate_feb2020 VALUES LESS THAN (TO_DATE('2020-03-01', 'YYYY-MM-DD')),
    PARTITION emp_hiredate_mar2020 VALUES LESS THAN (TO_DATE('2020-04-01', 'YYYY-MM-DD')),
    -- Add more partitions as needed
    PARTITION emp_hiredate_default VALUES LESS THAN (MAXVALUE)
);
```

Output:

```
SQL> CREATE TABLE partitioned_employees (
  2      employee_id NUMBER,
  3      ename VARCHAR2(50),
  4      department VARCHAR2(50),
  5      job VARCHAR2(50),
  6      salary NUMBER,
  7      hire_date DATE
  8  ) PARTITION BY RANGE (hire_date) (
  9      PARTITION emp_hiredate_jan2020 VALUES LESS THAN (TO_DATE('2020-02-01', 'YYYY-MM-DD')),
 10      PARTITION emp_hiredate_feb2020 VALUES LESS THAN (TO_DATE('2020-03-01', 'YYYY-MM-DD')),
 11      PARTITION emp_hiredate_mar2020 VALUES LESS THAN (TO_DATE('2020-04-01', 'YYYY-MM-DD')),
 12      -- Add more partitions as needed
 13      PARTITION emp_hiredate_default VALUES LESS THAN (MAXVALUE)
 14  );

Table created.

SQL>
```

```
INSERT INTO partitioned_employees VALUES (201, 'Robert King', 'HR', 'Manager', 5000, TO_DATE('2020-01-15', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (202, 'Olivia Clark', 'Finance', 'Analyst', 4500, TO_DATE('2020-02-10', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (203, 'Ethan Harris', 'IT', 'Developer', 6000, TO_DATE('2020-03-20', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (204, 'Sophia Turner', 'Marketing', 'Executive', 4800, TO_DATE('2020-01-25', 'YYYY-MM-DD'));
```

```
INSERT INTO partitioned_employees VALUES (205, 'William Scott', 'Sales', 'Lead', 5200,
TO_DATE('2020-02-05', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (206, 'Isabella Adams', 'HR', 'Assistant', 3500,
TO_DATE('2020-03-15', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (207, 'Lucas Carter', 'IT', 'Admin', 5600,
TO_DATE('2020-01-05', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (208, 'Mia Robinson', 'Finance', 'Manager', 7000,
TO_DATE('2020-04-10', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (209, 'Henry Evans', 'Sales', 'Executive', 4900,
TO_DATE('2020-02-28', 'YYYY-MM-DD'));

INSERT INTO partitioned_employees VALUES (210, 'Ava Mitchell', 'Marketing', 'Lead', 5300,
TO_DATE('2020-03-25', 'YYYY-MM-DD'));

CREATE TABLE partitioned_employees_rank (
    employee_id NUMBER,
    ename VARCHAR2(50),
    department VARCHAR2(50),
    job VARCHAR2(50),
    salary NUMBER,
    hire_date DATE
)
PARTITION BY LIST (department) (
    PARTITION emp_it VALUES ('IT'),
    PARTITION emp_hr VALUES ('HR'),
    PARTITION emp_sales VALUES ('Sales'),
    -- Add more partitions as needed
    PARTITION emp_other VALUES (DEFAULT)
);
```

Output:

```
SQL> SELECT department, job, SUM(salary) AS total_salary
  2  FROM employees
  3  GROUP BY CUBE (department, job);

DEPARTMENT                                         JOB                                                TOTAL_SALARY
-------------------------------------------------- -------------------------------------------------- ------------
                                                                                                              836000
                                                   HR Manager                                                 182000
                                                   Programmer                                                 162000
                                                   Sales Manager                                               95000
                                                   HR Coordinator                                              70000
                                                   Sales Associate                                            153000
                                                   Database Administrator                                     174000
HR                                                                                                            252000
HR                                                 HR Manager                                                 182000
HR                                                 HR Coordinator                                              70000
IT                                                                                                            336000

DEPARTMENT                                         JOB                                                TOTAL_SALARY
-------------------------------------------------- -------------------------------------------------- ------------
IT                                                 Programmer                                                 162000
IT                                                 Database Administrator                                     174000
Sales                                                                                                         248000
Sales                                              Sales Manager                                               95000
Sales                                              Sales Associate                                            153000

16 rows selected.
```

Practical 3

Aim: Implementation of ORDBMS concepts like ADT (Abstract Data Types), Object table and Inheritance

Theory:

1. Abstract Data Types (ADT)

What is ADT?

ADT is a custom data structure that encapsulates data attributes and behaviors (methods) into a single entity, resembling a class in object-oriented programming. It provides a reusable and structured way to represent real-world entities.

How to Implement in Oracle

Step 1: Define an ADT using the CREATE TYPE command.

Example:

```
CREATE TYPE EmployeeType AS OBJECT (
   EMPLOYEE_ID NUMBER,
   ENAME VARCHAR2(50),
   DEPARTMENT VARCHAR2(50),
   JOB VARCHAR2(50),
   SALARY NUMBER,
   MEMBER FUNCTION AnnualSalary RETURN NUMBER
);
```

- EMPLOYEE_ID, ENAME, etc., are attributes.
- MEMBER FUNCTION AnnualSalary is a method (behavior).

Step 2: Define the behavior (method) in the ADT body.

```
CREATE TYPE BODY EmployeeType AS
   MEMBER FUNCTION AnnualSalary RETURN NUMBER IS
   BEGIN
      RETURN SALARY * 12;
   END;
END;
```

- The body implements the functionality defined in the ADT.

2. Object Tables

What are Object Tables?

Object Tables store rows where each row is an object instance of a defined ADT. They bring a direct mapping of objects into the database, enabling a natural representation of data.

How to Implement in Oracle

Step 1: Create an object table based on the ADT.

Example:

```
CREATE TABLE Employees OF EmployeeType;
```

Step 2: Insert data into the object table.

```
INSERT INTO Employees VALUES (EmployeeType(101, 'John Doe', 'HR', 'Manager', 75000));

INSERT INTO Employees VALUES (EmployeeType(102, 'Jane Smith', 'IT', 'Developer', 65000));
```

Step 3: Query the object table.

```
SELECT VALUE(e) FROM Employees e;
```

- VALUE(e) fetches the entire object stored in each row.

Step 4: Use object methods in queries.

```
SELECT e.ENAME, e.AnnualSalary() AS ANNUAL_SALARY

FROM Employees e;
```

3. Inheritance

What is Inheritance?

Inheritance allows creating a new ADT (subtype) based on an existing ADT (supertype). The subtype inherits attributes and methods from the supertype and can add or override them.

How to Implement in Oracle

Step 1: Define a subtype using UNDER.

Example:

```
CREATE TYPE ManagerType UNDER EmployeeType (

   TEAM_SIZE NUMBER,

   OVERRIDE MEMBER FUNCTION AnnualSalary RETURN NUMBER

);
```

- UNDER specifies that ManagerType inherits from EmployeeType.
- TEAM_SIZE is an additional attribute in the subtype.
- OVERRIDE allows redefining the AnnualSalary method.

Step 2: Implement the overridden method.

```
CREATE TYPE BODY ManagerType AS

   OVERRIDE MEMBER FUNCTION AnnualSalary RETURN NUMBER IS

   BEGIN

      RETURN (SALARY * 12) + (TEAM_SIZE * 1000);

   END;

END;
```

Step 3: Create an object table for the subtype.

CREATE TABLE Managers OF ManagerType;

Step 4: Insert and query data in the subtype table.

Insert Data:

INSERT INTO Managers VALUES (ManagerType(201, 'Alice Johnson', 'HR', 'Manager', 85000, 5));

Query Data:

SELECT m.ENAME, m.AnnualSalary() AS ANNUAL_SALARY FROM Managers m;

Key Benefits of These Concepts

1. **Reusability**: ADTs and inheritance allow defining reusable structures and methods.
2. **Encapsulation**: ADTs bundle data and behavior together, improving clarity and maintainability.
3. **Polymorphism**: Subtypes can override methods and be queried alongside supertypes.

Code:

Q1. Creating tables with ADT and References for customer, purchase order ,stock, lineitems table.

a. **Create tables:**

CREATE TABLE Customer (

CustNo NUMBER NOT NULL,

CustName VARCHAR2(200) NOT NULL,

Street VARCHAR2(200) NOT NULL,

City VARCHAR2(200) NOT NULL,

State CHAR(2) NOT NULL,

Zip VARCHAR2(20) NOT NULL,

Phone1 VARCHAR2(20),

Phone2 VARCHAR2(20),

Phone3 VARCHAR2(20),

PRIMARY KEY (CustNo));

Output:

```
SQL> CREATE TABLE Customer (
  2  CustNo NUMBER NOT NULL,
  3  CustName VARCHAR2(200) NOT NULL,
  4  Street VARCHAR2(200) NOT NULL,
  5  City VARCHAR2(200) NOT NULL,
  6  State CHAR(2) NOT NULL,
  7  Zip VARCHAR2(20) NOT NULL,
  8  Phone1 VARCHAR2(20),
  9  Phone2 VARCHAR2(20),
 10  Phone3 VARCHAR2(20),
 11  PRIMARY KEY (CustNo));

Table created.
```

CREATE TABLE PurchaseOrder (

PONo NUMBER,

Custno NUMBER references Customer,

OrderDate DATE,

ShipDate DATE,

ToStreet VARCHAR2(200),

ToCity VARCHAR2(200),

ToState CHAR(2),

ToZip VARCHAR2(20),

PRIMARY KEY(PONo));

Output:

```
SQL> CREATE TABLE PurchaseOrder (
  2  PONo NUMBER,
  3  Custno NUMBER references Customer,
  4  OrderDate DATE,
  5  ShipDate DATE,
  6  ToStreet VARCHAR2(200),
  7  ToCity VARCHAR2(200),
  8  ToState CHAR(2),
  9  ToZip VARCHAR2(20),
 10  PRIMARY KEY(PONo));

Table created.
```

CREATE TABLE Stock (

StockNo NUMBER PRIMARY KEY,

Price NUMBER,

TaxRate NUMBER);

Output:

```
Table created.

SQL> CREATE TABLE Stock (
  2  StockNo NUMBER PRIMARY KEY,
  3  Price NUMBER,
  4  TaxRate NUMBER);

Table created.
```

CREATE TABLE LineItems (

LineItemNo NUMBER,

PONo NUMBER REFERENCES PurchaseOrder,

StockNo NUMBER REFERENCES Stock,

Quantity NUMBER,

Discount NUMBER,

PRIMARY KEY (PONo, LineItemNo));

```
SQL> CREATE TABLE LineItems (
  2  LineItemNo NUMBER,
  3  PONo NUMBER REFERENCES PurchaseOrder,
  4  StockNo NUMBER REFERENCES Stock,
  5  Quantity NUMBER,
  6  Discount NUMBER,
  7  PRIMARY KEY (PONo, LineItemNo));

Table created.
```

b. Inserting data into tables with ADT and references

1)Insert data into Stock Table:

INSERT INTO Stock VALUES(1004, 6750.00, 2);

INSERT INTO Stock VALUES(1011, 4500.23, 2);

INSERT INTO Stock VALUES(1534, 2234.00, 2);

INSERT INTO Stock VALUES(1535, 3456.23, 2);

Output:

```
SQL> INSERT INTO Stock VALUES(1004, 6750.00, 2);

1 row created.

SQL> INSERT INTO Stock VALUES(1011, 4500.23, 2);

1 row created.

SQL> INSERT INTO Stock VALUES(1534, 2234.00, 2);

1 row created.

SQL> INSERT INTO Stock VALUES(1535, 3456.23, 2);

1 row created.
```

2)Insert data into Customer Table:

INSERT INTO Customer VALUES (1, 'Jean Nance', '2 Avocet Drive','Redwood Shores', 'CA', '95054','415-555-1212', NULL, NULL);

INSERT INTO Customer VALUES (2, 'John Nike', '323 College Drive','Edison', 'NJ', '08820','609-555-1212', '201-555-1212', NULL);

Output:

```
SQL>
SQL> INSERT INTO Customer VALUES (1, 'Jean Nance', '2 Avocet Drive','Redwood Shores', 'CA', '95054','415-555-1212', NULL, NULL

1 row created.

SQL> INSERT INTO Customer VALUES (2, 'John Nike', '323 College Drive','Edison', 'NJ', '08820','609-555-1212', '201-555-1212',

1 row created.
```

3) Insert data into PurchaseOrder Table:

INSERT INTO PurchaseOrder VALUES (1001, 1, SYSDATE, '10-MAY-1997',NULL, NULL, NULL, NULL);

INSERT INTO PurchaseOrder VALUES (2001, 2, SYSDATE, '20-MAY-1997','55 Madison Ave', 'Madison', 'WI', '53715');

Output:

```
SQL> INSERT INTO PurchaseOrder VALUES (1001, 1, SYSDATE, '10-MAY-1997',NULL, NULL, NULL, NULL);

1 row created.

SQL> INSERT INTO PurchaseOrder VALUES (2001, 2, SYSDATE, '20-MAY-1997','55 Madison Ave', 'Madison', 'WI', '53715'

1 row created.
```

3) Insert data into LineItems Table:

INSERT INTO LineItems VALUES(01, 1001, 1534, 12, 0);

INSERT INTO LineItems VALUES(02, 1001, 1535, 10, 10);

INSERT INTO LineItems VALUES(01, 2001, 1004, 1, 0);

INSERT INTO LineItems VALUES(02, 2001, 1011, 2, 1);

INSERT INTO LineItems VALUES(01, 2001, 1534, 2, 1);

Output:

```
SQL>
SQL> INSERT INTO LineItems VALUES(01, 1001, 1534, 12, 0);

1 row created.

SQL> INSERT INTO LineItems VALUES(02, 1001, 1535, 10, 10);

1 row created.

SQL> INSERT INTO LineItems VALUES(01, 2001, 1004, 1, 0);

1 row created.

SQL> INSERT INTO LineItems VALUES(02, 2001, 1011, 2, 1);

1 row created.
```

c. simple query on table:

1. Get the Total Value of Purchase Orders

Query:

SELECT P.PONo, SUM(S.Price * L.Quantity)

FROM PurchaseOrder P,

LineItems L,

Stock S

WHERE P.PONo = L.PONo

AND L.StockNo = S.StockNo

GROUP BY P.PONo;

Output:

```
SQL> SELECT P.PONo, SUM(S.Price * L.Quantity)
  2  FROM PurchaseOrder P,
  3  LineItems L,
  4  Stock S
  5  WHERE P.PONo = L.PONo
  6  AND L.StockNo = S.StockNo
  7  GROUP BY P.PONo;

 Order
Number SUM(S.PRICE*L.QUANTITY)
------ -----------------------
  2001                15750.46
  1001                 61370.3
```

2.Get the Purchase Order and Line Item Data for Stock Item 1004

Query:

SELECT P.PONo, P.CustNo,

L.StockNo, L.LineItemNo, L.Quantity, L.Discount

FROM PurchaseOrder P,LineItems L

WHERE P.PONo = L.PONo

AND L.StockNo = 1004;

Output:

```
SQL> SELECT P.PONo, P.CustNo,
  2  L.StockNo, L.LineItemNo, L.Quantity, L.Discount
  3  FROM PurchaseOrder P,LineItems L
  4  WHERE P.PONo = L.PONo
  5  AND L.StockNo = 1004;

 Order Customer  Stock     Line
Number   Number Number Item No.    Qty Discount
------ -------- ------ -------- ------ --------
  2001        2   1004        1      1      .00
```

4. Updating Data Under the Relational Model. The application can execute statements like these to update the data:

Query:

UPDATE LineItems SET Quantity = 20 WHERE PONo = 1001 AND StockNo = 1534;

Output:

```
SQL> UPDATE LineItems SET Quantity = 20 WHERE PONo = 1001 AND StockNo = 1534;

1 row updated.

SQL> select * from LineItems;

    Line  Order  Stock
Item No. Number Number    Qty Discount
-------- ------ ------ ------ --------
       1   1001   1534     20      .00
       2   1001   1535     10    10.00
       1   2001   1004      1      .00
       2   2001   1011      2     1.00
```

5. Deleting Data Under the Relational Model:

Query:

DELETE FROM LineItems WHERE PONo = 1001;

DELETE FROM PurchaseOrder WHERE PONo = 1001;

Before deletion:

```
SQL> select * from LineItems;

    Line  Order  Stock
Item No. Number Number    Qty Discount
-------- ------ ------ ------ --------
       1   1001   1534     20      .00
       2   1001   1535     10    10.00
       1   2001   1004      1      .00
       2   2001   1011      2     1.00
```

After Deletion:

```
SQL> DELETE FROM LineItems WHERE PONo = 1001;

2 rows deleted.

SQL> DELETE FROM PurchaseOrder WHERE PONo = 1001;

1 row deleted.

SQL> select * from LineItems;

    Line  Order  Stock
Item No. Number Number    Qty Discount
-------- ------ ------ ------ --------
       1   2001   1004      1      .00
       2   2001   1011      2     1.00
```

```
SQL> COLUMN Order_Number FORMAT 99999 HEADING "Order|Number"
SQL> COLUMN Customer_Number FORMAT 99999 HEADING "Customer|Number"
SQL> COLUMN Order_Date FORMAT A12 HEADING "Order|Date"
SQL> COLUMN SHIPDATE FORMAT A12 HEADING "Ship|Date"
SQL> COLUMN TOSTREET FORMAT A30 HEADING "To|Street"
SQL> COLUMN TOCITY FORMAT A15 HEADING "To|City"
SQL> COLUMN TOZIP FORMAT A10 HEADING "To|ZIP"
SQL>
SQL> SET LINESIZE 120
SQL> SET PAGESIZE 50
SQL>
SQL> SELECT * FROM PurchaseOrder;

 Order Customer Order      Ship         To                             To              To
Number   Number Date       Date         Street                         City            TO ZIP
------ -------- ---------- ------------ ------------------------------ --------------- -- ----------
  2001        2 08-DEC-24  20-MAY-97    55 Madison Ave                 Madison         WI 53715

SQL>
```

Q2. Create an ADT Credit_type for purchase_amt, credit_grade(A,B,C)

Create a table Customer_Rating with attributes Cust_id, Name, and Credits

1. Display name of all customers.

2. Display the credit_grade of the customer whose purchase amount is greater than 25000

3. Display all customers with 'C' credit_grade

4. Display only the Name of customers with 'A' and 'B' Credit_grade

solution :

Create the ADT Credit_type:

```
CREATE TYPE Credit_type AS OBJECT (
   purchase_amt NUMBER,
   credit_grade CHAR(1)
);
```

Output:

```
SQL> CREATE TYPE Credit_type AS OBJECT (
  2      purchase_amt NUMBER,
  3      credit_grade CHAR(1)
  4  );
  5
  6
  7
  8
  9  /

Type created.
```

2)Create the table Customer_Rating

```
CREATE TABLE Customer_Rating (
   Cust_id NUMBER PRIMARY KEY,
   Name VARCHAR2(50),
   Credits Credit_type
);
```

Output:

```
SQL> CREATE TABLE Customer_Rating (
  2       Cust_id NUMBER PRIMARY KEY,
  3       Name VARCHAR2(50),
  4       Credits Credit_type
  5  );

Table created.
```

sample data

```
INSERT INTO Customer_Rating VALUES (1, 'vijay', Credit_type(30000, 'A'));
INSERT INTO Customer_Rating VALUES (2, 'ajay', Credit_type(20000, 'B'));
INSERT INTO Customer_Rating VALUES (3, 'muskaan', Credit_type(27000, 'C'));
INSERT INTO Customer_Rating VALUES (4, 'neel', Credit_type(18000, 'A'));
```

Output:

```
SQL> INSERT INTO Customer_Rating VALUES (1, 'vijay', Credit_type(30000, 'A'));

1 row created.

SQL> INSERT INTO Customer_Rating VALUES (2, 'ajay', Credit_type(20000, 'B'));

1 row created.

SQL> INSERT INTO Customer_Rating VALUES (3, 'muskaan', Credit_type(27000, 'C'));

1 row created.

SQL> INSERT INTO Customer_Rating VALUES (4, 'neel', Credit_type(18000, 'A'));

1 row created.
```

Q.Display name of all customers:

Query:

```
SELECT Name FROM Customer_Rating;
```

Output:

```
SQL> SELECT Name FROM Customer_Rating;

NAME
--------------------------------------------------
vijay
ajay
muskaan
neel
```

Q.Display all customers with 'C' credit_grade

SELECT * FROM Customer_Rating c WHERE c.Credits.credit_grade = 'C';

```
SQL> SELECT * FROM Customer_Rating c WHERE c.Credits.credit_grade = 'C';

   CUST_ID NAME
---------- --------------------------------------------------
CREDITS(PURCHASE_AMT, CREDIT_GRADE)
--------------------------------------------------------------------------------
         3 muskaan
CREDIT_TYPE(27000, 'C')


SQL>
```

Output:

Q.Display only the Name of customers with 'A' and 'B' Credit_grade

SELECT Name FROM Customer_Rating c WHERE c.Credits.credit_grade IN ('A', 'B');

Output:

```
SQL> SELECT Name FROM Customer_Rating c WHERE c.Credits.credit_grade IN ('A', 'B');

NAME
--------------------------------------------------
vijay
ajay
neel

SQL>
```

Practical 4

Aim: Implementation of ETL transformation with Pentaho .

Theory:

A Pentaho transformation is a sequence of steps that transform and manipulate data. It can be used toextract data from various sources, transform the data, and load it into a target system.

- **Step 1: Connect to MySQL Database**
- **MySQL Connection Step:** This step connects to the MySQL database using the providedconnection details.
- **Database Table:** Select the table from which you want to extract data.
- **Step 2: Filter and Select Data**
- **Filter Step:** This step filters the data based on specific conditions.
- **Select Values Step:** This step selects the required columns from the filtered data.
- **Step 3: Transform Data**
- **Transform Step:** This step transforms the data as required. For example, you can use this stepto convert date formats or perform calculations.
- **Step 4: Export Data to Excel**
- **Excel Output Step:** This step exports the transformed data to an Excel file.
- **Excel File Name:** Specify the name and location of the Excel file.
- **Sheet Name:** Specify the name of the sheet in the Excel file.

- **ETL Transformation with Pentaho**

  ETL (Extract, Transform, Load) is a process of extracting data from various sources, transforming it according to business requirements, and loading it into a target system. Pentaho provides a range of transformation steps that can be used to implement ETL processes.

  - **Transformation Steps**

    1. **Copy Data**
       - Source: Table/Excel/Oracle
       - Target: Table/Excel/Oracle
       - Example: Copy data from a MySQL table to an Excel file.
    2. **Sequence**
       - Used to generate a sequence of numbers or dates.
       - Example: Generate a sequence of numbers starting from 1 to 100.
    3. **Calculator**

- Used to perform mathematical operations on data.
- Example: Calculate the sum of two columns.

4. **Concatenation**
   - Used to concatenate two or more fields.
   - Example: Concatenate two columns, "First Name" and "Last Name", into a singlecolumn "Full Name".
5. **Split**
   - Used to split a single field into multiple fields.
   - Example: Split a field "Address" into separate fields "Street", "City", and "State".
6. **Number Range**
   - Used to perform range-based operations on numbers.
   - Example: Filter rows where the value is between 1 and 100.
7. **String Operations**
   - Used to perform string-based operations, such as trimming or converting touppercase.
   - Example: Trim leading and trailing spaces from a column.
8. **Sorting**
   - Used to sort data in ascending or descending order.
   - Example: Sort data by a specific column in ascending order.

Merge Join Transformation

The Merge Join transformation is used to combine data from two tables based on a common key.

•Example: Combine data from two tables, "orders" and "customers", based on the "customer_id" column.

Data Validations

Data validations are used to ensure that the data meets certain conditions before it is loaded into the target system.

•Example: Validate that the "price" column is greater than 0 before loading it into the target table.

Benefits of ETL Transformation with Pentaho

• Easy Data Integration: Pentaho provides an easy-to-use interface for integrating data from various sources.

• Flexible Data Transformation: Pentaho provides a wide range of transformation steps that can be used to manipulate data as required.

• Improved Data Quality: ETL transformation helps to ensure that data is accurate and consistent across systems.

By using Pentaho's ETL transformation capabilities, you can implement complex data transformations

and ensure that your data is accurate, consistent, and up-to-date.

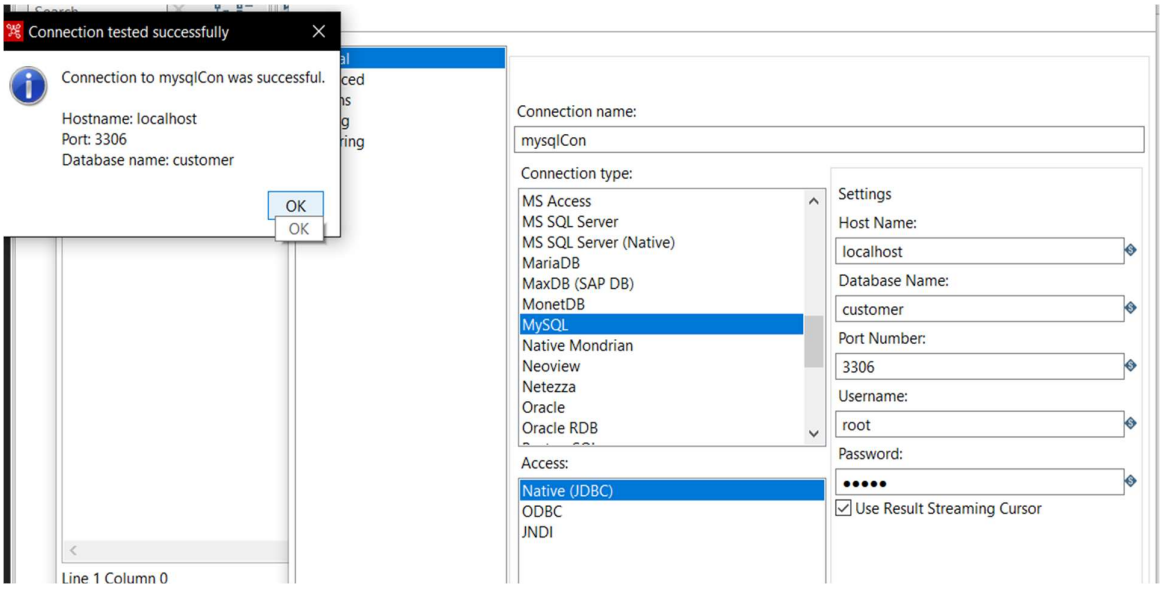
1. create a pentaho transformation to get data from mysql and push required data into an excel file

Create table customer:

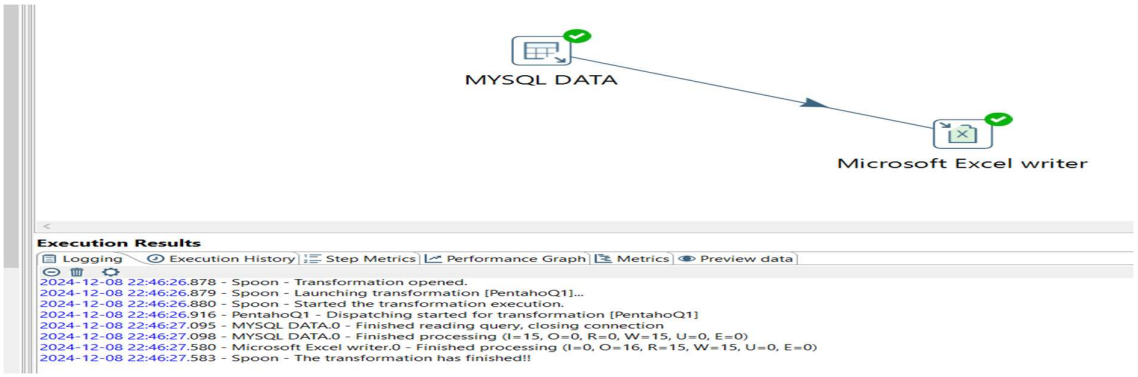
```
mysql> use customer;
Database changed
mysql> CREATE TABLE customer (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     firstname VARCHAR(50) NOT NULL,
    ->     middlename VARCHAR(50),
    ->     lastname VARCHAR(50) NOT NULL,
    ->     date_of_joining DATE NOT NULL,
    ->     pan_number VARCHAR(10) NOT NULL UNIQUE
    -> );
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> select * from customer;
+----+-----------+------------+----------+-----------------+------------+
| id | firstname | middlename | lastname | date_of_joining | pan_number |
+----+-----------+------------+----------+-----------------+------------+
|  1 | John      | Michael    | Doe      | 2022-01-15      | ABCDE1234F |
|  2 | Jane      | Elizabeth  | Smith    | 2023-03-22      | FGHIJ5678K |
|  3 | Alice     | Marie      | Johnson  | 2021-07-10      | LMNOP9876Z |
|  4 | Bob       | William    | Brown    | 2020-11-05      | QRSTU5432A |
|  5 | Charlie   | Edward     | Wilson   | 2022-06-30      | VWXYZ1234B |
|  6 | Diana     | Rose       | Taylor   | 2021-09-18      | CDEFG5678L |
|  7 | Ethan     | Oliver     | Martinez | 2023-02-01      | HIJKL4321M |
|  8 | Fiona     | Grace      | Davis    | 2022-08-25      | NOPQR8765X |
|  9 | George    | Henry      | Moore    | 2021-12-13      | STUVW3456P |
| 10 | Hannah    | Sophia     | Garcia   | 2020-05-07      | XYZAB2345Q |
| 11 | Ian       | Patrick    | Martinez | 2022-03-17      | BCDEF8901T |
| 12 | Jack      | Thomas     | White    | 2023-10-12      | GHIJK6789U |
| 13 | Katherine | Eve        | Miller   | 2022-04-08      | LMNOP1234V |
| 14 | Liam      | James      | Clark    | 2021-01-29      | QRSTU9876W |
| 15 | Mia       | Sophia     | Lewis    | 2020-09-15      | VWXYZ4321R |
+----+-----------+------------+----------+-----------------+------------+
15 rows in set (0.00 sec)
```

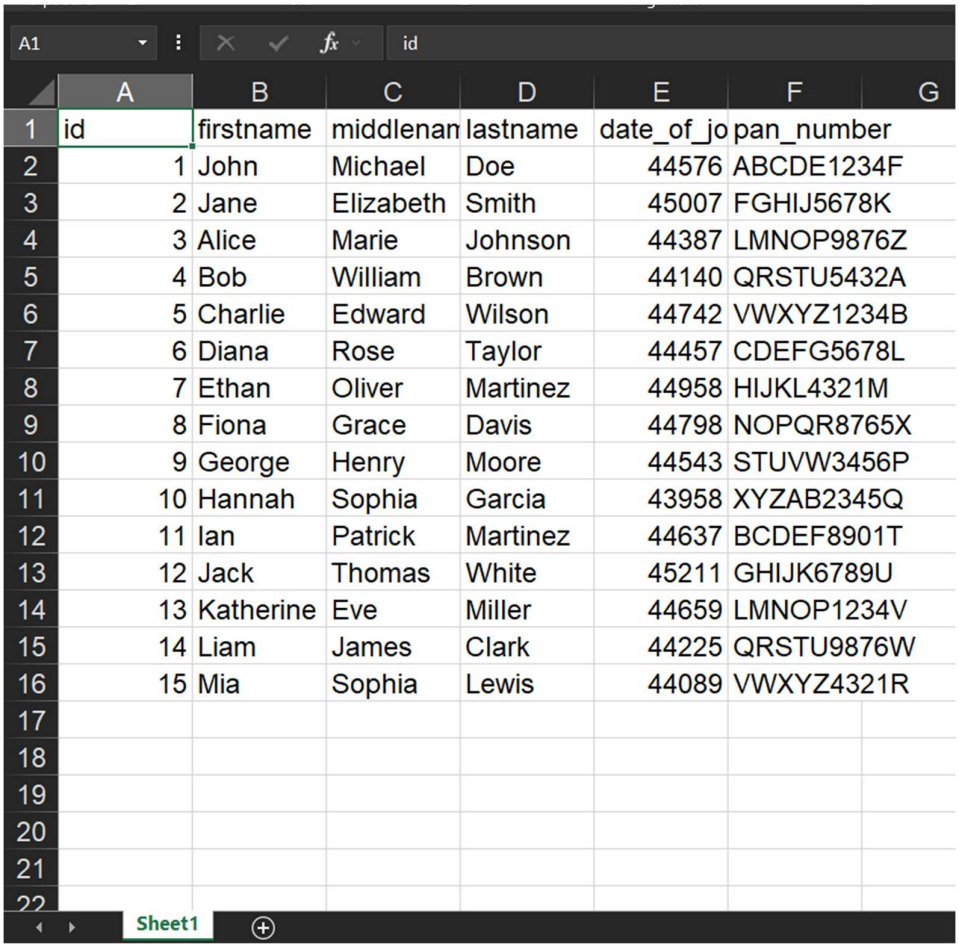
Establish connection with the database:



Transformation:



Data flushed to excel:



	A	B	C	D	E	F	G
1	id	firstname	middlenan	lastname	date_of_jo	pan_number	
2	1	John	Michael	Doe	44576	ABCDE1234F	
3	2	Jane	Elizabeth	Smith	45007	FGHIJ5678K	
4	3	Alice	Marie	Johnson	44387	LMNOP9876Z	
5	4	Bob	William	Brown	44140	QRSTU5432A	
6	5	Charlie	Edward	Wilson	44742	VWXYZ1234B	
7	6	Diana	Rose	Taylor	44457	CDEFG5678L	
8	7	Ethan	Oliver	Martinez	44958	HIJKL4321M	
9	8	Fiona	Grace	Davis	44798	NOPQR8765X	
10	9	George	Henry	Moore	44543	STUVW3456P	
11	10	Hannah	Sophia	Garcia	43958	XYZAB2345Q	
12	11	Ian	Patrick	Martinez	44637	BCDEF8901T	
13	12	Jack	Thomas	White	45211	GHIJK6789U	
14	13	Katherine	Eve	Miller	44659	LMNOP1234V	
15	14	Liam	James	Clark	44225	QRSTU9876W	
16	15	Mia	Sophia	Lewis	44089	VWXYZ4321R	
17							
18							
19							
20							
21							
22							

Sheet1

2. Create a pentaho transformation to get required data from excel file and push it into mysql

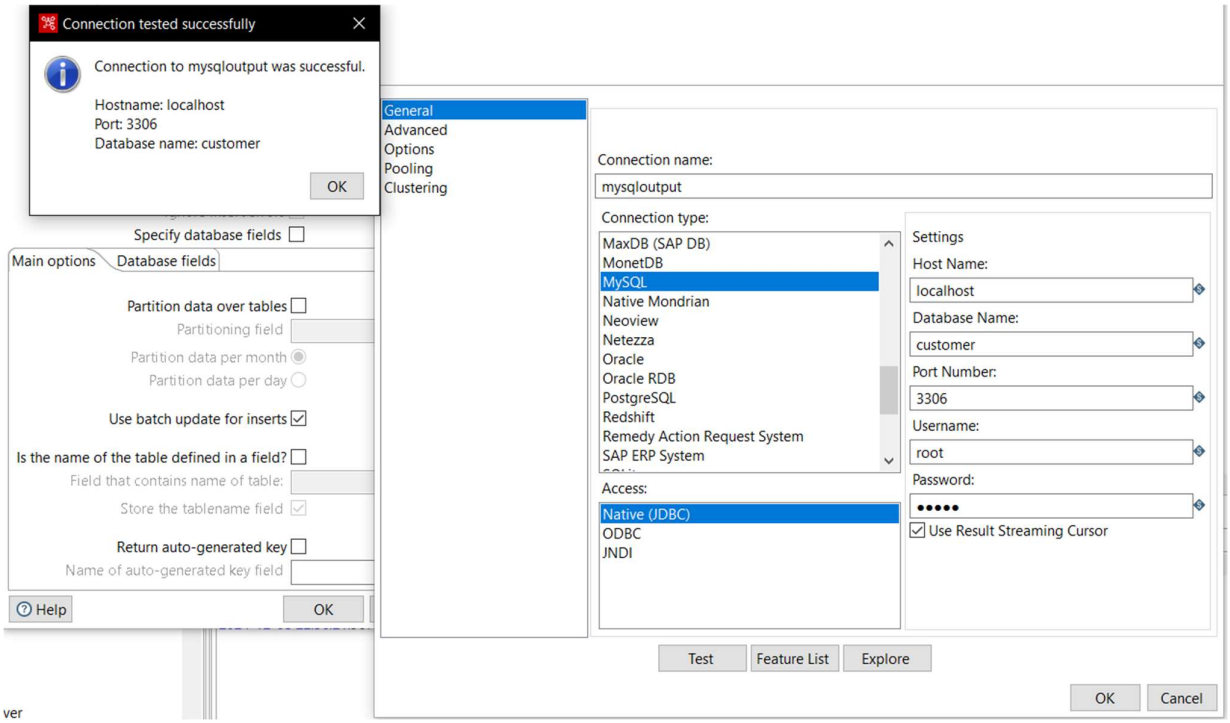
Excel data :

Examine preview data

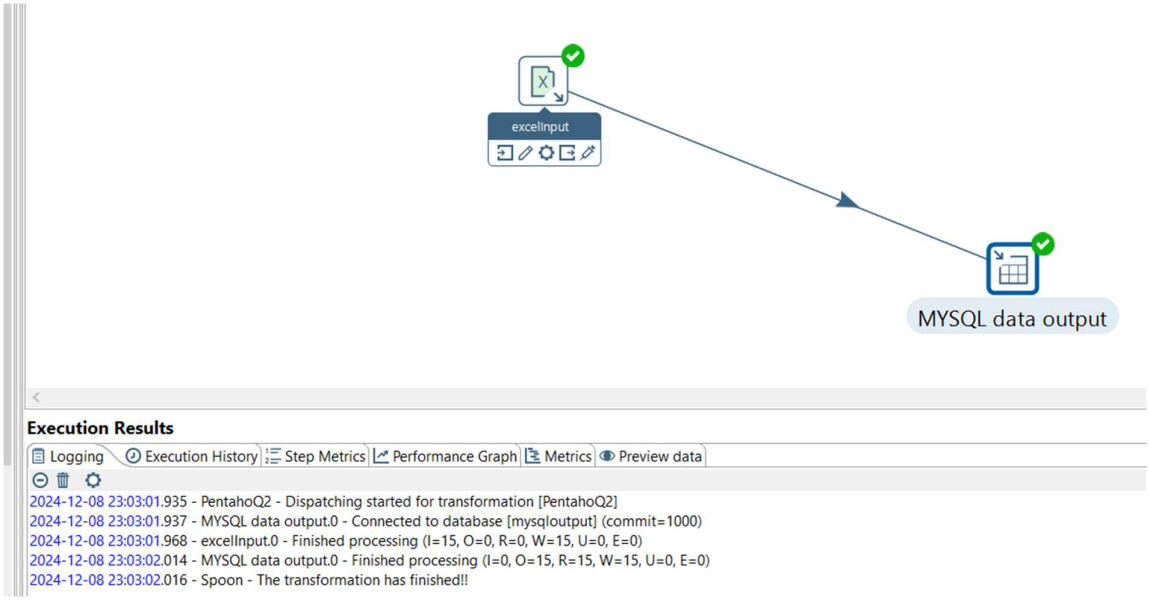
Rows of step: excelInput (15 rows)

#	EID	Full Name	Job Title	Department
1	1.0	Emily Davis	Sr. Manger	IT
2	4.0	Theodore Dinh	Technical Architect	IT
3	2.0	Luna Sanders	Director	Finance
4	3.0	Penelope Jordan	Computer Systems Manager	IT
5	5.0	Austin Vo	Sr. Analyst	Finance
6	7.0	Joshua Gupta	Account Representative	Sales
7	6.0	Ruby Barnes	Manager	IT
8	8.0	Luke Martin	Analyst	Finance
9	9.0	Easton Bailey	Manager	Accounting
1..	11.0	Madeline Walker	Sr. Analyst	Finance
1..	10.0	Savannah Ali	Sr. Manger	Human Resources
1..	13.0	Camila Rogers	Controls Engineer	Engineering
1..	12.0	Eli Jones	Manager	Human Resources
1..	15.0	Everleigh Ng	Sr. Manger	Finance
1..	14.0	Robert Yang	Sr. Analyst	Accounting
1..				

Establishing database connection:



Transformation:

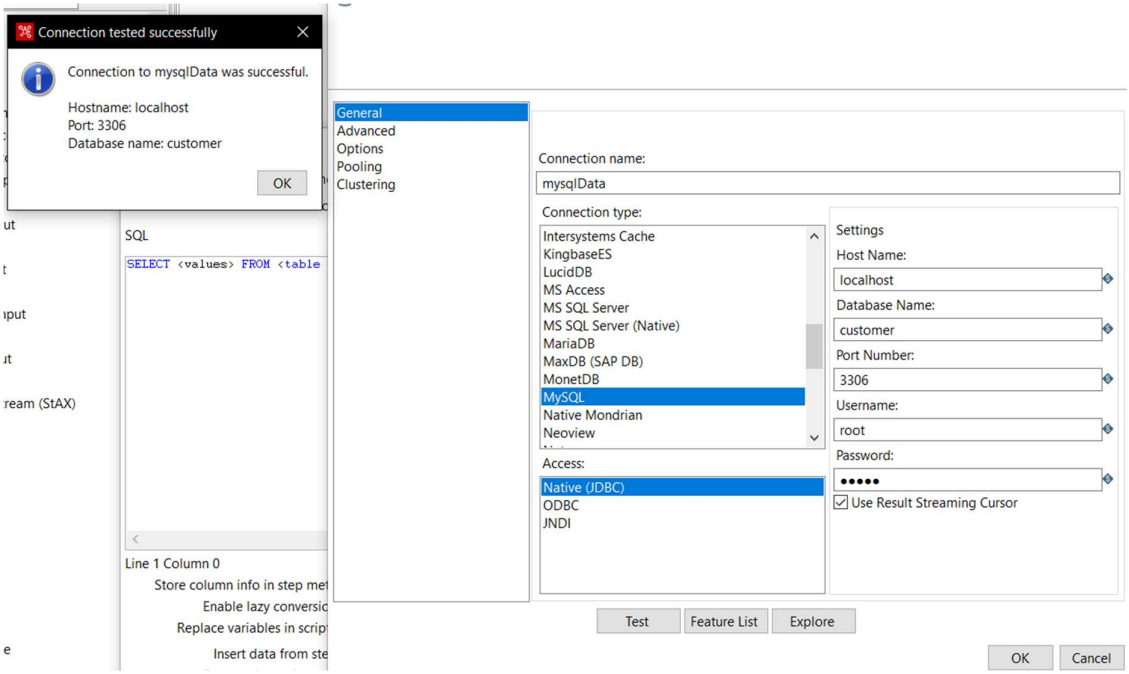


Flushed data to the database:

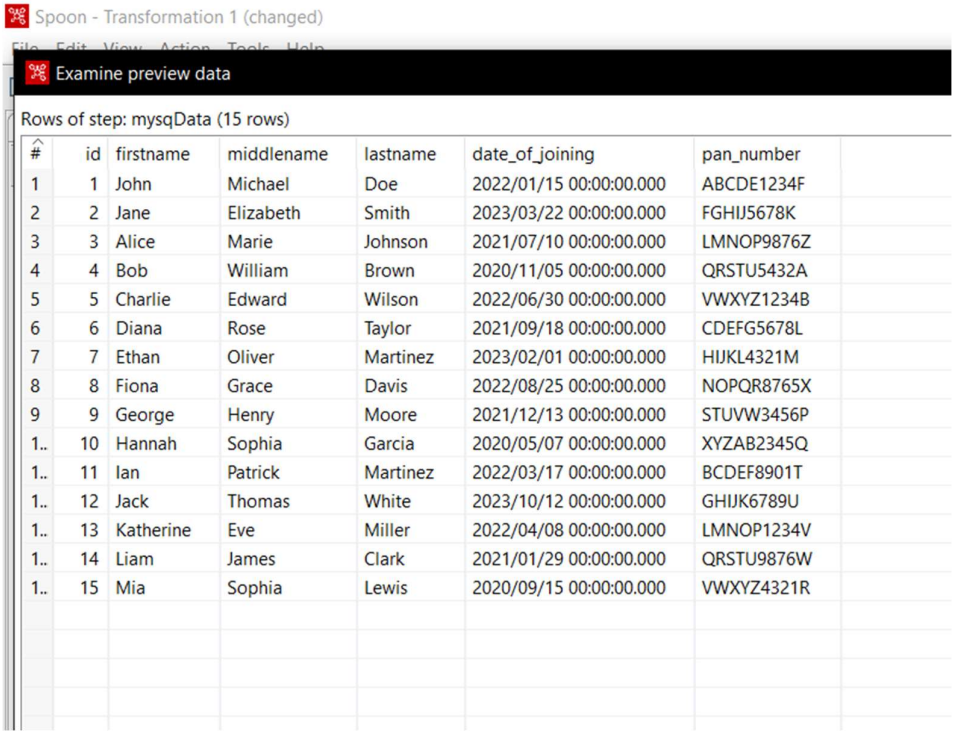
```
mysql> select * from employeeDetails;
+------+-----------------+---------------------------+-----------------+
| EID  | Full Name       | Job Title                 | Department      |
+------+-----------------+---------------------------+-----------------+
|    1 | Emily Davis     | Sr. Manger                | IT              |
|    4 | Theodore Dinh   | Technical Architect       | IT              |
|    2 | Luna Sanders    | Director                  | Finance         |
|    3 | Penelope Jordan | Computer Systems Manager  | IT              |
|    5 | Austin Vo       | Sr. Analyst               | Finance         |
|    7 | Joshua Gupta    | Account Representative    | Sales           |
|    6 | Ruby Barnes     | Manager                   | IT              |
|    8 | Luke Martin     | Analyst                   | Finance         |
|    9 | Easton Bailey   | Manager                   | Accounting      |
|   11 | Madeline Walker | Sr. Analyst               | Finance         |
|   10 | Savannah Ali    | Sr. Manger                | Human Resources |
|   13 | Camila Rogers   | Controls Engineer         | Engineering     |
|   12 | Eli Jones       | Manager                   | Human Resources |
|   15 | Everleigh Ng    | Sr. Manger                | Finance         |
|   14 | Robert Yang     | Sr. Analyst               | Accounting      |
+------+-----------------+---------------------------+-----------------+
15 rows in set (0.00 sec)
```

4. **Create a pentaho tranformation to get data from mysql table add serial no to the data using pentaho sequences and push data into a excel.**

Connection Established to the database:

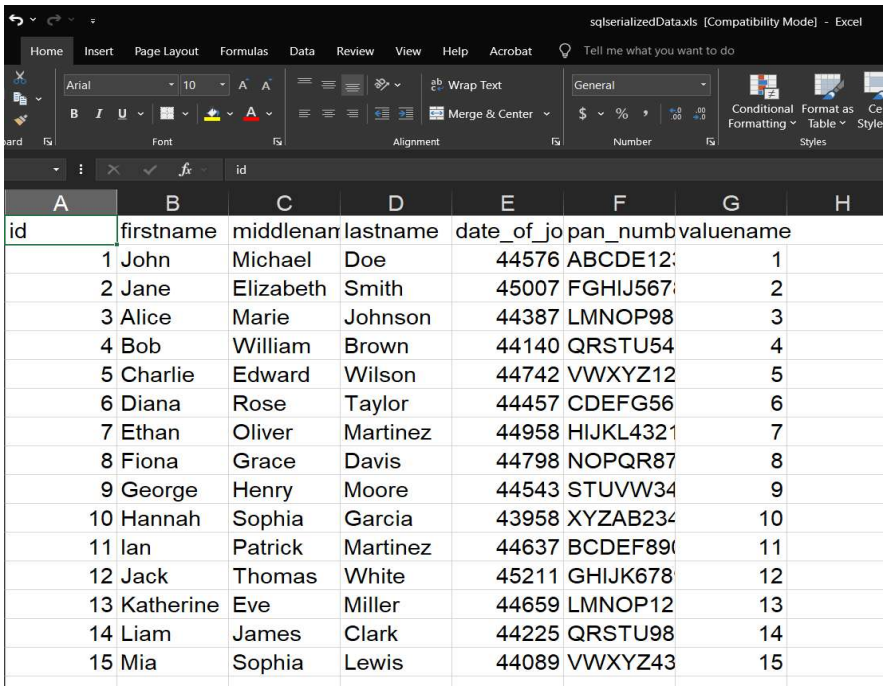


Data from database:



#	id	firstname	middlename	lastname	date_of_joining	pan_number
1	1	John	Michael	Doe	2022/01/15 00:00:00.000	ABCDE1234F
2	2	Jane	Elizabeth	Smith	2023/03/22 00:00:00.000	FGHIJ5678K
3	3	Alice	Marie	Johnson	2021/07/10 00:00:00.000	LMNOP9876Z
4	4	Bob	William	Brown	2020/11/05 00:00:00.000	QRSTU5432A
5	5	Charlie	Edward	Wilson	2022/06/30 00:00:00.000	VWXYZ1234B
6	6	Diana	Rose	Taylor	2021/09/18 00:00:00.000	CDEFG5678L
7	7	Ethan	Oliver	Martinez	2023/02/01 00:00:00.000	HIJKL4321M
8	8	Fiona	Grace	Davis	2022/08/25 00:00:00.000	NOPQR8765X
9	9	George	Henry	Moore	2021/12/13 00:00:00.000	STUVW3456P
1..	10	Hannah	Sophia	Garcia	2020/05/07 00:00:00.000	XYZAB2345Q
1..	11	Ian	Patrick	Martinez	2022/03/17 00:00:00.000	BCDEF8901T
1..	12	Jack	Thomas	White	2023/10/12 00:00:00.000	GHIJK6789U
1..	13	Katherine	Eve	Miller	2022/04/08 00:00:00.000	LMNOP1234V
1..	14	Liam	James	Clark	2021/01/29 00:00:00.000	QRSTU9876W
1..	15	Mia	Sophia	Lewis	2020/09/15 00:00:00.000	VWXYZ4321R

Flushed into the excel:



id	firstname	middlenam	lastname	date_of_jo	pan_numb	valuename
1	John	Michael	Doe	44576	ABCDE12	1
2	Jane	Elizabeth	Smith	45007	FGHIJ567	2
3	Alice	Marie	Johnson	44387	LMNOP98	3
4	Bob	William	Brown	44140	QRSTU54	4
5	Charlie	Edward	Wilson	44742	VWXYZ12	5
6	Diana	Rose	Taylor	44457	CDEFG56	6
7	Ethan	Oliver	Martinez	44958	HIJKL4321	7
8	Fiona	Grace	Davis	44798	NOPQR87	8
9	George	Henry	Moore	44543	STUVW34	9
10	Hannah	Sophia	Garcia	43958	XYZAB234	10
11	Ian	Patrick	Martinez	44637	BCDEF89	11
12	Jack	Thomas	White	45211	GHIJK678	12
13	Katherine	Eve	Miller	44659	LMNOP12	13
14	Liam	James	Clark	44225	QRSTU98	14
15	Mia	Sophia	Lewis	44089	VWXYZ43	15

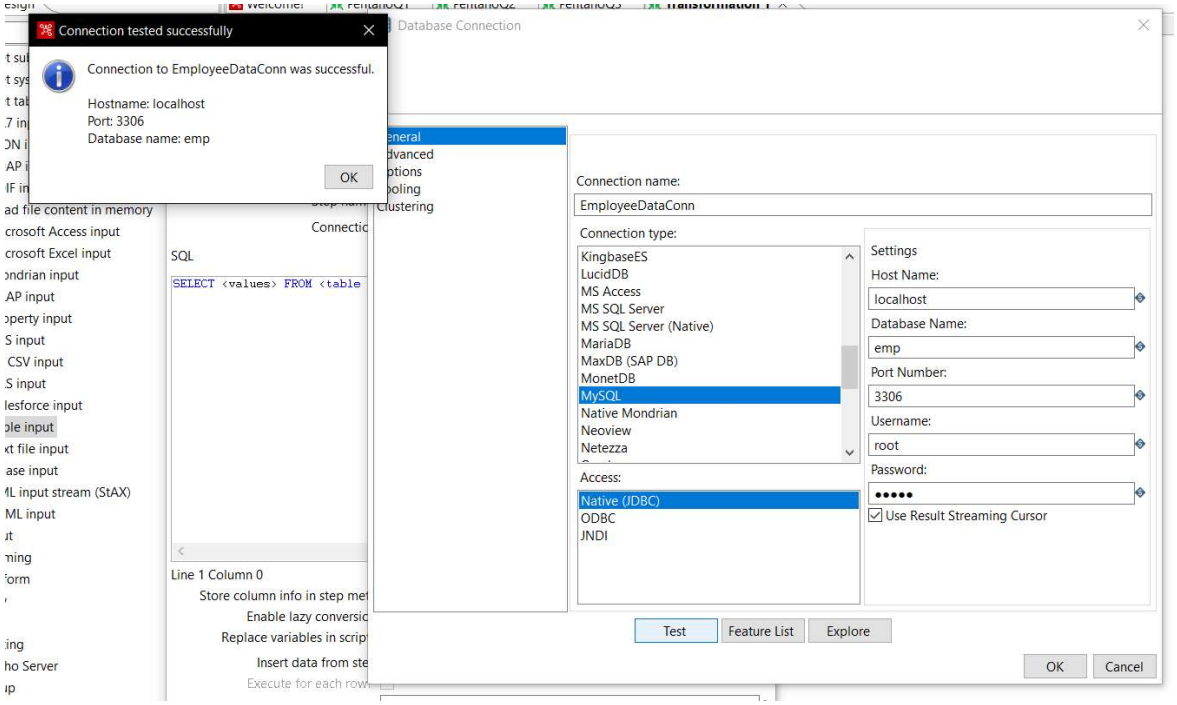
4. Create a pentaho tranformation to get data from mysql table named as employee have the following columns (emp_id,full name and job_id). Split the full name of the employee using pentaho split tranformation and store the data into a new table in mysql name as emp1 having the following columns (emp_id,fname,mname,lname and job_id).

SQL Queries:

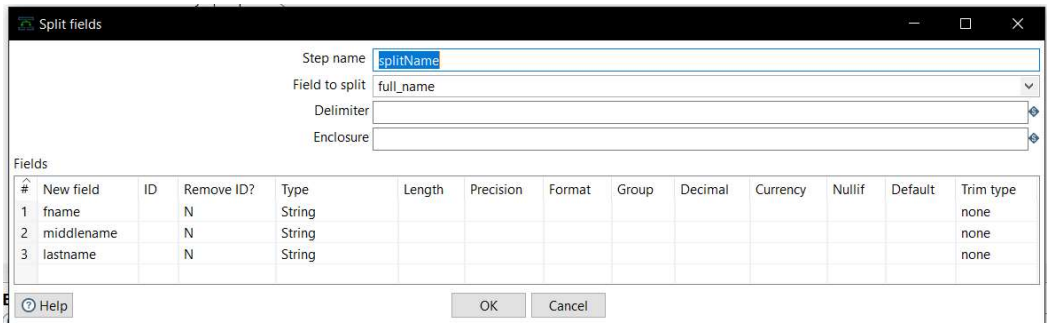
```
mysql> use emp;
Database changed
mysql> CREATE TABLE employee (
    ->     emp_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     full_name VARCHAR(150) NOT NULL,
    ->     job_id INT NOT NULL
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('John Adam Smith', 101);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Emma Beatrice Johnson', 102);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Michael Christopher Brown', 103);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Olivia Danielle Garcia', 104);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('William Edward Martinez', 105);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Sophia Frances Rodriguez', 106);
Query OK, 1 row affected (0.00 sec)
```

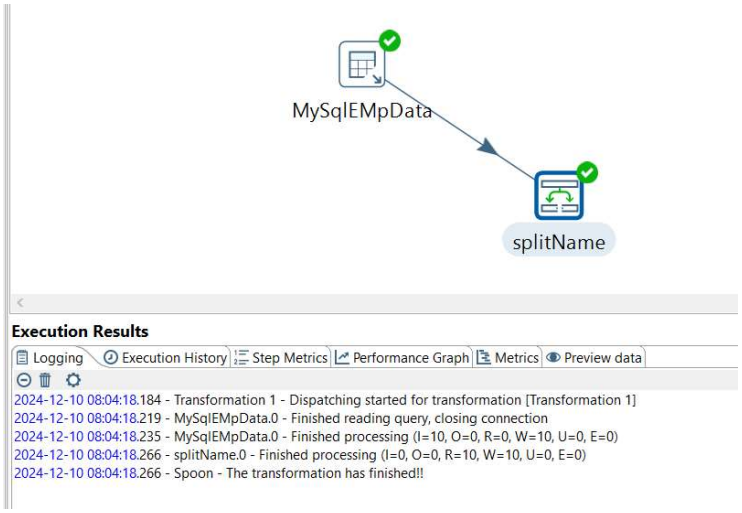
```
MySQL 9.1 Command Line Client - Unicode
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Sophia Frances Rodriguez', 106);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('James Gabriel Lopez', 107);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Isabella Hope Gonzalez', 108);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Alexander Isaac Wilson', 109);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee (full_name, job_id) VALUES ('Mia Juliet Anderson', 110);
Query OK, 1 row affected (0.00 sec)

mysql> select * from employee;
+--------+--------------------------+--------+
| emp_id | full_name                | job_id |
+--------+--------------------------+--------+
|      1 | John Adam Smith          |    101 |
|      2 | Emma Beatrice Johnson    |    102 |
|      3 | Michael Christopher Brown|    103 |
|      4 | Olivia Danielle Garcia   |    104 |
|      5 | William Edward Martinez  |    105 |
|      6 | Sophia Frances Rodriguez |    106 |
|      7 | James Gabriel Lopez      |    107 |
|      8 | Isabella Hope Gonzalez   |    108 |
|      9 | Alexander Isaac Wilson   |    109 |
|     10 | Mia Juliet Anderson      |    110 |
+--------+--------------------------+--------+
10 rows in set (0.00 sec)
```

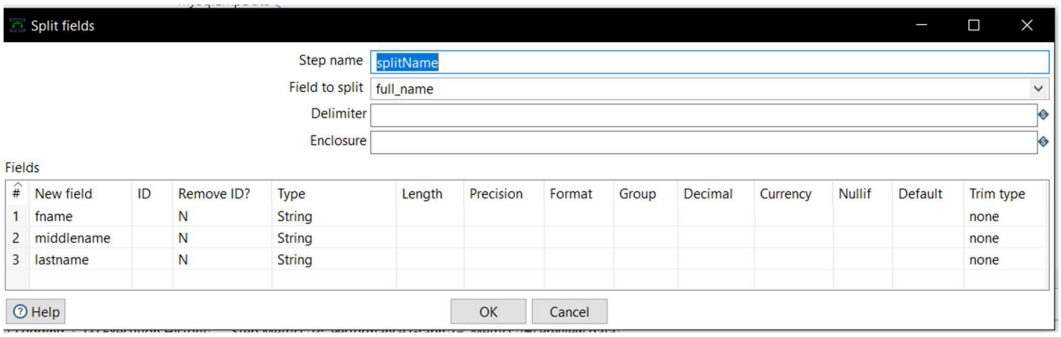
Database connection established:

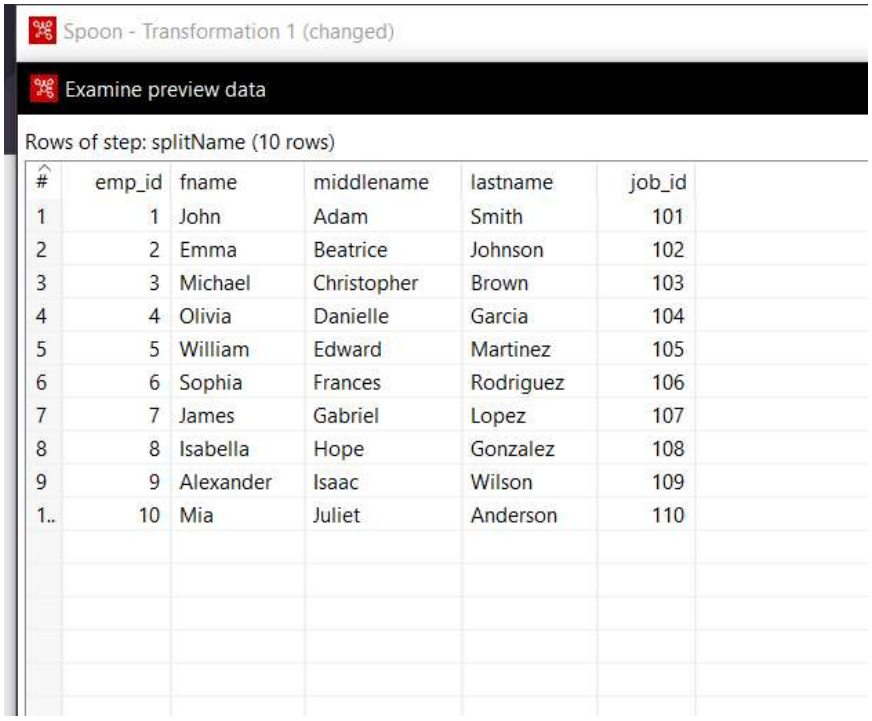


Splitting the fullname field into firstname, middlename, lastname .




MySqlEMpData
splitName
Execution Results
Logging
Execution History
Step Metrics
Performance Graph
Metrics
Preview data
2024-12-10 08:04:18.184 - Transformation 1 - Dispatching started for transformation [Transformation 1]
2024-12-10 08:04:18.219 - MySqlEMpData.0 - Finished reading query, closing connection
2024-12-10 08:04:18.235 - MySqlEMpData.0 - Finished processing (I=10, O=0, R=0, W=10, U=0, E=0)
2024-12-10 08:04:18.266 - splitName.0 - Finished processing (I=0, O=0, R=10, W=10, U=0, E=0)
2024-12-10 08:04:18.266 - Spoon - The transformation has finished!!


Split fields
Step name
splitName
Field to split
full_name
Delimiter
Enclosure
Fields
New field ID Remove ID? Type Length Precision Format Group Decimal Currency Nullif Default Trim type
1 fname N String none
2 middlename N String none
3 lastname N String none
Help
OK
Cancel


Spoon - Transformation 1 (changed)
Examine preview data
Rows of step: splitName (10 rows)
emp_id fname middlename lastname job_id
1 1 John Adam Smith 101
2 2 Emma Beatrice Johnson 102
3 3 Michael Christopher Brown 103
4 4 Olivia Danielle Garcia 104
5 5 William Edward Martinez 105
6 6 Sophia Frances Rodriguez 106
7 7 James Gabriel Lopez 107
8 8 Isabella Hope Gonzalez 108
9 9 Alexander Isaac Wilson 109
1.. 10 Mia Juliet Anderson 110

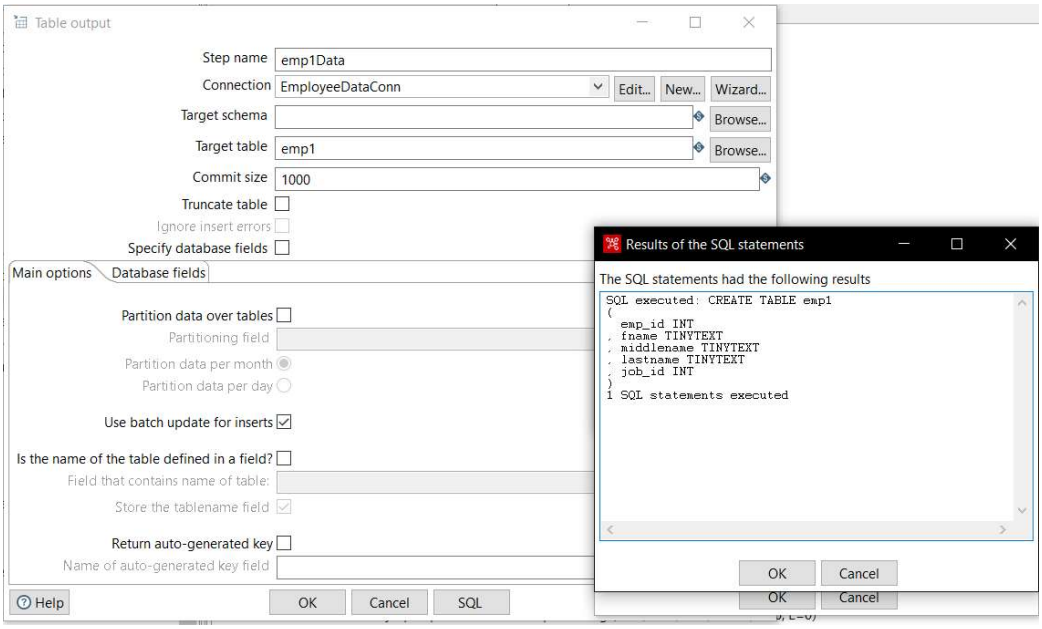


Table output
Step name emp1Data
Connection EmployeeDataConn
Edit...
New...
Wizard...
Target schema
Browse...
Target table emp1
Browse...
Commit size 1000
Truncate table
Ignore insert errors
Specify database fields
Main options
Database fields
Partition data over tables
Partitioning field
Partition data per month
Partition data per day
Use batch update for inserts
Is the name of the table defined in a field?
Field that contains name of table:
Store the tablename field
Return auto-generated key
Name of auto-generated key field
Help
OK
Cancel
SQL
Results of the SQL statements
The SQL statements had the following results
SQL executed: CREATE TABLE emp1
(
emp_id INT
, fname TINYTEXT
, middlename TINYTEXT
, lastname TINYTEXT
, job_id INT
)
1 SQL statements executed
OK
Cancel

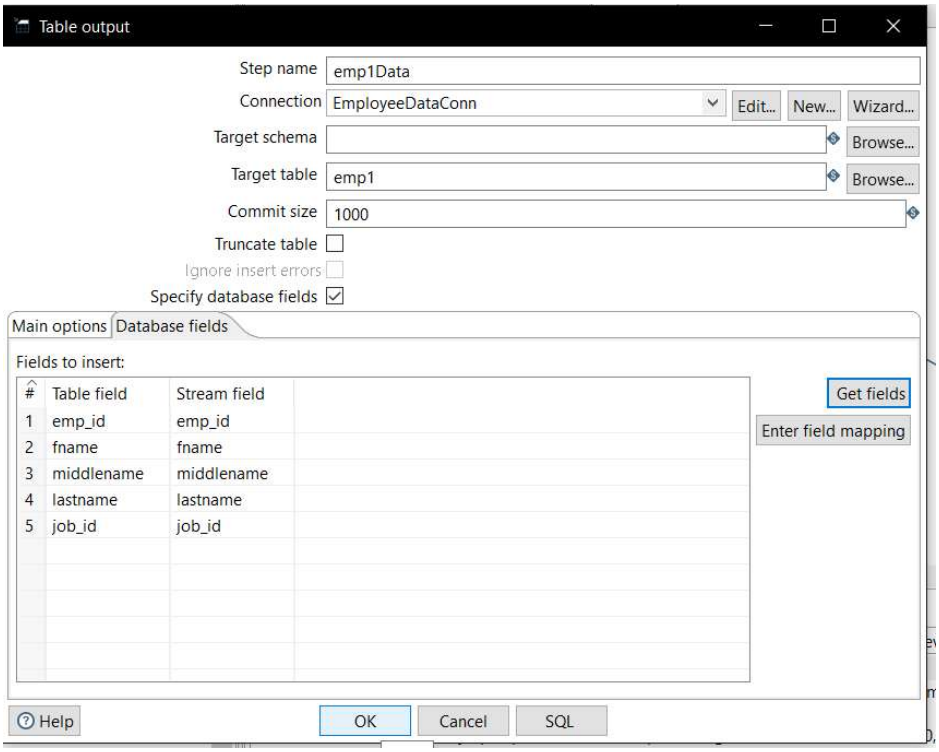
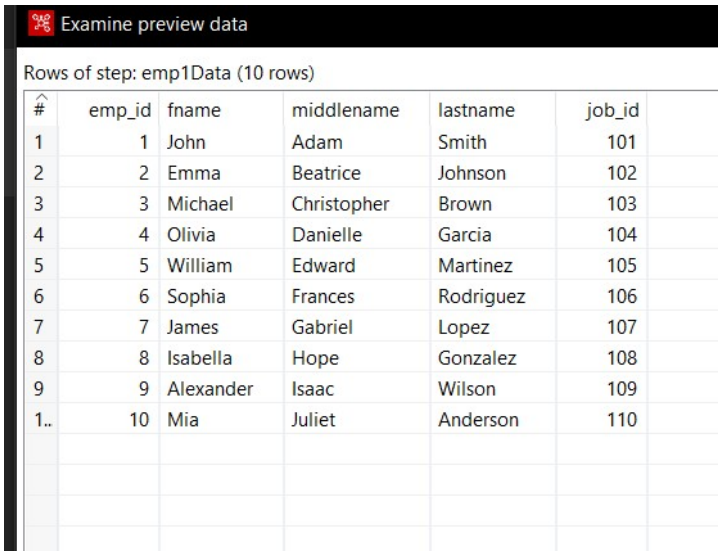


Table output
Step name emp1Data
Connection EmployeeDataConn
Edit...
New...
Wizard...
Target schema
Browse...
Target table emp1
Browse...
Commit size 1000
Truncate table
Ignore insert errors
Specify database fields
Main options
Database fields
Fields to insert:
Table field Stream field
1 emp_id emp_id
2 fname fname
3 middlename middlename
4 lastname lastname
5 job_id job_id
Get fields
Enter field mapping
Help
OK
Cancel
SQL


Examine preview data

Rows of step: emp1Data (10 rows)

#	emp_id	fname	middlename	lastname	job_id
1	1	John	Adam	Smith	101
2	2	Emma	Beatrice	Johnson	102
3	3	Michael	Christopher	Brown	103
4	4	Olivia	Danielle	Garcia	104
5	5	William	Edward	Martinez	105
6	6	Sophia	Frances	Rodriguez	106
7	7	James	Gabriel	Lopez	107
8	8	Isabella	Hope	Gonzalez	108
9	9	Alexander	Isaac	Wilson	109
1..	10	Mia	Juliet	Anderson	110

```
mysql> select * from emp1;
Empty set (0.00 sec)

mysql> drop table emp1;
Query OK, 0 rows affected (0.05 sec)

mysql> select * from emp1;
ERROR 1146 (42S02): Table 'emp.emp1' doesn't exist
mysql> select * from emp1;
+--------+-----------+-------------+-----------+--------+
| emp_id | fname     | middlename  | lastname  | job_id |
+--------+-----------+-------------+-----------+--------+
|      1 | John      | Adam        | Smith     |    101 |
|      2 | Emma      | Beatrice    | Johnson   |    102 |
|      3 | Michael   | Christopher | Brown     |    103 |
|      4 | Olivia    | Danielle    | Garcia    |    104 |
|      5 | William   | Edward      | Martinez  |    105 |
|      6 | Sophia    | Frances     | Rodriguez |    106 |
|      7 | James     | Gabriel     | Lopez     |    107 |
|      8 | Isabella  | Hope        | Gonzalez  |    108 |
|      9 | Alexander | Isaac       | Wilson    |    109 |
|     10 | Mia       | Juliet      | Anderson  |    110 |
+--------+-----------+-------------+-----------+--------+
10 rows in set (0.00 sec)
```

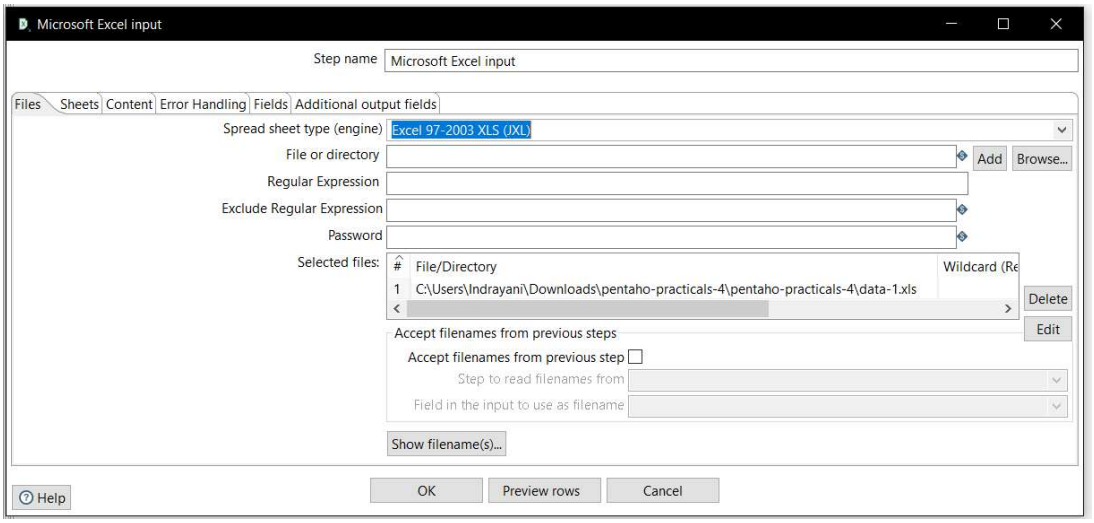
5. create a two excel files with following columns

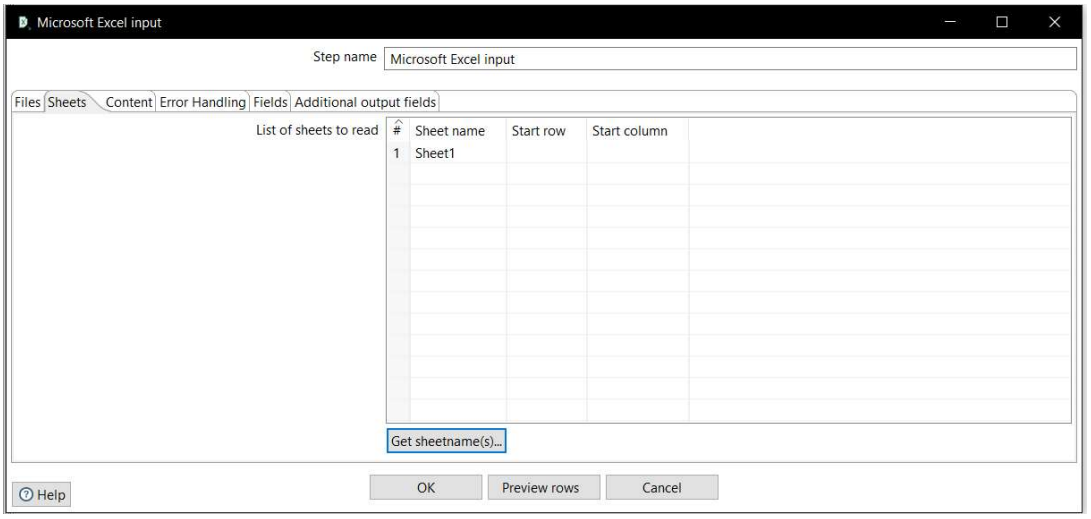
(a) data 1: EID, Full Name, Job, Title, Department.

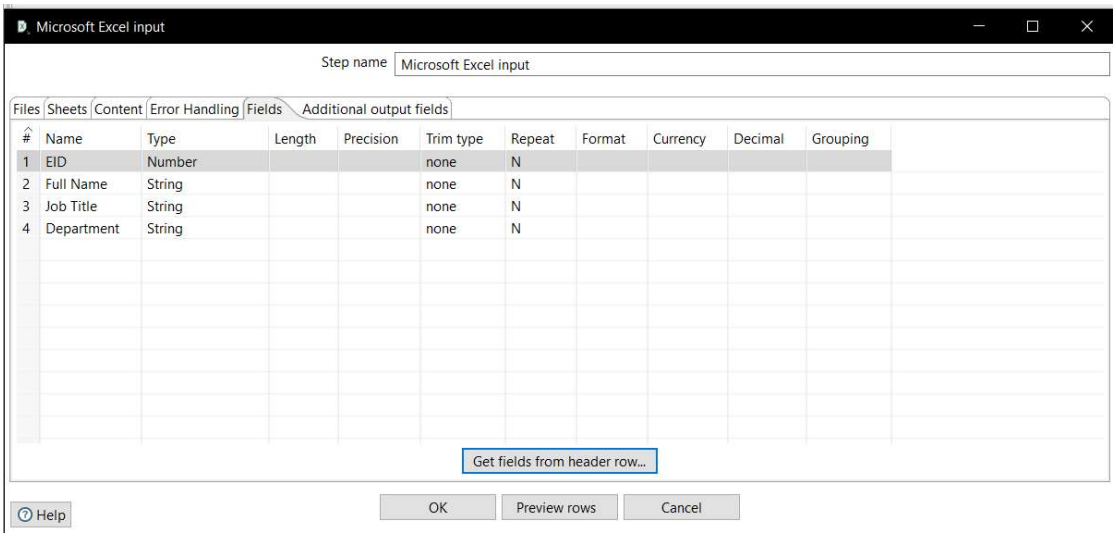
data 2:EID,Business,Unit, Gender, Ethnicity,Age.

EID in data 1 points to EID in data 2 . Make sure to have atleast 10 to 15 rows in the file. sort the data using sort tranformation of pentaho.

(b) Merge join the sorted in pentaho and remove repeated rows and save the data in mysql table.


Microsoft Excel input
Step name
Microsoft Excel input
Files
Sheets
Content
Error Handling
Fields
Additional output fields
Spread sheet type (engine)
Excel 97-2003 XLS (JXL)
File or directory
Add
Browse...
Regular Expression
Exclude Regular Expression
Password
Selected files:
File/Directory
Wildcard (Re
1 C:\Users\Indrayani\Downloads\pentaho-practicals-4\pentaho-practicals-4\data-1.xls
Delete
Edit
Accept filenames from previous steps
Accept filenames from previous step
Step to read filenames from
Field in the input to use as filename
Show filename(s)...
Help
OK
Preview rows
Cancel


Microsoft Excel input
Step name
Microsoft Excel input
Files
Sheets
Content
Error Handling
Fields
Additional output fields
List of sheets to read
Sheet name Start row Start column
1 Sheet1
Get sheetname(s)...
Help
OK
Preview rows
Cancel


Microsoft Excel input
Step name
Microsoft Excel input
Files
Sheets
Content
Error Handling
Fields
Additional output fields
Name Type Length Precision Trim type Repeat Format Currency Decimal Grouping
1 EID Number none N
2 Full Name String none N
3 Job Title String none N
4 Department String none N
Get fields from header row...
Help
OK
Preview rows
Cancel

Spoon - Transformation 2 (changed)

Examine preview data

Rows of step: Microsoft Excel input (15 rows)

#	EID	Full Name	Job Title	Department
1	1.0	Emily Davis	Sr. Manger	IT
2	4.0	Theodore Dinh	Technical Architect	IT
3	2.0	Luna Sanders	Director	Finance
4	3.0	Penelope Jordan	Computer Systems Manager	IT
5	5.0	Austin Vo	Sr. Analyst	Finance
6	7.0	Joshua Gupta	Account Representative	Sales
7	6.0	Ruby Barnes	Manager	IT
8	8.0	Luke Martin	Analyst	Finance
9	9.0	Easton Bailey	Manager	Accounting
1..	11.0	Madeline Walker	Sr. Analyst	Finance
1..	10.0	Savannah Ali	Sr. Manger	Human Resources
1..	13.0	Camila Rogers	Controls Engineer	Engineering
1..	12.0	Eli Jones	Manager	Human Resources
1..	15.0	Everleigh Ng	Sr. Manger	Finance
1..	14.0	Robert Yang	Sr. Analyst	Accounting

Sort rows

Step name: Sort rows

Sort directory: %%java.io.tmpdir%% Browse...

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files?

Only pass unique rows? (verifies keys only)

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	EID	Y				
2	Full Name	N				
3	Job Title	N				
4	Department	N				

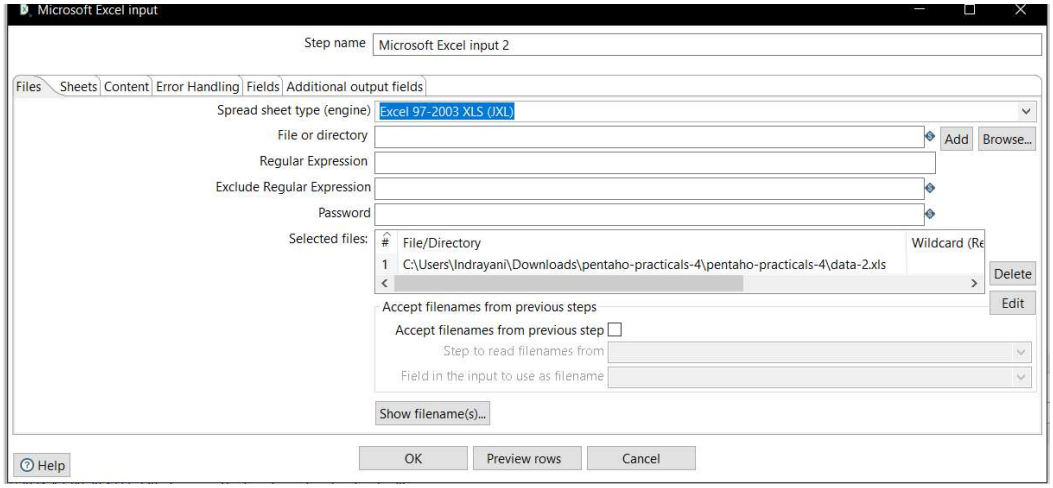
Help OK Cancel Get Fields

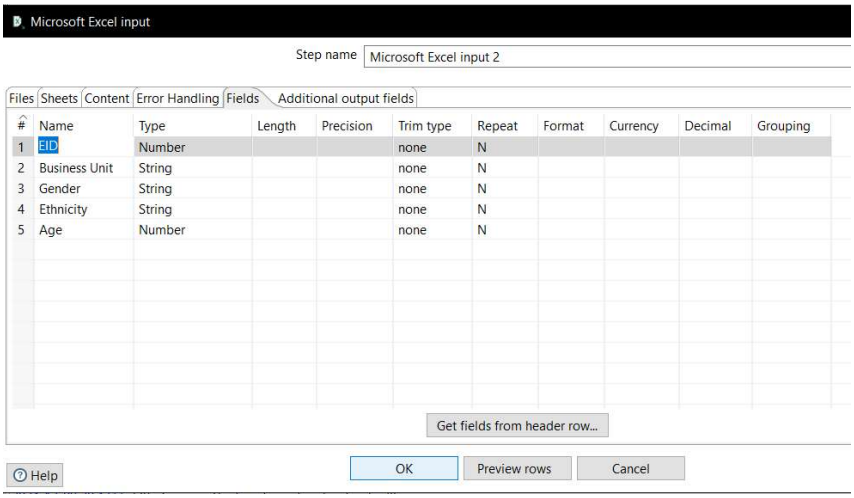
Spoon - Transformation 2 (changed)

Examine preview data

Rows of step: Sort rows (15 rows)

#	EID	Full Name	Job Title	Department
1	1.0	Emily Davis	Sr. Manger	IT
2	2.0	Luna Sanders	Director	Finance
3	3.0	Penelope Jordan	Computer Systems Manager	IT
4	4.0	Theodore Dinh	Technical Architect	IT
5	5.0	Austin Vo	Sr. Analyst	Finance
6	6.0	Ruby Barnes	Manager	IT
7	7.0	Joshua Gupta	Account Representative	Sales
8	8.0	Luke Martin	Analyst	Finance
9	9.0	Easton Bailey	Manager	Accounting
1..	10.0	Savannah Ali	Sr. Manger	Human Resources
1..	11.0	Madeline Walker	Sr. Analyst	Finance
1..	12.0	Eli Jones	Manager	Human Resources
1..	13.0	Camila Rogers	Controls Engineer	Engineering
1..	14.0	Robert Yang	Sr. Analyst	Accounting
1..	15.0	Everleigh Ng	Sr. Manger	Finance





Spoon - Transformation 2 (changed)

Examine preview data

Rows of step: Microsoft Excel input 2 (15 rows)

#	EID	Business Unit	Gender	Ethnicity	Age
1	1.0	Research & Development	Female	Black	55.0
2	4.0	Manufacturing	Male	Asian	59.0
3	2.0	Speciality Products	Female	Caucasian	50.0
4	3.0	Manufacturing	Female	Caucasian	26.0
5	5.0	Manufacturing	Male	Asian	55.0
6	7.0	Corporate	Male	Asian	57.0
7	6.0	Corporate	Female	Caucasian	27.0
8	8.0	Manufacturing	Male	Black	25.0
9	9.0	Manufacturing	Male	Caucasian	29.0
1..	11.0	Speciality Products	Female	Caucasian	34.0
1..	10.0	Manufacturing	Female	Asian	36.0
1..	13.0	Speciality Products	Female	Caucasian	27.0
1..	12.0	Manufacturing	Male	Caucasian	59.0
1..	15.0	Research & Development	Female	Asian	51.0
1..	14.0	Speciality Products	Male	Asian	31.0

Sort rows

Step name: Sort rows 2

Sort directory: %%java.io.tmpdir%% Browse...

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files?

Only pass unique rows? (verifies keys only)

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	EID	Y				
2	Business Unit	N				
3	Gender	N				
4	Ethnicity	N				
5	Age	N				

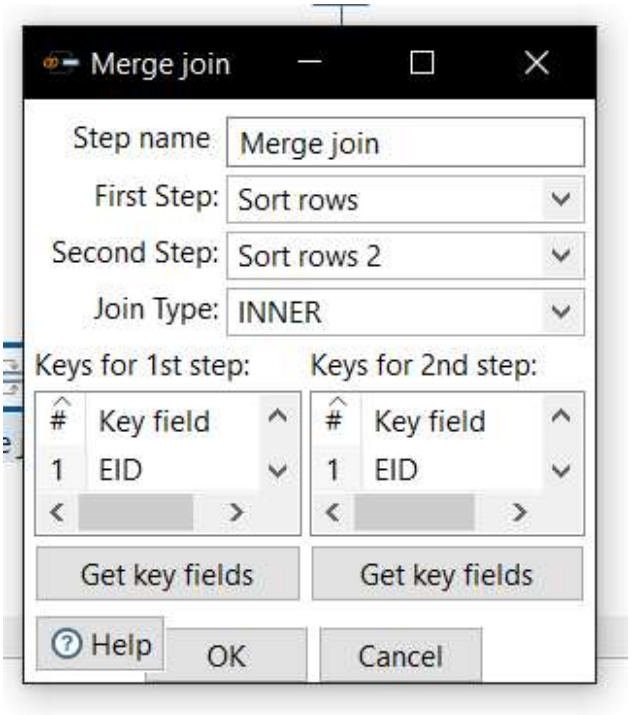
Help OK Cancel Get Fields

Spoon - Transformation 2 (changed)

Examine preview data

Rows of step: Sort rows 2 (15 rows)

#	EID	Business Unit	Gender	Ethnicity	Age
1	1.0	Research & Development	Female	Black	55.0
2	2.0	Speciality Products	Female	Caucasian	50.0
3	3.0	Manufacturing	Female	Caucasian	26.0
4	4.0	Manufacturing	Male	Asian	59.0
5	5.0	Manufacturing	Male	Asian	55.0
6	6.0	Corporate	Female	Caucasian	27.0
7	7.0	Corporate	Male	Asian	57.0
8	8.0	Manufacturing	Male	Black	25.0
9	9.0	Manufacturing	Male	Caucasian	29.0
1..	10.0	Manufacturing	Female	Asian	36.0
1..	11.0	Speciality Products	Female	Caucasian	34.0
1..	12.0	Manufacturing	Male	Caucasian	59.0
1..	13.0	Speciality Products	Female	Caucasian	27.0
1..	14.0	Speciality Products	Male	Asian	31.0
1..	15.0	Research & Development	Female	Asian	51.0



Merge Join:

Examine preview data

Rows of step: Merge join (15 rows)

#	EID	Full Name	Job Title	Department	EID_1	Business Unit	Gender	Ethnicity	Age
1	1.0	Emily Davis	Sr. Manger	IT	1.0	Research & Development	Female	Black	55.0
2	2.0	Luna Sanders	Director	Finance	2.0	Speciality Products	Female	Caucasian	50.0
3	3.0	Penelope Jordan	Computer Systems Manager	IT	3.0	Manufacturing	Female	Caucasian	26.0
4	4.0	Theodore Dinh	Technical Architect	IT	4.0	Manufacturing	Male	Asian	59.0
5	5.0	Austin Vo	Sr. Analyst	Finance	5.0	Manufacturing	Male	Asian	55.0
6	6.0	Ruby Barnes	Manager	IT	6.0	Corporate	Female	Caucasian	27.0
7	7.0	Joshua Gupta	Account Representative	Sales	7.0	Corporate	Male	Asian	57.0
8	8.0	Luke Martin	Analyst	Finance	8.0	Manufacturing	Male	Black	25.0
9	9.0	Easton Bailey	Manager	Accounting	9.0	Manufacturing	Male	Caucasian	29.0
1..	10.0	Savannah Ali	Sr. Manger	Human Resources	10.0	Manufacturing	Female	Asian	36.0
1..	11.0	Madeline Walker	Sr. Analyst	Finance	11.0	Speciality Products	Female	Caucasian	34.0
1..	12.0	Eli Jones	Manager	Human Resources	12.0	Manufacturing	Male	Caucasian	59.0
1..	13.0	Camila Rogers	Controls Engineer	Engineering	13.0	Speciality Products	Female	Caucasian	27.0
1..	14.0	Robert Yang	Sr. Analyst	Accounting	14.0	Speciality Products	Male	Asian	31.0
1..	15.0	Everleigh Ng	Sr. Manger	Finance	15.0	Research & Development	Female	Asian	51.0

Unique rows:

Spoon - Transformation 2 (changed)

Examine preview data

Rows of step: Unique rows (15 rows)

#	EID	Full Name	Job Title	Department	EID_1	Business Unit	Gender	Ethnicity	Age
1	1.0	Emily Davis	Sr. Manger	IT	1.0	Research & Development	Female	Black	55.0
2	2.0	Luna Sanders	Director	Finance	2.0	Speciality Products	Female	Caucasian	50.0
3	3.0	Penelope Jordan	Computer Systems Manager	IT	3.0	Manufacturing	Female	Caucasian	26.0
4	4.0	Theodore Dinh	Technical Architect	IT	4.0	Manufacturing	Male	Asian	59.0
5	5.0	Austin Vo	Sr. Analyst	Finance	5.0	Manufacturing	Male	Asian	55.0
6	6.0	Ruby Barnes	Manager	IT	6.0	Corporate	Female	Caucasian	27.0
7	7.0	Joshua Gupta	Account Representative	Sales	7.0	Corporate	Male	Asian	57.0
8	8.0	Luke Martin	Analyst	Finance	8.0	Manufacturing	Male	Black	25.0
9	9.0	Easton Bailey	Manager	Accounting	9.0	Manufacturing	Male	Caucasian	29.0
1..	10.0	Savannah Ali	Sr. Manger	Human Resources	10.0	Manufacturing	Female	Asian	36.0
1..	11.0	Madeline Walker	Sr. Analyst	Finance	11.0	Speciality Products	Female	Caucasian	34.0
1..	12.0	Eli Jones	Manager	Human Resources	12.0	Manufacturing	Male	Caucasian	59.0
1..	13.0	Camila Rogers	Controls Engineer	Engineering	13.0	Speciality Products	Female	Caucasian	27.0
1..	14.0	Robert Yang	Sr. Analyst	Accounting	14.0	Speciality Products	Male	Asian	31.0
1..	15.0	Everleigh Ng	Sr. Manger	Finance	15.0	Research & Development	Female	Asian	51.0

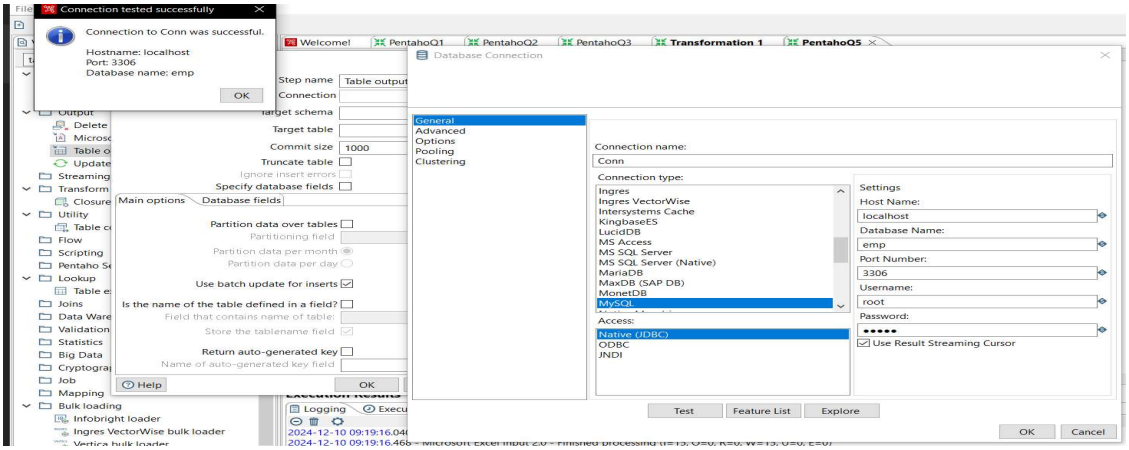
By using select values, we removed the EID_1 which was getting repeated.

Examine preview data

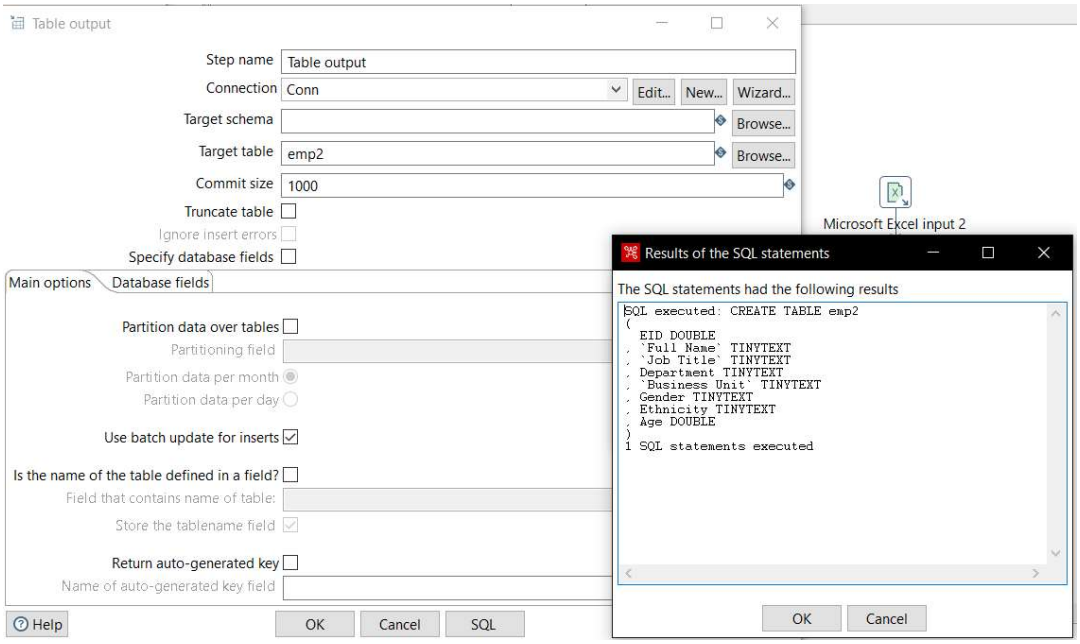
Rows of step: Select values (15 rows)

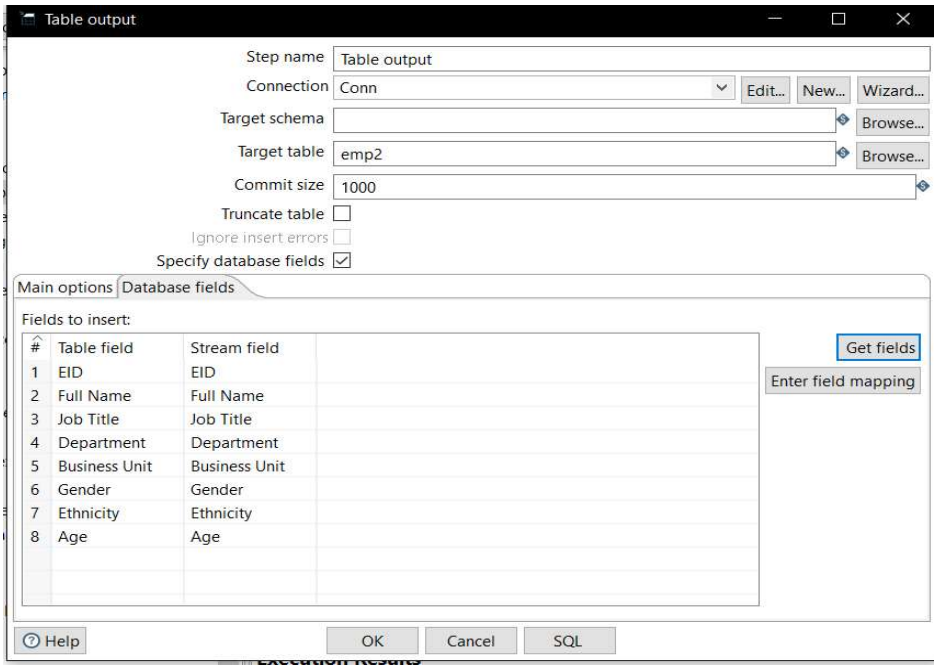
#	EID	Full Name	Job Title	Department	Business Unit	Gender	Ethnicity	Age
1	1.0	Emily Davis	Sr. Manger	IT	Research & Development	Female	Black	55.0
2	2.0	Luna Sanders	Director	Finance	Speciality Products	Female	Caucasian	50.0
3	3.0	Penelope Jordan	Computer Systems Manager	IT	Manufacturing	Female	Caucasian	26.0
4	4.0	Theodore Dinh	Technical Architect	IT	Manufacturing	Male	Asian	59.0
5	5.0	Austin Vo	Sr. Analyst	Finance	Manufacturing	Male	Asian	55.0
6	6.0	Ruby Barnes	Manager	IT	Corporate	Female	Caucasian	27.0
7	7.0	Joshua Gupta	Account Representative	Sales	Corporate	Male	Asian	57.0
8	8.0	Luke Martin	Analyst	Finance	Manufacturing	Male	Black	25.0
9	9.0	Easton Bailey	Manager	Accounting	Manufacturing	Male	Caucasian	29.0
1..	10.0	Savannah Ali	Sr. Manger	Human Resources	Manufacturing	Female	Asian	36.0
1..	11.0	Madeline Walker	Sr. Analyst	Finance	Speciality Products	Female	Caucasian	34.0
1..	12.0	Eli Jones	Manager	Human Resources	Manufacturing	Male	Caucasian	59.0
1..	13.0	Camila Rogers	Controls Engineer	Engineering	Speciality Products	Female	Caucasian	27.0
1..	14.0	Robert Yang	Sr. Analyst	Accounting	Speciality Products	Male	Asian	31.0
1..	15.0	Everleigh Ng	Sr. Manger	Finance	Research & Development	Female	Asian	51.0

lushing the data into the database:



Created table emp2 to store the data:





Data flushed into the database:

#	EID	Full Name	Job Title	Department	Business Unit	Gender	Ethnicity	Age
1	1.0	Emily Davis	Sr. Manger	IT	Research & Development	Female	Black	55.0
2	2.0	Luna Sanders	Director	Finance	Speciality Products	Female	Caucasian	50.0
3	3.0	Penelope Jordan	Computer Systems Manager	IT	Manufacturing	Female	Caucasian	26.0
4	4.0	Theodore Dinh	Technical Architect	IT	Manufacturing	Male	Asian	59.0
5	5.0	Austin Vo	Sr. Analyst	Finance	Manufacturing	Male	Asian	55.0
6	6.0	Ruby Barnes	Manager	IT	Corporate	Female	Caucasian	27.0
7	7.0	Joshua Gupta	Account Representative	Sales	Corporate	Male	Asian	57.0
8	8.0	Luke Martin	Analyst	Finance	Manufacturing	Male	Black	25.0
9	9.0	Easton Bailey	Manager	Accounting	Manufacturing	Male	Caucasian	29.0
1..	10.0	Savannah Ali	Sr. Manger	Human Resources	Manufacturing	Female	Asian	36.0
1..	11.0	Madeline Walker	Sr. Analyst	Finance	Speciality Products	Female	Caucasian	34.0
1..	12.0	Eli Jones	Manager	Human Resources	Manufacturing	Male	Caucasian	59.0
1..	13.0	Camila Rogers	Controls Engineer	Engineering	Speciality Products	Female	Caucasian	27.0
1..	14.0	Robert Yang	Sr. Analyst	Accounting	Speciality Products	Male	Asian	31.0
1..	15.0	Everleigh Ng	Sr. Manger	Finance	Research & Development	Female	Asian	51.0

```
mysql> select * from emp2;
+------+-----------------+--------------------------+-----------------+-------------------------+--------+-----------+------+
| EID  | Full Name       | Job Title                | Department      | Business Unit           | Gender | Ethnicity | Age  |
+------+-----------------+--------------------------+-----------------+-------------------------+--------+-----------+------+
|    1 | Emily Davis     | Sr. Manger               | IT              | Research & Development  | Female | Black     |   55 |
|    2 | Luna Sanders    | Director                 | Finance         | Speciality Products     | Female | Caucasian |   50 |
|    3 | Penelope Jordan | Computer Systems Manager | IT              | Manufacturing           | Female | Caucasian |   26 |
|    4 | Theodore Dinh   | Technical Architect      | IT              | Manufacturing           | Male   | Asian     |   59 |
|    5 | Austin Vo       | Sr. Analyst              | Finance         | Manufacturing           | Male   | Asian     |   55 |
|    6 | Ruby Barnes     | Manager                  | IT              | Corporate               | Female | Caucasian |   27 |
|    7 | Joshua Gupta    | Account Representative   | Sales           | Corporate               | Male   | Asian     |   57 |
|    8 | Luke Martin     | Analyst                  | Finance         | Manufacturing           | Male   | Black     |   25 |
|    9 | Easton Bailey   | Manager                  | Accounting      | Manufacturing           | Male   | Caucasian |   29 |
|   10 | Savannah Ali    | Sr. Manger               | Human Resources | Manufacturing           | Female | Asian     |   36 |
|   11 | Madeline Walker | Sr. Analyst              | Finance         | Speciality Products     | Female | Caucasian |   34 |
|   12 | Eli Jones       | Manager                  | Human Resources | Manufacturing           | Male   | Caucasian |   59 |
|   13 | Camila Rogers   | Controls Engineer        | Engineering     | Speciality Products     | Female | Caucasian |   27 |
|   14 | Robert Yang     | Sr. Analyst              | Accounting      | Speciality Products     | Male   | Asian     |   31 |
|   15 | Everleigh Ng    | Sr. Manger               | Finance         | Research & Development  | Female | Asian     |   51 |
+------+-----------------+--------------------------+-----------------+-------------------------+--------+-----------+------+
15 rows in set (0.00 sec)
```

PRACTICAL 5

Aim: Introduction to basics of R programming

Writeup:

Introduction to Basics of R Programming: R is a versatile programming language widely used for statistical computing and data analysis. Below is a concise guide to its basic functionalities:

1. Installing and Loading Packages

- Installing Packages: Use install.packages("package_name") to install packages.
- Loading Packages: Use library(package_name) to load installed packages for use.

2. Data Types and Variables

- Data Types: Common types include numeric, character, logical, and complex.
- Checking Variable Type: Use class(variable) or typeof(variable).
- Printing Variables: Use print(variable) or simply type the variable name.

3. Objects in R

- Vectors: Use c() to create a vector (e.g., c(1, 2, 3)).
- Matrices: Use matrix() to create a matrix.
- Lists: Use list() to create a list.
- Factors: Use factor() to create a categorical variable.
- Data Frames: Use data.frame() to create a data frame.
- Tables: Use table() to create frequency tables.

4. Combining Data

- Column Bind: Use cbind() to combine objects column-wise.
- Row Bind: Use rbind() to combine objects row-wise.

5. File Operations

- Reading Data:
  - CSV Files: Use read.csv("file.csv").
  - Excel Files: Use readxl::read_excel("file.xlsx") (requires readxl package).
- Writing Data:
  - Use write.csv(data, "file.csv") to save data to a CSV file.

6. Working Directory

- Set Directory: Use setwd("path/to/directory").
- Get Current Directory: Use getwd().

7. Managing Data

- Listing Data: Use data() to list available datasets.
- Removing Objects: Use rm(object_name) to remove an object.
- Attaching Data: Use attach(data_frame) to access columns directly.
- Detaching Data: Use detach(data_frame) to remove direct access.

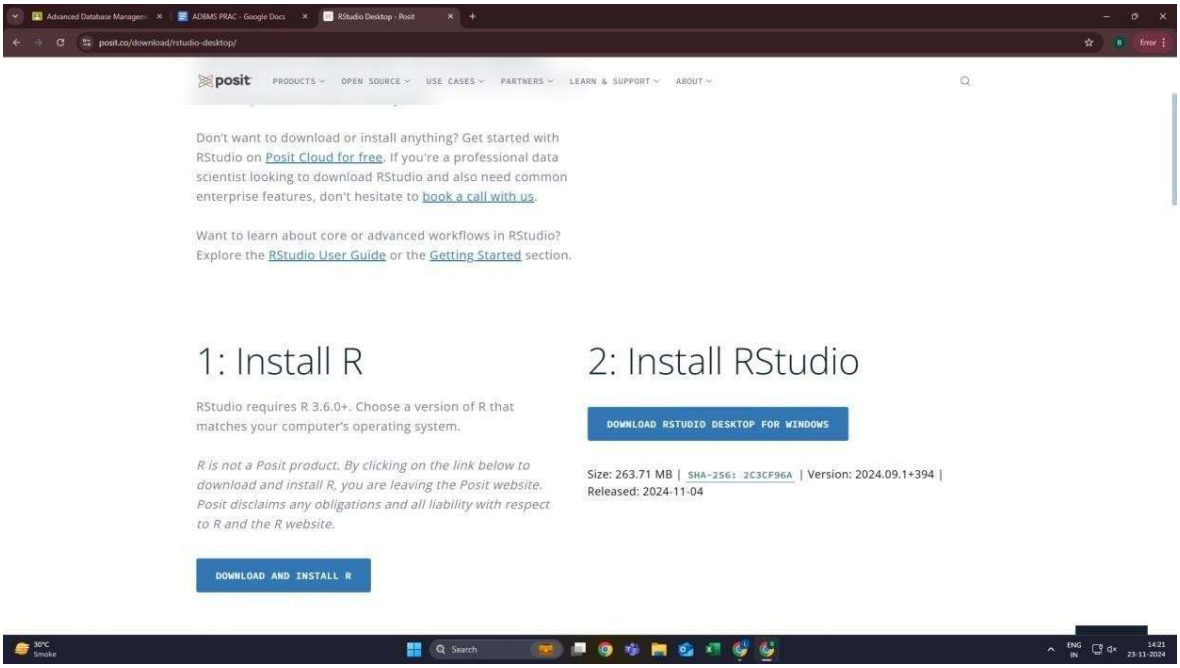
8. Reading Data from Console

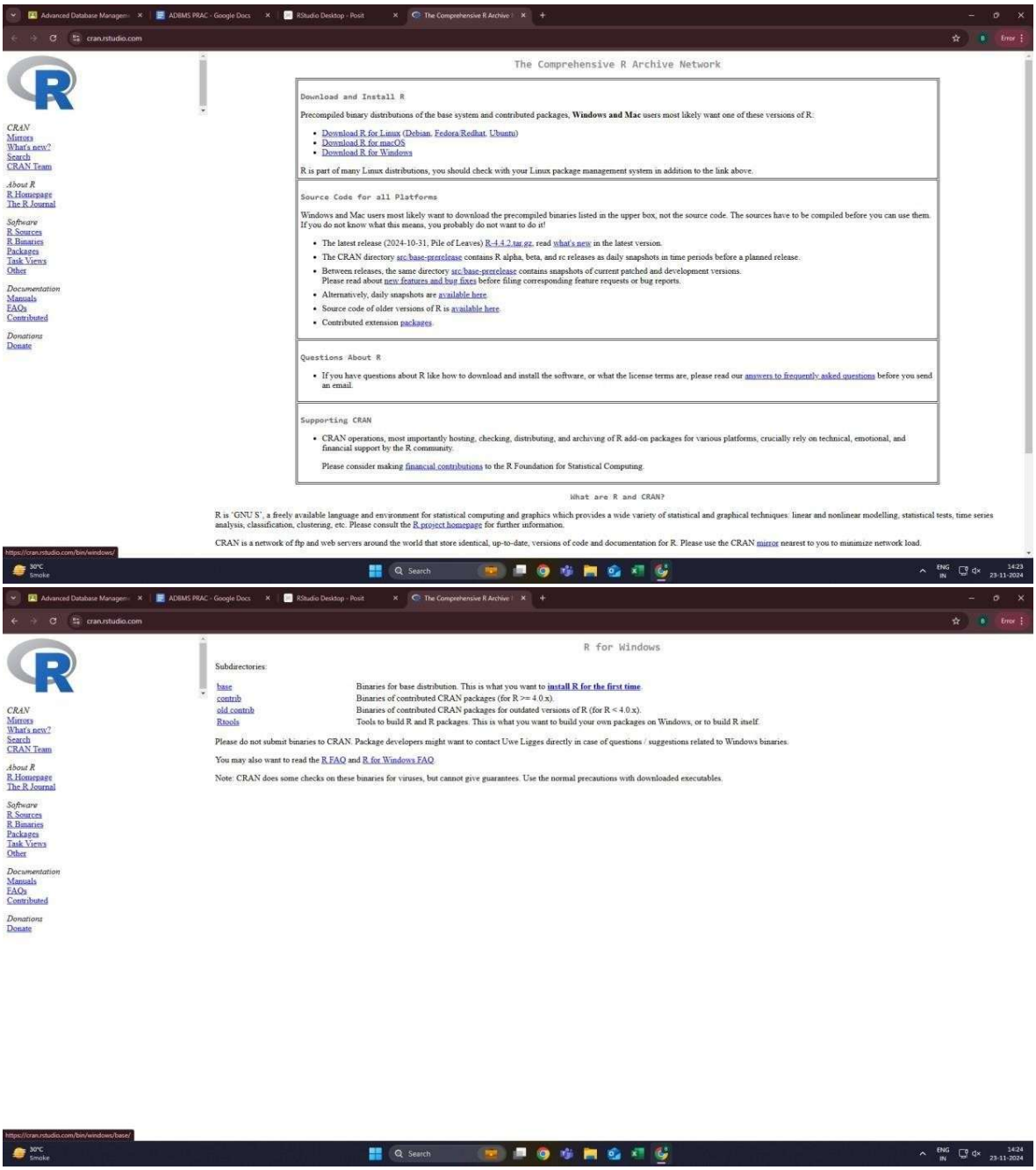
Use readline(prompt = "Enter value: ") for interactive inputs or scan() for bulk input.

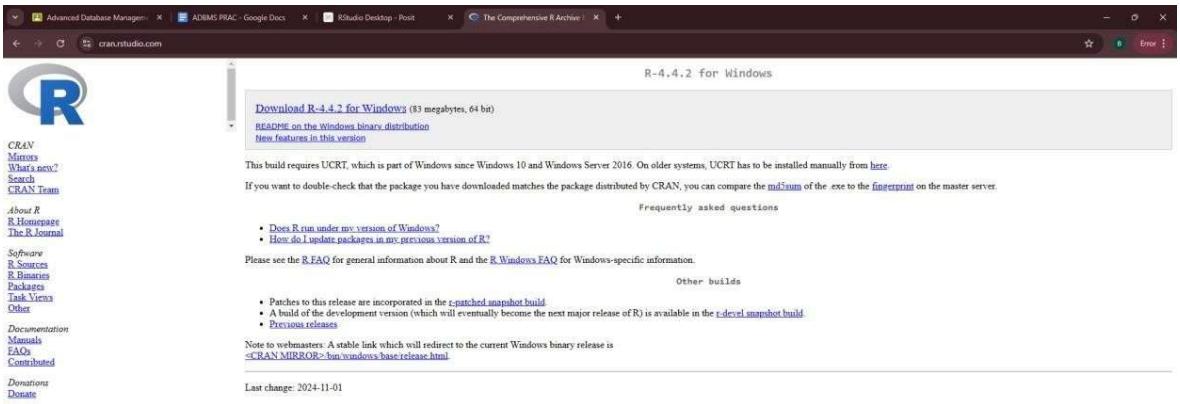
9. Summary

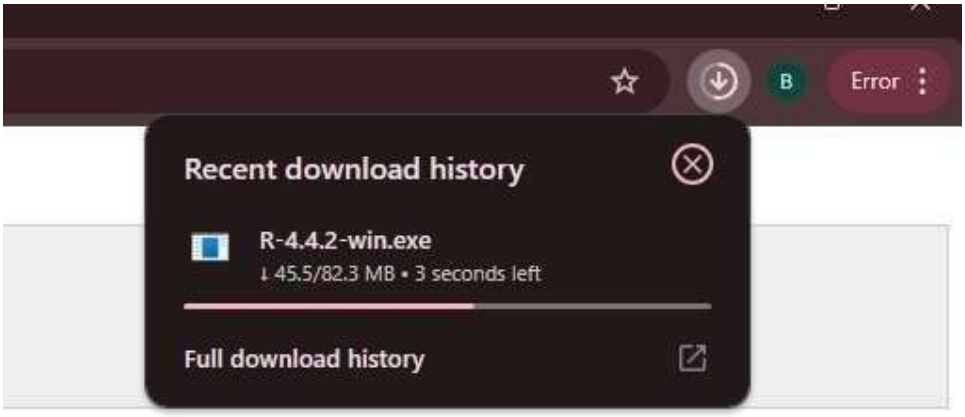
R provides a rich set of tools for data manipulation and analysis. By understanding its basic concepts, you can easily handle and analyze data from various sources.

Q.1 Installation and setting up R




cran.rstudio.com
The Comprehensive R Archive Network
Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:
Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
Download R for macOS
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!
The latest release (2024-10-31, Pile of Leaves) R-4.4.2.tar.gz, read what's new in the latest version.
The CRAN directory src/base-prerelease contains R alpha, beta, and rc releases as daily snapshots in time periods before a planned release.
Between releases, the same directory src/base-prerelease contains snapshots of current patched and development versions. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
Alternatively, daily snapshots are available here.
Source code of older versions of R is available here.
Contributed extension packages.
Questions About R
If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.
Supporting CRAN
CRAN operations, most importantly hosting, checking, distributing, and archiving of R add-on packages for various platforms, crucially rely on technical, emotional, and financial support by the R community.
Please consider making financial contributions to the R Foundation for Statistical Computing.
What are R and CRAN?
R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.
CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.
CRAN
Mirrors
What's new?
Search
CRAN Team
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Task Views
Other
Documentation
Manuals
FAQs
Contributed
Donations
Donate
https://cran.rstudio.com/bin/windows/
R for Windows
Subdirectories:
base Binaries for base distribution. This is what you want to install R for the first time.
contrib Binaries of contributed CRAN packages (for R >= 4.0.x).
old contrib Binaries of contributed CRAN packages for outdated versions of R (for R < 4.0.x).
Rtools Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.
Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.
You may also want to read the R FAQ and R for Windows FAQ
Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.
https://cran.rstudio.com/bin/windows/base/



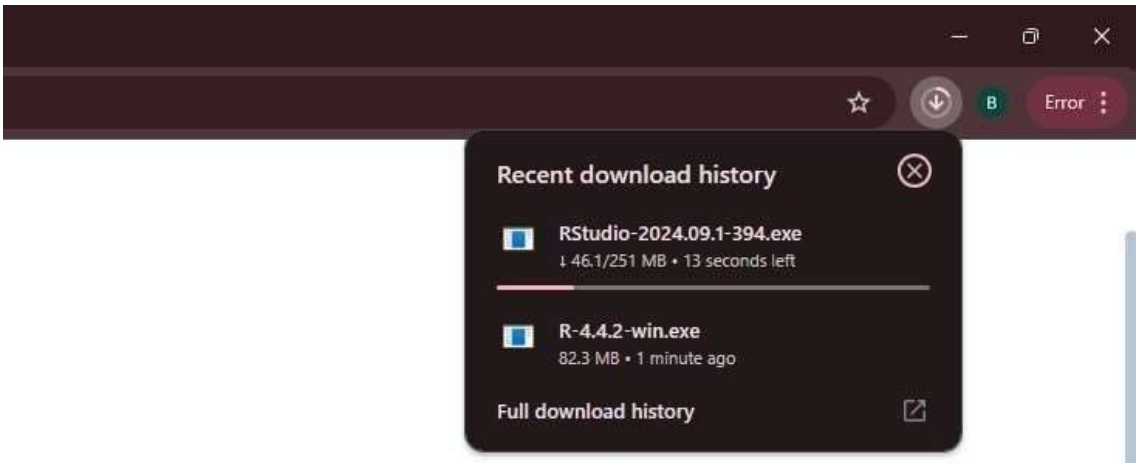


1 here.

e fingerprint on the master server.

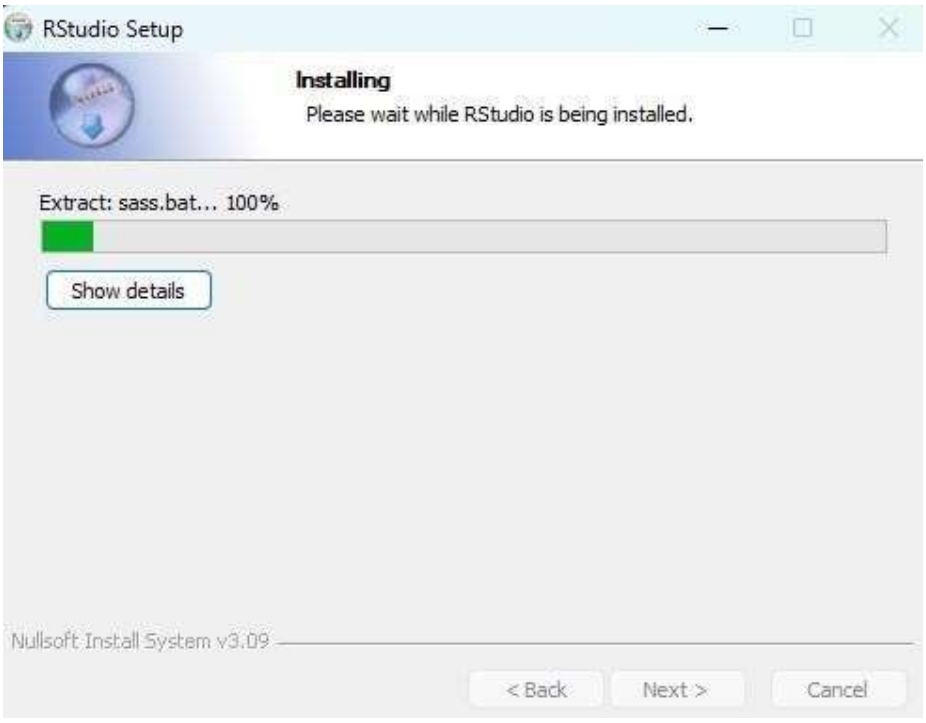
e.

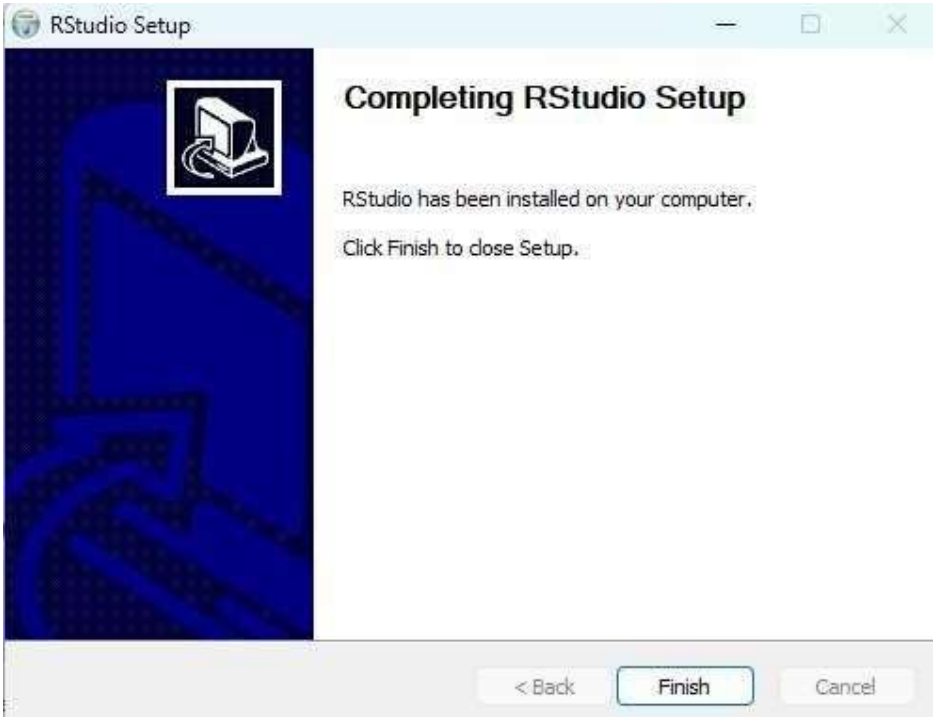
ect


Error
Recent download history
RStudio-2024.09.1-394.exe
↓ 46.1/251 MB • 13 seconds left
R-4.4.2-win.exe
82.3 MB • 1 minute ago
Full download history


RStudio Setup
Welcome to RStudio Setup
Setup will guide you through the installation of RStudio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


RStudio Setup
Choose Install Location
Choose the folder in which to install RStudio.
Setup will install RStudio in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files\RStudio
Browse...
Space required: 1.1 GB
Space available: 270.1 GB
Nullsoft Install System v3.09
< Back
Install
Cancel


RStudio Setup
Installing
Please wait while RStudio is being installed.
Extract: sass.bat... 100%
Show details
Nullsoft Install System v3.09
< Back
Next >
Cancel



Q.2.Data types in R Programming.

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

1.R Program to illustrate Numeric data type

Assign a decimal value to variable x

x = 5.6

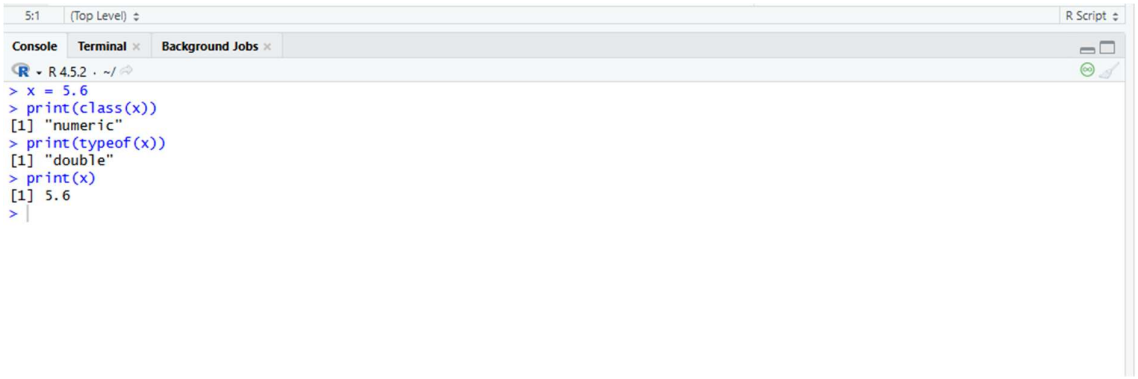
print the class name of variable

x print(class(x))

print the type of variable x

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

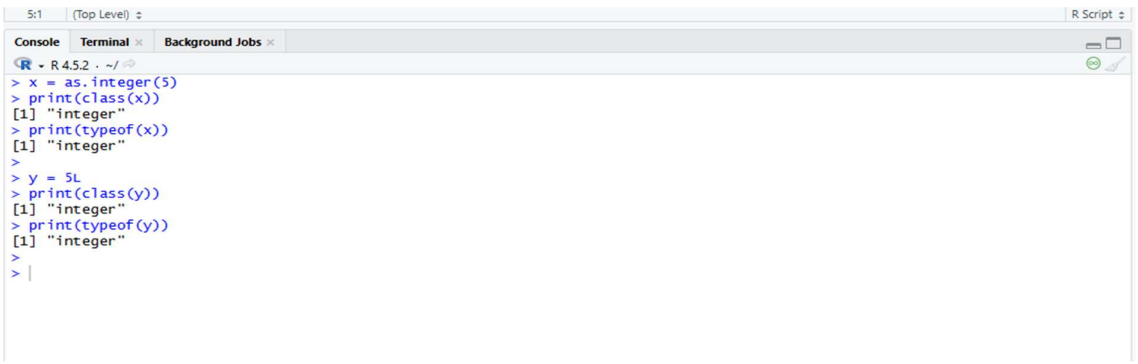
Output:



2.R program to illustrate integer data type

```
# Create an integer variable
x = as.integer(5)
# print the class name of variable x
print(class(x))
# print the type of variable x
print(typeof(x))
# Declare an integer by appending 'L' as suffix.
y = 5L
# print the class name of y
print(class(y))
# print the type of y
print(typeof(y))
```

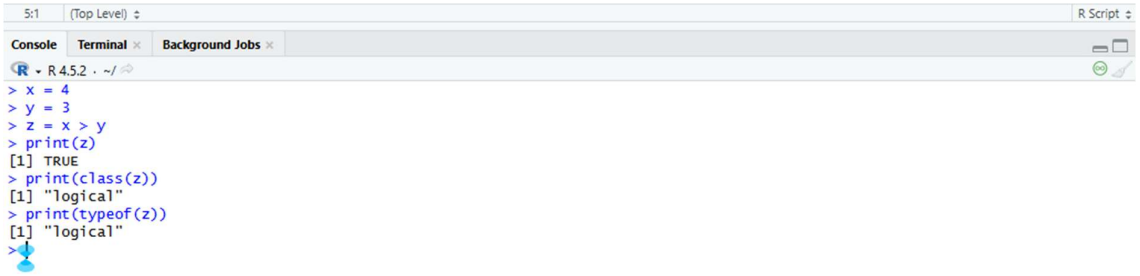
OUTPUT:



3.R program to illustrate logical data type

```
# Two variables
x = 4
y = 3
# Comparing two values
z = x > y
# print the logical value
print(z)
# print the class name of z
print(class(z))
# print the type of z
print(type of(z))
```

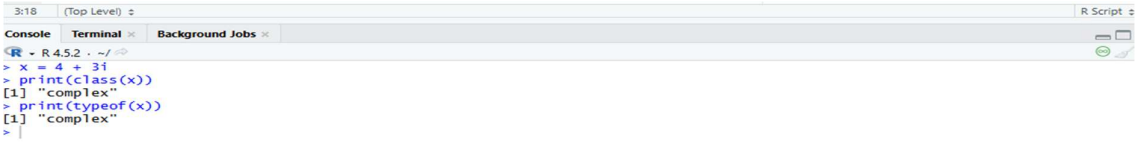
OUTPUT:



4.R program to illustrate complex data type

```
# Assign a complex value to variable x
x = 4 + 3i
# print the class name of variable x
print(class(x))
# print the type of variable x
print(typeof(x))
```

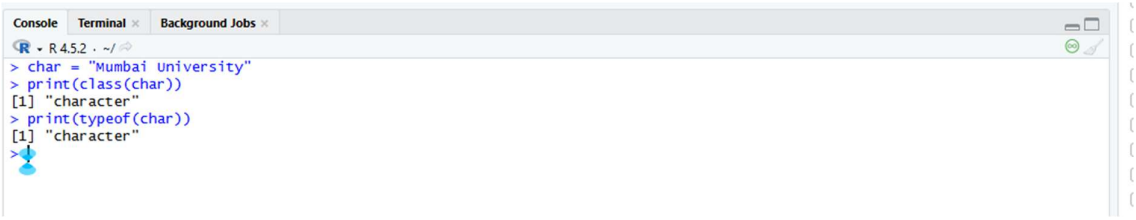
OUTPUT:



5.R program to illustrate character data type

```
# Assign a character value to char
char = "Mumbai University"
# print the class name of char
print(class(char))
# print the type of char
print(type of(char))
```

OUTPUT:



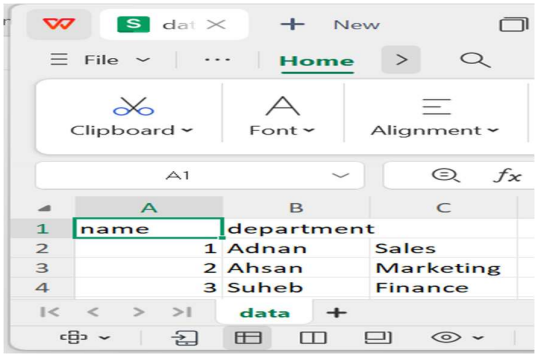
6.Reading and Writing data to and from R.

Code:

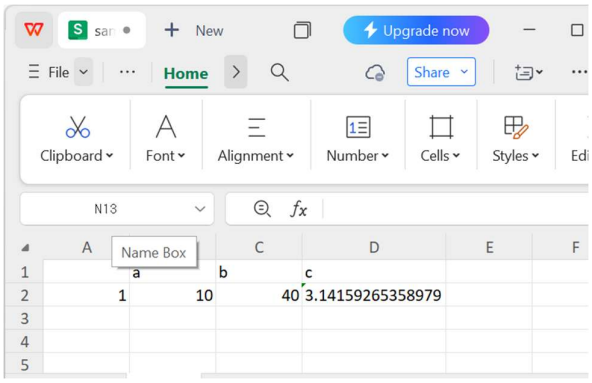
```
setwd("~/Adnan_FY31/Adbms/R studio practical")
setwd("~D:/Mca_Sem1/ADBMS")
x <- data.frame (name = c("Adnan", "Ahsan", "Suheb"), department = c("Sales", "Marketing", "Finance"))
x <- data.frame (name = c("Adnan", "Ahsan", "Suheb"), department = c("Sales", "Marketing", "Finance"))
write.table(x, file ="data.csv", sep = ",")
z <- data.frame(a = 10, b = 40, c = pi)
write.csv(z, file = "sample.csv")
```

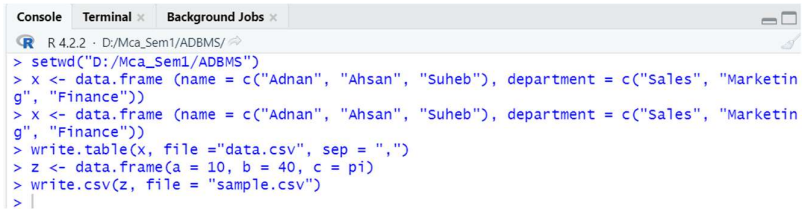
Output:

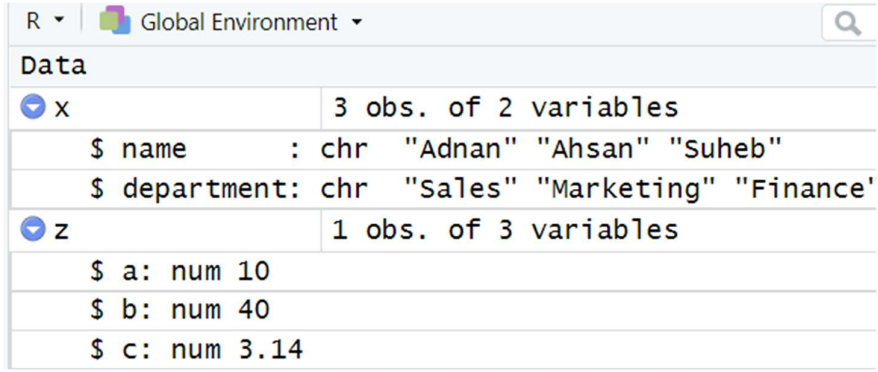
Data.csv



Sample.csv







7.Demonstrate Rstudio to insert data from a data frame into a mysql table.

Code:

```
install.packages("RMySQL")

library(RMySQL)

con <- dbConnect(MySQL(), user='root', password='gnims',
dbname='rstudiopractical',host='localhost',port=3306)

data <- data.frame( name = c("Adnan", "Suheb", "Ahsan"),age = c(22, 21, 15),city = c("Kurla",
"Wadala", "Goregaon"))

insert_query <- paste0("INSERT INTO emp1 (name, age, city) VALUES ")

values <- paste0("('", data$name, "', ", data$age, ", '", data$city, "')")

insert_query <- paste0(insert_query, paste(values, collapse = ", "))

dbSendQuery(con, insert_query)

dbDisconnect(con)
```

Output:

```
mysql> select * from emp1;
+-------+------+-----------+
| name  | age  | city      |
+-------+------+-----------+
| Adnan |   22 | New
Kurla |
| Suheb |   21 | Wadala    |
| Ahsan |   15 | Goregaon  |
+-------+------+-----------+
3 rows in set (0.00 sec)

mysql>
```

PRACTICAL NO: 6

AIM: Implementation of Data pre-processing techniques in RStudio like, Naming and Renaming variables, adding a new variable. Dealing with missing data. Dealing with categorical data. Data reduction using subsetting

Writeup:

Implementation of Data Pre-processing Techniques in RStudio

Data pre-processing is a critical step in data analysis to clean and prepare data for modeling. Below are some common techniques:

1. Naming and Renaming Variables

• Naming Variables: Use the names() function to assign or view variable names. Example: names(data) <- c("Var1", "Var2", "Var3").

• Renaming Variables: Modify specific names using indexing. Example: names(data)[2] <- "NewName".

2. Adding a New Variable

• Use the $ operator or indexing to add a new column. Example: data$new_var <- data$Var1 * 2

3. Dealing with Missing Data

• Identifying Missing Values: Use is.na(data) to check for missing values.

• Handling Missing Data:

- Replace with a value: data$Var1[is.na(data$Var1)] <- mean(data$Var1, na.rm = TRUE).
- Remove rows/columns: data <- na.omit(data) or data[complete.cases(data),].

4. Dealing with Categorical Data

• Encoding Categories: Use factor() to convert categorical data into factor variables. Example: data$Category <- factor(data$Category, levels = c("A", "B", "C"), labels = c(1, 2, 3)).

5. Data Reduction Using Subsetting

• Selecting Rows/Columns: Use indexing to subset data. Example: subset_data <- data[1:10, c("Var1", "Var2")].

• Using Conditions: Subset rows based on conditions. Example: subset_data <- data[data$Var1 > 50,].

These techniques help streamline the data cleaning process, making datasets more manageable and analysis-ready.

Code:

```
dataframe <- read.csv( "C:/Users/ADNAN/OneDrive/Documents/prac6.csv" ,sep = "\t", header = TRUE)

print(dataframe)
```

```
names(dataframe)[names(dataframe) == "Name"] <- "FirstName"

print(dataframe)

dataframe$emailaddress <- c('Adnan@gmail.com', 'Iqra@gmail.com', 'Saubiya@gmail.com',

                            'Shaban@gmail.com', 'Zaid@gmail.com','Safwan@gmail.com')

print(dataframe)

dataframe <- dataframe[complete.cases(dataframe), ]

print(dataframe)

dataframe <- read.csv( "C:/Users/ADNAN/OneDrive/Documents/prac6.csv" ,sep = "\t", header =
TRUE)

dataframe$Age <- ifelse(is.na(dataframe$Age), # Condition: If Age is NA (missing)

ave(dataframe$Age, FUN = function(x) mean(x, na.rm = TRUE)), # Calculate mean of non- missing
values in Age

dataframe$Age # Else, keep the original Age value

)

print(dataframe)

dataframe$Salary <- ifelse(

  is.na(dataframe$Salary), # Condition: If Salary is NA (missing)

  ave(dataframe$Salary, FUN = function(x) mean(x, na.rm = TRUE)), # Calculate mean of non-
missing values in Salary

  dataframe$Salary # Else, keep the original Salary value

)

print(dataframe)

dataframe$Gender <- as.numeric(factor(dataframe$Gender))

print(dataframe)

subset_data <- dataframe[dataframe$Age > 30, ]

print(subset_data)

dataframe <- read.csv("C:/Users/ADNAN/OneDrive/Documents/prac6.csv",sep = "\t", header =
TRUE)

print(dataframe)

sample_size <- 5 # Adjust the sample size as needed

sample_data <- dataframe[sample(nrow(dataframe), sample_size, replace = FALSE), ]

print(sample_data)
```

1.Load Data from CSV (make sure to make the CSV files)

```
Console  Terminal  Background Jobs
R 4.2.2 · D:/Mca_Sem1/ADBMS/
> dataframe <- read.csv( "C:/Users/ADNAN/OneDrive/Documents/prac6.csv" ,sep = "\t", header = TRUE)
> print(dataframe)
     Name Age Gender Occupation Salary
1   Adnan  21   Male   Engineer  60000
2    Iqra  20 Female     Doctor  80000
3 Saubiya  21 Female     Artist     NA
4  Shaban  NA Female   Designer  55000
5    Zaid  45   Male    Manager     NA
6  Safwan  NA   Male     Writer  45000
```

2.Naming and Renaming Variables

1. To Rename Variables

2. To Add a New Variable

```
> names(dataframe)[names(dataframe) == "Name"] <- "FirstName"
> print(dataframe)
  FirstName Age Gender Occupation Salary
1     Adnan  21   Male   Engineer  60000
2      Iqra  20 Female     Doctor  80000
3   Saubiya  21 Female     Artist     NA
4    Shaban  NA Female   Designer  55000
5      Zaid  45   Male    Manager     NA
6    Safwan  NA   Male     Writer  45000
> dataframe$emailaddress <- c('Adnan@gmail.com', 'Iqra@gmail.com', 'Saubiya@gmail.com',
+                             'Shaban@gmail.com', 'Zaid@gmail.com','Safwan@gmail.com')
> print(dataframe)
  FirstName Age Gender Occupation Salary      emailaddress
1     Adnan  21   Male   Engineer  60000   Adnan@gmail.com
2      Iqra  20 Female     Doctor  80000    Iqra@gmail.com
3   Saubiya  21 Female     Artist     NA Saubiya@gmail.com
4    Shaban  NA Female   Designer  55000  Shaban@gmail.com
5      Zaid  45   Male    Manager     NA    Zaid@gmail.com
6    Safwan  NA   Male     Writer  45000  Safwan@gmail.com
```

3.Dealing with Missing Data

1. Remove Rows with Missing Values

```
> dataframe <- dataframe[complete.cases(dataframe), ]
> print(dataframe)
  FirstName Age Gender Occupation Salary    emailaddress
1     Adnan  21   Male   Engineer  60000 Adnan@gmail.com
2      Iqra  20 Female     Doctor  80000  Iqra@gmail.com
```

2. Fill Missing Values with Mean

a. To calculate the missing Age

```
> dataframe <- read.csv( "C:/Users/ADNAN/OneDrive/Documents/prac6.csv" ,sep = "\t", header = TRUE)
> dataframe$Age <- ifelse(
+   is.na(dataframe$Age), # Condition: If Age is NA (missing)
+
+   ave(dataframe$Age, FUN = function(x) mean(x, na.rm = TRUE)), # Calculate mean of non- missing valu
es in Age
+
+   dataframe$Age # Else, keep the original Age value
+ )
> print(dataframe)
     Name   Age Gender Occupation Salary
1   Adnan 21.00   Male   Engineer  60000
2    Iqra 20.00 Female     Doctor  80000
3 Saubiya 21.00 Female     Artist     NA
4  Shaban 26.75 Female   Designer  55000
5    Zaid 45.00   Male    Manager     NA
6  Safwan 26.75   Male     Writer  45000
```

b. To calculate the missing salary

```
> dataframe$Salary <- ifelse(
+   is.na(dataframe$Salary), # Condition: If Salary is NA (missing)
+   ave(dataframe$Salary, FUN = function(x) mean(x, na.rm = TRUE)), # Calculate mean of non-missing valu
es in Salary
+   dataframe$Salary # Else, keep the original Salary value
+ )
> print(dataframe)
     Name   Age Gender Occupation Salary
1   Adnan 21.00   Male   Engineer  60000
2    Iqra 20.00 Female     Doctor  80000
3 Saubiya 21.00 Female     Artist  60000
4  Shaban 26.75 Female   Designer  55000
5    Zaid 45.00   Male    Manager  60000
6  Safwan 26.75   Male     Writer  45000
```

4.Dealing with Categorical Data

1. **Convert Categorical to Numerical (using factorization)**

```
> dataframe$Gender <- as.numeric(factor(dataframe$Gender))
> print(dataframe)
     Name   Age Gender Occupation Salary
1   Adnan 21.00      2   Engineer  60000
2    Iqra 20.00      1     Doctor  80000
3 Saubiya 21.00      1     Artist  60000
4  Shaban 26.75      1   Designer  55000
5    Zaid 45.00      2    Manager  60000
6  Safwan 26.75      2     Writer  45000
```

5.Data Reduction using Subsetting

1. **Subset Based on Condition**

```
> subset_data <- dataframe[dataframe$Age > 30, ]
> print(subset_data)
  Name Age Gender Occupation Salary
5 Zaid  45      2    Manager  60000
> dataframe <- read.csv("C:/Users/ADNAN/OneDrive/Documents/prac6.csv",sep = "\t", header = TRUE)
> print(dataframe)
     Name Age Gender Occupation Salary
1   Adnan  21   Male   Engineer  60000
2    Iqra  20 Female     Doctor  80000
3 Saubiya  21 Female     Artist     NA
4  Shaban  NA Female   Designer  55000
5    Zaid  45   Male    Manager     NA
6  Safwan  NA   Male     Writer  45000
>
```

2. **Random Sampling (to reduce data size for analysis)**

```
> sample_size <- 5 # Adjust the sample size as needed
> sample_data <- dataframe[sample(nrow(dataframe), sample_size, replace = FALSE), ]
> print(sample_data)
     Name Age Gender Occupation Salary
6  Safwan  NA   Male     Writer  45000
4  Shaban  NA Female   Designer  55000
3 Saubiya  21 Female     Artist     NA
1   Adnan  21   Male   Engineer  60000
2    Iqra  20 Female     Doctor  80000
>
```

Practical No:-7

Aim: Implementation and analysis of Linear Regression and Classification algorithms – Naïve Bayesian and K-Nearest Neighbor.

Write-up

1. Implementation and Analysis of Linear Regression in R Console

Linear regression is a statistical and machine learning technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between variables and is widely used for prediction and trend analysis.

1.1 Loading Data

A dataset is loaded or created for regression analysis.

```
data <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(2, 4, 5, 4, 5)
)
```

1.2 Fitting a Linear Regression Model

The lm() function is used to fit a linear regression model.

```
model <- lm(y ~ x, data = data)
```

1.3 Summarizing the Model

The summary() function provides information such as coefficients, R-squared value, and p-values.

```
summary(model)
```

1.4 Plotting the Results

The regression line is visualized using a scatter plot and fitted line.

```
plot(data$x, data$y, main="Linear Regression", xlab="X", ylab="Y")
abline(model, col="blue")
```

1.5 Making Predictions

The predict() function is used to predict values for new input data.

```
new_data <- data.frame(x = c(6, 7))
predictions <- predict(model, newdata = new_data)
```

1.6 Interpreting Results

Coefficients indicate the relationship between variables

R-squared shows how well the model fits the data

p-values indicate the statistical significance of predictors

2. Implementation and Analysis of Naïve Bayes Classification

Naïve Bayes is a probabilistic classification algorithm based on Bayes' Theorem. It assumes that features are independent, making it simple and efficient for classification tasks such as text classification and spam detection.

2.1 Loading Dataset

```
data <- data.frame(
  Outlook = c("Sunny","Sunny","Overcast","Rain"),
  Play = c("No","No","Yes","Yes")
)
```

2.2 Training the Model

```
library(e1071)
model <- naiveBayes(Play ~ ., data = data)
```

2.3 Making Predictions

```
predict(model, data)
```

2.4 Analysis

Fast and efficient classifier

Performs well with large datasets

Works best when feature independence assumption holds

3. Implementation and Analysis of K-Nearest Neighbor (KNN)

K-Nearest Neighbor is an instance-based classification algorithm that classifies a data point based on the majority class of its nearest neighbors.

3.1 Loading Dataset

```
data <- data.frame(
  x = c(1,2,3,6,7,8),
  y = c(1,2,3,6,7,8),
  class = c("A","A","A","B","B","B")
)
```

3.2 Applying KNN

```
library(class)

train <- data[,1:2]

test <- data[,1:2]

cl <- data$class

result <- knn(train, test, cl, k = 3)
```

3.3 Analysis

Simple and easy to understand

No training phase required

Performance depends on choice of K and distance measure

Conclusion

Linear Regression is effective for predicting continuous values, while Naïve Bayes and K-Nearest Neighbor are widely used classification algorithms. Naïve Bayes is fast and suitable for probabilistic classification, whereas KNN is a distance-based method that provides accurate results for small datasets. Together, these algorithms demonstrate both regression and classification techniques in machine learning.

7.1

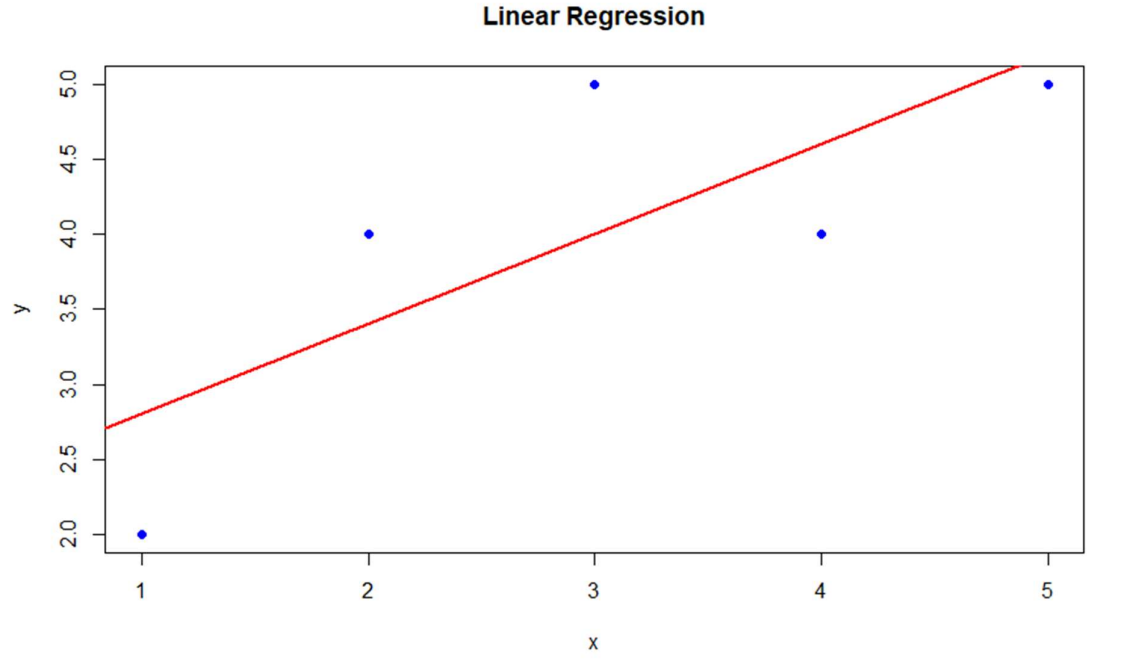
Code:

```
x <- c(1, 2, 3, 4, 5)

y <- c(2, 4, 5, 4, 5)

data <- data.frame(x, y)

model <- lm(y ~ x, data = data)

summary(model)

predicted <- predict(model)

print(predicted)

plot(x, y, main = "Linear Regression", col = "blue", pch = 16)

abline(model, col = "red", lwd = 2)
```

Output:

```
> data <- data.frame(x, y)
> model <- lm(y ~ x, data = data)
> summary(model)

Call:
lm(formula = y ~ x, data = data)

Residuals:
   1    2    3    4    5
-0.8  0.6  1.0 -0.6 -0.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2000     0.9381   2.345    0.101
x             0.6000     0.2828   2.121    0.124

Residual standard error: 0.8944 on 3 degrees of freedom
Multiple R-squared:    0.6,     Adjusted R-squared:  0.4667
F-statistic:   4.5 on 1 and 3 DF,  p-value: 0.124
```

```
> print(predicted)
  1   2   3   4   5
.8 3.4 4.0 4.6 5.2
```



7.2

Code:

```
install.packages("e1071")

library(e1071)

data <- data.frame(

Weather = c("Sunny", "Rainy", "Sunny", "Overcast", "Rainy", "Sunny"),

Play = c("No", "Yes", "Yes", "Yes", "No", "Yes")
```

```
)
data$Weather <- as.factor(data$Weather)
data$Play <- as.factor(data$Play)
model <- naiveBayes(Play ~ Weather, data = data)
print(model)
test <- data.frame(Weather = c("Sunny", "Rainy", "Overcast"))
prediction <- predict(model, test)
print(prediction)
```

Output:

```
> print(model)

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       No       Yes
0.3333333 0.6666667

Conditional probabilities:
     Weather
Y     Overcast Rainy Sunny
  No       0.00  0.50  0.50
  Yes      0.25  0.25  0.50

>
```

```
> print(prediction)
[1] Yes No  Yes
Levels: No Yes
>
>
```

7.3

Code:

```
install.packages("class")
library(class)
data <- data.frame(
Height = c(150, 160, 170, 180, 190),
Weight = c(50, 60, 70, 80, 90),
Category = c("Thin", "Normal", "Normal", "Overweight", "Overweight")
)
train <- data[, 1:2]
labels <- data$Category
test <- data.frame(Height = 175, Weight = 75)
```

result <- knn(train, test, labels, k = 3)

print(result)

Output:

Practical no.8

Aim: Implementation and analysis of Classification algorithms - ID3

Introduction

Classification is a supervised learning technique used to predict class labels based on input data. ID3 (Iterative Dichotomiser 3) is a decision tree–based classification algorithm that constructs a tree using information gain. It is widely used for understanding the fundamentals of decision tree learning due to its simplicity and interpretability.

Major Points

ID3 builds a decision tree using a top-down greedy approach.

It uses entropy to measure impurity in the dataset.

Information Gain is used to select the best attribute for splitting.

Works only with categorical attributes.

Produces easy-to-understand and interpretable classification rules.

Conclusion

ID3 is a simple and efficient classification algorithm suitable for small and structured datasets. Although it has limitations such as sensitivity to noise and inability to handle continuous data directly, it remains an important algorithm for understanding decision tree-based classification methods.

Code:

ID3 Algorithm Implementation in R

```
install.packages("rpart")

install.packages("rpart.plot")

library(rpart)

library(rpart.plot)

data <- data.frame(

Outlook = c("Sunny","Sunny","Overcast","Rain","Rain","Rain",

"Overcast","Sunny","Sunny","Rain","Sunny",

"Overcast","Overcast","Rain"),

Temperature = c("Hot","Hot","Hot","Mild","Cool","Cool",

"Mild","Cool","Mild","Mild","Mild",

"Hot","Mild","Hot"),

Humidity = c("High","High","High","High","Normal","Normal",
```

```
"High","High","Normal","Normal","Normal",
"High","Normal","High"),
Wind = c("Weak","Strong","Weak","Weak","Weak","Strong",
"Strong","Weak","Weak","Weak","Strong",
"Strong","Weak","Strong"),
PlayTennis = c("No","No","Yes","Yes","Yes","No",
"Yes","No","Yes","Yes","Yes",
"Yes","Yes","No")
)
print("Dataset Loaded Successfully")
print(data)
entropy <- function(target){
freq <- table(target) / length(target)
return(-sum(freq * log2(freq)))
}
base_entropy <- entropy(data$PlayTennis)
print(paste("Base Entropy:", round(base_entropy, 4)))
model <- rpart(PlayTennis ~ ., data = data, method = "class",
parms = list(split = "information"))
print("Decision Tree Model:")
print(model)
rpart.plot(model, type = 4, extra = 104,
fallen.leaves = TRUE,
main = "ID3 Decision Tree (Play Tennis Dataset)")
test <- data.frame(Outlook="Sunny", Temperature="Cool",
Humidity="Normal", Wind="Strong")
prediction <- predict(model, test, type="class")
print("Test Input:")
print(test)
print(paste("Predicted Output:", prediction))
predicted <- predict(model, data, type="class")
```

accuracy <- sum(predicted == data$PlayTennis) / nrow(data)

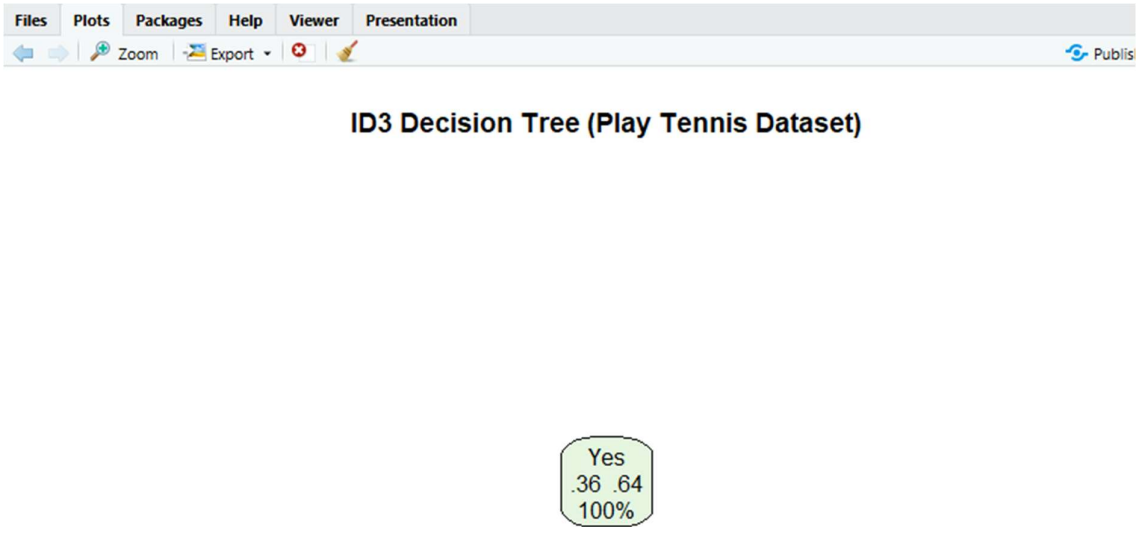
print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))

Output:

```
+                 "Yes","Yes","No")
+ )
> print("Dataset Loaded Successfully")
[1] "Dataset Loaded Successfully"
> print(data)
    Outlook Temperature Humidity   Wind PlayTennis
1     Sunny         Hot     High   Weak         No
2     Sunny         Hot     High Strong         No
3  Overcast         Hot     High   Weak        Yes
4      Rain        Mild     High   Weak        Yes
5      Rain        Cool   Normal   Weak        Yes
6      Rain        Cool   Normal Strong         No
7  Overcast        Mild     High Strong        Yes
8     Sunny        Cool     High   Weak         No
9     Sunny        Mild   Normal   Weak        Yes
10     Rain        Mild   Normal   Weak        Yes
11    Sunny        Mild   Normal Strong        Yes
12 Overcast         Hot     High Strong        Yes
13 Overcast        Mild   Normal   Weak        Yes
14     Rain         Hot     High Strong         No
>
```

```
R 4.5.2 · ~/
+   return(-sum(freq * log2(freq)))
+ }
> base_entropy <- entropy(data$PlayTennis)
> print(paste("Base Entropy:", round(base_entropy, 4)))
[1] "Base Entropy: 0.9403"
> model <- rpart(PlayTennis ~ ., data = data, method = "class",
+                parms = list(split = "information"))
> print("Decision Tree Model:")
[1] "Decision Tree Model:"
> print(model)
n= 14

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 14 5 Yes (0.3571429 0.6428571) *
> rpart.plot(model, type = 4, extra = 104,
+            fallen.leaves = TRUE,
+            main = "ID3 Decision Tree (Play Tennis Dataset)")
> test <- data.frame(Outlook="Sunny", Temperature="Cool",
+                    Humidity="Normal", Wind="Strong")
> prediction <- predict(model, test, type="class")
```

```
+            fallen.leaves = TRUE,
+            main = "ID3 Decision Tree (Play Tennis Dataset)")
> test <- data.frame(Outlook="Sunny", Temperature="Cool",
+                    Humidity="Normal", Wind="Strong")
> prediction <- predict(model, test, type="class")
> print("Test Input:")
[1] "Test Input:"
> print(test)
  Outlook Temperature Humidity   Wind
1   Sunny        Cool   Normal Strong
> print(paste("Predicted Output:", prediction))
[1] "Predicted Output: Yes"
> predicted <- predict(model, data, type="class")
> accuracy <- sum(predicted == data$PlayTennis) / nrow(data)
> print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))
[1] "Model Accuracy: 64.29 %"
>
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publis
ID3 Decision Tree (Play Tennis Dataset)
Yes
.36 .64
100%

PRACTICAL 9

AIM: Implementation and analysis of . C4.5 and Apriori Algorithm using Market Basket Analysis

1. Introduction

Data mining is the process of discovering meaningful patterns, correlations, and knowledge from large datasets using statistical, machine learning, and database techniques. Among various data mining tasks, classification and association rule mining play a crucial role in decision-making systems.

This study focuses on the implementation and analysis of two widely used algorithms:

C4.5 Algorithm for classification

Apriori Algorithm for association rule mining using Market Basket Analysis

Both algorithms are foundational techniques used in domains such as retail analytics, healthcare, finance, and customer behavior analysis.

2. Classification and the C4.5 Algorithm

2.1 Classification in Data Mining

Classification is a **supervised learning technique** that assigns predefined class labels to data instances based on their attributes. A classification model is trained using labeled data and then used to predict the class of unseen data.
Decision tree-based classifiers are popular due to their simplicity, interpretability, and efficiency.

2.2 C4.5 Algorithm

The **C4.5 algorithm**, developed by **Ross Quinlan**, is an extension of the ID3 algorithm and is used to generate a **decision tree** for classification purposes.

C4.5 constructs a decision tree by recursively splitting the dataset based on attribute values that provide the highest discriminatory power. The algorithm handles both **categorical and continuous attributes**, manages missing values, and performs **pruning** to avoid overfitting.

2.3 Working Principle of C4.5

1. Select the attribute that best splits the dataset using **Gain Ratio**.
2. Create a decision node based on the selected attribute.
3. Split the dataset into subsets.
4. Repeat the process recursively for each subset.
5. Stop when all instances belong to the same class or no attributes remain.

2.4 Information Gain and Gain Ratio

- **Entropy** measures the impurity or randomness in the dataset:

$$Entropy(S) = -\sum p_i \log_2(p_i)$$

- **Information Gain** measures the reduction in entropy after splitting:

$$Gain(S, A) = Entropy(S) - \sum \frac{|S_v|}{|S|} Entropy(S_v)$$

- **Gain Ratio** adjusts Information Gain to avoid bias toward attributes with many values:

$$GainRatio(A) = \frac{Gain(S, A)}{SplitInfo(A)}$$

C4.5 selects the attribute with the **highest Gain Ratio**.

2.5 Advantages of C4.5

- Handles continuous and discrete data
- Deals with missing values
- Performs tree pruning
- Produces easily interpretable rules

3. Association Rule Mining and Market Basket Analysis

3.1 Association Rule Mining

Association rule mining aims to discover interesting relationships, patterns, or associations among items in large transactional databases. It is widely used in **Market Basket Analysis**, where customer purchasing behavior is analyzed.

An association rule is represented as:

$$X \rightarrow Y$$

where $X$and $Y$are itemsets.

3.2 Market Basket Analysis

Market Basket Analysis identifies patterns in customer purchases to understand which products are frequently bought together.
It helps businesses in:

- Product placement
- Cross-selling strategies
- Recommendation systems
- Inventory management

4. Apriori Algorithm

4.1 Overview

The **Apriori algorithm** is a classical algorithm for mining frequent itemsets and generating association rules. It is based on the **Apriori Principle**, which states:

"If an itemset is frequent, all of its subsets must also be frequent."

4.2 Working of Apriori Algorithm

1. Generate frequent 1-itemsets by scanning the database.
2. Use frequent k-itemsets to generate candidate $(k+1)$-itemsets.
3. Prune candidate itemsets whose subsets are not frequent.
4. Repeat until no further frequent itemsets are found.
5. Generate association rules from frequent itemsets.

4.3 Measures of Association Rules

- **Support**
  Indicates how frequently an itemset appears in the dataset.

$$Support(X) = \frac{\text{Number of transactions containing } X}{\text{Total transactions}}$$

- **Confidence**
  Measures the reliability of the rule.

$$Confidence(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

- **Lift**
  Measures the strength of association.

$$Lift(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{Support(Y)}$$

4.4 Advantages of Apriori Algorithm

- Simple and easy to understand
- Effective for small and medium datasets
- Generates interpretable association rules

6. Conclusion

The C4.5 and Apriori algorithms represent two important paradigms in data mining: **classification** and **association analysis**.
C4.5 is effective for predictive modeling and decision-making, while Apriori provides valuable insights into customer behavior through Market Basket Analysis.

Together, these algorithms enable comprehensive data-driven analysis for various real-world applications.

Code:

1. C4.5 Algorithm Implementation in R

```
install.packages("RWeka")
library(RWeka)
data <- data.frame(
  Weather = c("Sunny", "Rainy", "Sunny", "Overcast", "Rainy", "Sunny"),
  Temperature = c("Hot", "Mild", "Mild", "Cool", "Cool", "Hot"),
  Play = c("No", "Yes", "Yes", "Yes", "No", "Yes")
)
data <- data.frame(lapply(data, factor))
model_c45 <- J48(Play ~ Weather + Temperature, data = data)
print(model_c45)
test <- data.frame(
  Weather = factor("Sunny", levels = levels(data$Weather)),
  Temperature = factor("Cool", levels = levels(data$Temperature))
)
predict_c45 <- predict(model_c45, test)
print(predict_c45)
predicted <- predict(model_c45, data)
accuracy <- sum(predicted == data$Play) / nrow(data)
print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))
```

Output:

```
> data <- data.frame(lapply(data, factor))
> model_c45 <- J48(Play ~ Weather + Temperature, data = data)
> print(model_c45)
J48 pruned tree
------------------
: Yes (6.0/2.0)

Number of Leaves  :     1

Size of the tree :      1

> test <- data.frame(
+   Weather = factor("Sunny", levels = levels(data$Weather)),
+   Temperature = factor("Cool", levels = levels(data$Temperature))
+ )
> predict_c45 <- predict(model_c45, test)
> print(predict_c45)
[1] Yes
Levels: No Yes
> predicted <- predict(model_c45, data)
> accuracy <- sum(predicted == data$Play) / nrow(data)
> print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))
[1] "Model Accuracy: 66.67 %"
>
>
```

PRACTICAL NO.9.2

Code:

```
install.packages("arules" , dependencies = TRUE)

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

library(arules)

library(arulesViz)

transactions <- list(

c("Milk", "Bread", "Eggs"),

c("Bread", "Butter"),

c("Milk", "Bread", "Butter", "Eggs"),

c("Milk", "Eggs"),

c("Bread", "Eggs"),

c("Milk", "Bread", "Butter")

)

trans <- as(transactions, "transactions")

summary(trans)

inspect(trans)

rules <- apriori(trans, parameter = list(supp = 0.5, conf = 0.60))
```

inspect(rules)

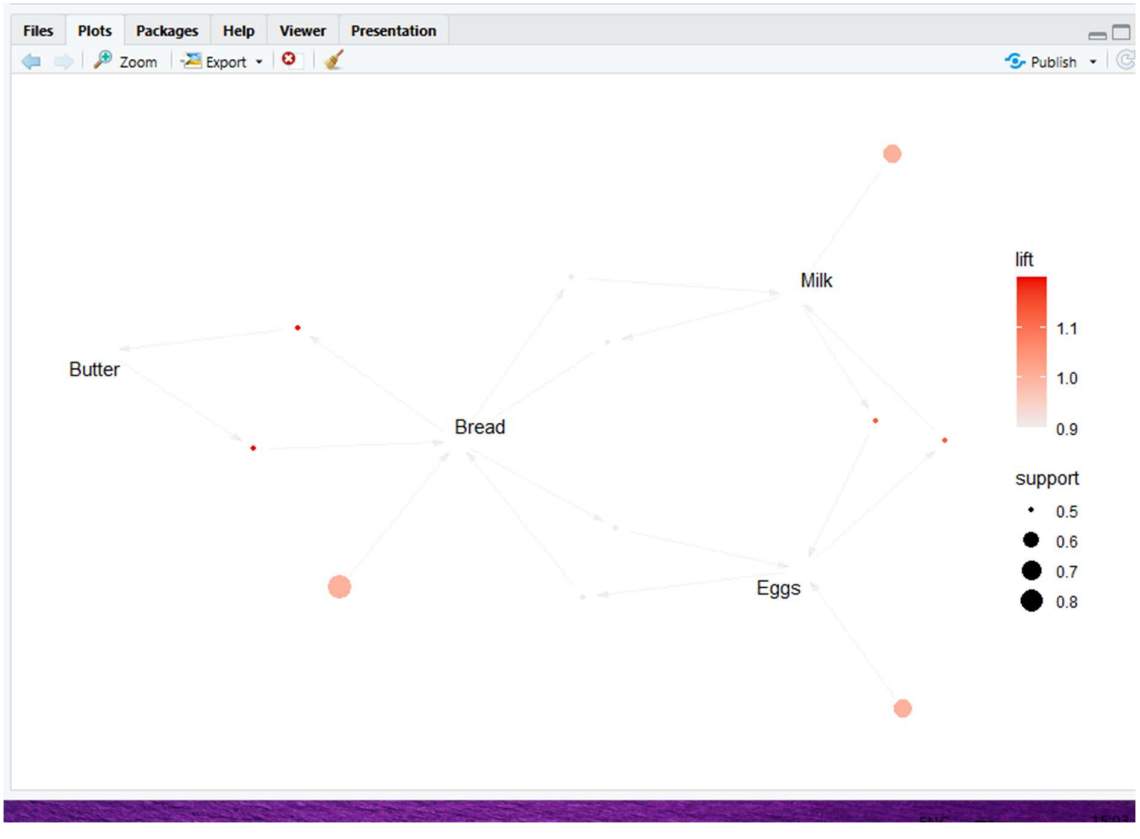
inspect(sort(rules, by = "lift"))

plot(rules, method = "graph")

Output:

```
+ )
> trans <- as(transactions, "transactions")
> summary(trans)
transactions as itemMatrix in sparse format with
 6 rows (elements/itemsets/transactions) and
 4 columns (items) and a density of 0.6666667

most frequent items:
  Bread    Eggs    Milk  Butter (Other)
      5       4       4       3       0

element (itemset/transaction) length distribution:
sizes
2 3 4
3 2 1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.500   2.667   3.000   4.000

includes extended item information - examples:
  labels
1  Bread
2 Butter
3   Eggs

> inspect(trans)
    items
[1] {Bread, Eggs, Milk}
[2] {Bread, Butter}
[3] {Bread, Butter, Eggs, Milk}
[4] {Eggs, Milk}
[5] {Bread, Eggs}
[6] {Bread, Butter, Milk}
> rules <- apriori(trans, parameter = list(supp = 0.5, conf = 0.60))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.6    0.1    1 none FALSE            TRUE       5     0.5      1     10  rules TRUE

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

Absolute minimum support count: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[4 item(s), 6 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].
```

```
creating S4 object  ... done [0.00s].
> inspect(rules)
     lhs         rhs      support   confidence coverage  lift  count
[1]  {}       => {Eggs}   0.6666667 0.6666667  1.0000000 1.000 4
[2]  {}       => {Milk}   0.6666667 0.6666667  1.0000000 1.000 4
[3]  {}       => {Bread}  0.8333333 0.8333333  1.0000000 1.000 5
[4]  {Butter} => {Bread}  0.5000000 1.0000000  0.5000000 1.200 3
[5]  {Bread}  => {Butter} 0.5000000 0.6000000  0.8333333 1.200 3
[6]  {Eggs}   => {Milk}   0.5000000 0.7500000  0.6666667 1.125 3
[7]  {Milk}   => {Eggs}   0.5000000 0.7500000  0.6666667 1.125 3
[8]  {Eggs}   => {Bread}  0.5000000 0.7500000  0.6666667 0.900 3
[9]  {Bread}  => {Eggs}   0.5000000 0.6000000  0.8333333 0.900 3
[10] {Milk}   => {Bread}  0.5000000 0.7500000  0.6666667 0.900 3
[11] {Bread}  => {Milk}   0.5000000 0.6000000  0.8333333 0.900 3
> inspect(sort(rules, by = "lift"))
     lhs         rhs      support   confidence coverage  lift  count
[1]  {Butter} => {Bread}  0.5000000 1.0000000  0.5000000 1.200 3
[2]  {Bread}  => {Butter} 0.5000000 0.6000000  0.8333333 1.200 3
[3]  {Eggs}   => {Milk}   0.5000000 0.7500000  0.6666667 1.125 3
[4]  {Milk}   => {Eggs}   0.5000000 0.7500000  0.6666667 1.125 3
[5]  {}       => {Eggs}   0.6666667 0.6666667  1.0000000 1.000 4
[6]  {}       => {Milk}   0.6666667 0.6666667  1.0000000 1.000 4
[7]  {}       => {Bread}  0.8333333 0.8333333  1.0000000 1.000 5
[8]  {Eggs}   => {Bread}  0.5000000 0.7500000  0.6666667 0.900 3
[9]  {Milk}   => {Bread}  0.5000000 0.7500000  0.6666667 0.900 3
[10] {Bread}  => {Eggs}   0.5000000 0.6000000  0.8333333 0.900 3
[11] {Bread}  => {Milk}   0.5000000 0.6000000  0.8333333 0.900 3
> plot(rules, method = "graph")
>
>
```



PRACTICAL 10

A. K-MEANS CLUSTERING

B. AGGLOMERATIVE HIERARCHICAL CLUSTERING

1. Introduction

Clustering is an unsupervised machine learning technique used to group data points such that objects within the same group are more similar to each other than to those in other groups. Unlike supervised learning, clustering does not require labeled data. It is widely used in data analysis, pattern recognition, customer segmentation, image processing, and bioinformatics.

This journal focuses on the implementation and analysis of two popular clustering algorithms:

1. **K-Means Clustering**
2. **Agglomerative (Hierarchical) Clustering**

2. K-Means Clustering

2.1 Definition

K-Means is a partition-based clustering algorithm that divides a dataset into K distinct non-overlapping clusters, where each data point belongs to the cluster with the nearest mean (centroid).

2.2 Objective of K-Means

The main objective of K-Means is to minimize intra-cluster variance, also known as Within Cluster Sum of Squares (WCSS).

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} || \, x - \mu_i \, ||^2$$

Where:

- k**= number of clusters**
- C_i**= i-th cluster**
- μ_i**= centroid of cluster** C_i

2.3 Working of K-Means Algorithm

Step 1: Choose the number of clusters K.
Step 2: Initialize Kcentroids randomly.
Step 3: Assign each data point to the nearest centroid using distance measures such as Euclidean distance.

Step 4: Recalculate centroids as the mean of all points in a cluster.
Step 5: Repeat steps 3 and 4 until centroids no longer change or maximum iterations are reached.

2.4 Advantages of K-Means

- Simple and easy to implement
- Fast and computationally efficient
- Works well for large datasets

2.5 Limitations of K-Means

- Requires predefined value of K
- Sensitive to initial centroid selection
- Performs poorly with non-spherical clusters
- Sensitive to outliers

2.6 Applications of K-Means

- Customer segmentation
- Market basket analysis
- Image compression
- Document clustering

3. Agglomerative (Hierarchical) Clustering

3.1 Definition

Agglomerative clustering is a **hierarchical clustering technique** that follows a **bottom-up approach**. Initially, each data point is treated as a separate cluster, and clusters are **merged step by step** based on similarity until a single cluster is formed.

3.2 Types of Linkage Methods

Agglomerative clustering uses linkage criteria to decide which clusters to merge:

1. **Single Linkage** – Minimum distance between points
2. **Complete Linkage** – Maximum distance between points
3. **Average Linkage** – Average distance between all points
4. **Ward's Method** – Minimizes variance within clusters

3.3 Working of Agglomerative Clustering

Step 1: Treat each data point as an individual cluster.
Step 2: Compute the distance matrix between clusters.

Step 3: Merge the two closest clusters based on linkage criteria.
Step 4: Update the distance matrix.
Step 5: Repeat steps 3 and 4 until one cluster remains or desired clusters are obtained.

3.4 Dendrogram

A **dendrogram** is a tree-like diagram used to visualize the hierarchical clustering process. It helps in deciding the optimal number of clusters by cutting the tree at a certain height.

3.5 Advantages of Agglomerative Clustering

- No need to specify number of clusters initially
- Produces a clear hierarchical structure
- Works well for small datasets

3.6 Limitations of Agglomerative Clustering

- Computationally expensive
- Not suitable for large datasets
- Once clusters are merged, they cannot be split

4. Conclusion

K-Means and Agglomerative clustering are powerful unsupervised learning algorithms with different strengths. K-Means is efficient and suitable for large datasets, whereas Agglomerative clustering provides better interpretability through hierarchical relationships. The choice of algorithm depends on dataset size, structure, and application requirements.

10.1

Aim: K Means clustering

Code:

```
install.packages("factoextra")
library(factoextra)
data(iris)
df <- iris[, 1:4]
df_scaled <- scale(df)
set.seed(123)
kmeans_model <- kmeans(df_scaled, centers = 3, nstart = 25)
cat("Cluster Centers:\n")
print(kmeans_model$centers)
```

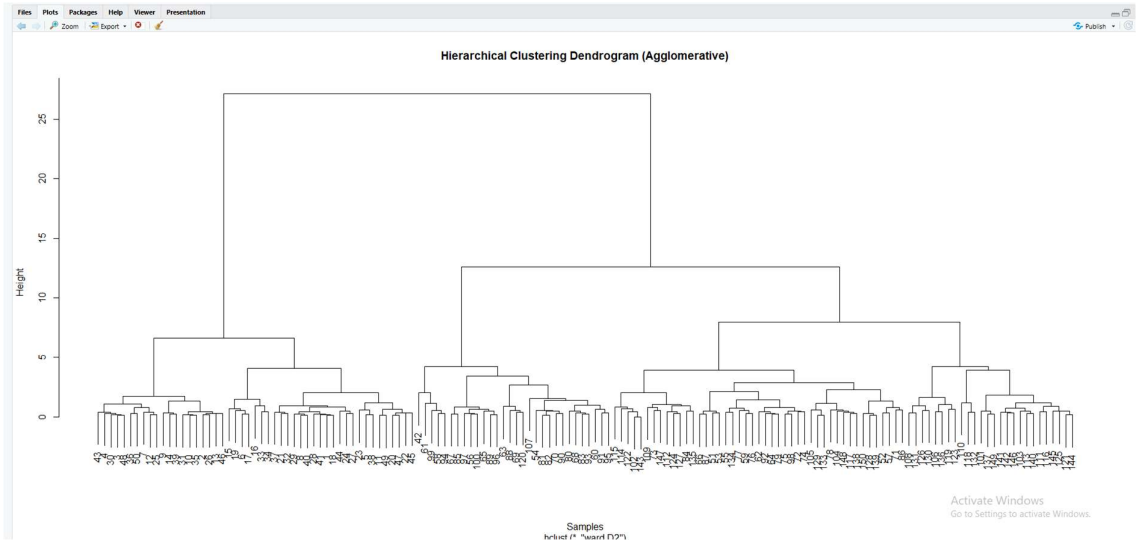
```
cat("\nCluster Sizes:\n")
print(kmeans_model$size)
cat("\nCluster Assignments (first 20 rows):\n")
print(kmeans_model$cluster[1:20])
library(ggplot2)
library(cluster)
fviz_cluster(
  list(data = df_scaled, cluster = kmeans_model$cluster),
  geom = "point",
  ellipse = TRUE,
  main = "K-Means Clustering of Iris Data"
)
data(iris)
df <- iris[, 1:4]
df_scaled <- scale(df)
dist_matrix <- dist(df_scaled, method = "euclidean")
hclust_model <- hclust(dist_matrix, method = "ward.D2")
plot(
  hclust_model,
  main = "Hierarchical Clustering Dendrogram (Agglomerative)",
  xlab = "Samples",
  ylab = "Height"
)
clusters <- cutree(hclust_model, k = 3)
cat("Cluster Assignments (first 20 rows):\n")
print(clusters[1:20])
df_clustered <- data.frame(df_scaled, Cluster = as.factor(clusters))
ggplot(df_clustered, aes(x = Sepal.Length, y = Petal.Length, color = Cluster)) +
  geom_point() +
  labs(title = "Agglomerative Clustering Results")
```

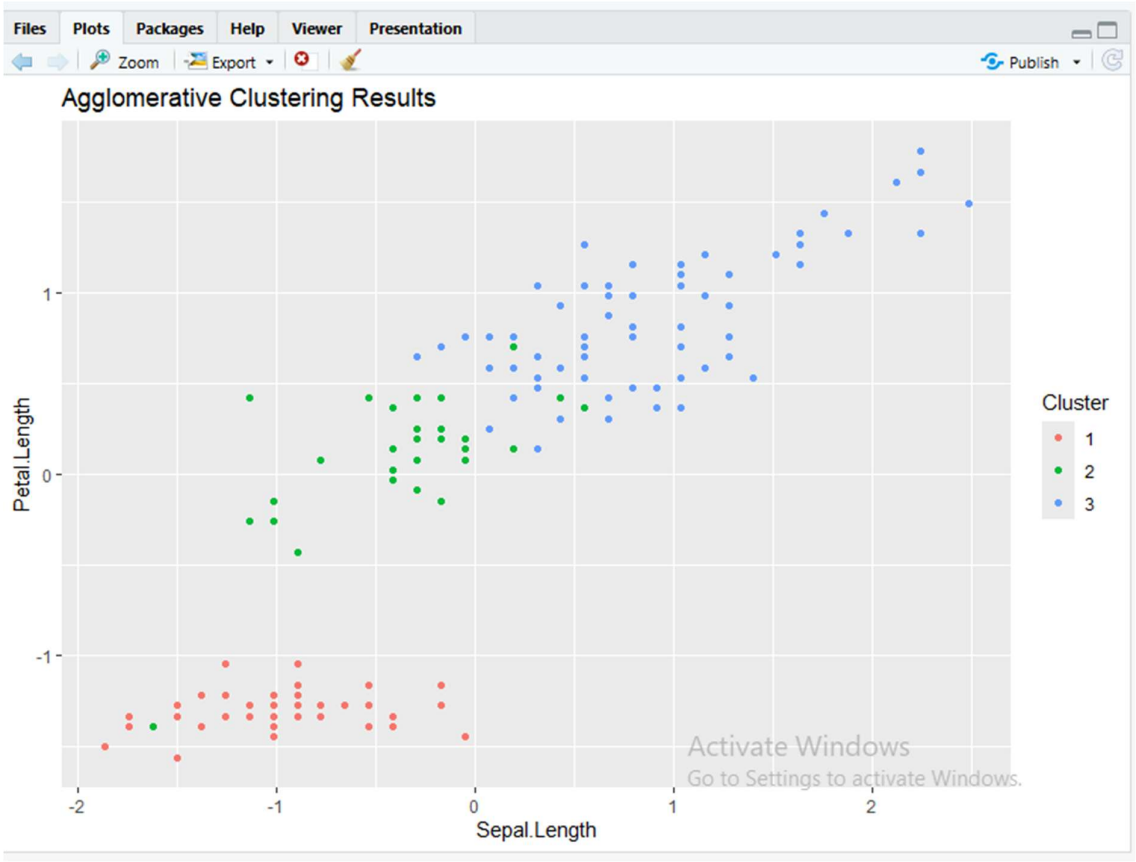
Output:

```
Console  Terminal  Background Jobs
R 4.5.2 · ~/
> library(factoextra)
> data(iris)
> df <- iris[, 1:4]
> df_scaled <- scale(df)
> set.seed(123)
> kmeans_model <- kmeans(df_scaled, centers = 3, nstart = 25)
> cat("Cluster Centers:\n")
Cluster Centers:
> print(kmeans_model$centers)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287
> cat("\nCluster Sizes:\n")

Cluster Sizes:
> print(kmeans_model$size)
[1] 50 53 47
> cat("\nCluster Assignments (first 20 rows):\n")

Cluster Assignments (first 20 rows):
> print(kmeans_model$cluster[1:20])
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> library(ggplot2)
> library(cluster)
> fviz_cluster(
+   list(data = df_scaled, cluster = kmeans_model$cluster),
+   geom = "point",
+   ellipse = TRUE,
+   main = "K-Means Clustering of Iris Data"
+ )
> data(iris)
> df <- iris[, 1:4]
> df_scaled <- scale(df)
> dist_matrix <- dist(df_scaled, method = "euclidean")
> hclust_model <- hclust(dist_matrix, method = "ward.D2")
> plot(
+   hclust_model,
+   main = "Hierarchical Clustering Dendrogram (Agglomerative)",
+   xlab = "Samples",
+   ylab = "Height"
+ )
> clusters <- cutree(hclust_model, k = 3)
> cat("Cluster Assignments (first 20 rows):\n")
Cluster Assignments (first 20 rows):
> print(clusters[1:20])
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> df_clustered <- data.frame(df_scaled, Cluster = as.factor(clusters))
> ggplot(df_clustered, aes(x = Sepal.Length, y = Petal.Length, color = Cluster)) +
+   geom_point() +
+   labs(title = "Agglomerative Clustering Results")
>
>
```

Agglomerative (Hierarchical) Clustering

Output:




Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Agglomerative Clustering Results
Petal.Length
Sepal.Length
1
0
-1
-2
-1
0
1
2
Cluster
1
2
3
Activate Windows
Go to Settings to activate Windows.