SREENIDHI EDUCATIONAL GROUP | SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

SNIST Sreenidhi Institute of Science and Technology
Because, life is all about taking the right decisions
AUTONOMOUS

(An Autonomous Institution approved by UGC and affiliated to JNTUH))
(Accredited by NAAC with 'A" Grade, Accredited by NBA of AICTE and Recipient of World Bank under TEQIP-I and II)
Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301

LAB MANUALS
For

DATA WAREHOUSING & DATA MINING
(From Page No. 4 to 72)

&

COMPUTER NETWORKS
(From Page No. 73 to)

&

DESIGN & ANALYSIS OF ALGORITHMS
(From Page No. 96 to End)

FOR
B. Tech. III year - I Semester
CSE Branch

SNIST
Sreenidhi Institute of Science and Technology
Because, Life is all about taking the right decisions
AUTONOMOUS

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
JUNE-2020

B. Tech (Computer Science and Engineering)

Program objective:

B. Tech in Computer Science and Engineering program emphasizes the use of computer as a sophisticated problem solving tool.

The first two years of this program begins with a set of introductory courses, like Mathematics, physics, English, computer languages (C,C++,Java), Database Management Systems, which provide students with a firm foundation in mathematics, computer science, as well as communication skills. These courses include weekly labs in which students use state-of-the art software development techniques to create solutions to interesting problems.

The last two years of study focuses on the concepts and techniques used in the design and development of advanced software systems. In addition, students choose from a rich set of electives, which covers skills in demand. These advanced courses give broad opening for research and help them to choose specialization in their higher studies. A generous allotment of open electives allows students to learn foreign languages like French, German, Spanish; and it includes computing with a business focus.

Students in this program pursue an inter-disciplinary course of study that combines strong foundation in computer science with a focus on interdisciplinary areas. This program is designed for students who seek to blend their computer science abilities with skills in demand and skills specific to another domain to solve problems in that domain.

Having completed this course, a student is prepared to work independently within a well structured design frame work in the job and for higher studies.

DEPARTMENT OF COMPUTER SCIENCE and ENGINEERING

Vision

To emerge as a leading department in Technical Education and Research in Computer Science and Engineering with focus to produce professionally competent and socially sensitive engineers capable of working in global environment.

Mission

I. To prepare Computer Science and Engineering graduates to be a life long learner with competence in basic science & engineering and professional core, multidisciplinary areas , with continuous update of the syllabus, so that they can succeed in industry as an individual and as a team or to pursue higher studies or to become an entrepreneur.

II. To enable the graduates to use modern tools, design and create novelty based products required for the society and communicate effectively with professional ethics.

III. To continuously engage in research and projects development with financial management to promote scientific temper in the graduates and attain sustainability.

Programme Educational Objectives

I Graduates will have a strong foundation in fundamentals of mathematics, science, computer science and basic engineering with abilities to analyze problems, design and development of optimal solutions to address societal problems.

II Apply knowledge of modern tools to solve the complex problems and enable graduates to be professionally competent engineers to sensitize towards societal, health, safety legal, environmental and sustainable issues by following the ethical ideologies and makes them globally employable.

III Ability to work effectively as an individual, team member or a leader or pursue entrepreneurial skills and be aware of gender sensitization with good communication, practice project and finance management skills.

IV Encouraging students to pursue higher studies in internationally reputed institutes thus making them life-long learners.

Programme Outcomes

The Programme Outcomes (**POs**) of the B.Tech (CSE) programme as stated by the NBA, India are listed below:

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

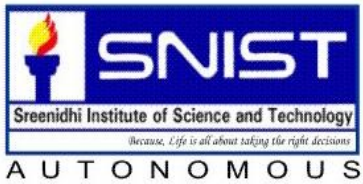
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Correlation between the POs and the PEOs

PEOs	Programme Outcomes											
	1	2	3	4	5	6	7	8	9	10	11	12
I	✓	✓	✓	✓								
II			✓	✓	✓	✓	✓	✓				
III									✓	✓	✓	
IV							✓					✓

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

(An Autonomous Institution approved by UGC and affiliated to JNTUH))
(Accredited by NAAC with 'A" Grade, Accredited by NBA of AICTE and Recipient of World Bank under TEQIP-I and II)
Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301.



Lab Manual

FOR

DATA WAREHOUSING AND DATA MINING

FOR

B.Tech - III year - I Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JULY 2020

1	2	3	4	5	6	7	8	9	10	11	12
			H	M							

Syllabus for B. Tech. III Year I semester
Computer Science and Engineering
Data Warehousing and Data Mining Lab

L	T	P	C
0	0	4	2

Code: 7E574

Course Objectives: Learn how to build a data warehouse and query it. Learn to perform data mining tasks using a data mining toolkit. Understand the data sets and data preprocessing. Demonstrate the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression. Exercise the data mining techniques with varied input values for different parameters. To obtain Practical Experience Working with all real data sets. Emphasize hands-on experience working with all real data sets.

Course outcomes:
At the end of this course the student will be able to

1. Ability to work with the ETL and Mining tools.
2. Demonstrate the classification, clustering techniques on the data sets.
3. Comprehend the results obtained in the clustering, Association and Classification techniques applied on the data sets with varied input parameters.
4. Ability to apply mining techniques for realistic data.

Exercices

1. Build a Data Warehouse to perform filter transformation for the employee database.
2. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.
3. Using Aggregator transformation display the average salary of employees in each departments.
4. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.
5. How to load top 2 salaries for each department without using Rank Transformation and SQL queries in Source Qualifier.
6. Implement the following Multidimensional Data Models

i.Star Schema

ii.Snowflake Schema

iii.Fact Constellation.

7. Compare the GRI and Apriori usage (Prepare a sample data set in Spread Sheet).
8. Determine the Drugs importance w.r.t. Age, Cholestrol and BP using C 5.0.
9. Predict the accuracy of the test data set using Neural Net model using a Case Study of Botanical data set.
10. Compare the C 5.0 and Neural Net using the sample data.
11. Using BASKETS1n dataset select the data as given below.
    a) Customer age < 35 and count the customers who buy dairy and VEG products
    b) Find the AVG income of customers who buy atleast 5 products
12. Using BASKETS1n dataset select the data as given below.

a) Derive the field whose homeown is 'YES' and Age > 30 and sort data w.r.t. income in Ascending order, and output only the item fields.

b) Find the mean value of salary w.r.t age={Young, Middle, Senior}.

Exercises/ Case Studies:

WEEK 1 : Exploring and understanding Informatica tool

WEEK 2 :

1. Build a Data Warehouse to perform filter transformation for the employee database.

WEEK 3:

1. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.

1 Using Aggregator transformation display the average salary of employees in each departments.
2 Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.

WEEK 4:

1. Using Aggregator transformation display the average salary of employees in each departments.
2. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table

WEEK 5: Exploring understanding Clementine Tool

WEEK 6:

1. Compare the GRI and Apriori usage (Prepare a sample data set in Spread Sheet)
2. Determine the Drugs importance w.r.t. Age, Cholestrol and BP using C 5.0

WEEK 7:

1. Predict the accuracy of the test data set using Neural Net model using a Case Study of Botanical data set.
2. Compare the C 5.0 and Neural Net using the sample data.

WEEK 8:

1. Using BASKETS1n dataset select the data as given below

   a) Customer age < 35 and count the customers who buy dairy and VEG products

   b) Find the AVG income of customers who buy atleast 5 products

   c) Derive the field whose homeown is 'YES' and Age > 30 and sort data w.r.t. income in Ascending order, and output only the item fields.

INTRODUCTION TO DATA MINING

Data Mining is a general term, which describes a number of techniques used to identify pieces of information or decision-making knowledge in data. A common misconception is that it involves passing huge amounts of data through intelligent techonologies that find patterns and give solutions to business problems.

Data Mining is an interactive and iterative process.

Many of the techniques used in Data Mining are referred to as "machine learning" or "modeling". Historical data are used to generate models, which can be applied at a later date to areas such as prediction, forecasting, estimation and decision support.

The data mining process model recommended for use with Clementine is the Cross-Industry Standard Process for Data Mining (CRISP-DM). The first version of the CRISP-DM process model is now available. It is included with Clementine and can be downloaded from www.crisp-dm.org.

WEEK 1 : Exploring and understanding Informatica tool

PREREQUISITES OF DATA WAREHOUSING USING INFORMATICA POWER CENTER EXPRESS (PERSONAL EDITION) TOOL

I. Installing the Tool:

Make sure that you have the latest version of Java installed on your system and the PATH variable set in the System Environment Variables sub-section of Advanced System Settings of your computer. Then install Informatica PowerCenter Express (Personal Edition) version 9.6.1 as per the instructions given by the manual. Remember the username and password that you entered for the server while installation as these settings will be required for the tool to work properly.

II. Setting up the Data Source Name (DSN):

A **Data Source Name** (DSN) is the logical **name** that is used by Open Database Connectivity (ODBC) to refer to the drive and other information that is required to access **data**. The **name** is use by Internet Information Services for a connection to an ODBC **data source**, such as a Microsoft SQL Server database or in this case **Oracle Database 10g Express Edition.**

The steps to configure a Data Source are as follows:

1. Type in **ODBC Data Sources** into the Search Bar and run it with Administrator privileges.

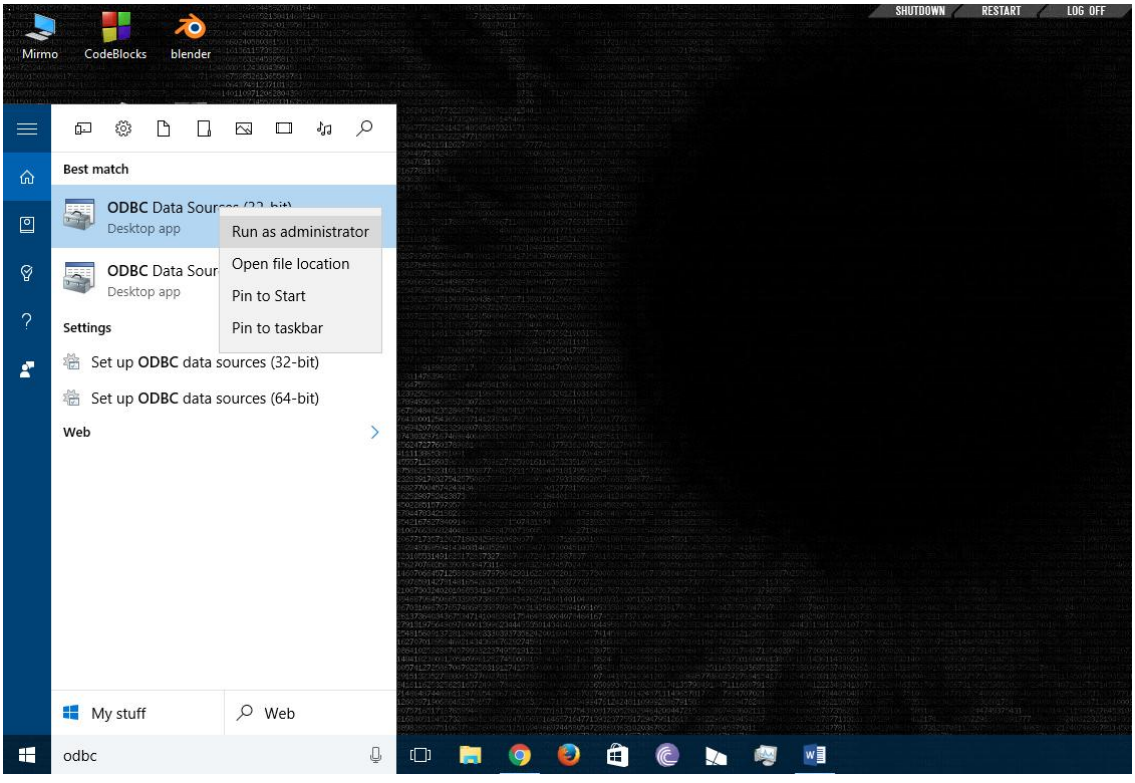


Fig 2.1: Step 2.1

2. Click on **Add** button to add a DSN and select **Oracle in XE** from the list that appears and then select **Finish**.

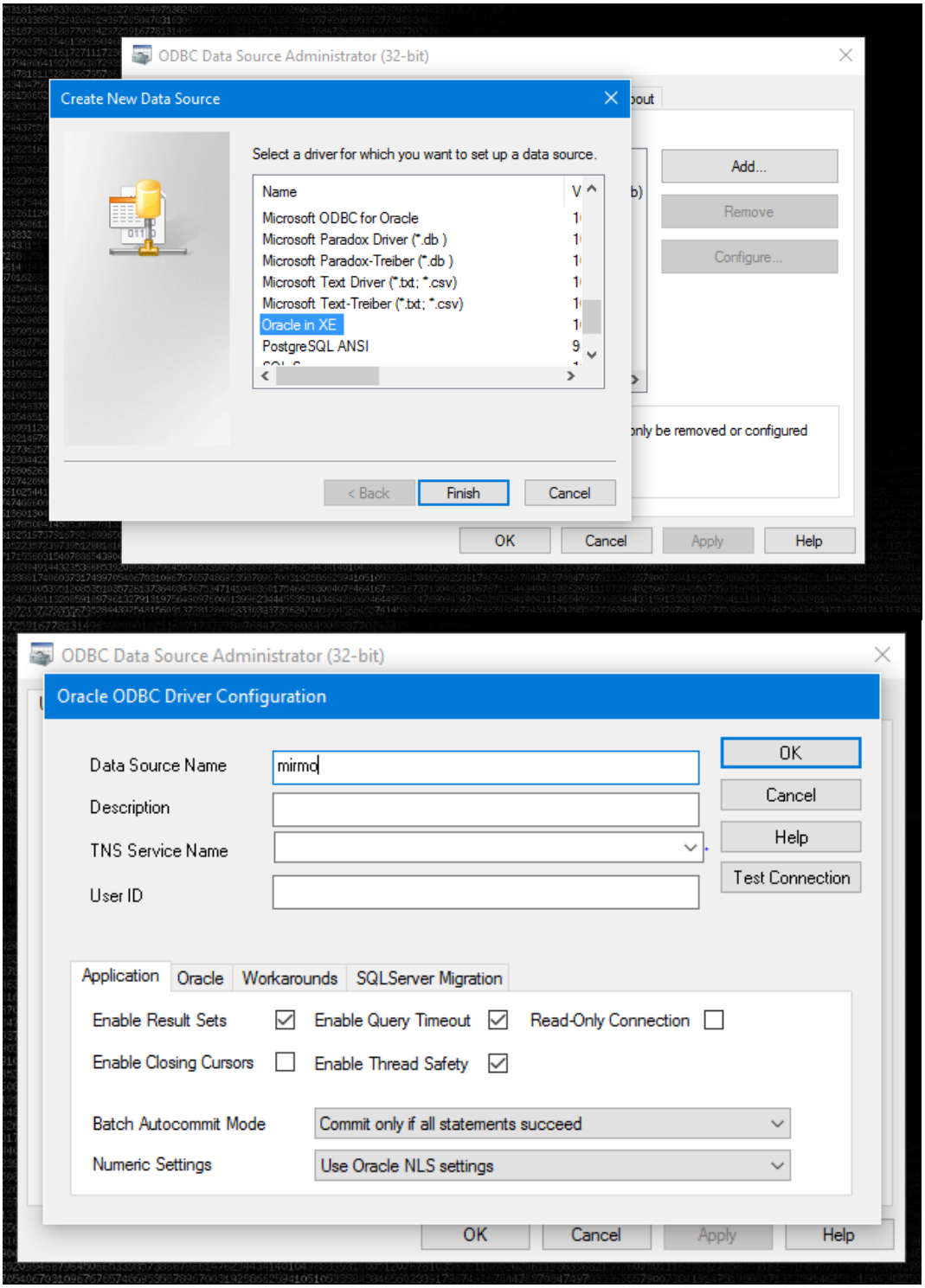
3. 

Fig 2.3: Step 2.3

Now you have created a new DSN for use in the tool.

III. Starting Informatica PowerCenter Express

Step 1:After booting the system up, go to Start Menu and run **Start Informatica Services** with Administrator privileges.

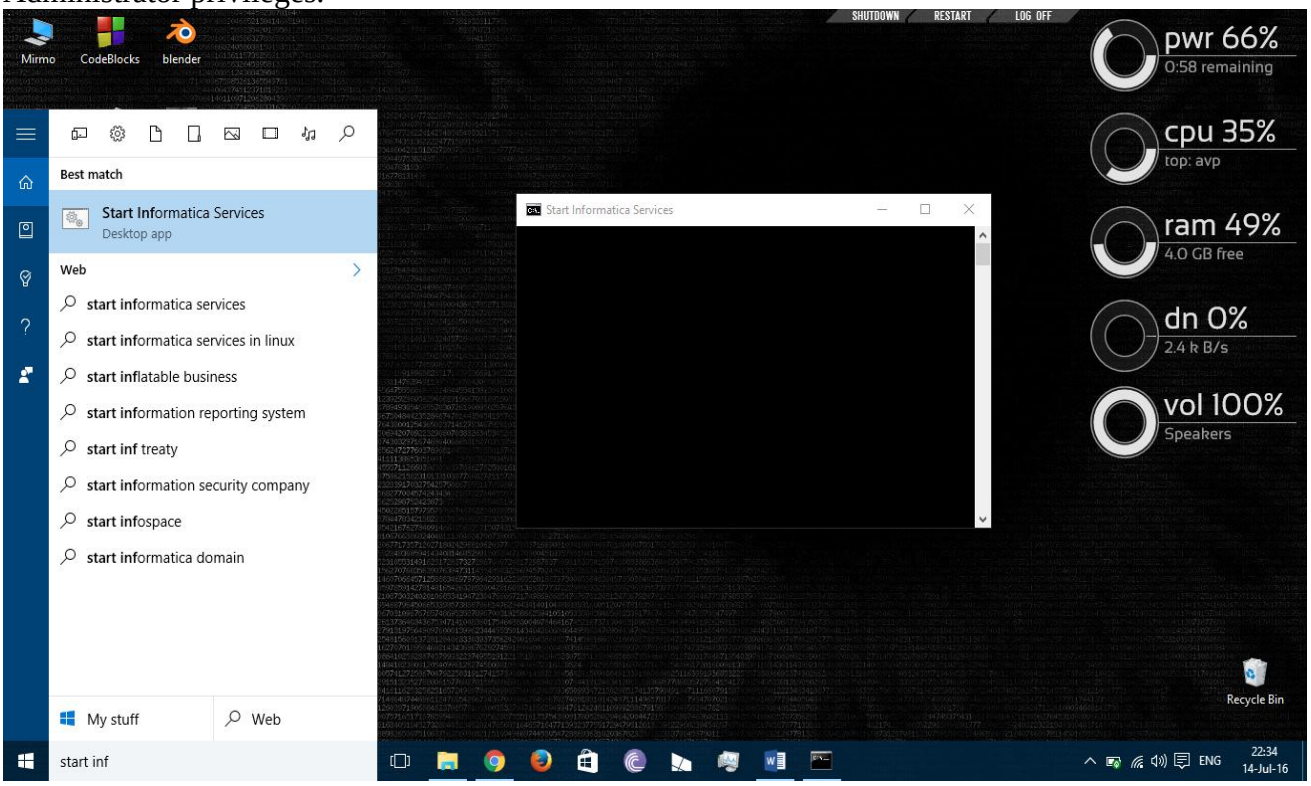


Fig. 3.1: Step 3.1

Step 2:Once the batch file runs its course, navigate to **localhost:7009** in your favourite browser. This is the login page for the Administrator Server. Login with the credentials that were supplied during the install process.

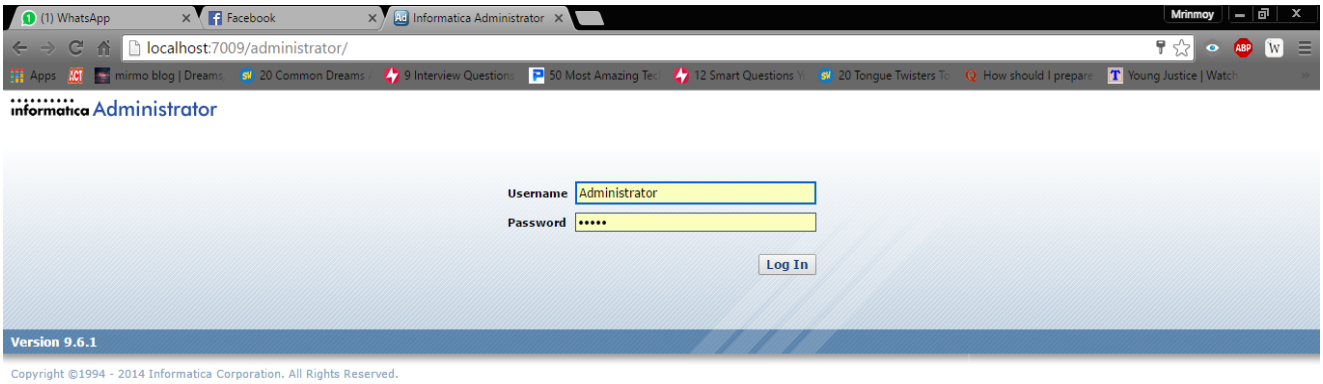


Fig 3.2: Step 3.2

*Step 3:*Once the server page starts up, we are ready to **Launch Informatica Developer** from the Search Bar.

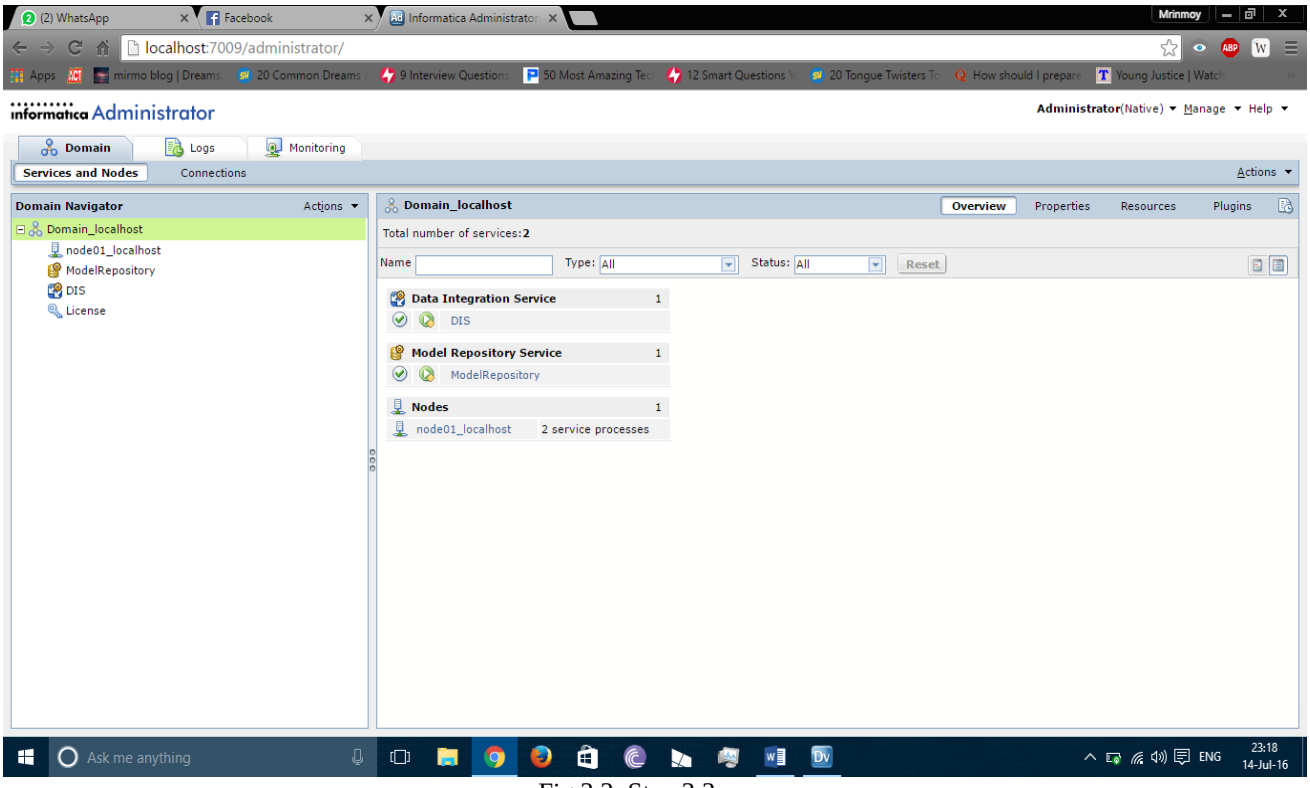


Fig 3.3: Step 3.3

*Step 4:*Once Informatica Developer loads up, connect to Model Repository using the credentials as specified in the server and you are good to go.

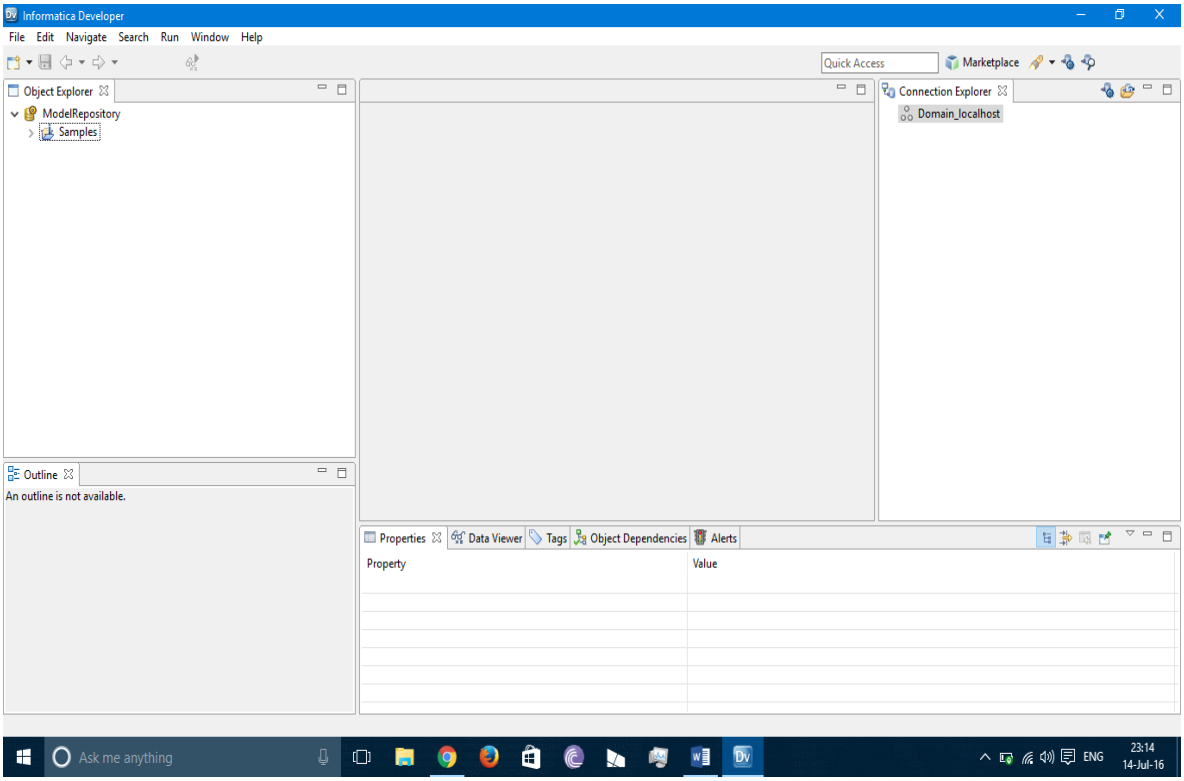
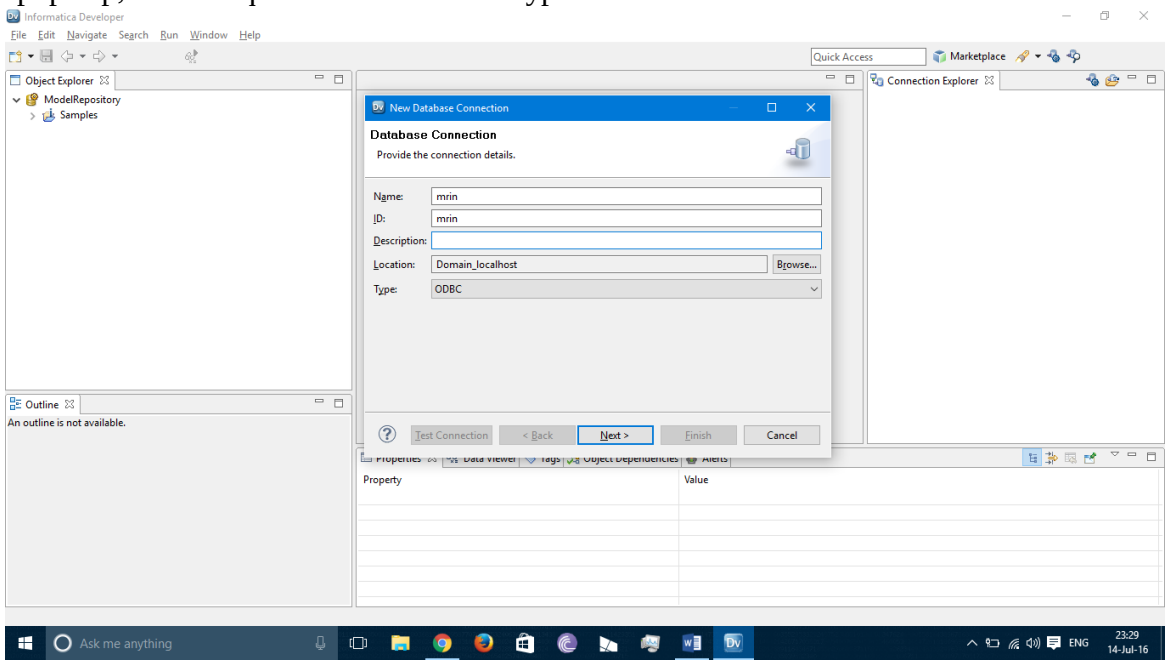


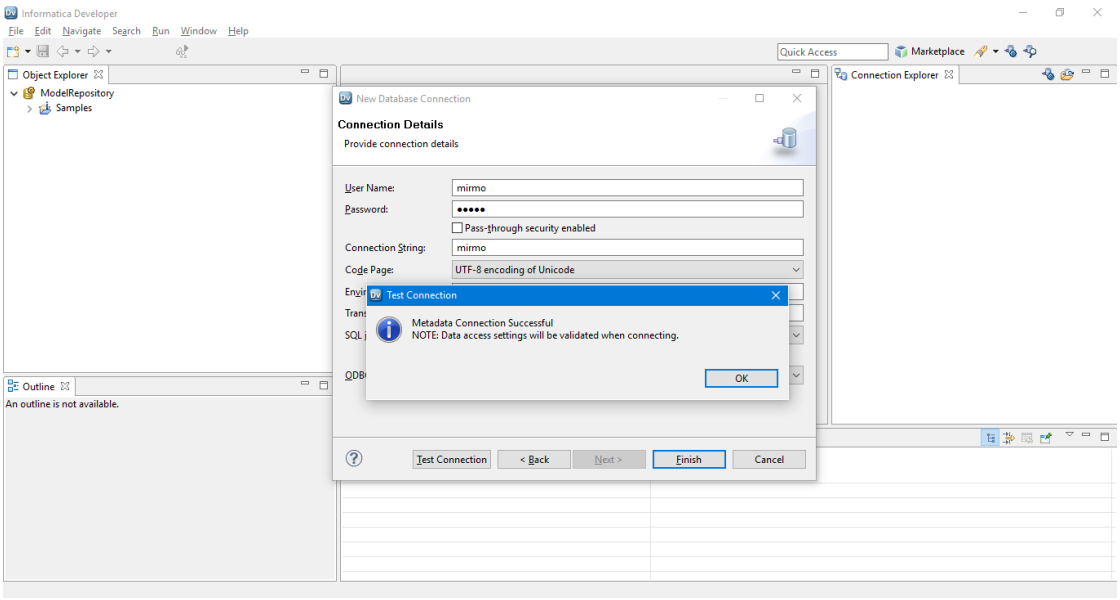
Fig 3.4: Step 3.4

IV. Connecting to Data Source

*Step 1:*Click on the **Create Connection** Icon in the **Connection Explorer.** In the dialog window that pops up, set a unique name and set the type as **ODBC**



Step 2: In the following dialog box that appears, set the **Username** and **Password** of your database. Set the **Connection String** to the name given in the DSN. This will connect the Tool using ODBC Drivers to your Oracle Database. Click on **Test Connection** to verify connection status.



Step 3: If the Connection Test is successful, click on **Finish**. You can now start using the resources that are available in the database.
Note: Unless there is a necessity to connect to another database, the same connection can be reused. Hence these steps are required to be performed only once.

Informatica Tool

Week-2:

1. Build a Data Warehouse to perform filter transformation for the employee database.
2. Add the commission of 1000 Rs in the Salary field of Employee table using Expression Transformation.

Filter transformation

• Active Transformation

• Used to filter the data or reduce the amount of data.

• Single condition can be applied at a time.

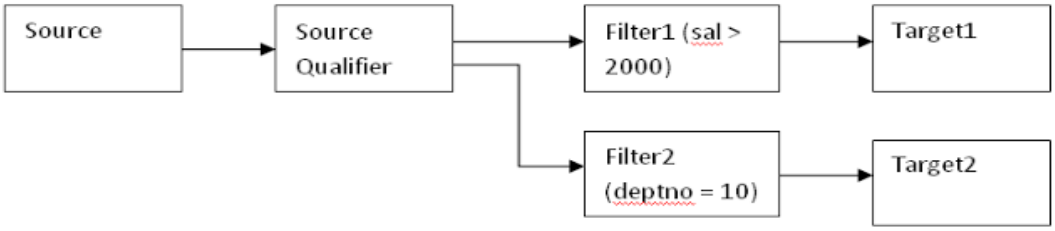
• Filter transformation accepts only two types of data - Boolean (True or False)

• Filter accepts condition satisfied data.

• The false data is rejected or dropped to the bad files. **(Drawback)**

Scenario 1: More than on targets.

We can't combine multiple conditions in one filter



Scenario 2: Only one target.

Here we can combine multiple conditions in one filter using logical operators (AND, OR)

Steps to use Filter Transformation in Informatica

Define the Source (EMP)

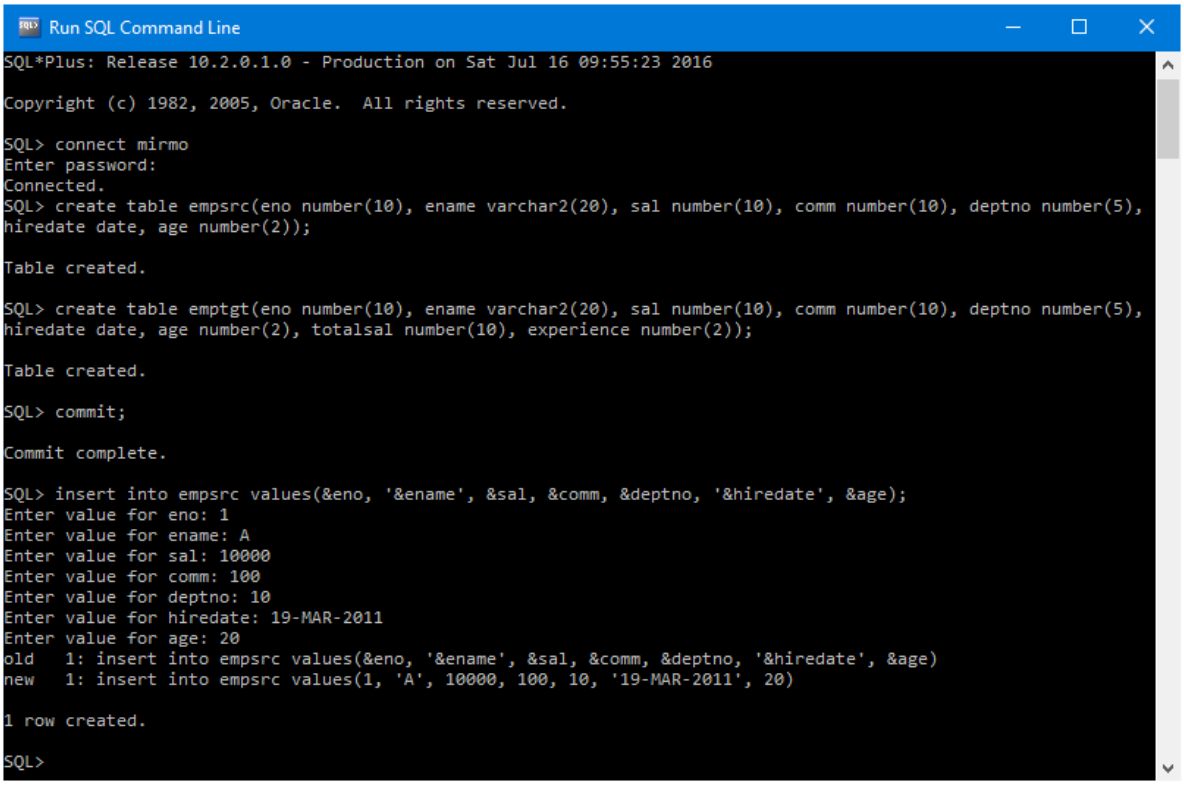
Define the Target (DIM_EMP)

Aim: To apply Filter Transformation to a Data Set and filter out tuples matching a certain criterion.

Theory: The filter transformation takes one table as an input and writes into another table all the values that match a certain criterion. The tuples which do not meet the criteria are left as they are.

Procedure:

Step 1: Create two tables. The first table is the source table for all the employee information and populate it with values and the second table is an empty table which will be the target table for the filter transformation. Commit when done.

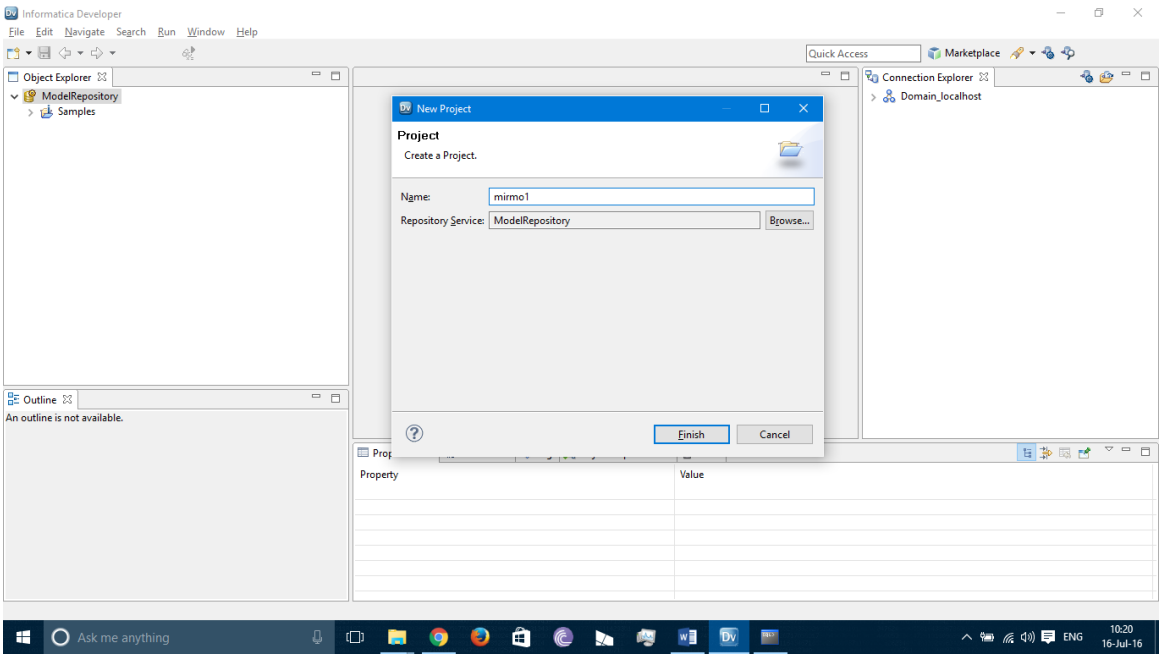


Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.

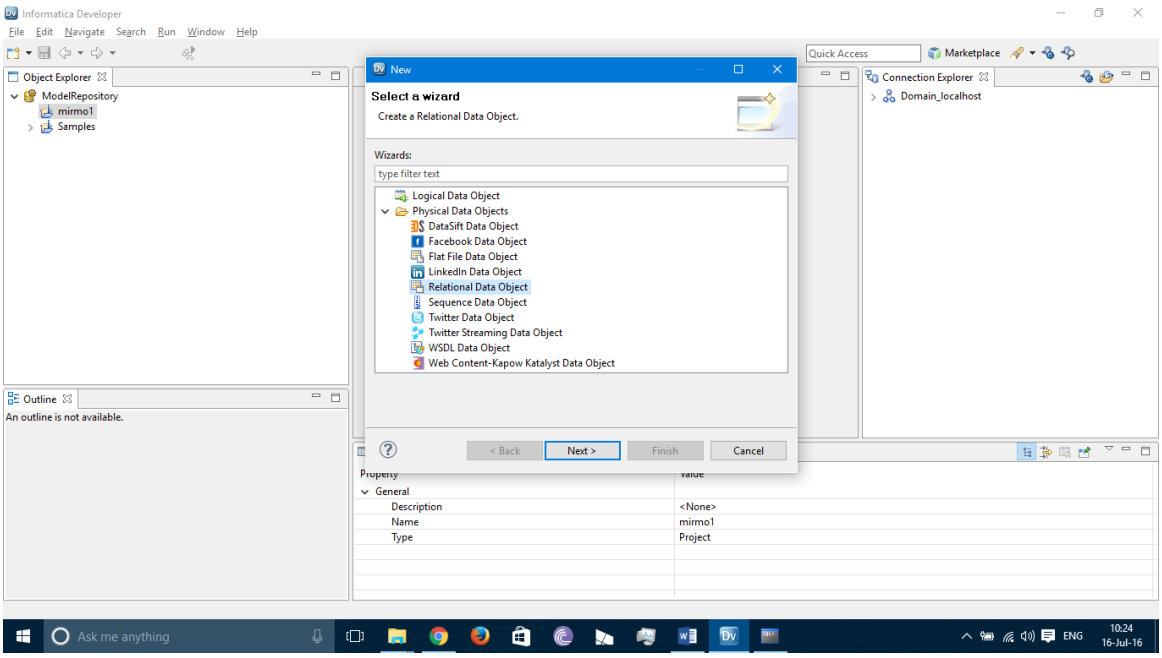
Step 3: Double click on Model Repository under **Object Explorer** to connect it to the Server and also on the DSN under Domain_localhost tab of the **Connection Explorer** to establish connection to the Database.

Step 4: Right click on **Model Repository**. Select **New ➔Project**. Enter the project name (mirmo1) and then click on **Finish**. A new project is created.

Step 5: Right click on mirmo1, select **New ➔ Data Object.** From the dialog box that pops up, select Relational Data Object because we are working with Relational Data (Oracle DB). Then click **Next.**

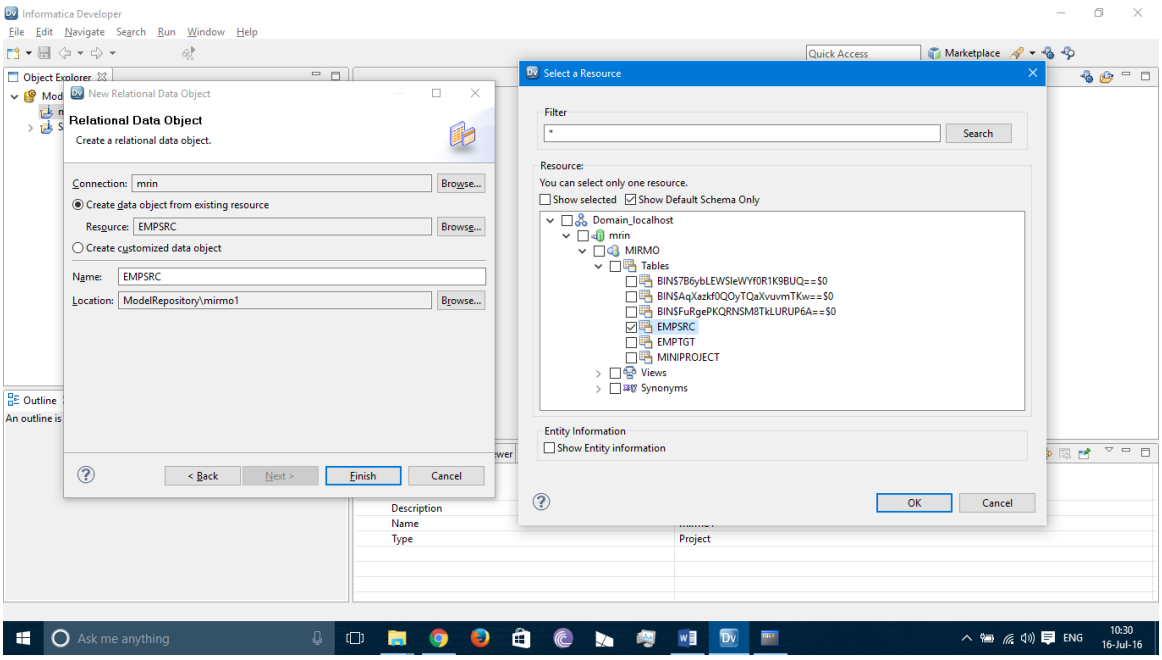


Step 4



Step 5

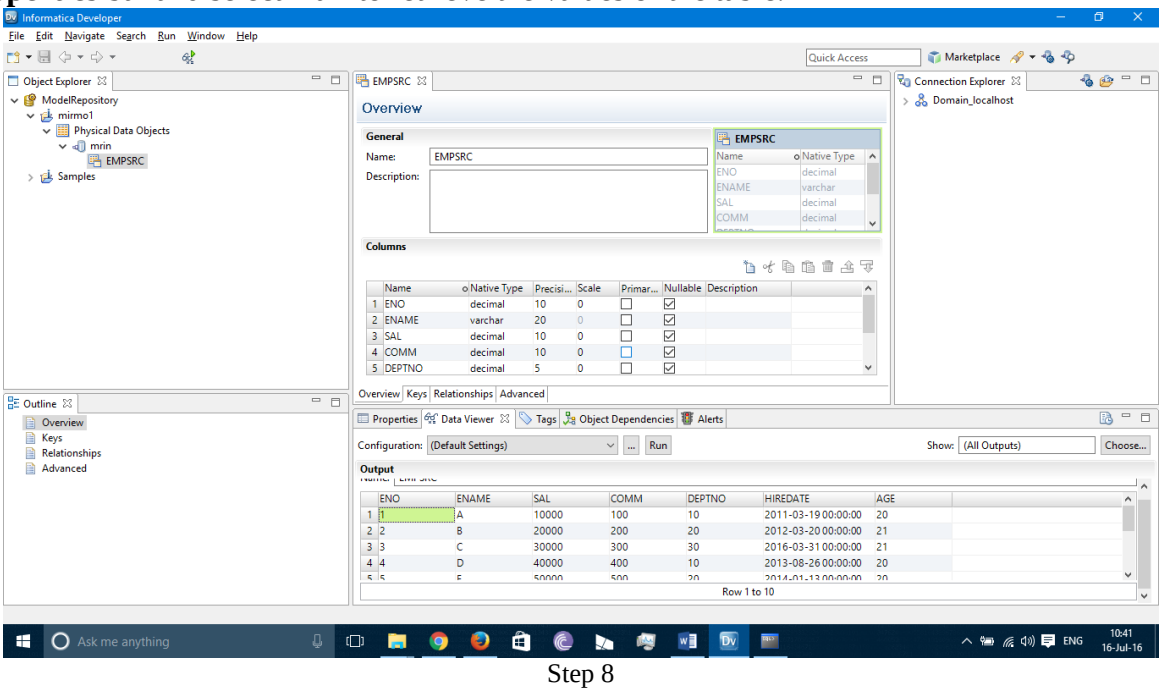
Step 6: In the following dialog box that appears, set the **Connection** to mrin (created earlier). Make sure that **Create data object from existing source** is selected and browse to the table you want to import. Click on **OK** and then click on **Finish.**



Step 6

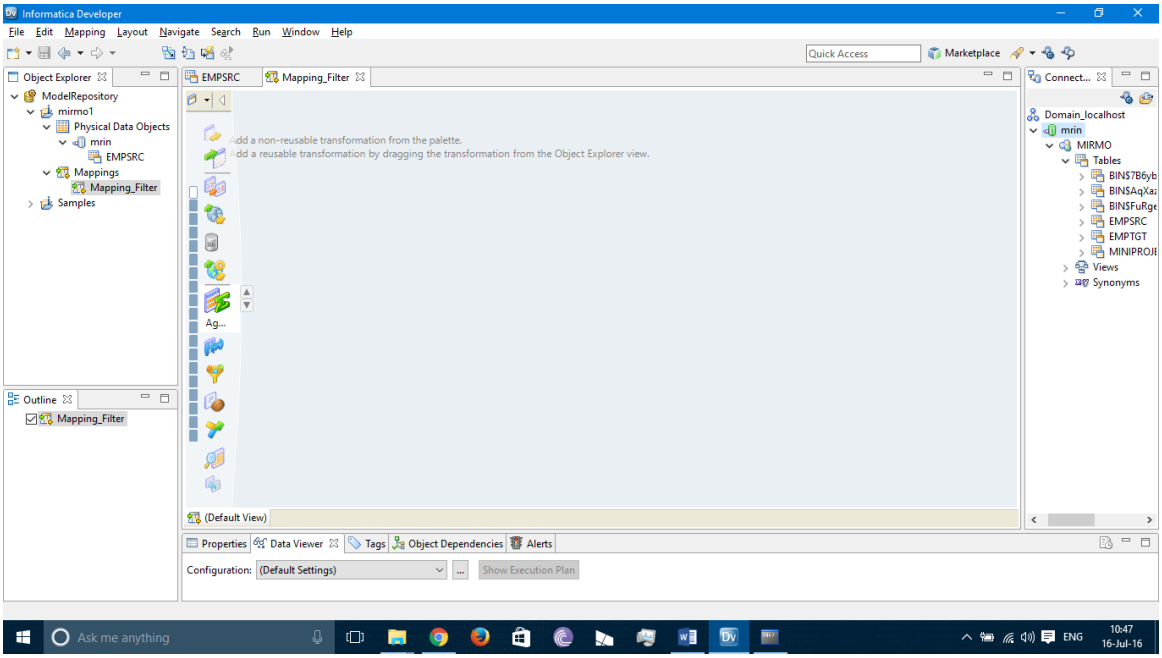
Step 7: If you want to import any more tables, right click on **Physical Data Objects➔New ➔ Physical Data Object** under mirmo1 and go through step 5 and 6 to add tables.

Step 8: Select EMPSRC table in the Object Explorer. Click on the **Data Viewer** tab on the **Properties** bar and select **Run** to retrieve the values of the table.



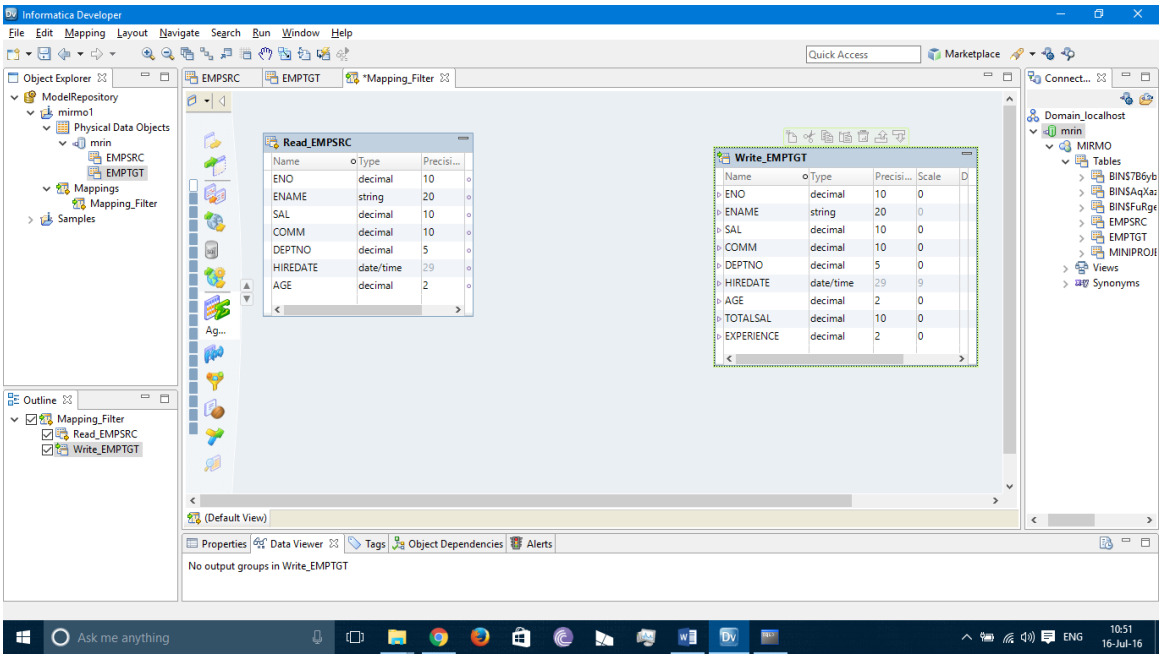
Step 8

Step 9: This step is to create a new mapping for performing transformations. Right Click on mirmo1 **➔New ➔ Mapping.** Name it as **Mapping_Filter** and click on **Finish.**



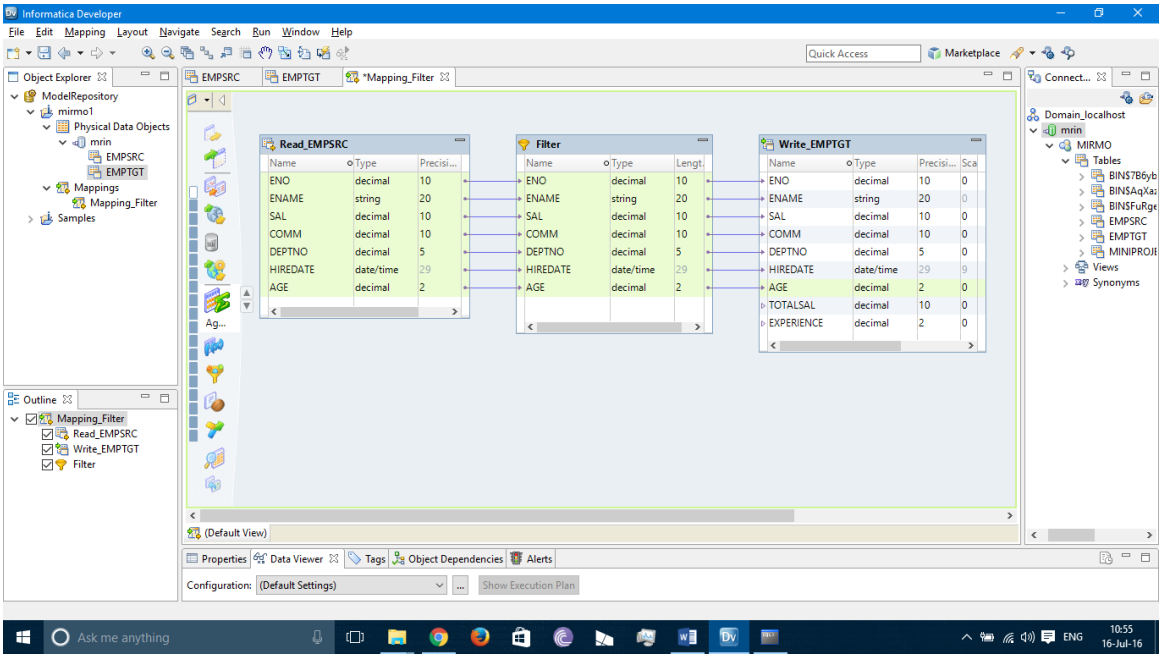
Step 9

Step 10: Drag EMPSRC table from Object Explorer into the Mapping_Filter tab. Set **Physical Data Object Access** to **Read** as we use this table only to take inputs. Then drag EMPTGT table and set Physical **Data Object Access** to **Write** as we use this table only to store the output of the filter transformation.



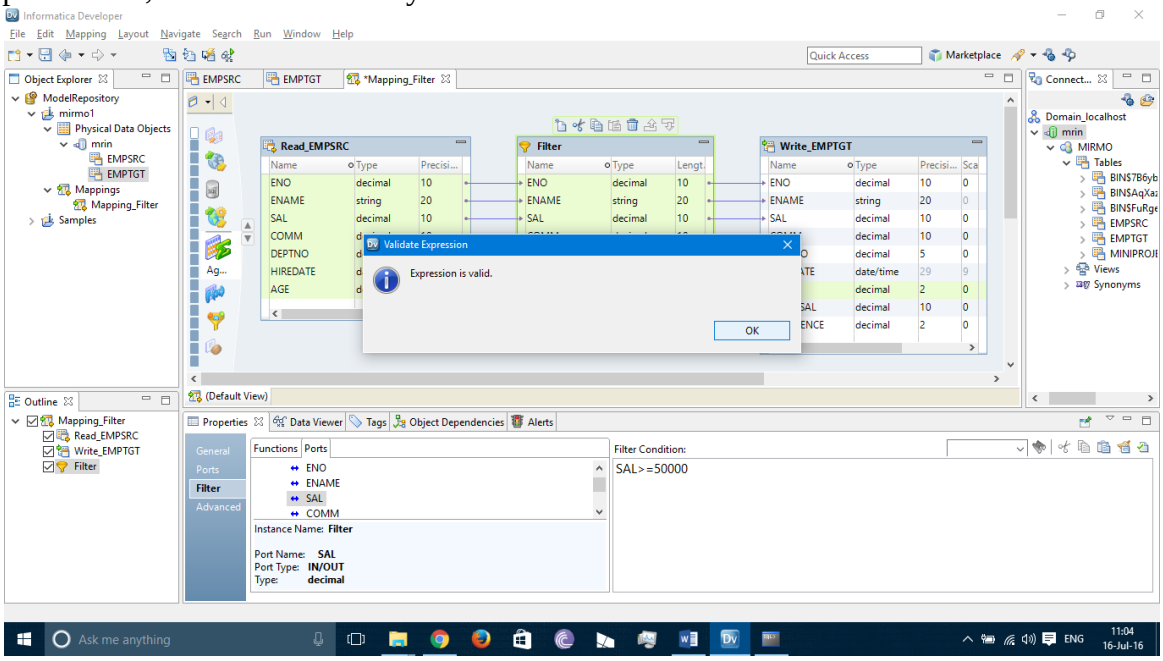
Step 10

Step 11: Select **Filter** transformation from the Palette and drag it onto the Mapping Tab. Select all entries from **Read_EMPSRC** and drop it onto the Filter. Now select all entries of **Filter** and drop it onto **Write_EMPTGT.** This creates a connection between the three. Make sure the entries correspond to each other.



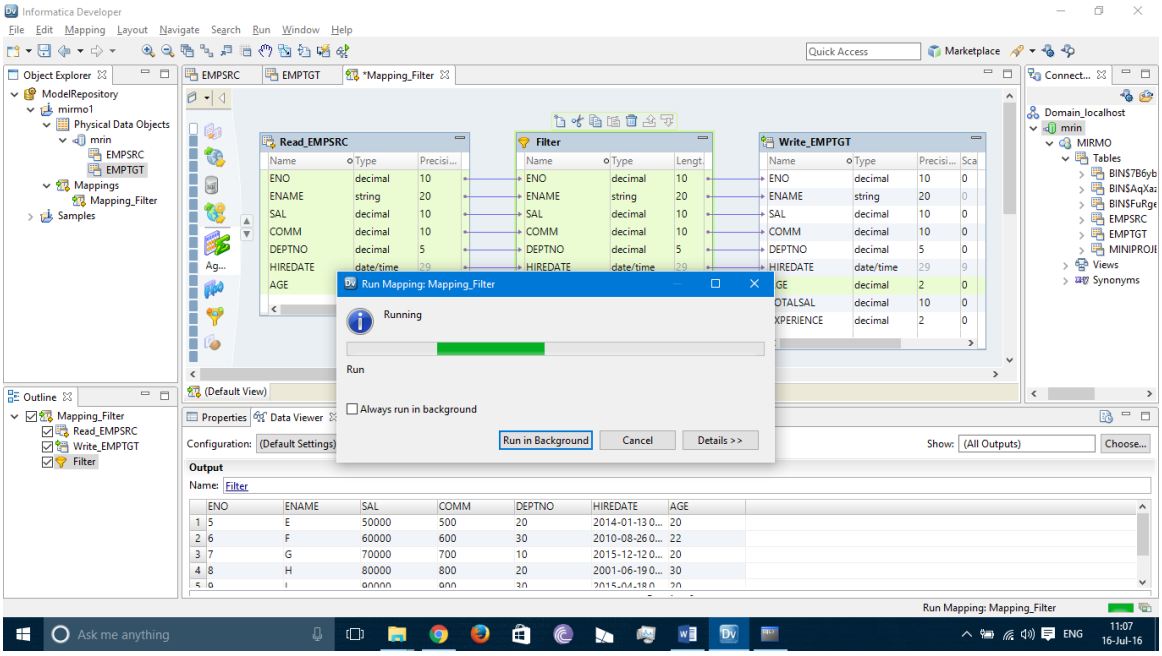
Step 11

Step 12: Make sure that **Filter** is selected. Then navigate to **Properties ➔ Filter** and specify the condition in the **Filter Condition** box. Click on **Validate Expression** at the right corner of the Properties Tab, to check the validity of the Filter Condition.



Step 12

Step 13: Click on the **Data Viewer** Tab of the Properties Explorer and select Run. If the Transformation is successful, then all entries matching the condition are displayed. Then click on **Run** in the menu bar and select **Run Mapping.**

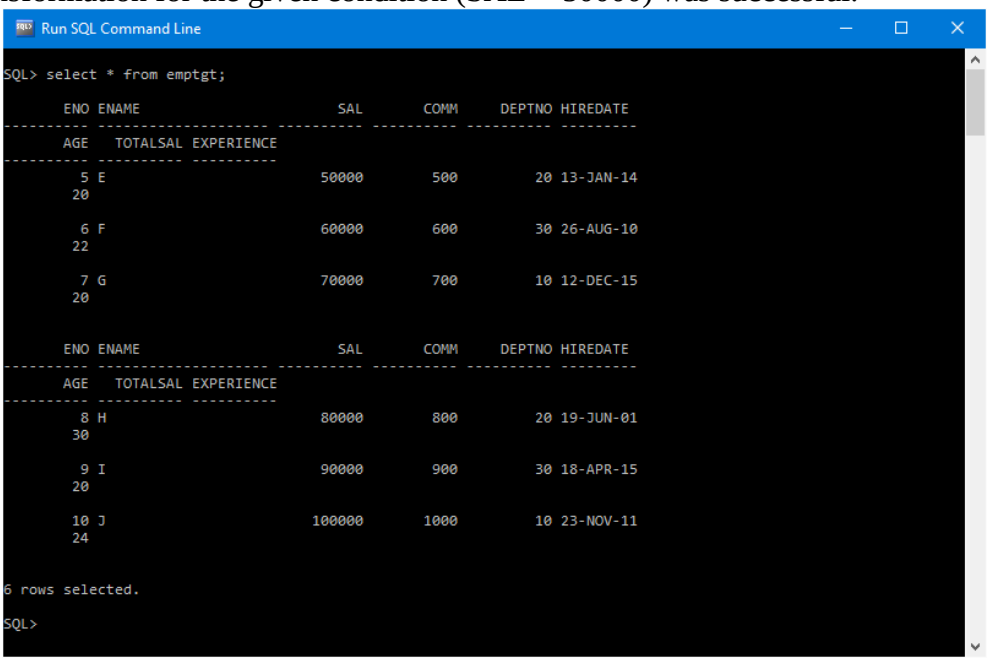


Step 13

Step 14: Verify the output of the Filter Transformation in SQL Command Prompt.

Output:

Filter Transformation for the given condition (SAL>=50000) was successful.



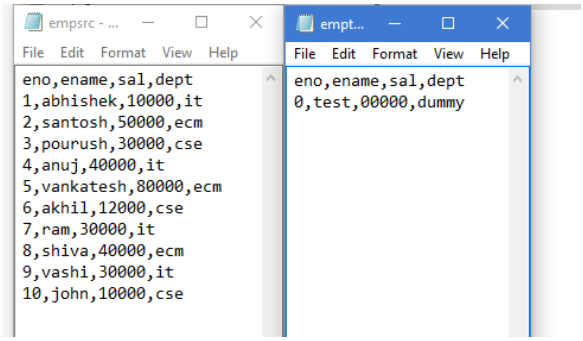
Output

Aim: To apply Filter Transformation to a Data Set and filter out tuples matching a certain criterion.

Theory: The filter transformation takes one table as an input and writes into another table all the values that match a certain criterion. The tuples which do not meet the criteria are left as they are.

Procedure:

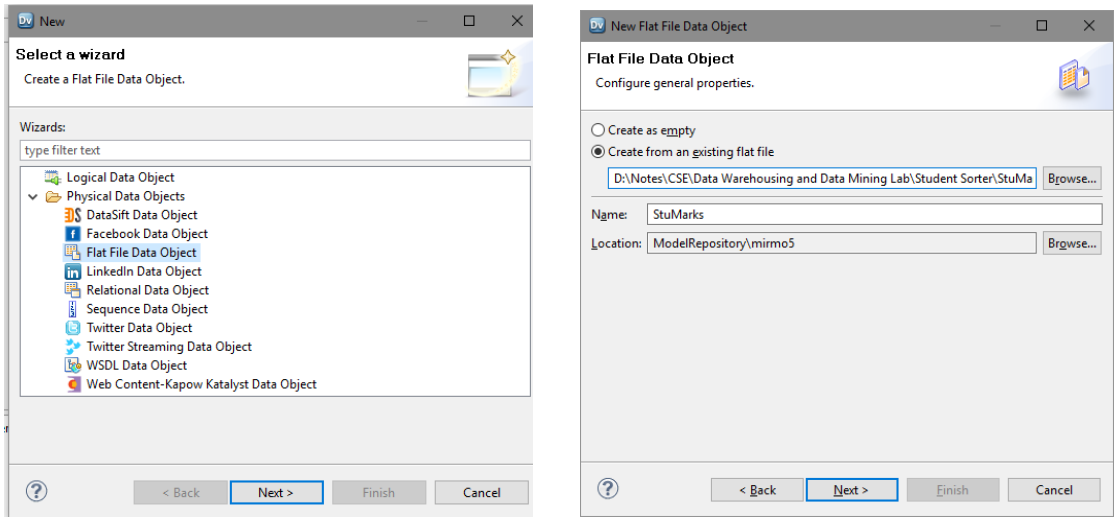
Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ',' as a delimiter.



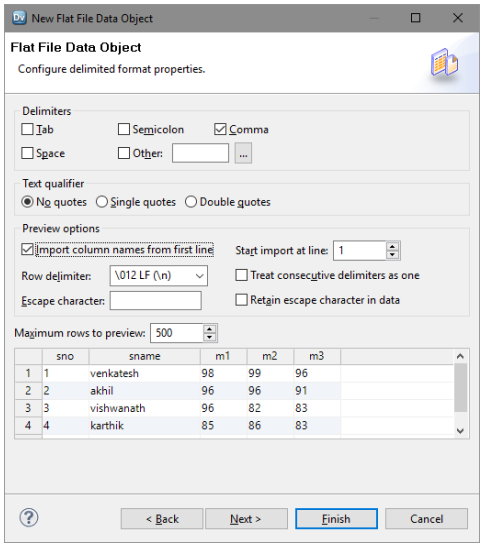
Step 1

Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Import your flat files (both source and target files) into **mirmo1** project. Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.

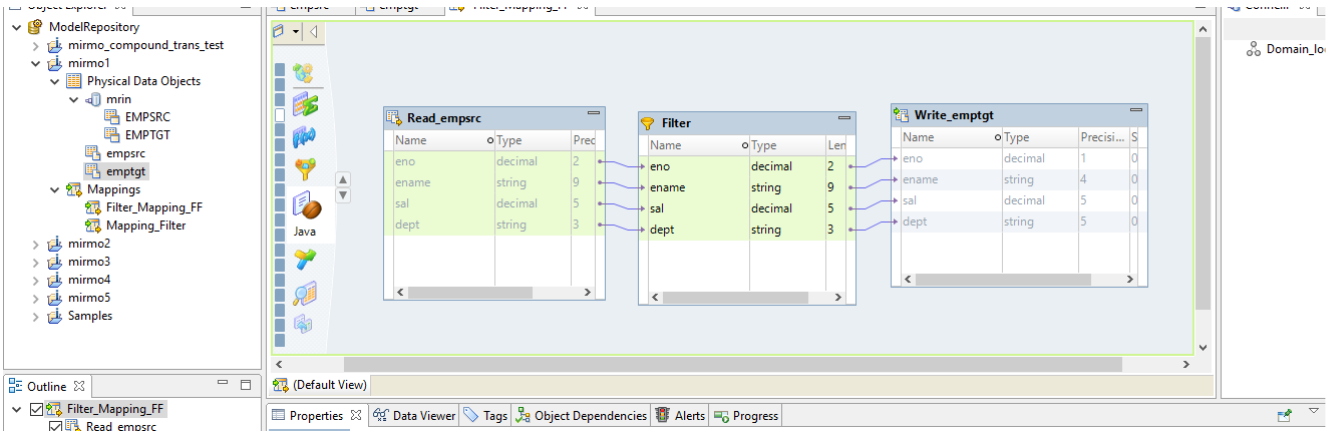


Step 3



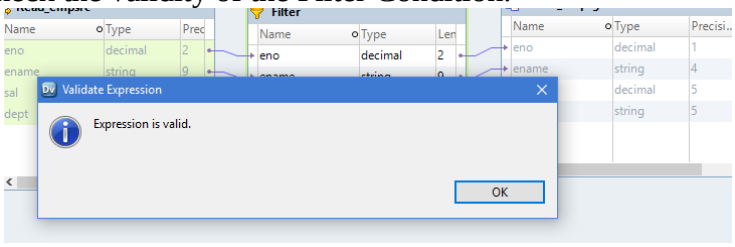
Step 3

Step 4: Once the files have been imported, create a new mapping and name it **Filter_Mapping**.
Step 5: Import the source file with read mode and target file as write mode and drag and drop the Filter Transformation from the Palette into the workspace. Give the necessary connections.



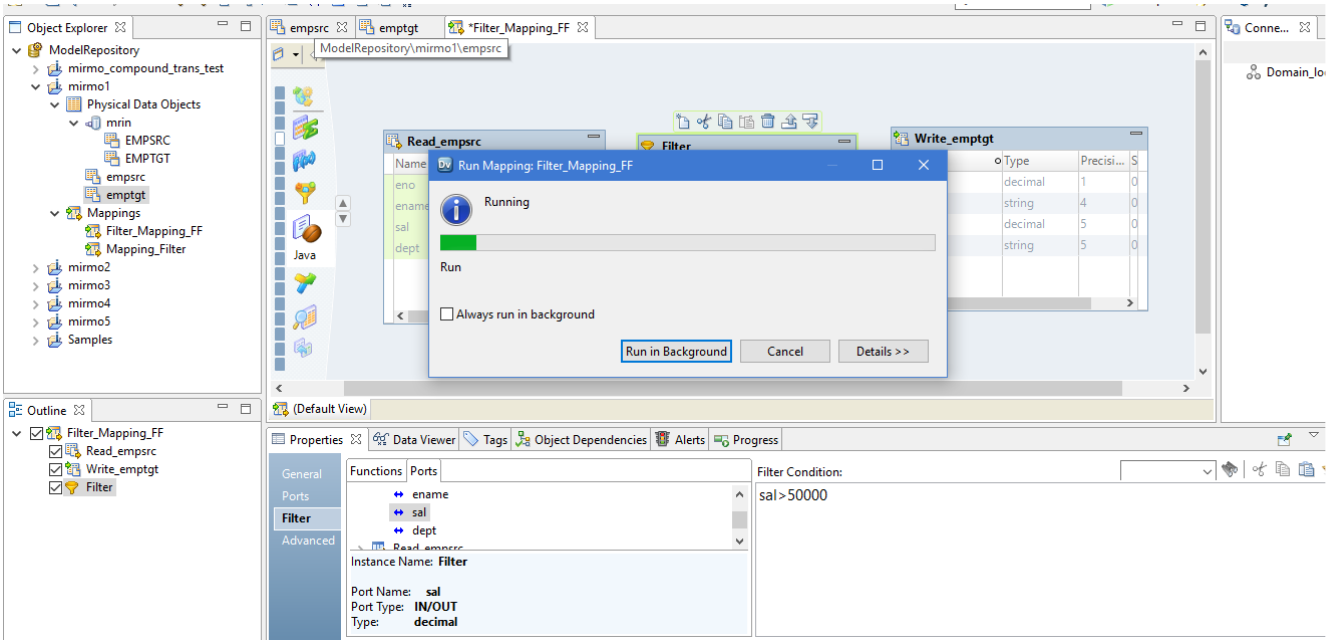
Step 5

Step 6: Make sure that **Filter** is selected. Then navigate to **Properties ➔ Filter** and specify the condition in the **Filter Condition** box. Click on **Validate Expression** at the right corner of the Properties Tab, to check the validity of the Filter Condition.



Step 6

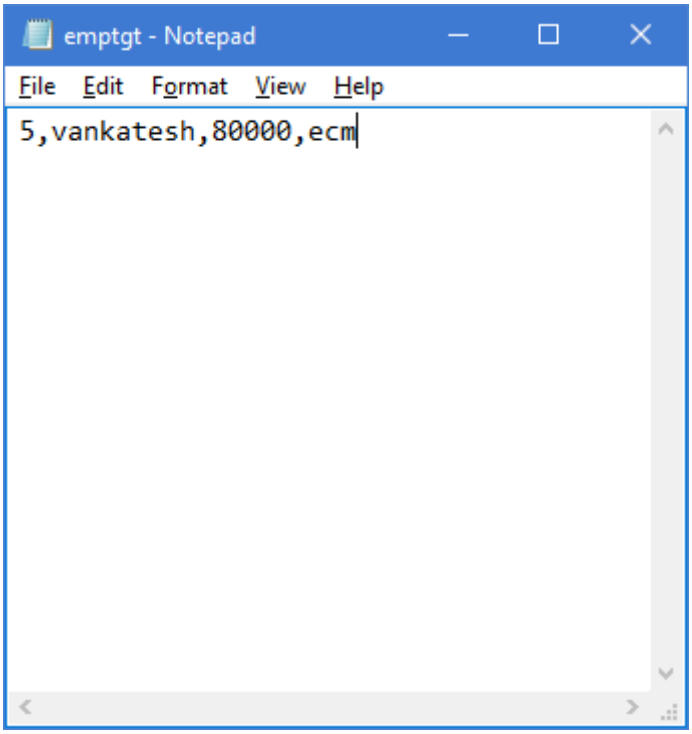
Step 7: Click on the **Data Viewer** Tab of the Properties Explorer and select Run. If the Transformation is successful, then all entries matching the condition are displayed. Then click on **Run** in the menu bar and select **Run Mapping.**



Step 7

Step 8: Make sure that the connections are intact. Now Run the Mapping. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Output

Filter Transformation for the given condition (SAL>50000) was successful.

EXPRESSION TRANSFORMATION:

Aim: To apply an Expression Transformation on a Data Set to evaluate two columns into an output column in the target table.

Theory: Expression transformation is a connected, passive transformation used to calculate values on a single row. Examples of calculations are concatenating the first and last name, adjusting the employee salaries, converting strings to date etc. Expression transformation can also be used to test conditional statements before passing the data to other transformations.

Procedure:

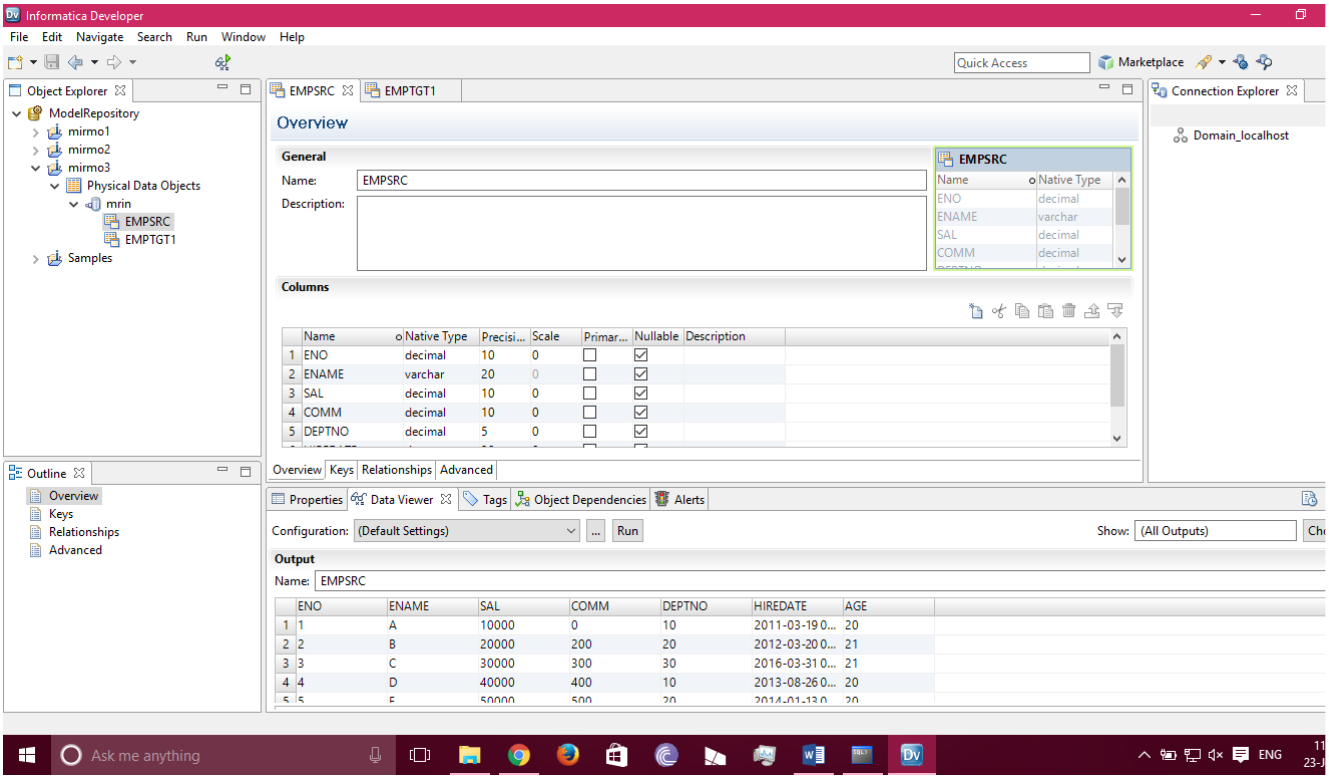
*Step 1:*Create the required tables and populate the source table with values. Make sure that the fields you perform the Expression Transformation on, are of the type Integer/Number as we will be adding two columns and storing the value in another column in the target table.

```
SQL> desc source;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ENO                                                NUMBER(10)
 ENAME                                              VARCHAR2(20)
 SAL                                                NUMBER(10)
 COMM                                               NUMBER(10)
 DEPTNO                                             NUMBER(5)
 HIREDATE                                           DATE
 AGE                                                NUMBER(2)

SQL> desc emptgt1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ENO                                                NUMBER(10)
 ENAME                                              VARCHAR2(20)
 SAL                                                NUMBER(10)
 COMM                                               NUMBER(10)
 DEPTNO                                             NUMBER(5)
 HIREDATE                                           DATE
 AGE                                                NUMBER(2)
 TOTALSAL                                           NUMBER(10)
 EXPERIENCE                                         NUMBER(2)
```

Step 1

Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.

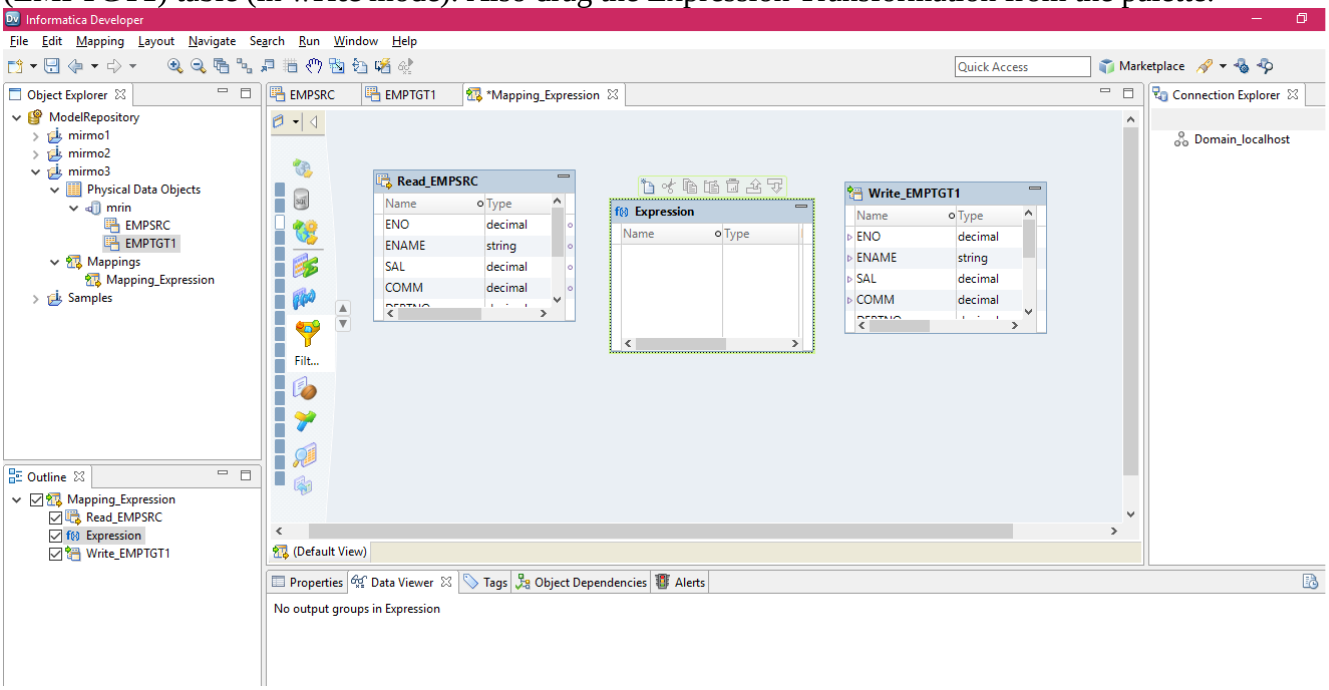
Step 3: Once Informatica Developer has started, create a **new project** (mirmo3) and import all the required tables as done in the previous experiments.



Step 3

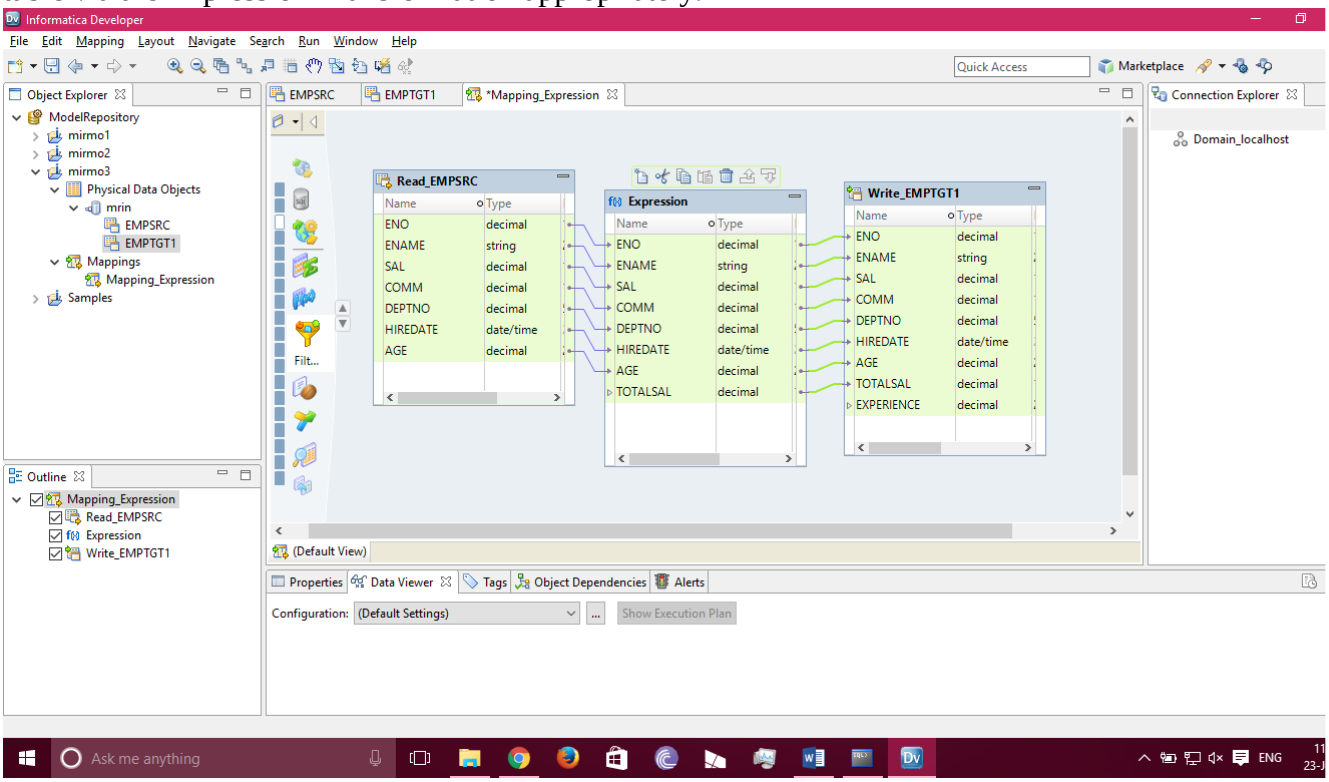
Step 4: Create a new mapping and name it Mapping_Expression.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (EMPTGT1) table (in **write** mode). Also drag the Expression Transformation from the palette.



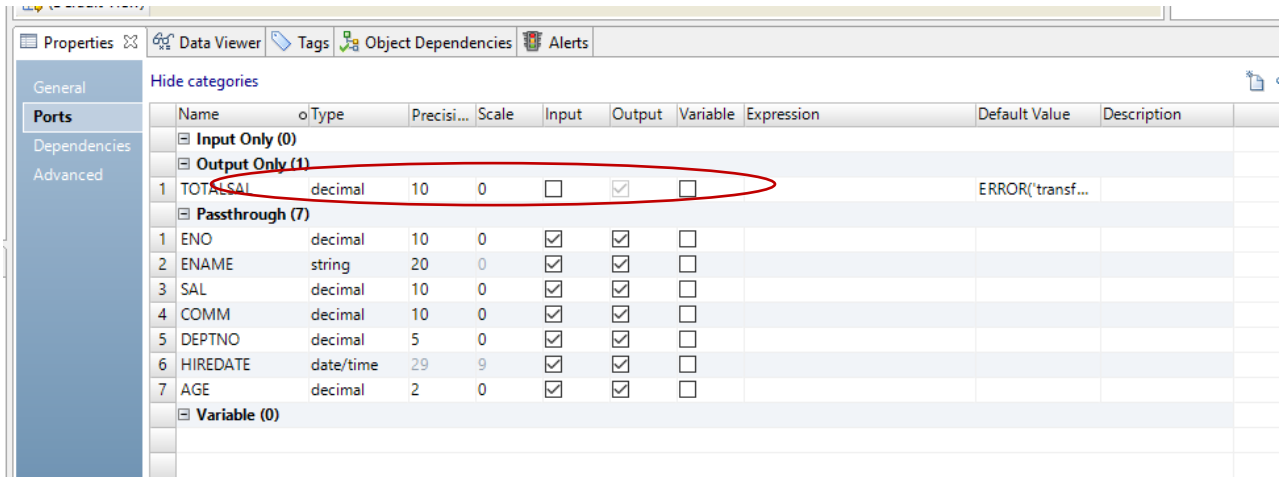
Step 5

Step 6: Drag and drop the values of the target table into the **Expression Transformation**. Delete the **Experience**entry as it is not required for this task. Also link up the input table and the output table via the Expression Transformation appropriately.



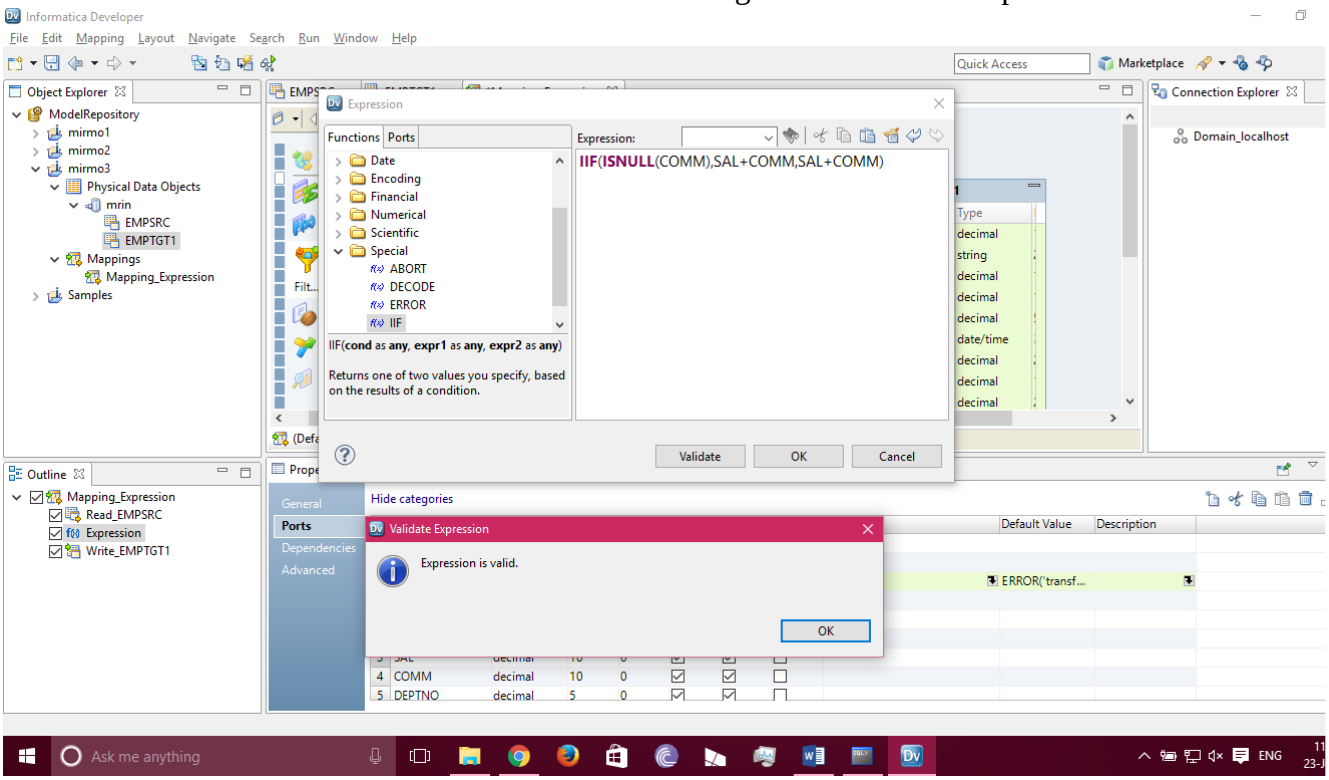
Step 6

Step 7: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **TOTALSAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



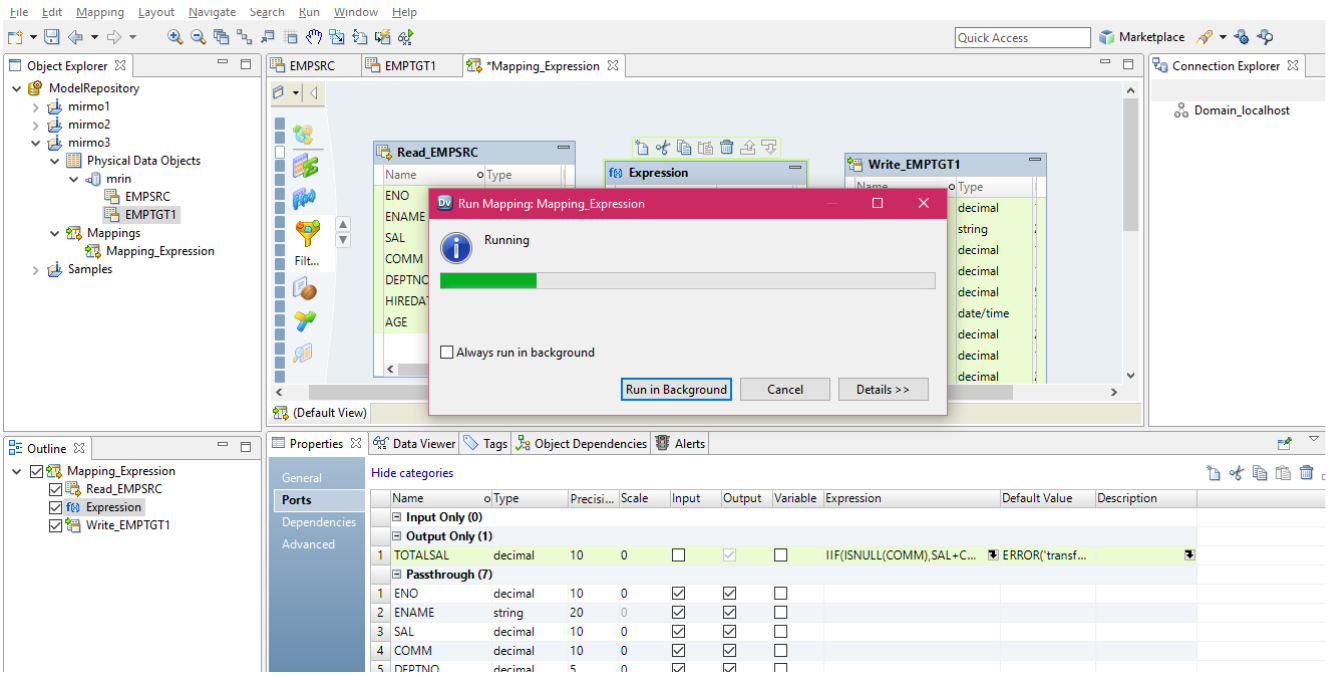
Step 7

Step 8: In the Expression Section of the entry, click on the small arrow head to open up the **Expression Dialog Box**and enter the following expression: "**IIF(ISNULL(COMM),SAL+COMM,SAL+COMM)"** This expression checks if the Commission field is NULL, then returns only salary as the result, else returns Salary + Commission as result to the TOTALSAL field. Don't forget to validate the expression for errors.



Step 8

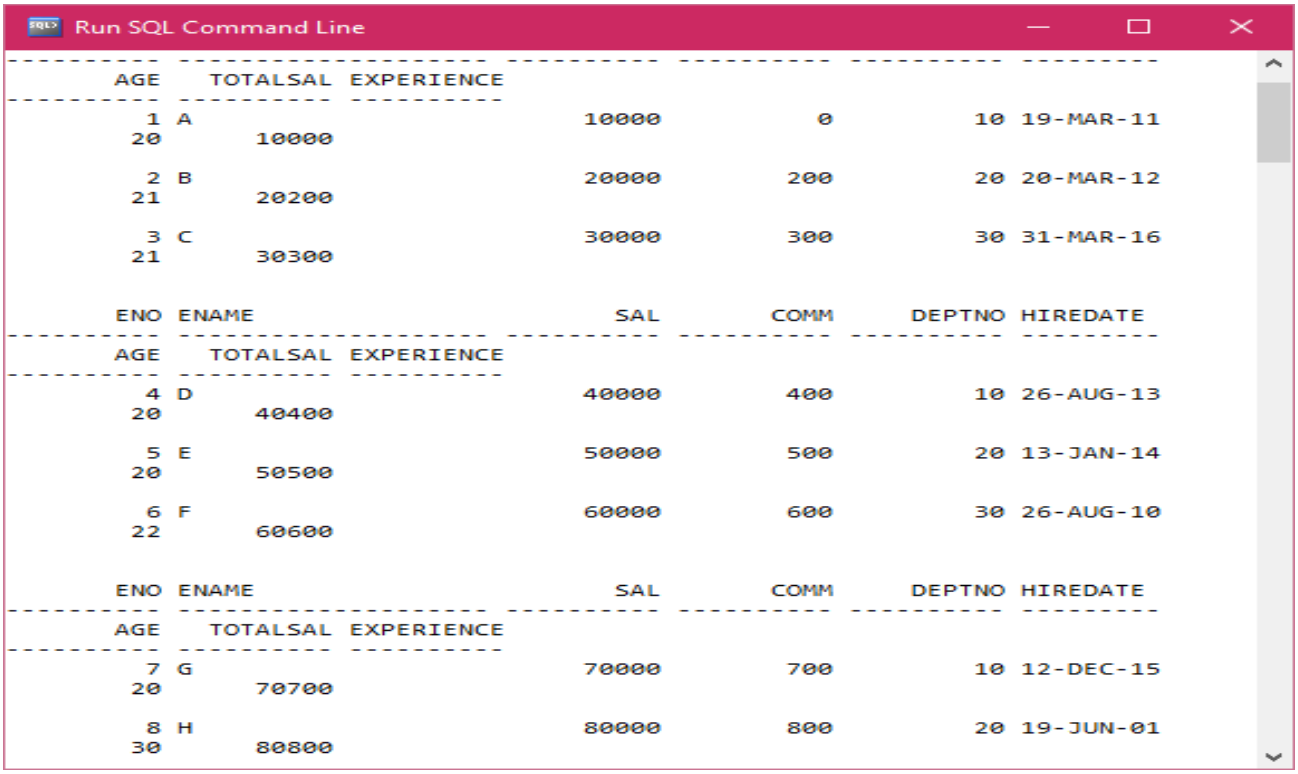
Step 9: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.



Step 9

Step 10: Verify the output of Expression Transformation in the SQL Command Prompt.

Output:



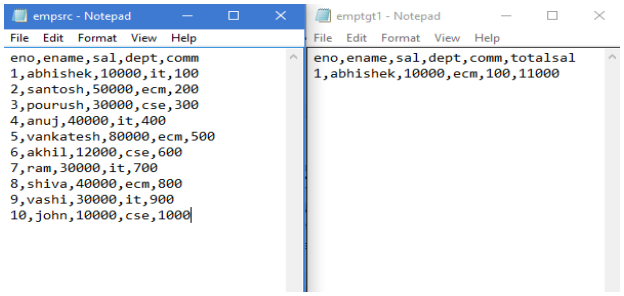
Expression Transformation was successful.

Aim: To apply an Expression Transformation on a flat file to evaluate two columns into an output column in the target table.

Theory: Expression transformation is a connected, passive transformation used to calculate values on a single row. Examples of calculations are concatenating the first and last name, adjusting the employee salaries, converting strings to date etc. Expression transformation can also be used to test conditional statements before passing the data to other transformations.

Procedure:

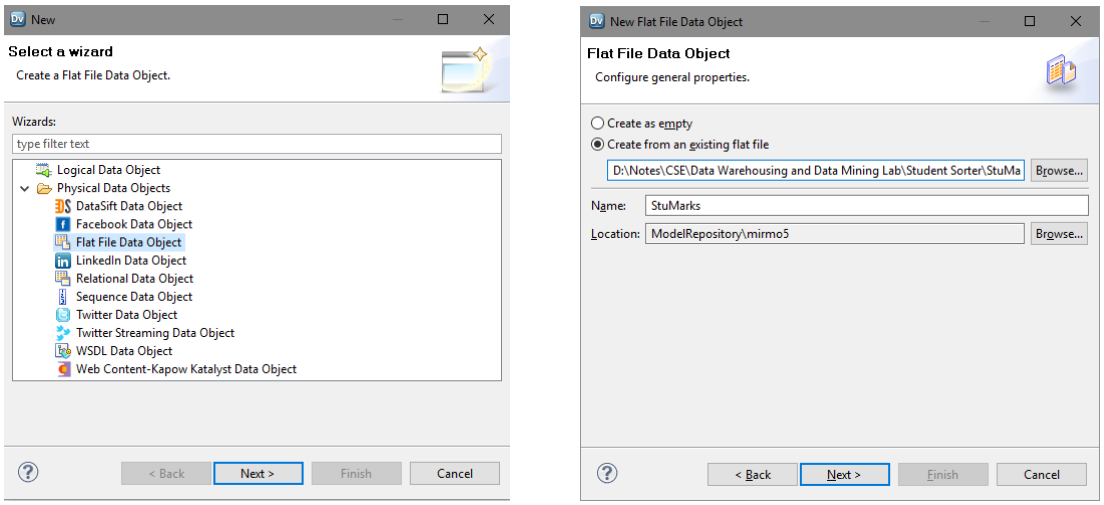
Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ',' as a delimiter.



Step 1

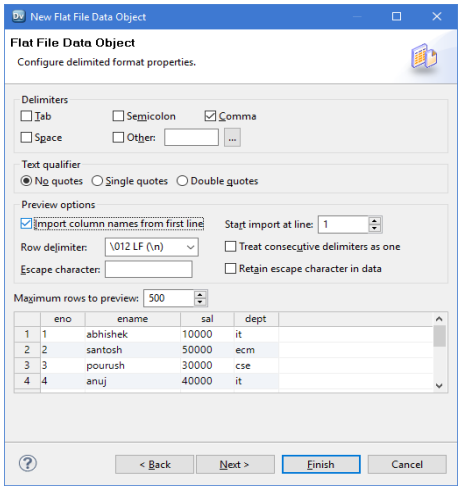
Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Import your flat files (both source and target files) into **mirmo2** project.



Step 3

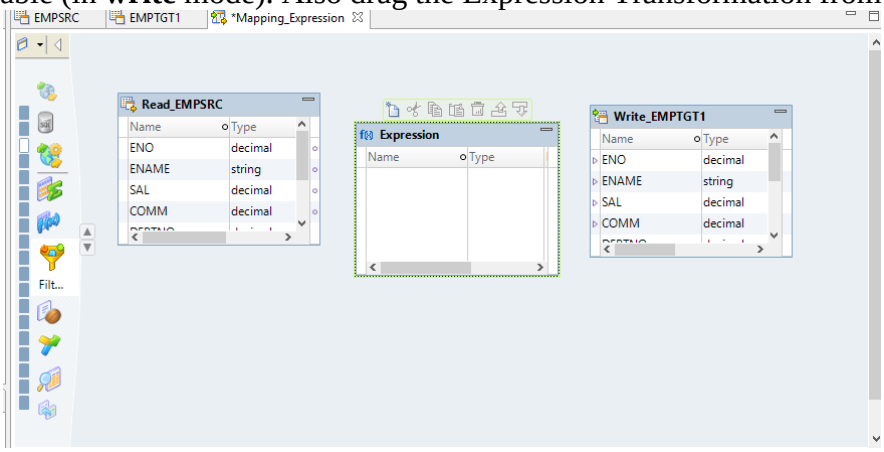
Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.



Step 3

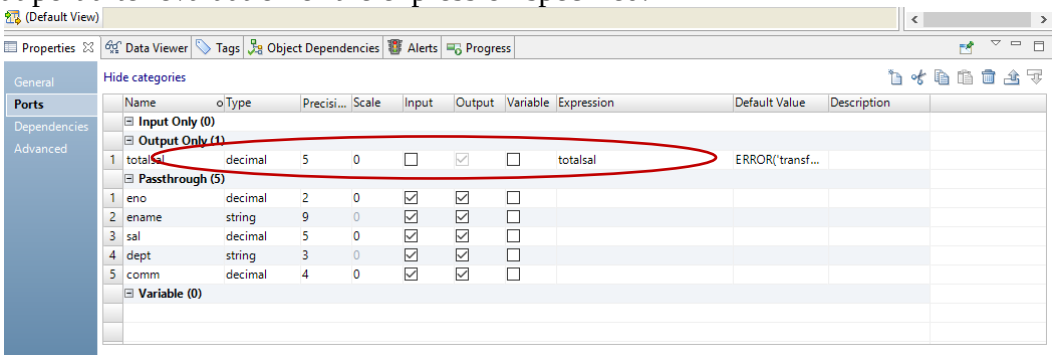
Step 4: Once the files have been imported, create a new mapping and name it **Expression_Mapping**.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (EMPTGT1) table (in **write** mode). Also drag the Expression Transformation from the palette.



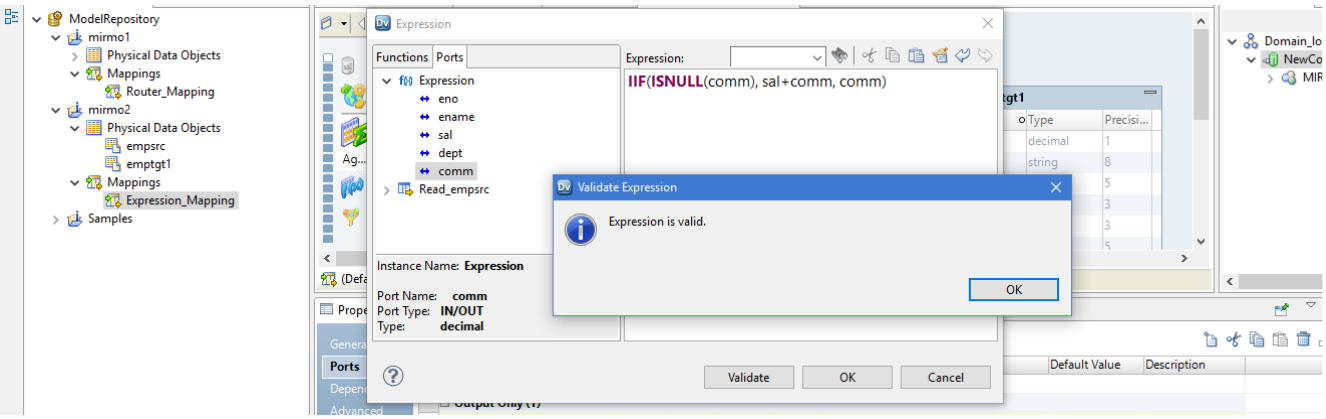
Step 5

Step 6: Drag and drop the values of the target table into the **Expression Transformation**. Also link up the input table and the output table via the Expression Transformation appropriately. Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **TOTALSAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



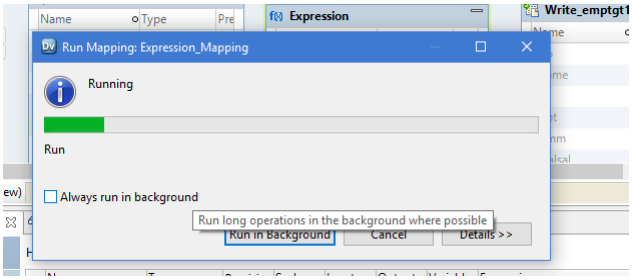
Step 6

Step 7: In the Expression Section of the entry, click on the small arrow head to open up the **Expression Dialog Box**and enter the following expression: "**IIF(ISNULL(COMM),SAL+COMM,SAL+COMM)"** This expression checks if the Commission field is NULL, then returns only salary as the result, else returns Salary + Commission as result to the TOTALSAL field. Don't forget to validate the expression for errors.



Step 7

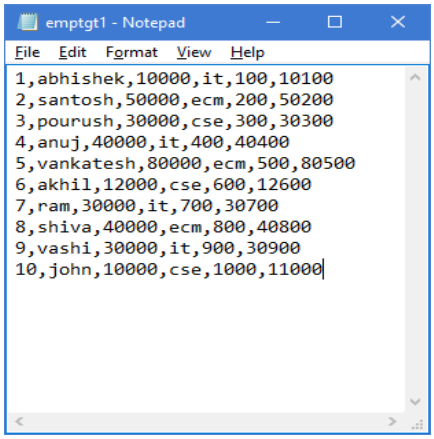
Step 8: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.



Step 8

Step 9: Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target files. Open them up to see the output.

Output:



Expression Transformation was successful.

WEEK-3:

1. Using Aggregator transformation display the average salary of employees in each departments.
2. Using Joiner transformation display the Sailor_Name form Sailors table and Boat_Name from Boats table in a new table.

Aggregator Transformation in Informatica

Aggregator is an active transformation used to calculate group values.

Ports:

· Input

· Output

· Expression

· Variable

· Group By

Properties:

· Data Cache

· Index cache

· Sorted Input

Steps:

1. Define Source

2. Define Target

Write the Target requirements and Generate/Execute SQL.

AGG_FACT

· DEPTNO

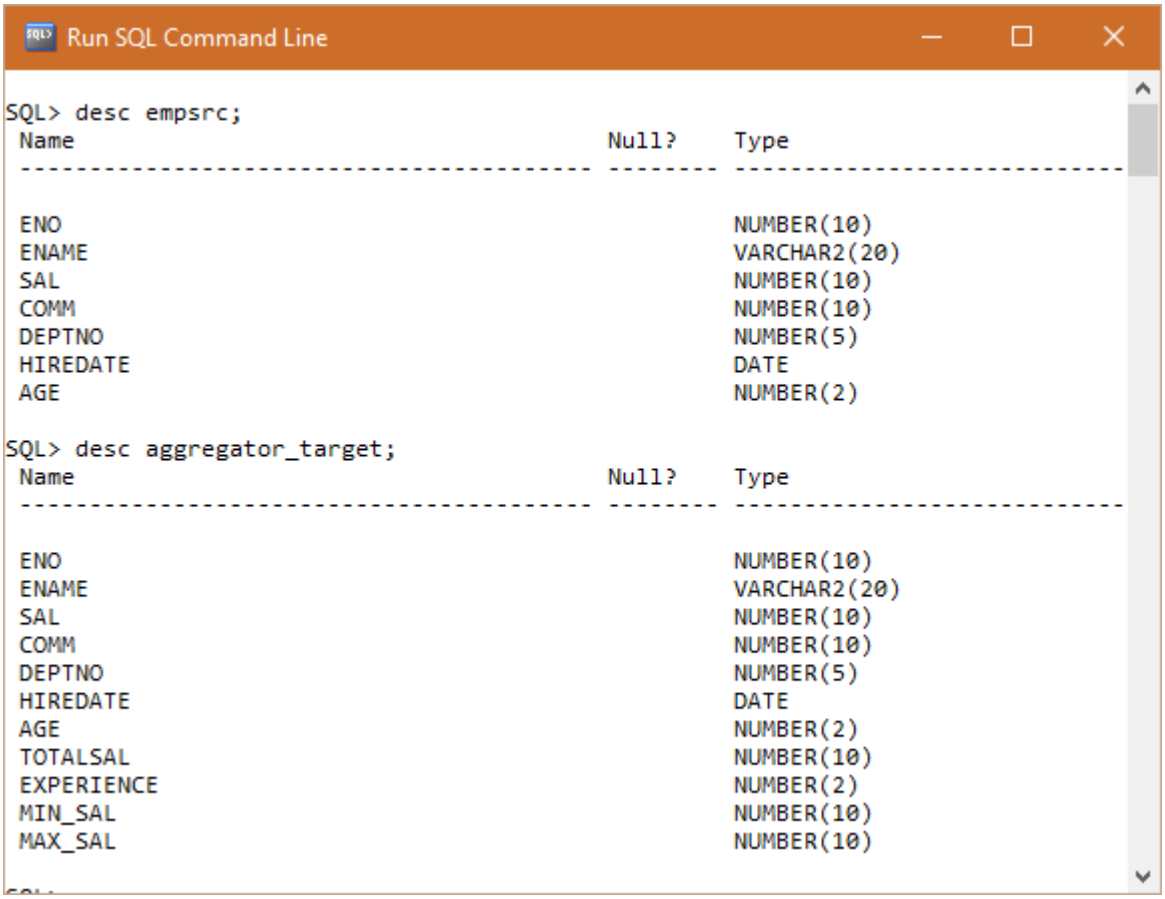
· TOT_SAL

· MAX_SAL

· MIN_SAL

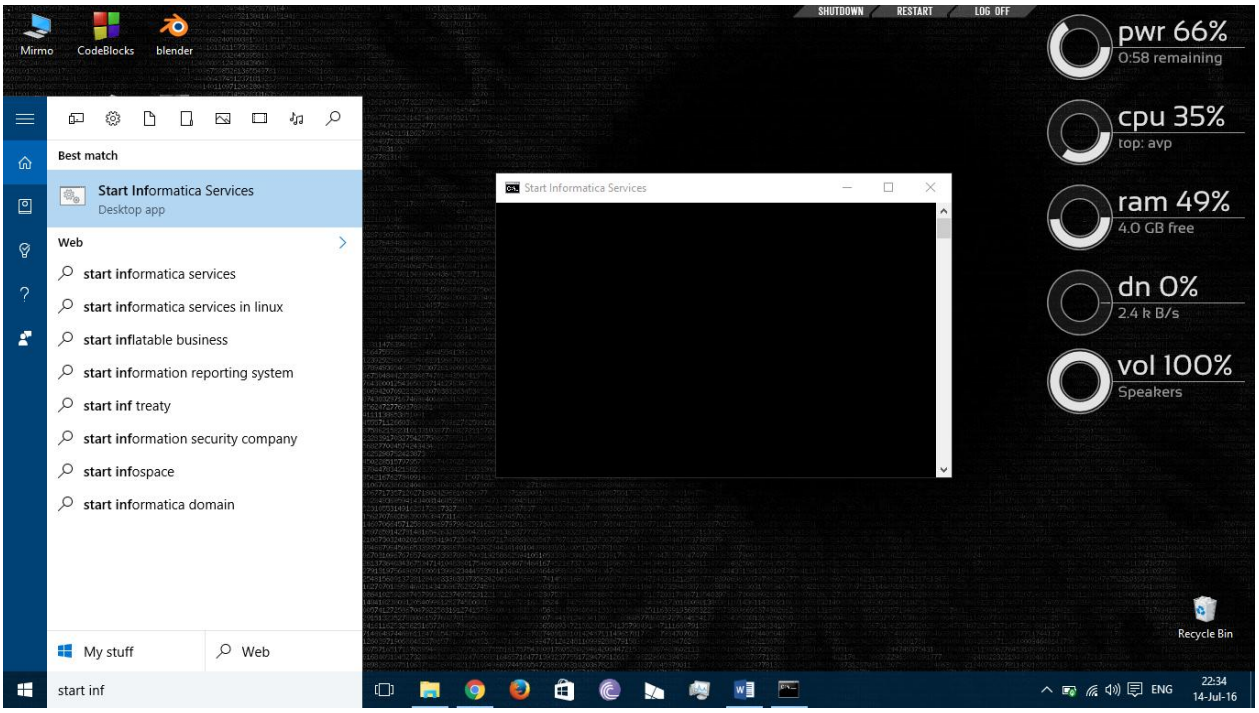
· AVG_SAL
· COUNT_EMP

Aim: To apply an Aggregator Transformation on a Data Set to calculate Minimum and Maximum Salary in the table.
Theory: Aggregator transformation is an active transformation used to perform calculations such as sums, averages, counts on groups of data. The integration service stores the data group and row data in aggregate cache. The Aggregator Transformation provides more advantages than the SQL, you can use conditional clauses to filter rows.
Procedure:
*Step 1:*Create the required tables and populate the source table with values. Add two extra columns namely min_sal and max_sal to the target table (aggregator_target) to store the output values.



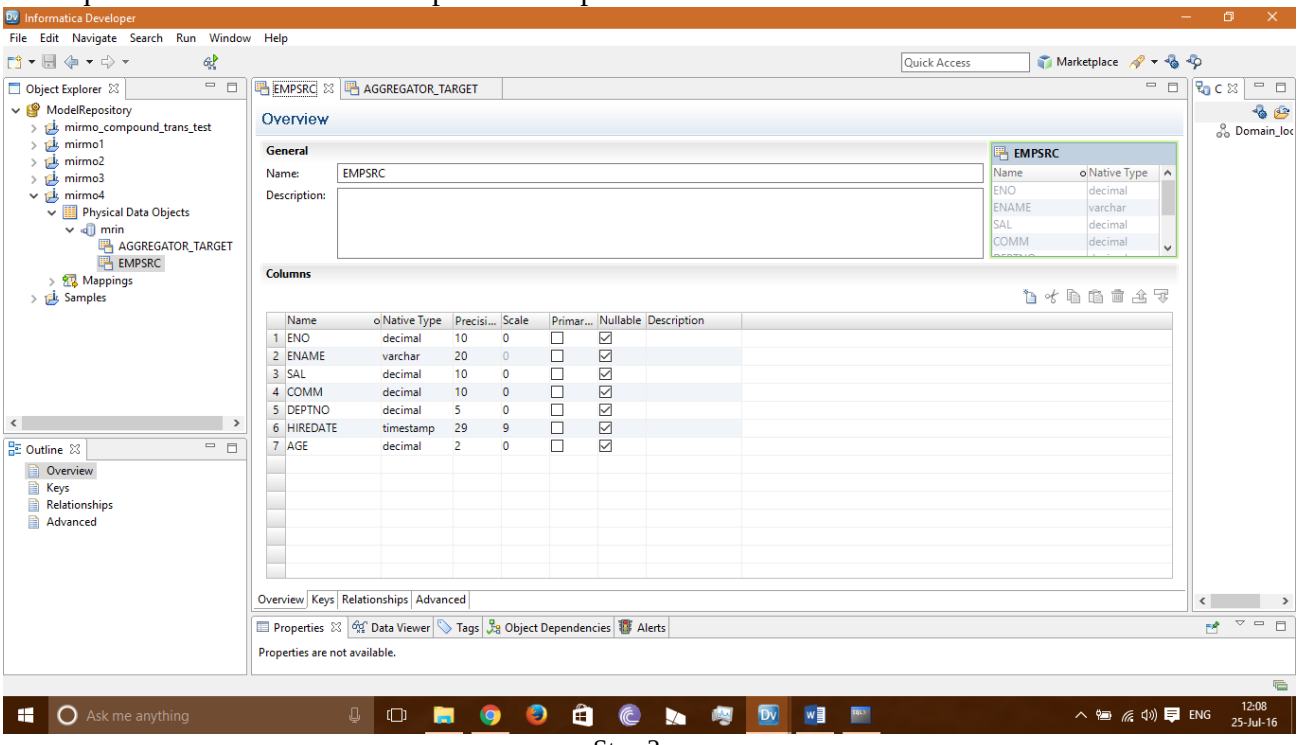
Step 1

Step 2: Start Informatica Services and Launch Informatica Administrator and Developer.



Step 2

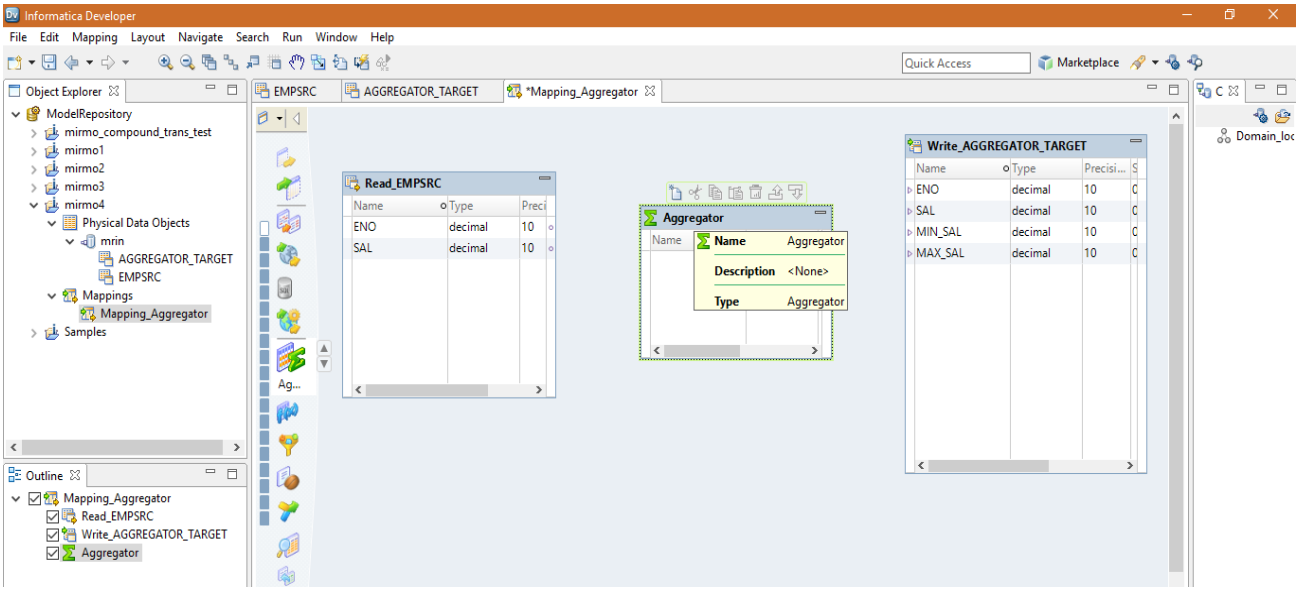
Step 3: Once Informatica Developer has started, create a **new project** (mirmo4) and import all the required tables as done in the previous experiments.



Step 3

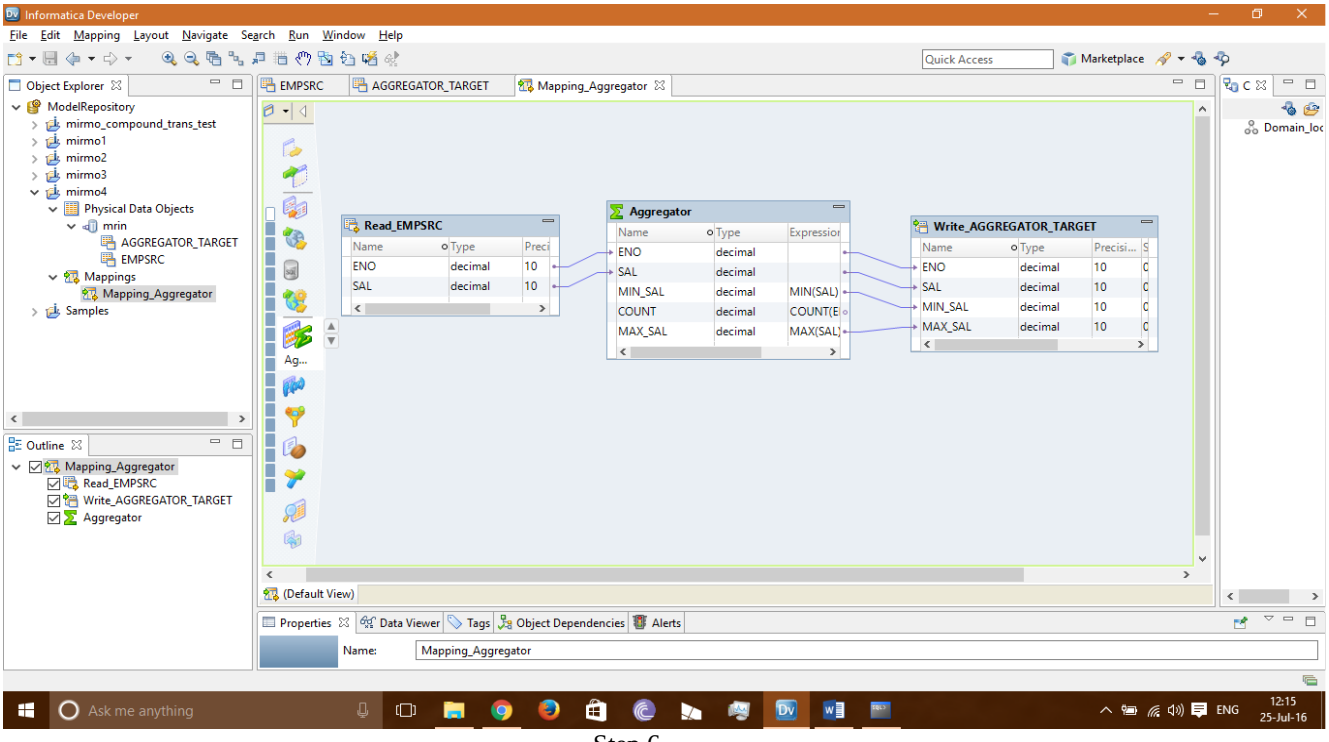
Step 4: Create a new mapping and name it Mapping_Aggregator

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (AGGREGATOR_TARGET) table (in **write** mode). Delete all the non-required entries from the tables. Also drag the Aggregator Transformation from the palette.



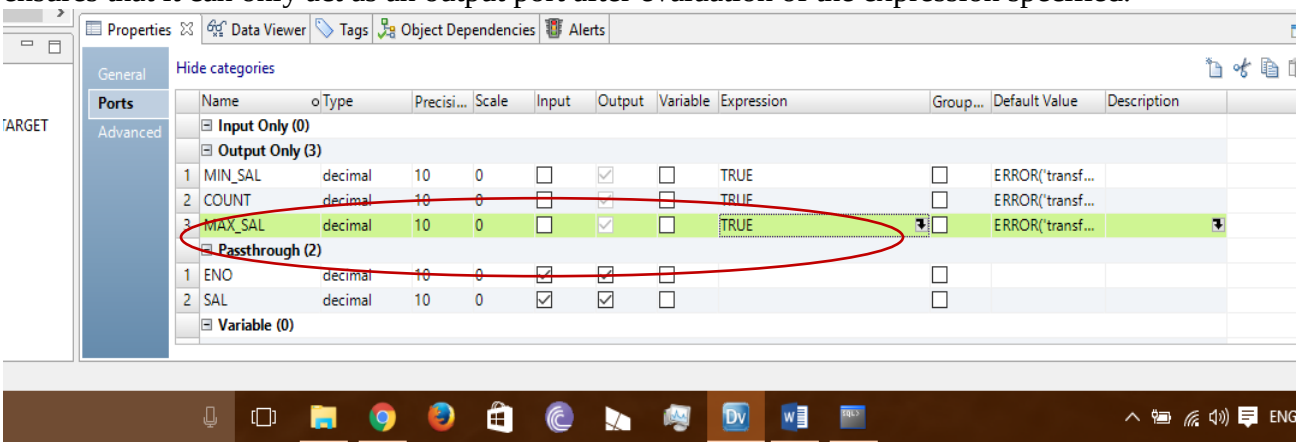
Step 5

Step 6: Drag and drop the values of the target table into the **Aggregator Transformation**. Link up the input table and the output table via the Aggregator Transformation appropriately. Add MIN_SAL, MAX_SAL and COUNT entries to the Transformation as they are required for the evaluation.



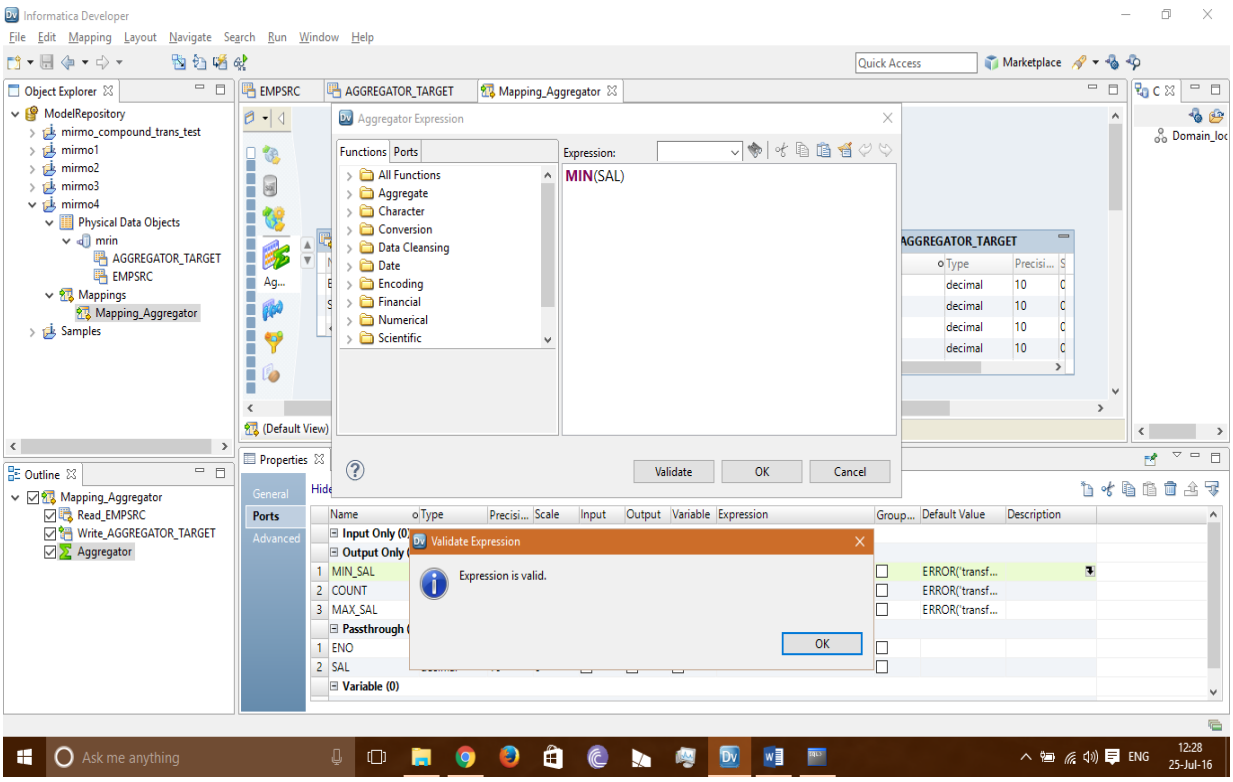
Step 6

Step 7: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **MIN_SAL, MAX_SAL and COUNT**. This ensures that it can only act as an output port after evaluation of the expression specified.



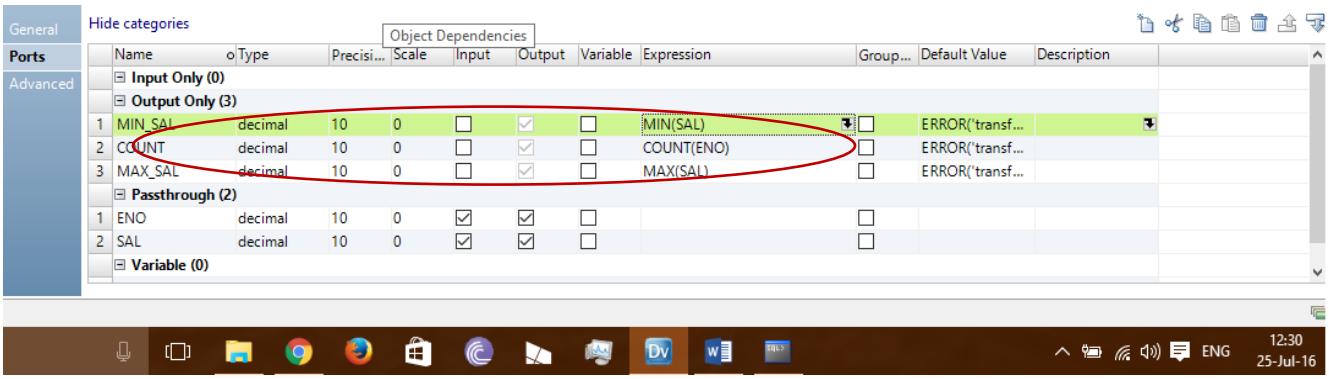
Step 7

Step 8: In the Expression Section of the MIN_SAL entry, click on the small arrow head to open up the **Expression Dialog Box,** navigate to **Functions ➔ Aggregate** and select **MIN** function.This function checks for the least value among all the values and returns it. Then navigate to **Ports** and select **SAL**, so that the expression reads "**MIN(SAL)".** Don't forget to validate the expression for errors.



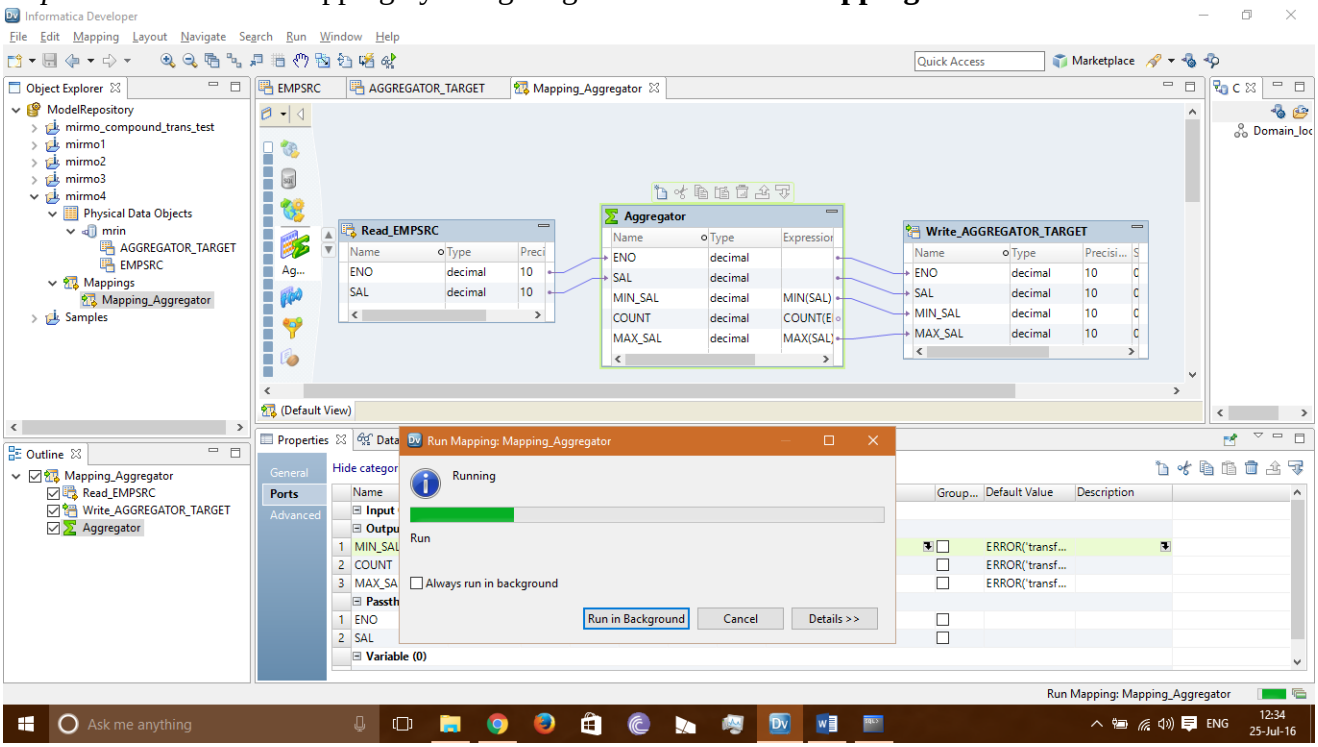
Step 8.1

Similarly, do it for **MAX_SAL** and **COUNT** appropriately.



Step 8.2

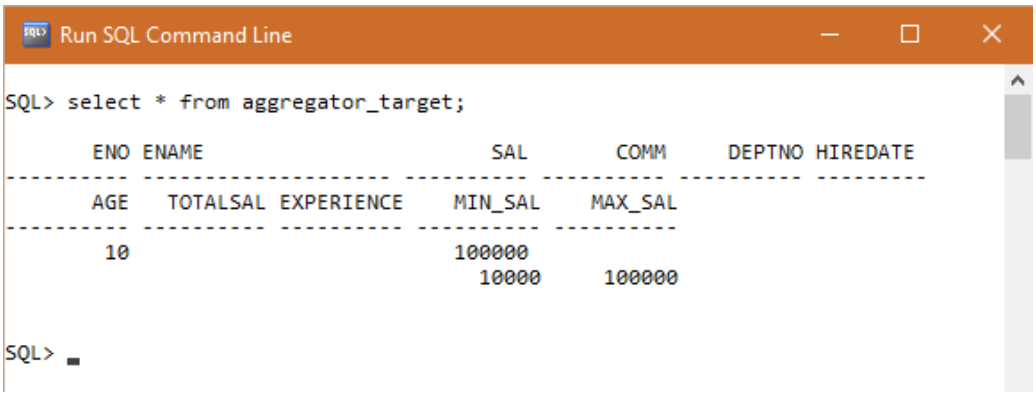
Step 9: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.



Step 9

Step 10: Verify the output of Aggregator Transformation in the SQL Command Prompt.

Output:



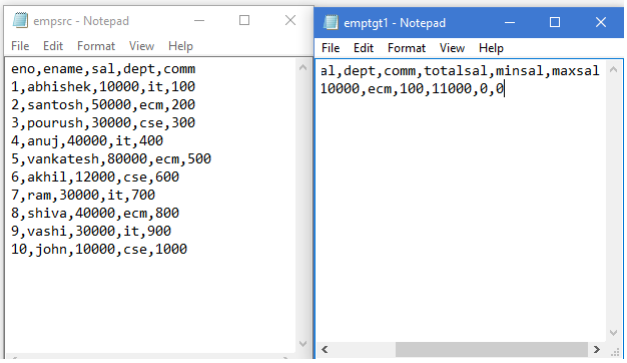
Aggregator Transformation was successful. Max and Min salaries for the table were obtained.

Aim: To apply an Aggregator Transformation on a Data Set to calculate Minimum and Maximum Salary in the table.

Theory: Aggregator transformation is an active transformation used to perform calculations such as sums, averages, counts on groups of data. The integration service stores the data group and row data in aggregate cache. The Aggregator Transformation provides more advantages than the SQL, you can use conditional clauses to filter rows.

Procedure:

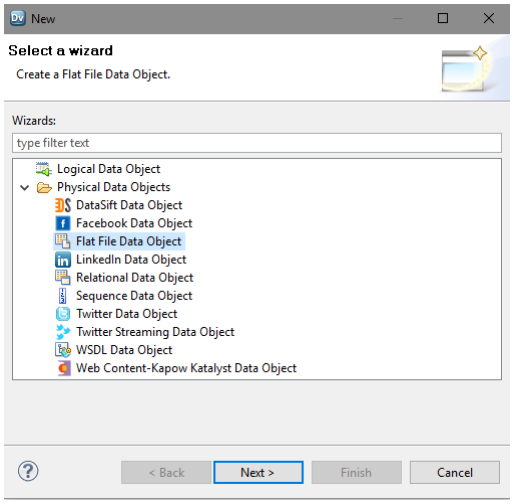
Step 1: Create the source and target flat files (text files), with the first row being column name and subsequent rows being records, separated by ',' as a delimiter.

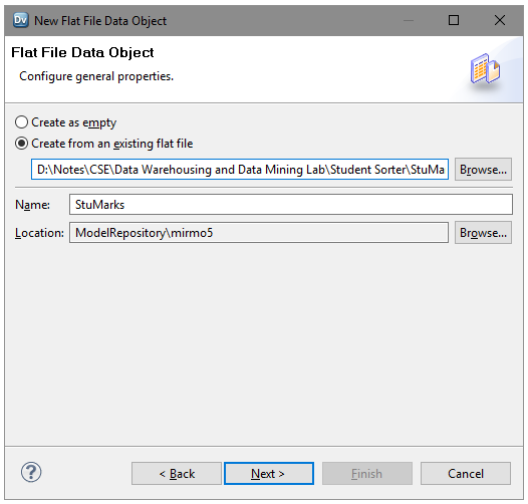


Step 1

Step 2: Start up Informatica Services and Launch Informatica Developer.

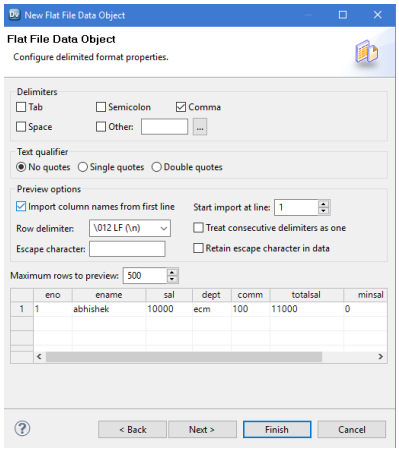
Step 3: Import your flat files (both source and target files) into **mirmo2** project.





Step 3

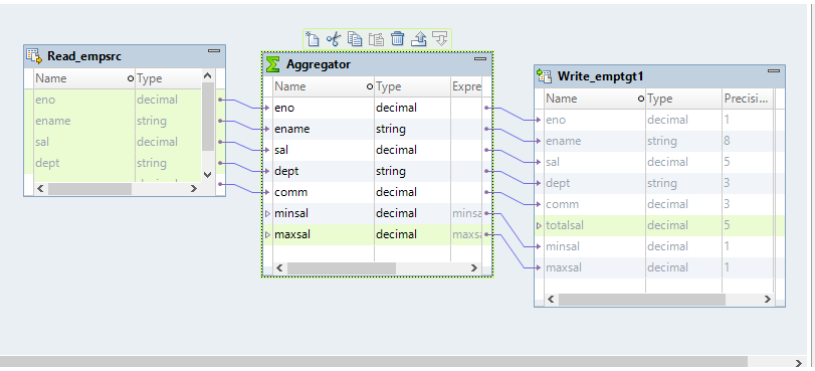
Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.



Step 3

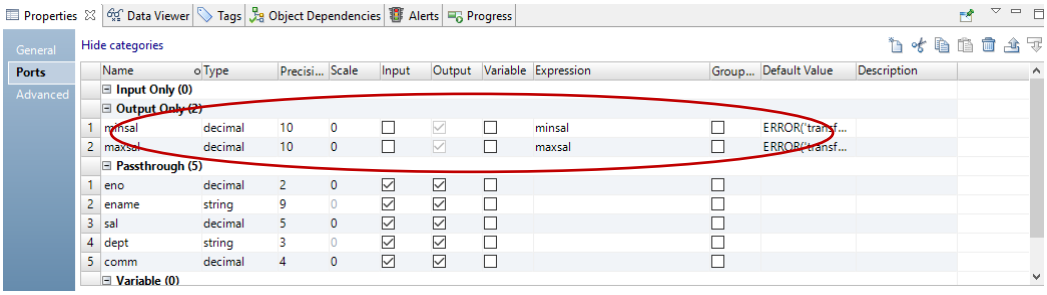
Step 4: Once the files have been imported, create a new mapping and name it **Aggregator_Mapping**.

*Step 5:*Drag the source (EMPSRC) table into the workspace (in **read** mode) and the target (AGGREGATOR_TARGET) table (in **write** mode). Delete all the non-required entries from the tables. Also drag the Aggregator Transformation from the palette. Drag and drop the values of the target table into the **Aggregator Transformation**. Link up the input table and the output table via the Aggregator Transformation appropriately. Add MIN_SAL, MAX_SAL and COUNT entries to the Transformation as they are required for the evaluation.



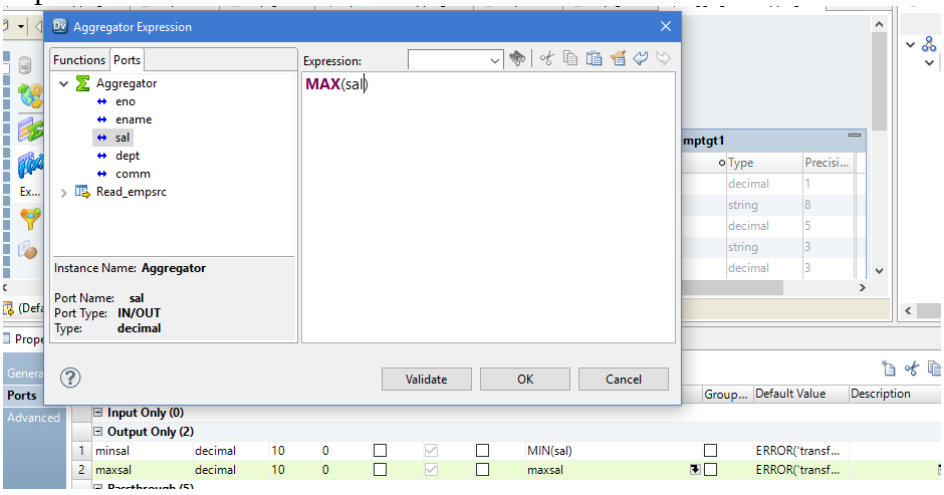
Step 5

Step 6: Make sure that the Transformation is selected. Then navigate to **Properties ➔ Ports** in the **Properties Bar.** Unselect the **Input Mark** for **MIN_SAL, MAX_SAL**. This ensures that it can only act as an output port after evaluation of the expression specified.



Step 6

Step 7: In the Expression Section of the MIN_SAL entry, click on the small arrow head to open up the **Expression Dialog Box,** navigate to **Functions ➔ Aggregate** and select **MIN** function.This function checks for the least value among all the values and returns it. Then navigate to **Ports** and select **SAL**, so that the expression reads "**MIN(SAL)".** Don't forget to validate the expression for errors.

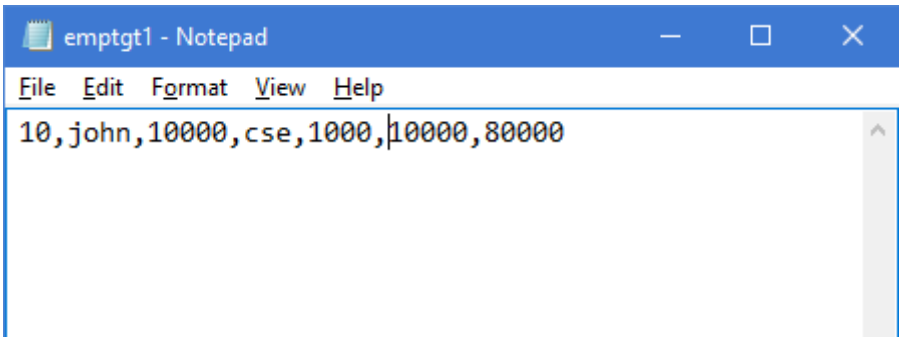


Step 7

Step 8: Now Run the Mapping by navigating to **Run ➔ Run Mapping** in the Menu Bar.

Step 9: Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Aggregator Transformation was successful. Max and Min salaries for the table were obtained.

Joiner Transformation in Informatica

The joiner transformation is an active and connected transformation used to join two heterogeneous sources. The joiner transformation joins sources based on a condition that matches one or more pairs of columns between the two sources. The two input pipelines include a master and a detail pipeline or branch. To join more than two sources, you need to join the output of the joiner transformation with another source. To join n number of sources in a mapping, you need n-1 joiner transformations.

Creating Joiner Transformation

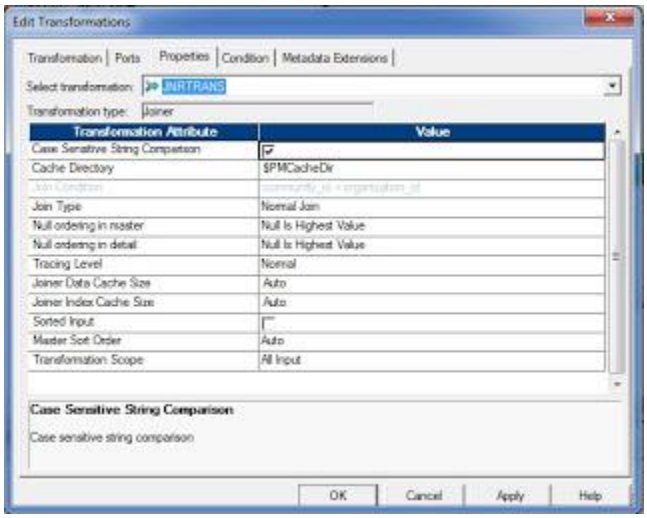
Follow the below steps to create a joiner transformation in informatica

- Go to the mapping designer, click on the Transformation->Create.
- Select the joiner transformation, enter a name and click on OK.
- Drag the ports from the first source into the joiner transformation. By default the designer creates the input/output ports for the source fields in the joiner transformation as detail fields.
- Now drag the ports from the second source into the joiner transformation. By default the designer configures the second source ports as master fields.
- Edit the joiner transformation, go the ports tab and check on any box in the M column to switch the master/detail relationship for the sources.
- Go to the condition tab, click on the Add button to add a condition. You can add multiple conditions.
- Go to the properties tab and configure the properties of the joiner transformation.

Configuring Joiner Transformation

Configure the following properties of joiner transformation:

- **Case-Sensitive String Comparison**: When performing joins on string columns, the integration service uses this option. By default the case sensitive string comparison option is checked.
- **Cache Directory**: Directory used to cache the master or detail rows. The default directory path is $PMCacheDir. You can override this value.
- **Join Type**: The type of join to be performed. Normal Join, Master Outer Join, Detail Outer Join or Full Outer Join.
- **Tracing Level**: Level of tracing to be tracked in the session log file.
- **Joiner Data Cache Size**: Size of the data cache. The default value is Auto.
- **Joiner Index Cache Size**: Size of the index cache. The default value is Auto.
- **Sorted Input**: If the input data is in sorted order, then check this option for better performance.
- **Master Sort Order**: Sort order of the master source data. Choose Ascending if the master source data is sorted in ascending order. You have to enable Sorted Input option if you choose Ascending. The default value for this option is Auto.
- **Transformation Scope**: You can choose the transformation scope as All Input or Row.



Join Condition

The integration service joins both the input sources based on the join condition. The join condition contains ports from both the input sources that must match. You can specify only the equal (=) operator between the join columns. Other operators are not allowed in the join condition. As an example, if you want to join the employees and departments table then you have

to specify the join condition as department_id1= department_id. Here department_id1 is the port of departments source and department_id is the port of employees source.

Join Type

The joiner transformation supports the following four types of joins.

- Normal Join
- Master Outer Join
- Details Outer Join
- Full Outer Join

We will learn about each join type with an example. Let say i have the following students and subjects tables as the source.

```
Table Name: Subjects
Subject_Id subject_Name
-----------------------
1          Maths
2          Chemistry
3          Physics

Table Name: Students
Student_Id  Subject_Id
--------------------
10           1
20           2
30           NULL
```

Assume that subjects source is the master and students source is the detail and we will join these sources on the subject_id port.

Normal Join:

The joiner transformation outputs only the records that match the join condition and discards all the rows that do not match the join condition. The output of the normal join is

```
Master Ports       |   Detail Ports
-------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
-------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
```

Master Outer Join:

In a master outer join, the joiner transformation keeps all the records from the detail source and only the matching rows from the master source. It discards the unmatched rows from the master source. The output of master outer join is

```
Master Ports        |   Detail Ports
--------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
--------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
NULL       NULL         30         NULL
```

Detail Outer Join:

In a detail outer join, the joiner transformation keeps all the records from the master source and only the matching rows from the detail source. It discards the unmatched rows from the detail source. The output of detail outer join is

```
Master Ports        |   Detail Ports
--------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
--------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
3          Physics      NULL       NULL
```

Full Outer Join:

The full outer join first brings the matching rows from both the sources and then it also keeps the non-matched records from both the master and detail sources. The output of full outer join is

```
Master Ports        |   Detail Ports
--------------------------------------------
Subject_Id Subject_Name Student_Id Subject_Id
--------------------------------------------
1          Maths        10         1
2          Chemistry    20         2
3          Physics      NULL       NULL
NULL       NULL         30         NULL
```

Sorted Input

Use the sorted input option in the joiner properties tab when both the master and detail are sorted on the ports specified in the join condition. You can improve the performance by using the sorted input option as the integration service performs the join by minimizing the number of disk IOs.

you can see good performance when worked with large data sets.

Steps to follow for configuring the sorted input option

- Sort the master and detail source either by using the source qualifier transformation or sorter transformation.
- Sort both the source on the ports to be used in join condition either in ascending or descending order.
- Specify the Sorted Input option in the joiner transformation properties tab.

Why joiner transformation is called as blocking transformation

The integration service blocks and unblocks the source data depending on whether the joiner transformation is configured for sorted input or not.

Unsorted Joiner Transformation

In case of unsorted joiner transformation, the integration service first reads all the master rows before it reads the detail rows. The integration service blocks the detail source while it caches the all the master rows. Once it reads all the master rows, then it unblocks the detail source and reads the details rows.

Sorted Joiner Transformation

Blocking logic may or may not possible in case of sorted joiner transformation. The integration service uses blocking logic if it can do so without blocking all sources in the target load order group. Otherwise, it does not use blocking logic.

Joiner Transformation Performance Improve Tips

To improve the performance of a joiner transformation follow the below tips

- If possible, perform joins in a database. Performing joins in a database is faster than performing joins in a session.
- You can improve the session performance by configuring the Sorted Input option in the joiner transformation properties tab.
- Specify the source with fewer rows and with fewer duplicate keys as the master and the other source as detail.

Limitations of Joiner Transformation

The limitations of joiner transformation are

- You cannot use joiner transformation when the input pipeline contains an update strategy transformation.
- You cannot connect a sequence generator transformation directly to the joiner transformation.

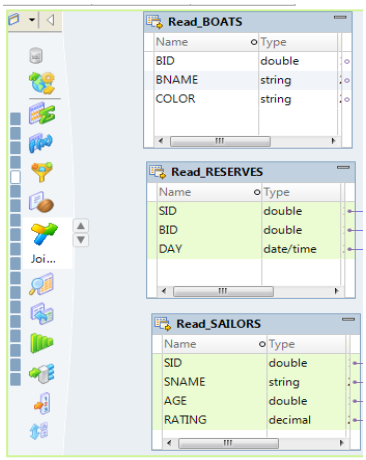
Aim: To apply Joiner Transformation on a Data Set to perform Normal Join.
Theory: The joiner transformation provides you the option to create joins in Informatica. The joins created using joiner transformation are similar to the joins in databases. The advantage of joiner transformation is that joins can be created for heterogeneous systems (different databases). In joiner transformation, there are two sources which we are going to use it for joins. These two sources are called

- Master Source
- Detail Source

In the properties of joiner transformation, you can select which data source can be Master and which source can be detail source.During execution, the master source is cached into the memory for joining purpose. So it is recommended to select the source with less number of records as the master source.
The following joins can be created using joiner transformation

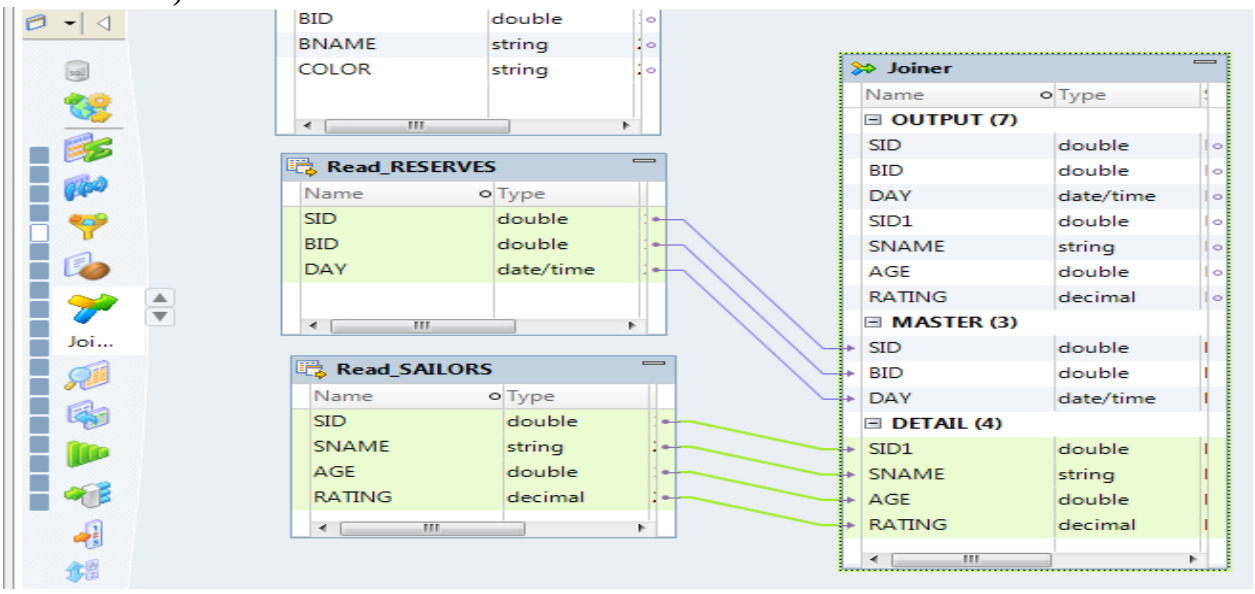
1. **Master outer join:**In Master outer join, all records from the Detail source are returned by the join and only matching rows from the master source are returned.
2. **Detail outer join:**In detail outer join only matching rows are returned from the detail source, and all rows from the master source are returned.
3. **Full outer join:**In full outer join, all records from both the sources are returned. Master outer and Detail outer joins are equivalent to left outer joins in SQL.
4. **Normal join:**In normal join only matching rows are returned from both the sources.

Procedure:
Step 1: Create the following tables Boats(bid number(2), bname varchar2(20), color varchar2(20)), Reserves(sid number(2), bid number(2), day date)), Sailors(sid number(2), sname varchar2(20), age number(2), rating number(2)) and populate them with values.
Step 2: Start up Informatica Services and Launch Informatica Developer.
Step 3: Create a new project (mirmo5) and import your tables from the database (both source and target files).
Step 4: Once the files have been imported, create a new mapping and name it **Joiner_Mapping**.
Step 5: Drag all the three tables in read mode and place them in a vertical column.



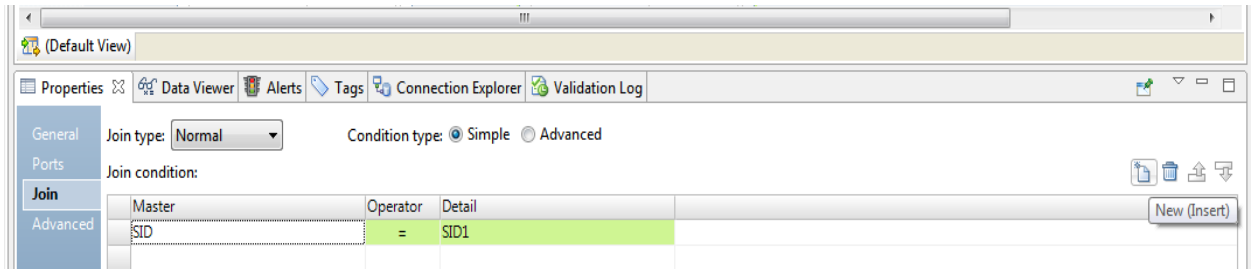
Step 5

Step 6: Now drag joiner from the palette and load two tables into it. (Both tables must have a common row)



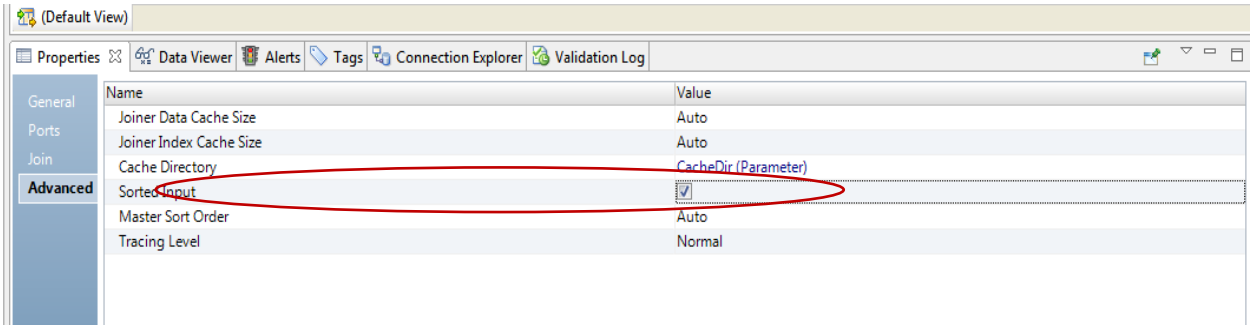
Step 6

Step 7: Navigate to **Properties ➔ Join** in the Properties Tab. Specify the join type. In this case, we use Normal Join. And specify the join condition sid of Reserves = sid of Sailors i.e., Master SID = Detail SID1.



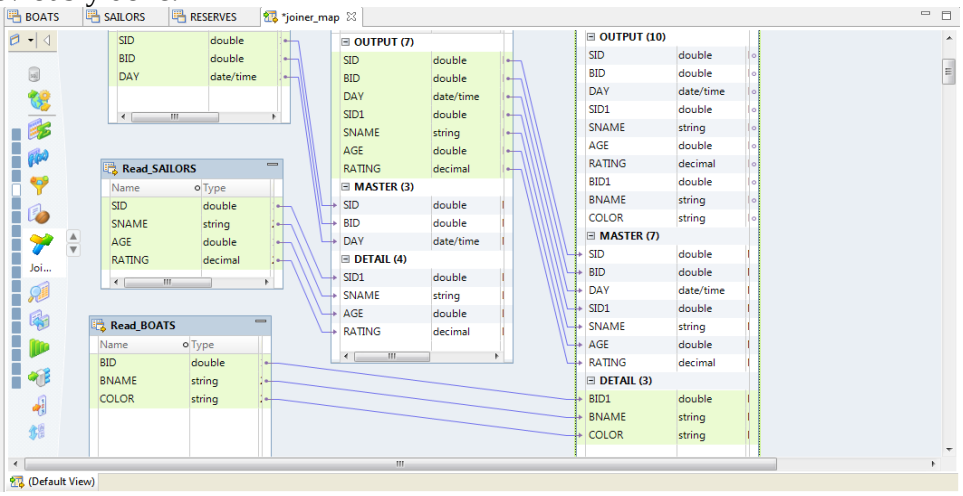
Step 7.1

Navigate to **Properties ➔ Advanced** of the Properties Tab and put a check mark to the Sorted Input option so that the input is sorted.

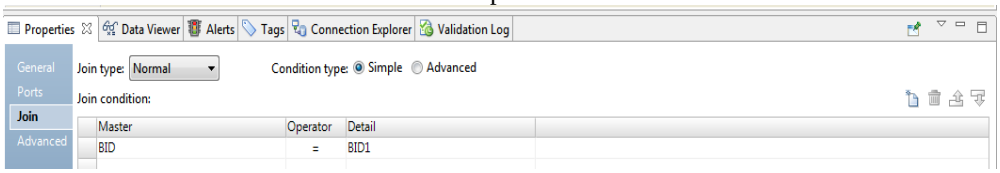


Step 7.2

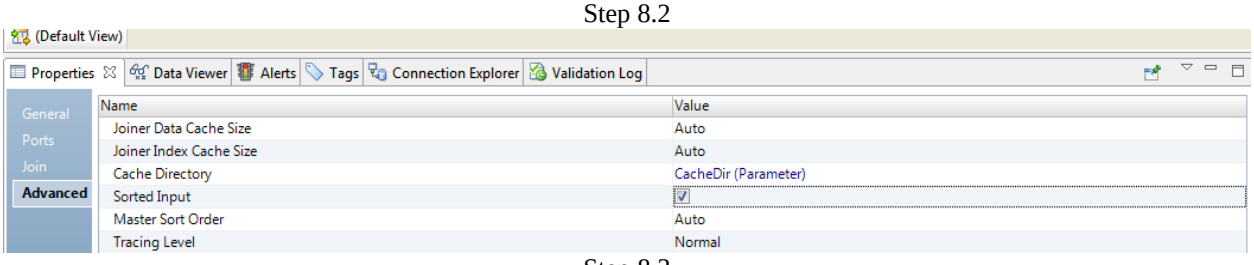
Step 8: Now Drag another Joiner from the palette and give the output of the previous Joiner as the Master and Boats as the Detail. Give the condition as Master BID = Detail BID1 and Sorted Input as previously done.



Step 8.1

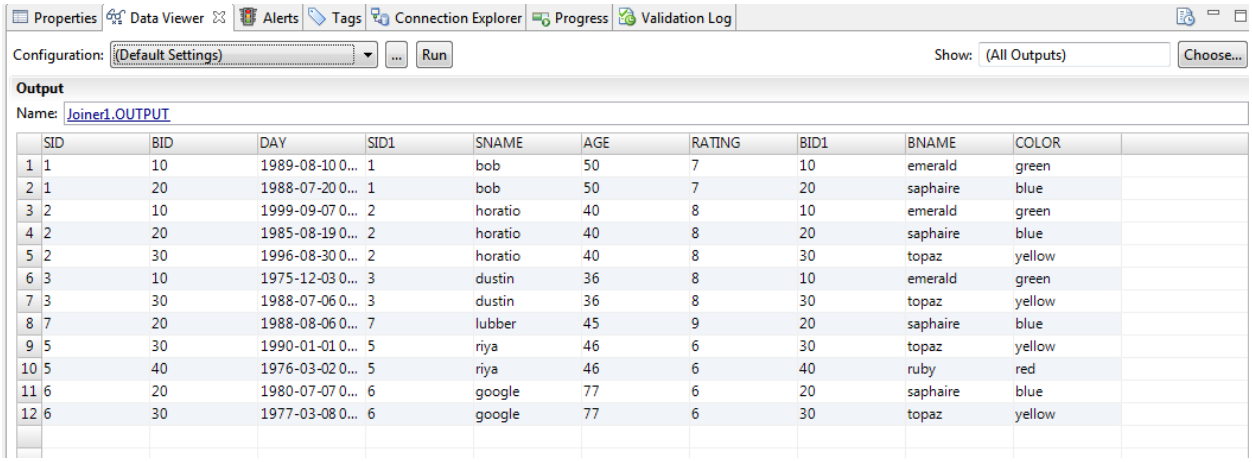


Step 8.2



Step 8.3

Step 9: Now Run the Mapping. Check the Database or the DataViewer for the Output.
Output:



	SID	BID	DAY	SID1	SNAME	AGE	RATING	BID1	BNAME	COLOR
1	1	10	1989-08-10 0...	1	bob	50	7	10	emerald	green
2	1	20	1988-07-20 0...	1	bob	50	7	20	saphaire	blue
3	2	10	1999-09-07 0...	2	horatio	40	8	10	emerald	green
4	2	20	1985-08-19 0...	2	horatio	40	8	20	saphaire	blue
5	2	30	1996-08-30 0...	2	horatio	40	8	30	topaz	yellow
6	3	10	1975-12-03 0...	3	dustin	36	8	10	emerald	green
7	3	30	1988-07-06 0...	3	dustin	36	8	30	topaz	yellow
8	7	20	1988-08-06 0...	7	lubber	45	9	20	saphaire	blue
9	5	30	1990-01-01 0...	5	riya	46	6	30	topaz	yellow
10	5	40	1976-03-02 0...	5	riya	46	6	40	ruby	red
11	6	20	1980-07-07 0...	6	google	77	6	20	saphaire	blue
12	6	30	1977-03-08 0...	6	google	77	6	30	topaz	yellow

Output

Normal Join has been successfully performed.

Aim: To apply Joiner Transformation on a Flat File Database to perform Full Outer Join.

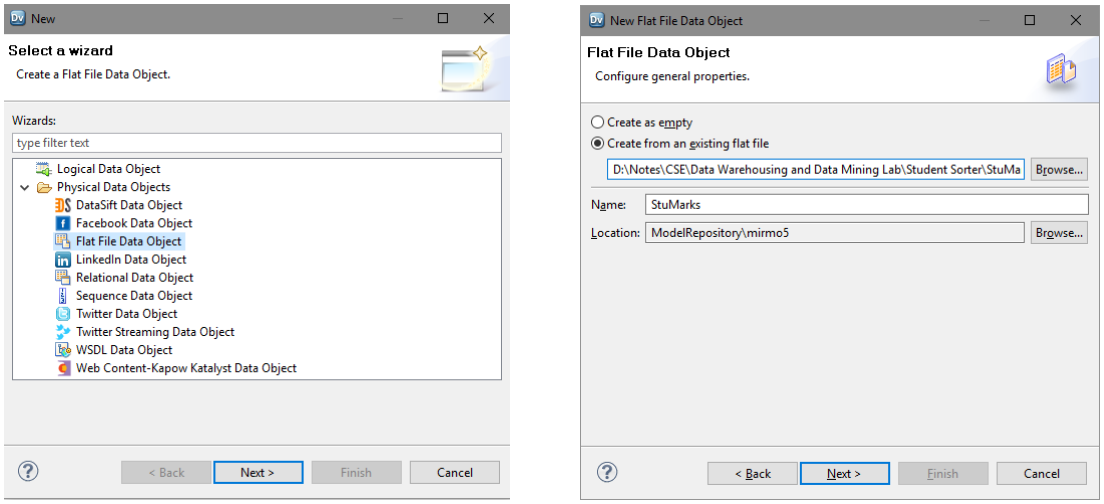
Theory: The joiner transformation provides you the option to create joins in Informatica. The joins created using joiner transformation are similar to the joins in databases. The advantage of joiner transformation is that joins can be created for heterogeneous systems (different databases).

Procedure:

Step 1: Create the followingflatfilesReserves(sid,bid,day), Sailors(sid,sname,age,rating) and populate them with values.

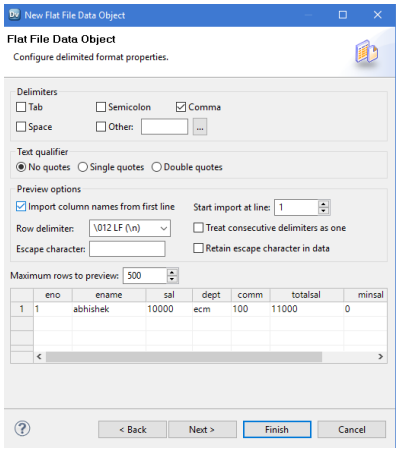
Step 2: Start up Informatica Services and Launch Informatica Developer.

Step 3: Import your flat files into **mirmo2** project.



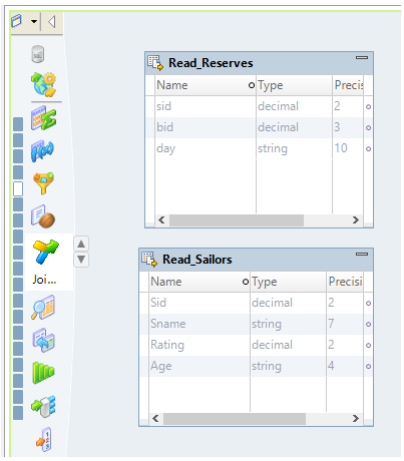
Step 3

Click on Next and then Next again. In the next dialog box that opens, check the **"Import Column Names from the first line"** checkbox and make sure the delimiter selected is '**,**'. Then click on **Finish**.



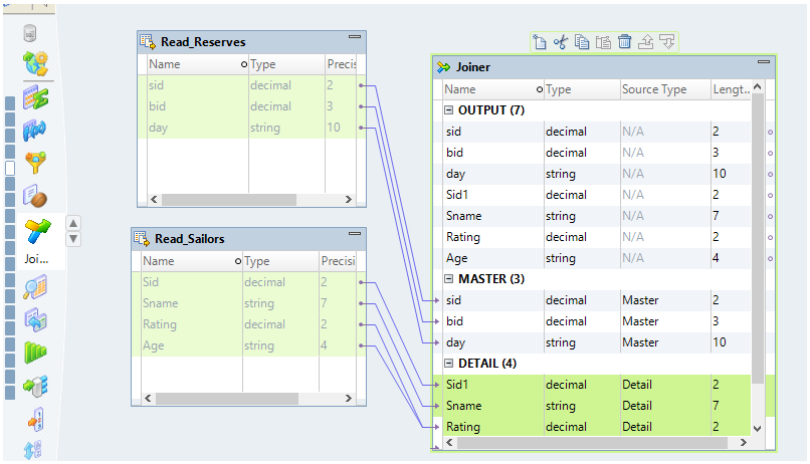
Step 3

Step 4: Once the files have been imported, create a new mapping and name it **Joiner_Mapping**.
Step 5: Drag all the files in read mode and place them in a vertical column.



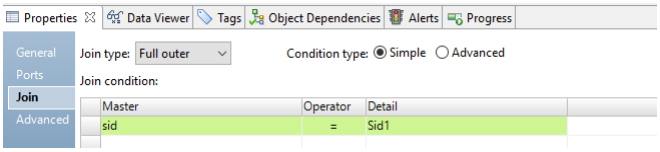
Step 5

Step 6: Now drag joiner from the palette and load the tables into it. (Both tables must have a common row)



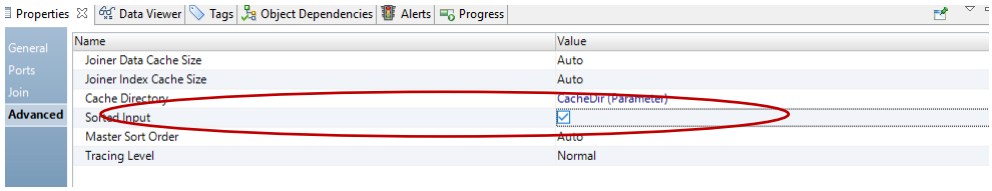
Step 6

Step 7: Navigate to **Properties ➔ Join** in the Properties Tab. Specify the join type. In this case, we use Full Outer Join. And specify the join condition sid of Reserves = sid of Sailors i.e., Master SID = Detail SID1.



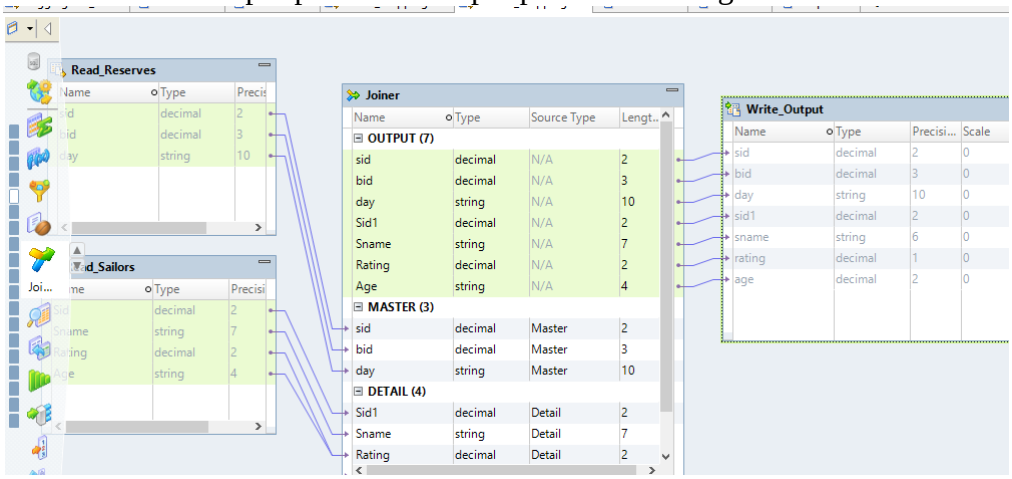
Step 7.1

Navigate to **Properties ➔ Advanced** of the Properties Tab and put a check mark to the Sorted Input option so that the input is sorted.



Step 7.2

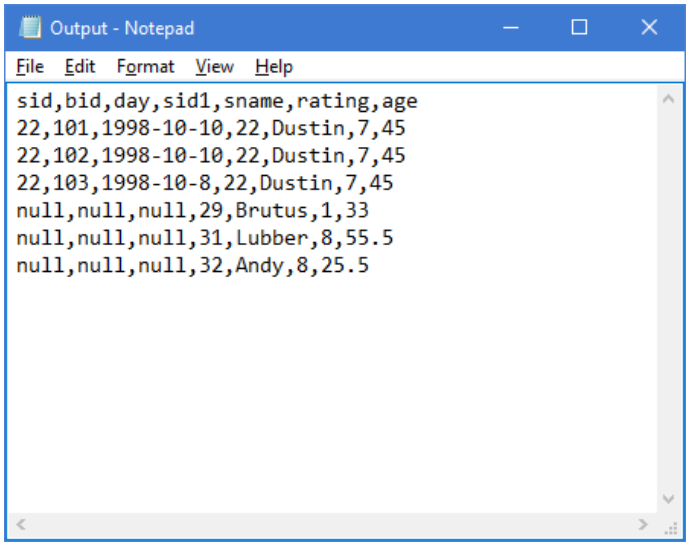
Step 8: Now connect the output ports to the input ports of the Target File.



Step 8

Step 9: Now Run the Mapping. Then navigate to **"F:\Informatica\PCExpress\tomcat\bin\target"** (that's where Informatica is installed on my system, F: Drive), and you can find your output target file. Open it up to see the output.

Output:



Output

Full Outer Join has been successfully performed.

1. List all the categorical (or nominal) attributes and the real-valued attributes separately.

Categorical or Nomianal attributes:-

1. checking_status
2. credit history
3. purpose
4. savings_status
5. employment
6. personal status
7. debtors
8. property
9. installment plans
10. housing
11. job
12. telephone
13. foreign worker

Real valued attributes:-

1. duration
2. credit amount
3. credit amount
4. residence
5. age
6. existing credits
7. num_dependents

2. What attributes do you think might be crucial in making the credit assessement ? Come up with some simple rules in plain English using your

selected attributes.

According to me the following attributes may be crucial in making the credit risk assessment.

1. duration
2. credit_history
3. credit_amount
4. employment
5. other_parties
6. job
7. class

Basing on the above attributes, we can make a decision whether to give credit or not.

Some of the simple rules are

1.if (credit_history =”allpaid” && employment>4)
then
credit_amount=XXX;

else
credit_amount=0

2.if(credit_history=” existing credits paid back duly till now
“ && job=”officer”)
then
credit_history=XXX
else
credit_amount=0

3. One type of model that you can create is a Decision Tree - train a Decision Tree using the complete dataset as the training data. Report the model obtained after training.

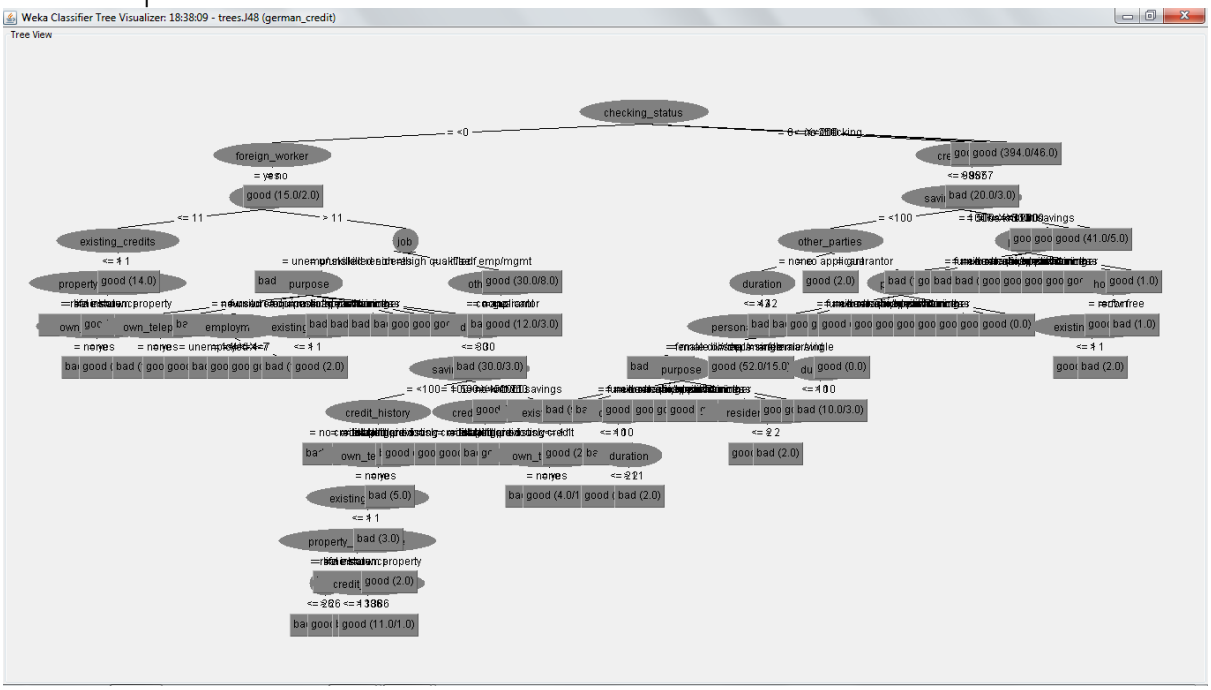
Using J48 classifier

Number of Leaves : 103
Size of the tree : 140
Time taken to build model: 0.03 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 855 85.5 %
Incorrectly Classified Instances 145 14.5 %
Kappa statistic 0.6251
Mean absolute error 0.2312

Root mean squared error 0.34
Relative absolute error 55.0377 %
Root relative squared error 74.2015 %
Total Number of Instances 1000

=== Confusion Matrix ===

a	b	<-- classified as
669	31	a = good
114	186	b = bad



Model tree obtained after applying J48 classifier

4. Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly? (This is also called testing on the training set) Why do you think you cannot get 100 % training accuracy?

In the above model we trained complete dataset and we classified credit good/bad for each of the examples in the dataset.

For example:

IF

purpose=vacation THEN

credit=bad

ELSE

purpose=business THEN

Credit=good

In this way we classified each of the examples in the dataset.
We classified 85.5% of examples correctly and the remaining 14.5% of examples are incorrectly classified. We can't get 100% training accuracy because out of the 20 attributes, we have some unnecessary attributes which are also been analyzed and trained.Due to this the accuracy is affected and hence we can't get 100% training accuracy.

5. Is testing on the training set as you did above a good idea? Why or Why not?

According to the rules, for the maximum accuracy, we have to take 2/3 of the dataset as training set and the remaining 1/3 as test set. But here in the above model we have taken complete dataset as training set which results only 85.5% accuracy.This is done for the analyzing and training of the unnecessary attributes which does not make a crucial role in credit risk assessment. And by this complexity is increasing and finally it leads to the minimum accuracy.
If some part of the dataset is used as a training set and the remaining as test set then it leads to the accurate results and the time for computation will be less. This is why, we prefer not to take complete dataset as training set.

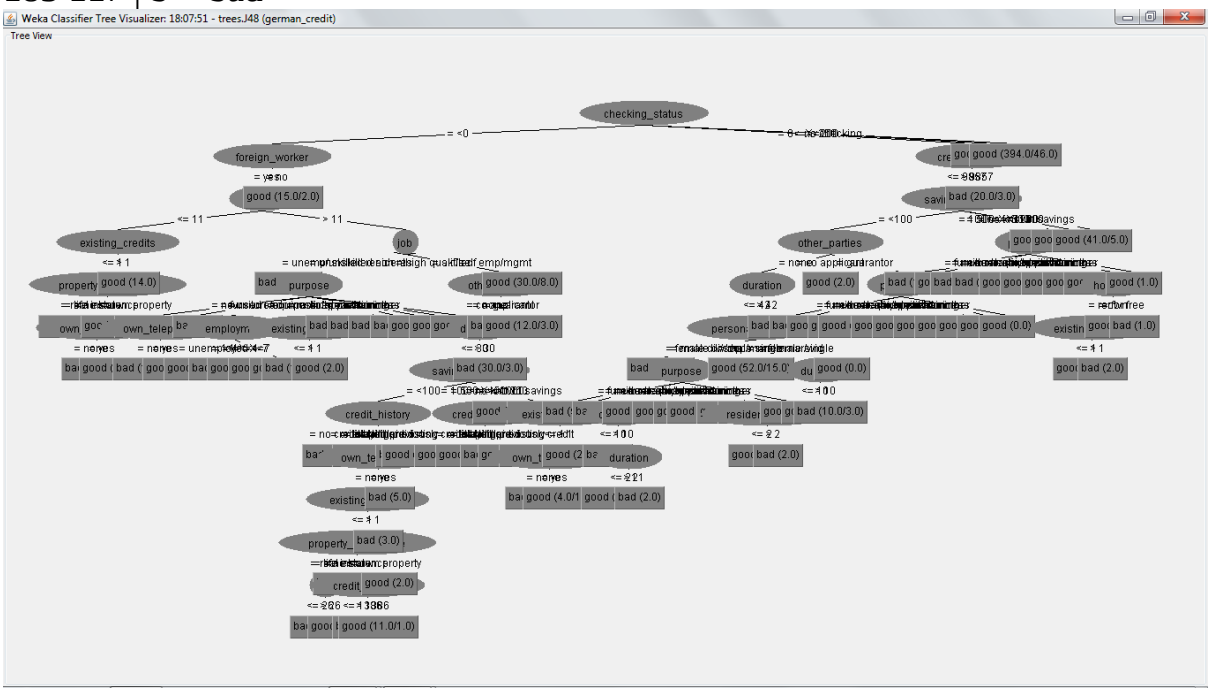
6. One approach for solving the problem encountered in the previous question is using cross-validation? Describe what cross-validation is briefly. Train a Decision Tree again using cross-validation and report your results. Does your accuracy increase/decrease? Why?

Cross validation:-

In k-fold cross-validation, the initial data are randomly portioned into 'k' mutually exclusive subsets or folds D1, D2, D3, ..., Dk. Each of approximately equal size. Training and testing is performed 'k' times. In iteration I, partition Di is reserved as the test set and the remaining partitions are
collectively used to train the model. That is in the first iteration subsets D2, D3, ..., Dk collectively serve as the training set in order to obtain as first model. Which is tested on Di. The second trained on the subsets D1, D3, ..., Dk and test on the D2 and so on….
Using J48 classifier and test option is cross-validation

Number of Leaves : 103
Size of the tree : 140
Time taken to build model: 0.07 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 705 70.5 %
Incorrectly Classified Instances 295 29.5 %
Kappa statistic 0.2467

Mean absolute error 0.3467
Root mean squared error 0.4796
Relative absolute error 82.5233 %
Root relative squared error 104.6565 %
Total Number of Instances 1000
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.84 0.61 0.763 0.84 0.799 0.639 good
0.39 0.16 0.511 0.39 0.442 0.639 bad
Weighted Avg. 0.705 0.475 0.687 0.705 0.692 0.639
=== Confusion Matrix ===
a b <-- classified as
588 112 | a = good
183 117 | b = bad



Decision tree obtained using J48 classifier (cross validation)

Accuracy increased since the attributes are selected uniquely for each fold in cross validation where some of the unnecessary attributes are filtered while choosing.This increases accuracy compared to the accuracy obtained using the test option training dataset where all the attributes are selected irrespective of their behaviour.

7. Check to see if the data shows a bias against "foreign workers" (attribute 20),or "personal-status"(attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the

decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the reprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect? Discuss.

This increase in accuracy is because thus two attributes are not much important in training and analyzing by removing this, the time has been reduced to some extent and then it results in increase in the accuracy.

The decision tree which is created is very large compared to the decision tree which we have trained now. This is the main difference between these two decision trees.

8. Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). Try out some combinations. (You had removed two attributes in problem 7.Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)

To get good results we may not require all the attributes,only a few would do. Below examples shows the results by considering all the attributes once and considering few attributes then.

Using J48 classifier(Considering all attributes)

Number of Leaves : 103

Size of the tree : 140

Time taken to build model: 0.03 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	855	85.5 %
Incorrectly Classified Instances	145	14.5 %
Kappa statistic	0.6251	
Mean absolute error	0.2312	
Root mean squared error	0.34	
Relative absolute error	55.0377 %	
Root relative squared error	74.2015 %	
Total Number of Instances	1000	

=== Confusion Matrix ===

a	b	<-- classified as
669	31	a = good
114	186	b = bad

Using J48 classifier(After removing certain attributes)

Number of Leaves : 109
Size of the tree : 147
Time taken to build model: 0.04 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	834	83.4 %
Incorrectly Classified Instances	166	16.6 %
Kappa statistic	0.5677	
Mean absolute error	0.2547	
Root mean squared error	0.3569	
Relative absolute error	60.6181 %	
Root relative squared error	77.8724 %	
Total Number of Instances	1000	

== Confusion Matrix ===

a	b	<-- classified as
662	38	a = good
128	172	b = bad

9. Sometimes, the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who has bad credit (case 2).Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and cross-validation results. Are they significantly different from results obtained in problem 6 (using equal cost)?

In the Problem 6, we used equal cost and we trained the decision tree. But here, we consider two cases with different cost.

Let us take cost 5 in case 1 and cost 2 in case 2.

When we give such costs in both cases and after training the decision tree, we can observe that almost equal to that of the decision tree obtained in problem 6. But we find some difference in cost factor which is in summary in the difference in cost factor.

	Case1 (cost 5)	Case2 (cost 5)
Total Cost	3820	1705
Average cost	3.82	1.705

We don’t find this cost factor in problem 6. As there we use equal cost. This is the major difference between the results of problem 6 and problem 9.

The cost matrices we used here:
Case 1:
5 1
1 5
Case 2:
2 1
1 2

10. Do you think it is a good idea to prefer simple decision trees instead of having long complex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?

When we consider long complex decision trees, we will have many unnecessary attributes in the tree which results in increase of the bias of the model. Because of this, the accuracy of the model can also effected.This problem can be reduced by considering simple decision tree. The attributes will be less and it decreases the bias of the model. Due to this the result will be more accurate.So it is a good idea to prefer simple decision trees instead of long complex trees.

11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning - Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross-validation (you can do this in Weka) and report the Decision Tree you obtain ? Also, report your accuracy using the pruned model. Does your accuracy increase ?

Reduced-error pruning :-

The idea of using a separate pruning set for pruning—which is applicable to decision trees as well as rule sets—is called reduced-error pruning. The variant described previously prunes a rule immediately after it has been grown and is called incremental reduced-error pruning.Another possibility is to build a full, unpruned rule set first, pruning it afterwards by discarding individual tests.However, this method is much slower. Of course, there are many different ways to assess the worth of a rule based on the pruning set. A simple measure is to consider how well the rule would do at discriminating the predicted class from other classes if it were the only rule in the theory,operating under the closed world assumption. If it gets p instances right out of the t instances that it covers, and there are P instances of this class out of a total T of instances altogether, then it gets p positive instances right. The instances that it does not cover include N - n negative ones, where n = t – p is the number of negative instances that the rule covers and N = T - P is the total number of negative

instances. Thus the rule has an overall success ratio of [p +(N - n)] T , and this quantity, evaluated on the test set, has been used to evaluate the success of a rule when using reduced-error pruning.
Number of Leaves : 47
Size of the tree : 64
Time taken to build model: 0.49 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 725 72.5 %
Incorrectly Classified Instances 275 27.5 %
Kappa statistic 0.2786
Mean absolute error 0.3331
Root mean squared error 0.4562
Relative absolute error 79.2826 %
Root relative squared error 99.5538 %
Total Number of Instances 1000

12. How can you convert a Decision Trees into "if-then-else rules". Make up your own small Decision Tree consisting of 2-3 levels and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules - one such classifier in Weka is rules.PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this dataset? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a one R classifier. Rank the performance of j48, PART and oneR.

In weka, rules.PART is one of the classifier which converts the decision trees into "IF-THEN-ELSE" rules.

Converting Decision trees into "IF-THEN-ELSE" rules using rules.PART classifier:-

PART decision list
outlook = overcast: yes (4.0)
windy = TRUE: no (4.0/1.0)
outlook = sunny: no (3.0/1.0)
: yes (3.0)
Number of Rules : 4
Yes, sometimes just one attribute can be good enough in making the decision.
In this dataset (Weather), Single attribute for making the decision is **"outlook"**
outlook:
sunny -> no

overcast -> yes
rainy -> yes
(10/14 instances correct)
With respect to the **time**, the one R classifier has higher ranking and J48 is in 2nd place and PART gets 3rd place.

	J48	PART	one R
TIME (sec)	0.12	0.14	0.04
RANK	II	III	I

But if you consider the **accuracy,** The J48 classifier has higher ranking, PART gets second place and oneR gets lst place

	J48	PART	oneR
ACCURACY (%)	70.5	70.2%	66.8%

Using J48 classifier

Number of Leaves : 103

Size of the tree : 140

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	705	70.5 %
Incorrectly Classified Instances	295	29.5 %
Kappa statistic	0.2467	
Mean absolute error	0.3467	
Root mean squared error	0.4796	
Relative absolute error	82.5233 %	
Root relative squared error	104.6565 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.84	0.61	0.763	0.84	0.799	0.639	good
	0.39	0.16	0.511	0.39	0.442	0.639	bad
Weighted Avg.	0.705	0.475	0.687	0.705	0.692	0.639	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 588 112 |   a = good
 183 117 |   b = bad
```

Using PART classifier

Number of Rules : 78

Time taken to build model: 0.22 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	702	70.2 %
Incorrectly Classified Instances	298	29.8 %
Kappa statistic	0.2767	
Mean absolute error	0.3245	
Root mean squared error	0.4974	
Relative absolute error	77.2302 %	
Root relative squared error	108.5412 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.801	0.53	0.779	0.801	0.79	0.658	good
	0.47	0.199	0.504	0.47	0.486	0.658	bad
Weighted Avg.	0.702	0.431	0.696	0.702	0.699	0.658	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 561 139 |   a = good
 159 141 |   b = bad
```

Using oneR classifier

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	668	66.8 %
Incorrectly Classified Instances	332	33.2 %
Kappa statistic	0.0839	
Mean absolute error	0.332	
Root mean squared error	0.5762	
Relative absolute error	79.0142 %	
Root relative squared error	125.7359 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.866	0.793	0.718	0.866	0.785	0.536	good
	0.207	0.134	0.397	0.207	0.272	0.536	bad
Weighted Avg.	0.668	0.596	0.622	0.668	0.631	0.536	

=== Confusion Matrix ===

a	b	<-- classified as
606	94	a = good
238	62	b = bad

Exploring understanding Clementine Tool

INTRODUCING CLEMENTINE

CLEMENTINE AND CLEMENTINE SERVER

By default, Clementine will run in local mode on your desktop machine. If Clementine Server has been installed, then Clementine can be run in local mode or in distributed (Client-Server) mode. In this latter mode, Clementine streams are built on the client machine, but executed by Clementine Server.

To determine in which mode Clementine is running on your machine, examine the connection status area of the Clementine status bar (left-most area of status bar) or click Tools..Server Login(from within Clementine) if the choice is active; if it is not active, then

Clementine Server is not licensed. See that the Server Login dialog whether the Connection is set to Local or Network.

Note :

Data for this course are assumed to be stored in the directory c:\programfiles\SPSSClementine\10.1 .

STARTING CLEMENTINE

To run Clementine:

From the Start button, click **Programs..Clementine..Clementine**

Clementine enables you to mine data by visual programming techniques using the Stream Canvas.This is the main work area in Clementine and can be thought of as a surface on which to place icons. These icons represent operations to be carried out on the data and are often referred to as nodes.

The nodes are contained in palettes, located across the bottom of the Clementine window. Once nodes have been placesd on the Stream Canvas, they can be linked together to form a stream.

READING DATA FILES

READING DATA FILES INTO CLEMENTINE

Clementine reads a variety of different file types, including data stored in spreadsheets and databases, using the nodes within the Sources palette.

Data can be read in from text files, in either free-field or fixed-field format, using the Var. File and Fixed File source nodes.

Reading Data from Free-Field Text Files

The Var. File node reads data from a free-field (delimited) text file.

We will read this file into Clementine, using the Var File source node.

If the Stream Canvas isn't empty, clear the Stream Canvas by choosing **Edit..Clear Stream**

Click the **Var. File** node in the Sources palette

Position the cursor on the left side of the Stream Canvas and click once(not shown)
Right-click on the **Var. File** node, then click **Edit**

Click the file list button ☐ , and then move to the
c:\programfiles\SPSS\clementine\10.1 directory
Click Drug.txt and then click **Open**
Click **OK**
Click the **Table** node from the Output palette
Click to the right of the Var. File node named Drug.txt

Right-click the **Var. File** node, select **Connect** from the context pop-up menu, and then click the **Table** node in the Stream Canvas

Click the **Execute** button in the Toolbar

After having executed the Table using any of the methods mentioned, the table window opens.

Exercise : 1

1. Execute the above steps and at each step display the screen.
2. Read the Fixed data file and display the data output on the screen in the table format.
3. Read the data using ODBC.

DATA QUALITY

In Clementine there are a number of different types of representation of missing data. First, a field may be completely blank. Clementine calls such missing information white space string if the field is symbolic and null value if the field is numeric. Clementine also refers to this as a null value or non-numeric missing. Finally, when entering data, predefined codes may be used to represent missing or invalid information.Clementine refers to such codes as value balnks.

Assessing Data Quality

To illustrate the handling of missing data we will open Drug1n.txt and assess the quality of the data.

If the Stream Canvas is not empty, start a new stream by clicking **File..New Stream**
Select the **Var. File** node and place it on the Stream Canvas

Edit the node and set the file to **Drug1n.txt** held in the
c:\programfiles\SPSS\Clementine\10.1\Demos directory

Make sure the **Read field names from file** option is checked
Click the **Types** tab, then right-click any field and click **Select All** from the context menu
Right-click any field, and then click **Set Values..<Read>** from the context menu

Click **OK**
Add a Table node and connect the **Var. File** node to the **Type** node

Execute the **Table** node
Click **File..Close** to close the **Table** node
Double-click the **Var. File** node and click the **Types** tab
Place a **Quality** node from the Output palette into the Strem Canvas and connect the **Var.File** node to it.
Edit the **Quality** node
Deselect the **White space** option
Deselect **the Empty string** option
Click the **Execute** button

Exercise : 2

1. Execute the above steps and display the screens for each step.
2. For the above data check for all missing value types i.e Null value, Empty String, White Space, Blank Value and display the screens
3. Examine the distribution of data fields for drug4n.txt in Clementine by using Data Audit node.

DATA MANIPULATION

Once checking the quality of data has been completed it is often necessary to manipulate the data further.

Such techniques are available within Clementine and can be found in either the Record Ops palette (containing tools for manipulating records) or Field Ops palette (Containing tools for manipulating fields).

Record operations and the Select Node

The select node, which allows you to either select or eliminate a group of records based on a specified condition.

If the Stream Canvas is not empty, choose **File..Close Stream**

Click **File..Open Stream**, move to the **c:\programfiles\SPSS\Clementine\10.1\demos** directory and double-click **Drug4n**

Place a **Select** node from the Record Ops palette to the right of the **Type** node

Connect the **Type** node to the **Select** node

Right-click the **Select** node, then click **Edit**
Type **Age < 25** in the **Condition** text box
Check that **Mode:** is set to **Include**

Place a **Table** node from the Output palette to the right of the **Select** node
Connect the **Select** node to the new **Table** node
Right –click the new **Table** node, then click **Execute**

Next, using a Distribution node, we will compare the distribution of risk for the entire sample to the subgroup with Age < 25.

Click **File..Close** to close the Table window
Delete the **Table** node

Drag the **select** node below the **Type** node
Place **a Distribution** node from the Graphs palette to the right of the **Type** node
Connect the **type** node to the **Distribution** node
Double-click the **Distribution** node
Click the field list button and select **Drug4n**
Click **OK**
Copy the **Distribution** node and **Paste** it to the right of the **Select** node
Connect the **Select** node to the pasted **Distribution** node
Execute the two **Distribution** nodes.

Field Operations and the Filter Node

The Filter node allows data to pass through it and has two main functions:

- To filter out unwanted fields
- To rename fields

Place a **Filter** node from the **Field Ops** palette to the right of the **Type** node
Connect the **Type** node to the **filter** node
Right-click on the **Filter** node, and then click **Edit**

To change the Name of a Field

Click the text in the right column
Type the new name in the text box

To Filter Out Fields

Click on the arrow next to **Age**

Click **OK** to close the **Filter** dialog box

Field Reordering

Place a **Field Reorder** node from the Field Ops . Palette to the right of the **Filter** node
Connect the **Filter** node to **Field reorder** node
Right –click the **Field Reorder** node, and then click **Edit**
Click the field list button
Select **Na, Drug** in the Select Fields dialog
Click **OK**

Click **Drug** in the Field Reorder dialog
Click **Ok**

Place a **Table** node from the Output palette to the right of the **Field Reorder** node
Connect the **Field Rorder** node to the **Table** node
Right-click the new **Table** node, then click **Execute.**

The Derive Node

The Derive node calculates a new value, based on a CLEM expression for every record passed through it. To enter the CLEM expression and the new field name you need to edit the Derive node.

Click **Insert..Field Ops..Derive**
Right-click the **Derive** node, then click **Edit**
Select **Formula from the Derive as :** drop-down list
Type **SUM NAK** in the **Derive field**: text box
Click the Expression Builder button
Click **Na** in the Fields list box, then click the **Insert** button
Click the plus button
Click **Ki**n the Fields list box, then click the **Insert** button
Click the plus button

Click **OK**

Click **OK**

Exercise :3

1. For the above steps display the screens step wise.
2. Write the steps for Derive Type Flag and display the screens
3. Write the steps for Derive Type set and display the screens
4. Write the steps for Derive Type Conditional and display the screens
5. Write the steps for Derive Type Flag and display the screens

Reclassify Node

The Reclassify node allows you to recode or reclassiry the data values for fields of set or flag type.

Place a **Reclassify** node from the **Field Ops** palette to the right of the **Derive** node named **Drug** in the Stream Canvas
Connect the **Derive** node named **Drug** to the **Reclassify** node
Right-click the **Reclassify** node, then click **Edit**
Click the field list button for **Reclassify field** and select **Drug**

Type **Drug_Cat** in the **New field name** box
Click the **Get** button
Click the For unspecified values use: **Default value** option button
Type **Da** in the **New value box** for the **DrugA** row
Click in the **right side** of the **New value box** for the **DrugB** row, then click the drop-**down arrow**, and select **Db** from the drop-down value list
Type **Dc** in the **New value** box for the **DrugC** row

Click **OK**

Place a **Table** from the Output palette above the **Reckassify** node
Connect the **Reclassify** node to new **Table** node
Right-click the new **Tabl**e node, then click **Execute**

Exercise : 4

1. Generate the Screens at each step of Reclassify node.

RELATIONSHIPS IN DATA

There are two methods for examining whether symbolic fields are related. The first is the Matrix node used to display the relation between a pair of symbolic fields. We will extend this to more than two symbolic fields with the graphical Web node.

Matrix Node : Relating Two Symbolic Fields

The Matrix node performs crosstabulations of two symbolic fields within the data, and shows how values of one field relate to those of a second field.

Click File..Open Stream, move to c:\programsfile\SPSS\Clementine\10.1\demos directory and double-click on **Drug4n**
Place a **Matrix** node from the **Output** palette to the right of the **Type** node
Connect the **Type** node to the **Matrix** node
Double-click on the **Matrix** node

Click the Fields list button in the **Rows**: list and select **Drug**
Click the Fields list button in the **Columns:** list and select **Gender**

Click the **Appearance t**ab
Select **Counts**
Select **Percentage of column**

Click the **Execute** button

Click **file..Close** to close the Matrix output window
Double-click on the **Matrix** node, and then click the **settings** tab
Click the Fields list button in the Columns: list and select **Na**
Click **Execute**

Web Node

The Web node, located in the Graphs pallete, can be used to show the strength of connection between values of two or more symbolic fields.

Place a **Web** node from the **Graphs** palette near the **Type** node

Connect the **Type** node to the **Web** node
Double-click the **Web** node

Click the Field List button

Select **Age, Sex** and **Drug**
Click **Ok** to return to the Web dialog box

Click the **Line values are** drop-down list

Click the **options** tab

Click **show only links above** option button
Type **300** in the **Show only links** above box
Type **400** in the **Weak links below**: box
Type **600** in the **Strong links below**: box

Click **Execute**
Drag the lower corner of the resulting Web graph window to enlarge it.

Click >>
Use **Slider control** to discard links weaker than 450

Click the **Controls** tab in the Web Summary output
Set the **Web Display** option to **Size shows strong/normal/weak**
Use slider control or text box to set the value for **strong links above** to 1500 and the value for **weak links below to 1000**

Click the link connecting **DrugA and F**
Click the link connecting **DrugB and F**
Click **Generate..Derive Node("And")**

Click **File..Close** to close the Web graph window

Exercise : 5

1. Generate the screens at each step of Matrix node .
2. Generate the screens at each step of Web node .

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

(An Autonomous Institution approved by UGC and affiliated to JNTUH))

(Accredited by NAAC with 'A" Grade, Accredited by NBA of AICTE)

Yamnampet, Ghatkesar Mandal, Hyderabad - 501 301.

LAB MANUAL

For

COMPUTER NETWORKS

B. Tech. III year - I Semester

CSE Branch

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

JULY, 2020

1	2	3	4	5	6	7	8	9	10	11	12
	M	M									

Syllabus for B. Tech. III Year I semester
Computer Science and Engineering
Computer Networks Lab

	L	T	P	C
Code:7EC75	**0**	**0**	**4/2***	**1**

Course Objective:

To provide an understanding of the design concepts of framing Error Detection & correction, Routing, Congestion concepts and Network tools.

Course Outcomes:

At the end of this course, the student will be able to

1. Implement and analyze framing methods of data link layer.
2. Illustrate and implement error detection & correction techniques.
3. Implement and analyze different Routing Algorithm.
4. Demonstrate the execution of basic Network Commands.

Computer Networks Lab Exercises:

1. Implement the data link layer framing methods such as
   a) Character / Byte stuffing b) Bit stuffing.
2. Implement on a data set of characters the three CRC polynomials
   a) CRC 12 b) CRC 16 c) CRC CCITT.
3. Implement Hamming code for error detection and error correction
4. Implement Dijkstra's algorithm to compute the shortest path through a graph.
5. Take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table for each node using distance vector routing algorithm.
6. Implement Congestion control using Leaky-Bucket Algorithm
7. Execute the basic Networking Commands

   i. Arp
   ii. Hostname
   iii. ipconfig
   iv. ipconfig/all
   v. Ipconfig/renew
   vi. Ipconfig/release
   Vii. Ipconfig/flushdns
   viii. Pathping
   ix. Ping
   x. Route
   xi. tracert

<u>Beyond Syllabus</u>

1. Installation of NS-2
2. Demonstration of NS-2

INDEX

COMPUTER NETWORKS

Exercise-1

Aim: Implement the data link layer framing methods - such as character/byte stuffing and bit stuffing

Character/byte Stuffing

Algorithm: Stuffing

Steps:

1. Read in the bit stream.
2. Scan the bit stream from left to right.
3. Divide stream into 8-bit (ASCII) frames.
4. Append this ASCII character sequence with Start and End FLAG.
5. When there is accidental DLE character found, then insert ESC before each occurrence of it.
6. When there is accidental ESC character found, then insert ESC before each occurrence of it.

Algorithm: DeStuffing

Steps:

1. Read the character sent from sender
2. Scan characters from left to right.
3. When there are two consecutive ESC found or one ESC found before FLAG, simple discard one extra ESC from the frame.

Bit Stuffing

Algorithm: Stuffing

Steps:

1. Read in the bit stream.
2. Add FLAG byte with bit pattern 01111110 at start and end of frame.
3. Scan bit stream from left to right.
4. When stream of 5 consecutive 1's found, then 0 bit inserted into outgoing bit stream.

Algorithm: DeStuffing

Steps:

i) Read the bit stream from sender.

ii) Scan the bit stream from left to right.

iii) When stream for 5 consecutive 1's followed by 0 found, then delete that extra 0 bit from bit stream.

Exercise - 2

Aim: Implement on a data set of characters, CRC Technique–

Algorithm: Sending process:

Steps:

1) Read data (text/numeric) T.

2) Convert T into binary format B.

3) Shift the contents of B left by 12/16 times and store it in C.

4) Perform modulus division of C by P and store remainder in R.

5) Perform XOR of C with R and store result in X.

6) Send X to receiver

Algorithm: Receiving Process:

Steps:

1) Read X from Sending Device.

2) Perform modulus division of X by P and store remainder in R.

3) If R = 0 Then display "Received data is correct" else display "Received data is wrong".

Note: P is one of following generated polynomial

i) CRC-12, $P = x^{12} + x^4 + x^3 + x^2 + x + 1$ = 0000 1000 0000 1111

ii) CRC-16, $P = x^{16} + x^{12} + x^2 + 1$ = 0001 0000 0000 0101

iii) CRC-CCITT, $P = x^{16} + x^{12} + x^2 + 1$ = 0001 0000 0010 0001

Exercise - 3

Implement Hamming code for error detection and error correction

Hamming code is a popular error detection and error correction method in data communication. Hamming code can only detect 2 bit error and correct a single bit error which means it is unable to correct burst errors if may occur while transmission of data.

Hamming code uses redundant bits (extra bits) which are calculated according to the below formula:-

$2^r \geq m+r+1$
Where **r** is the number of redundant bits required and **m** is the number of data bits.
R is calculated by putting **r = 1, 2, 3 …** until the above equation becomes true.
R1 bit is appended at position 2^0
R2 bit is appended at position 2^1
R3 bit is appended at position 2^2 and so on.

These redundant bits are then added to the original data for the calculation of error at receiver's end.

At receiver's end with the help of even parity (generally) the erroneous bit position is identified and since data is in binary we take complement of the erroneous bit position to correct received data.

Respective index parity is calculated for **r1, r2, r3, r4** and so on.

Advantages of Hamming Code

1. Easy to encode and decode data at both sender and receiver end.
2. Easy to implement.

Disadvantages of Hamming Code

1. Cannot correct burst errors.
2. Redundant bits are also sent with the data therefore it requires more bandwidth to send the data.

Exercise - 4

Aim: Implement Dijkstar's algorithm to Compute the short term path thru a graph

Algorithm:

Steps:

1) Define S as set of nodes with initially it contain node A.

2) Define Cost (X) as the cost of cheapest route from A to X.

3) Initially cost(X) is cost of link from A to X.. If no such link exist, then it is infinity (say its value is –1)

4) For these nodes linked to A define prior (X) = A

5) Repeat

   1. Determine set of nodes not in S, but are connected to a node in S, call this set as W.
   2. Chose a node X in W for which cost(X) is minimizes.
   3. Add X to the set S
   4. For each v not in S do

      Cost(v) = minimum {cost(v), cost(x) + cost of link connecting X to V}

      If cost (v) is changed, define prior(v) = X

6) Until not all nodes in S

Exercise 5

Aim: Take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table for each node using distance vector routing algorithm.

Data Structures Used:
Create 6 structures each of which represents a node

```
Struct node
{
int neighbour1
int neighbour2
int neighbour3
}
```

i) Three elements of the structure represent 3 neighbouring node costs (Read from command prompt [6 x 2]
ii) For each node populate the array / distance vector with costs of neighbors and rest of the elements are undefined / (-1)
iii) Create a module where each node reads the distance vectors of all the other nodes and makes modifications to its own distance vector.
iv) Committee this process till exists.

Exercise 6

Implement Congestion control using Leaky-Bucket Algorithm

Algorithm:

1. Start
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)
6. Stop

```
1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<unistd.h>
4.
5. #define NOF_PACKETS 10
6.
7. int rand(int a)
8. {
9.     int rn = (random() % 10) % a;
10.    return  rn == 0 ? 1 : rn;
11. }
12.
13. int main()
14. {
15.    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
16.    for(i = 0; i<NOF_PACKETS; ++i)
17.        packet_sz[i] = rand(6) * 10;
18.    for(i = 0; i<NOF_PACKETS; ++i)
19.        printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
20.    printf("\nEnter the Output rate:");
21.    scanf("%d", &o_rate);
22.    printf("Enter the Bucket Size:");
23.    scanf("%d", &b_size);
24.    for(i = 0; i<NOF_PACKETS; ++i)
25.    {
26.        if( (packet_sz[i] + p_sz_rm) > b_size)
27.            if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
28.                printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
29.            else
```

```
            printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
        else
        {
            p_sz_rm += packet_sz[i];
            printf("\n\nIncoming Packet size: %d", packet_sz[i]);
            printf("\nBytes remaining to Transmit: %d", p_sz_rm);
            p_time = rand(4) * 10;
            printf("\nTime left for transmission: %d units", p_time);
            for(clk = 10; clk <= p_time; clk += 10)
            {
                sleep(1);
                if(p_sz_rm)
                {
                    if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                    printf("\nPacket of size %d Transmitted", op);
                    printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
                else
                {
                    printf("\nTime left for transmission: %d units", p_time-clk);
                    printf("\nNo packets to transmit!!");
                }
            }
        }
    }
}
```

Exercise 7

Execute the basic Networking Commands

i. Arp
ii. Hostname
iii. ipconfig
iv. ipconfig/all
v. Ipconfig/renew
vi. Ipconfig/release
Vii. Ipconfig/flushdns
viii. Pathping
ix. Ping
x. Route
xi. tracert

Exercise 8

Demonstrate the Downloading and Installation of Wire shark

Activity 1 - Download Wireshark

To download Wireshark:

1. Open a web browser.
2. Navigate to **http://www.wireshark.org**.
3. Select **Download Wireshark**.
4. Select the Wireshark Windows Installer matching your system type, either 32-bit or 64-bit as determined in Activity 1. Save the program in the Downloads folder.
5. Close the web browser.

Activity 3 - Install Wireshark

To install Wireshark:

1. Open Windows Explorer.
2. Select the Downloads folder.
3. Locate the version of Wireshark you downloaded in Activity 2. Double-click on the file to open it.
4. If you see a User Account Control dialog box, select **Yes** to allow the program to make changes to this computer.
5. Select **Next >** to start the Setup Wizard.
6. Review the license agreement. If you agree, select **I Agree** to continue.
7. Select **Next >** to accept the default components.
8. Select the shortcuts you would like to have created. Leave the file extensions selected. Select **Next >** to continue.
9. Select **Next >** to accept the default install location.
10. Select **Install** to begin installation.
11. Select **Next >** to install WinPcap.
12. Select **Next >** to start the Setup Wizard.
13. Review the license agreement. If you agree, select **I Agree** to continue.
14. Select **Install** to begin installation.
15. Select **Finish** to complete the installation of WinPcap.
16. Select **Next >** to continue with the installation of Wireshark.
17. Select **Finish** to complete the installation of Wireshark.

Exercise 9

Use Wire shark for packet capture and traffic analysis

i. How many packets passed through interface?
ii. Study of parameter in IP Header
iii. Filters
iv. Capture ARP & ICMP.

i. How many packets passed through interface?

Select a Network Interface to Capture Packets through. Start the Wireshark application. When Wireshark is first run, a default, or blank window is shown. To list the available network interfaces, select the Capture->Interfaces menu option.

Wireshark should display a popup window such as the one shown in Figure 2. To capture network traffic click the Start button for the network interface you want to capture traffic on. Windows can have a long list of virtual interfaces, before the Ethernet Network Interface Card (NIC).

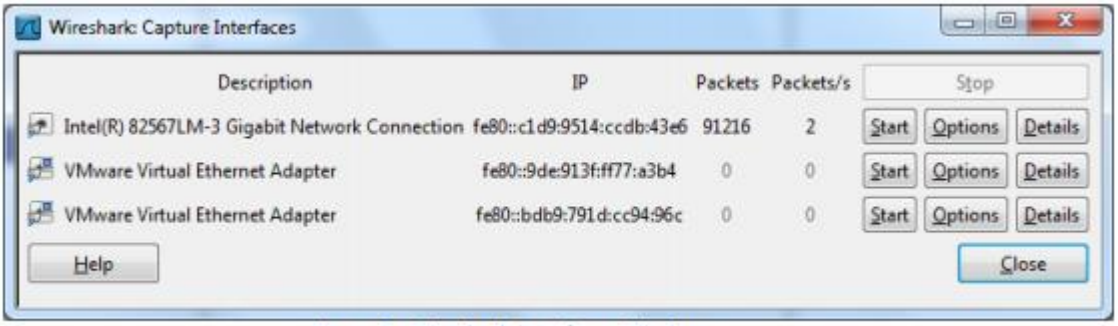


Figure 2 - Wireshark Interfaces Window

Generate some network traffic with a Web Browser, such as Internet Explorer or Chrome. Your Wireshark window should show the packets, and now look something like.

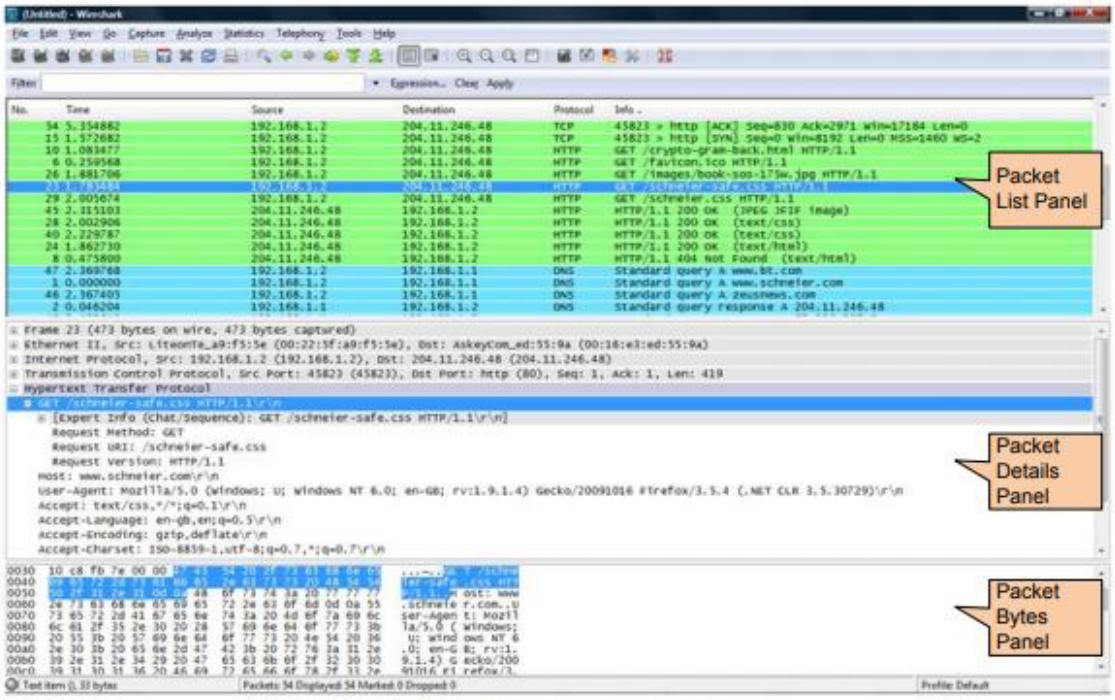
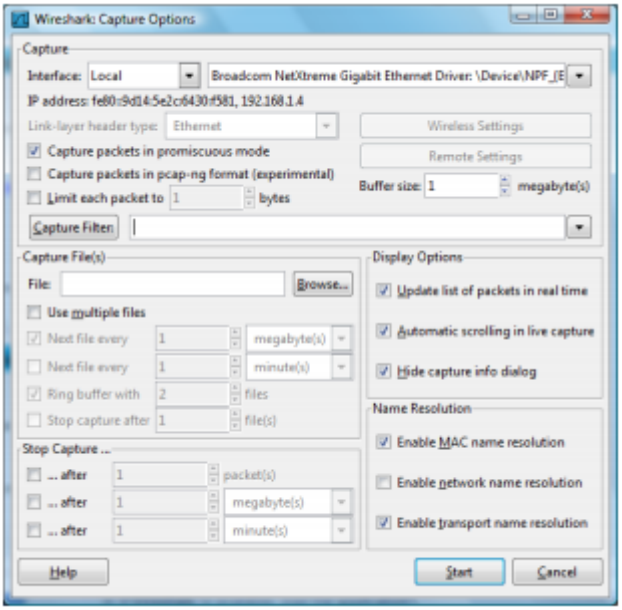


Figure 3 - Wireshark capuring traffic

To stop the capture, select the Capture->Stop menu option, Ctrl+E, or the Stop toolbar button. What you have created is a Packet Capture or 'pcap', which you can now view and analyse using the Wireshark interface, or save to disk to analyse later. The capture is split into 3 parts: 1. Packet List Panel – this is a list of packets in the current capture. It colours the packets based on the protocol type. When a packet is selected, the details are shown in the two panels below. 2. Packet Details Panel – this shows the details of the selected packet. It shows the different protocols making up the layers of data for this packet. Layers include Frame, Ethernet, IP, TCP/UDP/ICMP, and application protocols such as HTTP. 3. Packet Bytes Panel – shows the packet bytes in Hex and ASCII encodings.

To select more detailed options when starting a capture, select the Capture->Options menu option, or Ctrl+K, or the Capture Options button on the toolbar (the wrench). This should show a window such as shown in Figure 4. Figure 4 - Wireshark Capture Options Some of the more interesting options are: Capture Options• > Interface - Again the important thing is to select the correct Network Interface to capture traffic through. Capture Options• > Capture File – useful to save a file of the packet capture in real time, in case of a system crash. Display Options• > Update list of packets in real time – A display option, which should be checked if you want to view the capture as it happens (typically switched off to capture straight to a file, for later analysis). Name Resolution• > MAC name resolution – resolves the first 3 bytes of the MAC Address, the Organisation Unique Identifier (OUI), which represents the Manufacturer of the Card. Name Resolution• > Network name resolution – does a DNS lookup for the IP Addresses captured, to display the network name. Set to off by default, so covert scans do not generate this DNS traffic, and tip off who's packets you are sniffing. Make sure the MAC name resolution is selected. Start the capture, and generate some Web traffic again, then stop the capture.



iii. Filters

Wireshark Display Filters. Right click on the Source Port field in the Packet Details Panel. Select Prepare a Filter->Selected.

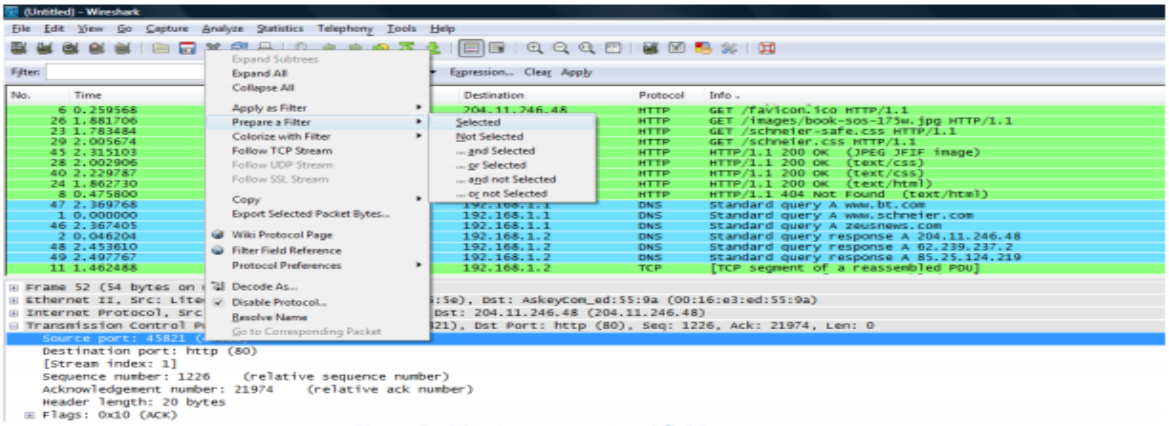


Figure 5 - Filtering on a protocol field

Wireshark automatically generates a **Display Filter**, and applies it to the capture. The filter is shown in the **Filter Bar**, below the button toolbar. Only packets captured with a Source Port of the value selected should be displayed. The window should be similar to that shown in Figure 6. This same process can be performed on most fields within Wireshark, and can be used to include or exclude traffic.

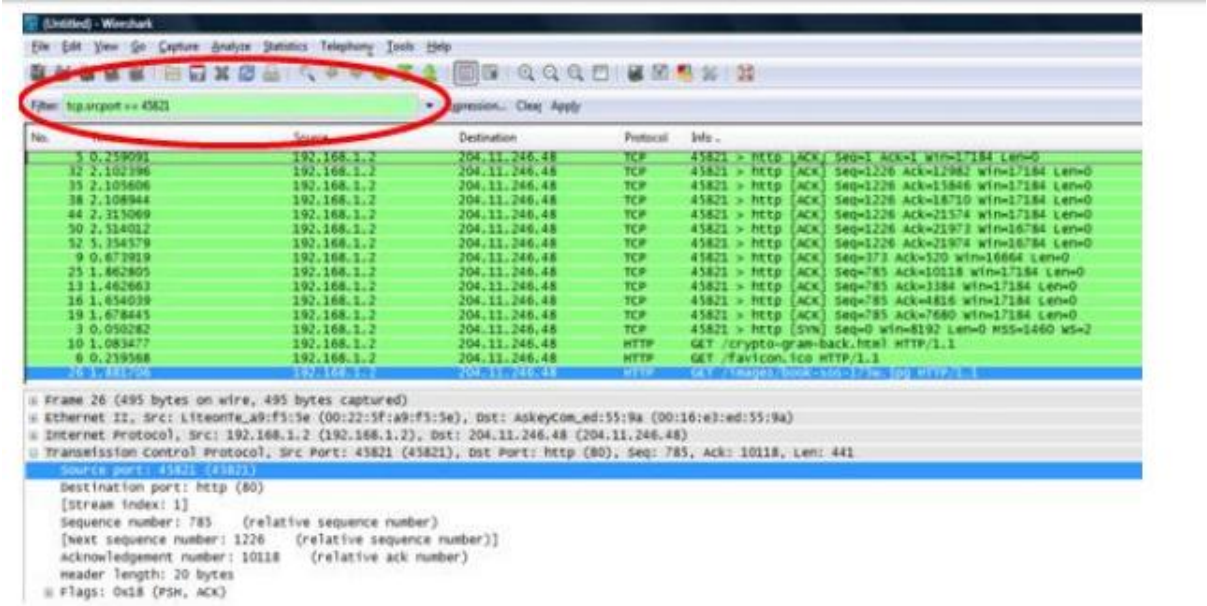
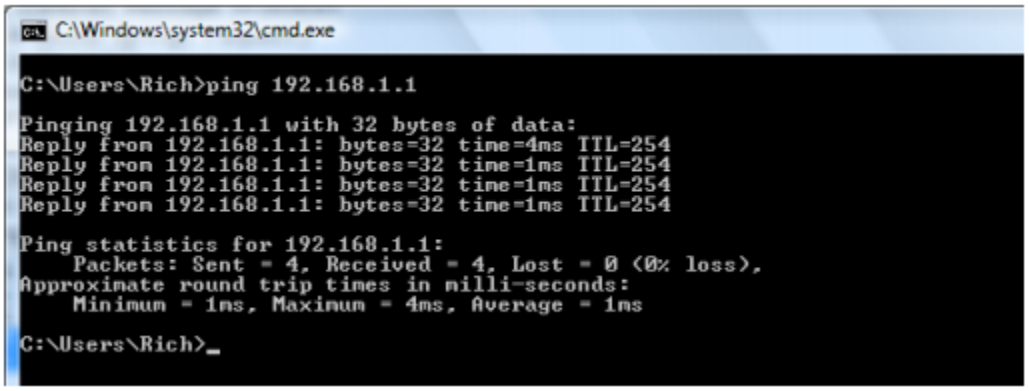


Figure 6 - Wireshark Display Filter

iv. Capture ARP & ICMP.

Start a Wireshark capture. Open a Windows console window, and generate some ICMP traffic by using the Ping command line tool to check the connectivity of a neighbouring machine (or your home router).

Stop the capture and Wireshark should now look something like Figure 10. The Address Resoloution Protocol (ARP) and ICMP packets are difficult to pick out, create a display filter to only show ARP or ICMP packets.



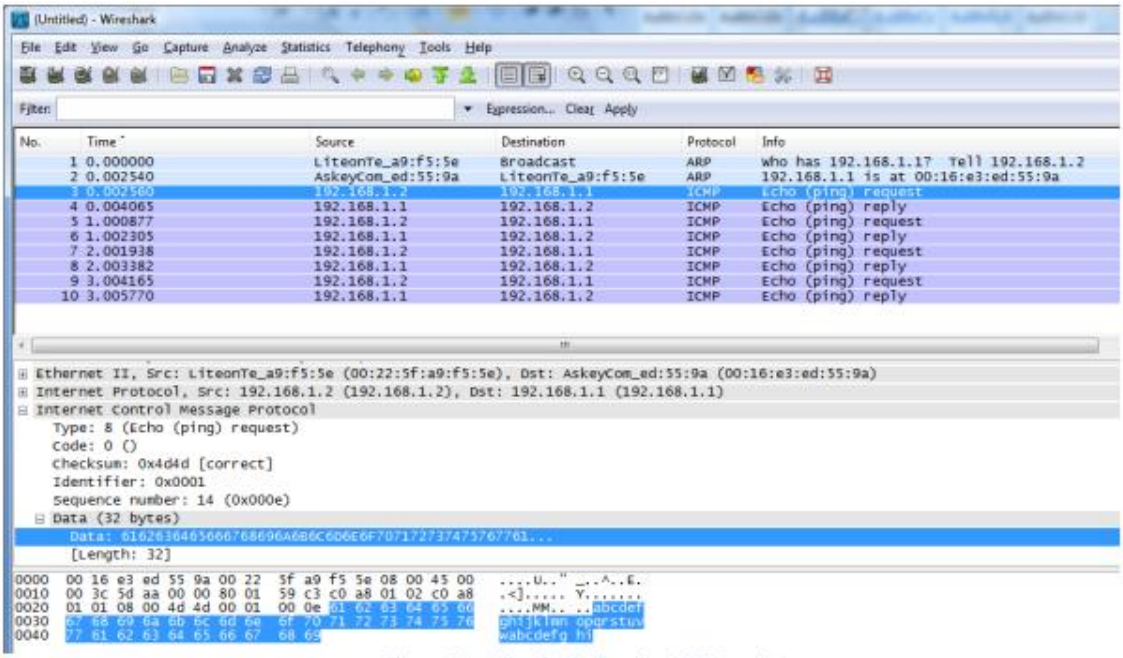


Figure 10 - Wireshark showing ICMP packets

Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address 192.168.1.1, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

If pinging the same system more than once, delete the ARP cache on your system, using the arp command, as shoen below, so an new ARP request will be generated. C:\> ping 192.168.1.1 ... ping output ... C:\> arp –d * Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address 192.168.1.1, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

Exercise 10 & 11

Installation and demonstration of NS-2

Aim: To study NS2 simulator, setup simulation scenario and run basic TCL script.

Overview

NS is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UPD, traffic source behavior such

as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations.

The NS project is a part of the VINT project that develops tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. NS (version 2) written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT) .

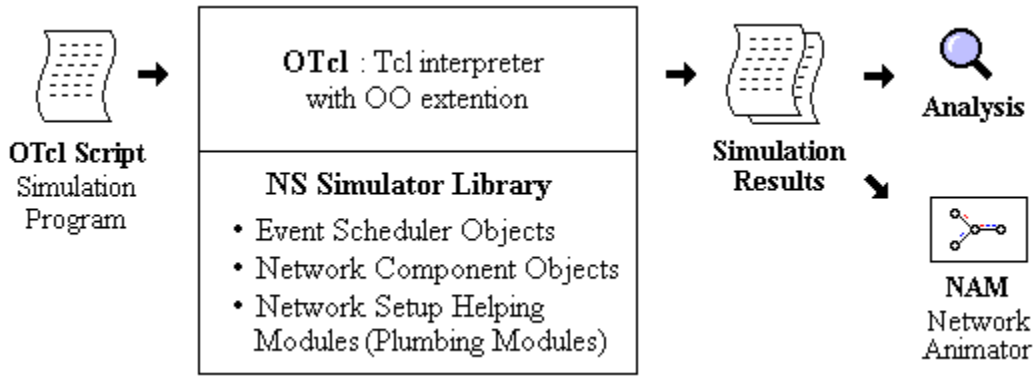


Figure 1. Simplified User's View of NS

NS is Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries (actually, plumbing modules are implemented as member functions of the base simulator object). OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler.

The term "plumbing" is used for a network setup, because setting up a network is plumbing possible data paths among network objects by setting the "neighbor" pointer of an object to the address of an appropriate object.

Another major component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event.

Network components communicate with one another passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling a packet (i.e. need a delay) use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet.

For example, a network switch component that simulates a switch with 20 microseconds of switching delay issues an event for a packet to be switched to the scheduler as an event 20 microsecond later. The scheduler after 20 microsecond dequeues the event and fires it to the switch component, which then passes the packet to an appropriate output link component.

Another use of an event scheduler is timer. For example, TCP needs a timer to keep track of a packet transmission time out for retransmission (transmission of a packet with the same TCP packet number but different NS packet ID). Timers use event schedulers in a similar manner that delay does. The only difference is that timer measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay.

NS is written not only in OTcl but in C++ also. For efficiency reason, NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the data path are written and compiled using C++.

These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object.

In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl.

PROGRAM:

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the NAM trace file
close $nf
```

```
#Execute NAM on the trace file
exec nam out.nam &
exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Setup a UDP connection
set udp [new Agent/UDP]
```

```
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"


#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

DESCRIPTION:

The following is the explanation of the script above. In general, an NS script starts with making a Simulator object instance.

- **set *ns* [new Simulator]:** generates an NS simulator object instance, and assigns it to variable *ns* (italics is used for variables and values in this section). What this line does is

the following:

- Initialize the packet format (ignore this for now)
- Create a scheduler (default is calendar scheduler)
- Select the default address format (ignore this for now)

The "Simulator" object has member functions that do the following

- Create compound objects such as nodes and links (described later)
- Connect network component objects created (ex. attach-agent)
- Set network component parameters (mostly for compound objects)
- Create connections between agents (ex. make connection between a "tcp" and "sink")
- Specify NAM display options

Most of member functions are for simulation setup (referred to as plumbing functions) and scheduling, however some of them are for the NAM display.

- ***$ns* color *fid color*:** is to set color of the packets for a flow specified by the flow id (fid). This member function of "Simulator" object is for the NAM display,and has no effect on the actual simulation.

- ***$ns* namtrace-all *file-descriptor***: This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command *$ns* flush-trace. Similarly, the member function trace-all is for recording the simulation trace in a general format.

- **proc *finish* {}:** is called after this simulation is over by the command *$ns* at 5.0 "*finish*". In this function, post-simulation processes are specified.

- **set *n0* [*$ns* node]:** The member function node creates a node. A node in NS is compound object made of address and port classifiers. Users can create a node by separately creating an address and a port classifier objects and connecting them together.
- ***$ns* duplex-link *node1 node2 bandwidth delay queue-type***: creates two simplex links of specified bandwidth and delay, and connects the two specified nodes. In NS, the output queue of a node is implemented as a part of a link, therefore users should specify the queue-type when creating links. In the above simulation script, DropTail queue is used.. One thing to note is that you can insert error modules in a link component to simulate a lossy link (actually users can make and insert any network objects).
- ***$ns* queue-limit *node1 node2 number***: This line sets the queue limit of the two simplex links that connect node1 and node2 to the number specified.
- ***$ns* duplex-link-op *node1 node2 ...***: The next couple of lines are used for the NAM display.

- Now that the basic network setup is done, the next thing to do is to setup traffic agents such as TCP and UDP, traffic sources such as FTP and CBR, and attach them to nodes and agents respectively.

- **set *tcp* [new *Agent/TCP*]:** This line shows how to create a TCP agent. But in general, users can create any agent or traffic sources in this way. Agents and traffic sources are in fact basic objects mostly implemented in C++ and linked to OTcl. Therefore, there are no specific Simulator object member functions that create these object instances. To create agents or traffic sources, a user should know the class names these objects (Agent/TCP, Agnet/TCPSink, Application/FTP and so on). This file contains the default configurable parameter value settings for available network objects. Therefore, it works as a good indicator of what kind of network objects are available in NS and what are the configurable parameters.

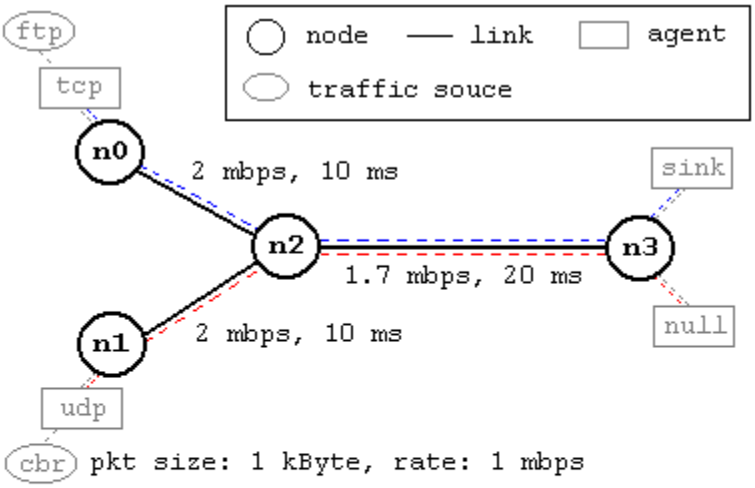
- ***$ns* attach-agent *node agent***: The attach-agent member function attaches an agent object created to a node object. Actually, what this function does is call the attach member function of specified node, which attaches the given agent to itself. Therefore, a user can do the same thing by, Similarly, each agent object has a member function attach-agent that attaches a traffic source object to itself.

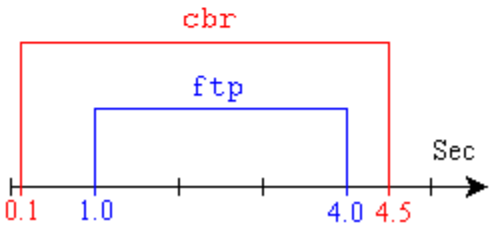
- ***$ns* connect *agent1 agent2***: After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination address to each others' network and port address pair.

  Assuming that all the network configuration is done, the next thing to do is write a simulation scenario (i.e. simulation scheduling). The Simulator object has many scheduling member functions. However, the one that is mostly used is the following:

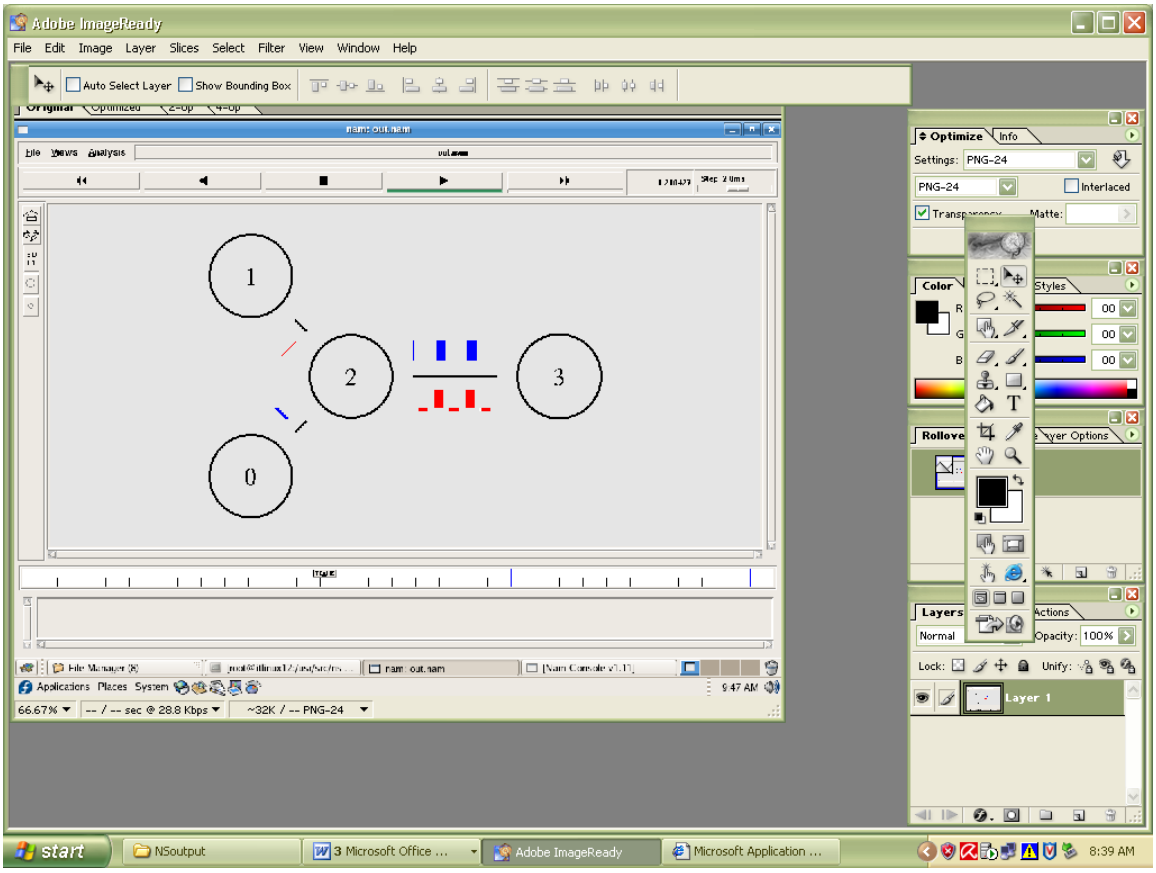
- ***$ns* at *time "string"***: This member function of a Simulator object makes the scheduler to schedule the execution of the specified string at given simulation time. For example, *$ns* at *0.1 "$cbr start"* will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.

MODEL:


ftp
tcp
node
link
agent
traffic souce
n0
2 mbps, 10 ms
sink
n2
n3
1.7 mbps, 20 ms
null
n1
2 mbps, 10 ms
udp
cbr
pkt size: 1 kByte, rate: 1 mbps


cbr
ftp
Sec
0.1
1.0
4.0
4.5

NAM OUTPUT:


Adobe ImageReady
File Edit Image Layer Slices Select Filter View Window Help
Auto Select Layer
Show Bounding Box
1
2
3
0
Optimize
Info
Settings: PNG-24
Interlaced
Color
Styles
Layers
Actions
Normal
Opacity: 100%
Lock:
Unify:
Layer 1
start
NSoutput
3 Microsoft Office ...
Adobe ImageReady
Microsoft Application ...
8:39 AM

Design and Analysis of Algorithms

List of Programs for Lab

1. Write a C program to implement Merge sort algorithm for sorting a list of integers in ascending order.
2. Write a C program to implement Dijkstra's algorithm for the single source shortest path problem.
3. Write a C program that implements Prim's algorithm to generate minimum cost spanning tree.
4. Write a C program to implement greedy algorithm for job sequencing with deadlines.
5. Write a C program to implement Dynamic Programming algorithm for the 0/1 knapsack problem.
6. Write a C program to implement Dynamic programming algorithm for the Optimal Binary search Tree problem.
7. Write a C program to implement backtracking algorithm for n-queens problems.

Other Practice Programs:

1. Write a C program to implement Quick Sort algorithm for sorting a list of integers in ascending order.
2. Write a C program to implement the DFS algorithm for a Graph.
3. Write a C program to implement the BFS algorithm for a graph.
4. Write a C program that implements kruskal's algorithm to generate minimum cost spanning tree.
5. Write a C program to implement Floyd;s algorithm for all pairs shortest path problem.
6. Write a C program to implement the backtracking algorithm for the Hamiltonian circuit's problem.
7. Write a C program to implement backtracking algorithm for the sum of subsets problem.

1. Write a C program to implement Merge sort algorithm for sorting a list of integers in ascending order.

```
#include<stdio.h>
#include<stdlib.h>

void mergesort(int[], int, int);

void merge(int[], int, int);

void main()

{
	int n, a[10], i, j;

	printf("enter the size of the array\n");

	scanf("%d",&n);

	printf("enter the array elements\n");

	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}

	mergesort(a,0,n-1);

	printf("array after sorting is \n");

	for(i=0;i<n;i++)
	{
		printf("%d\n",a[i]);
	}
}

void mergesort(int a[], int low, int high)

{
	int mid;

	if(low>=high)
	{
		return;
```

```
        }
        else
        {
              mid = (low + high)/2;

               mergesort(a, low, mid);
               mergesort(a, mid+1, high);
               merge(a, low, high);

        }
}

void merge(int a[], int low, int high)

{
    int mid = (low + high)/2;

    int k,i, j, temp[10];

    i = low;

    j = mid+1;

    k = low;

    while(i<=mid && j<= high)
    {
        if(a[i] < a[j])
        {
            temp[k++] = a[i++];
        }

        else
        {
            temp[k++] = a[j++];
        }
    }

    while(i<=mid)
    {
        temp[k++] = a[i++];
    }

    while(j<=high)
    {
        temp[k++] = a[j++];
```

```
    }


    for(i=low;i<=high;i++)
    {
        a[i] = temp[i];
    }
}
```

Output

```
enter the size of the array
5
enter the array elements
1
4
2
8
7
array after sorting is
1
2
4
7
8
```

2. Write a C program to implement Dijkstra's algorithm for the single source shortest path problem.

```
#include<stdio.h>
#include<stdlib.h>
 void main()

{
	int i, j, n, startn, nextn, count, minc, mindistance;

	int c[10][10], dist[10], visit[10], pre[10];

	printf("enter the no. of vertices\n");

	scanf("%d",&n);

	printf("enter the matrix\n");

	for(i=0;i<n;i++)

	{
		for(j=0;j<n;j++)
		{
			scanf("%d",&c[i][j]);

			if(c[i][j] == 0)
			{
				c[i][j] = 9999;
			}

		}
	}

	printf("enter the startnode\n");

	scanf("%d",&startn);



	for(i=0;i<n;i++)
	{
		dist[i] = c[startn][i];

		visit[i] = 0;
```

```
        pre[i] = startn;
}

count = 1;

dist[startn] = 0;

visit[startn] = 1;

while(count < n)

{
        mindistance = 9999;

        for(i=0;i<n;i++)
        {
                if(dist[i] < mindistance && !visit[i])
                {
                        mindistance = dist[i];

                        nextn = i;
                }
        }

        visit[nextn] = 1;

        for(i=0;i<n;i++)
        {
                if(!visit[i])
                {
                        if(mindistance + c[nextn][i] < dist[i])
                        {
                                dist[i] = mindistance + c[nextn][i];

                                pre[i] = nextn;
                        }
                }
        }

        count++;
}

for(i=0;i<n;i++)
{
        if(i!=startn)
```

```
            {
                printf("\ndistance of %d is %d\n", i, dist[i]);

                printf("path = %d",i);

                 j = i;

                do
                {
                    j = pre[j];

                    printf("<-- %d",j);
                }
                while(j!=startn);
            }
        }
}
```

Output

```
enter the no. of vertices
4

enter the matrix
0
10
0
2
10
0
6
0
0
6
0
8
2
0
8
0
```

```
enter the startnode
0

distance of 1 is 10

path = 1<-- 0

distance of 2 is 10

path = 2<-- 3<-- 0

distance of 3 is 2

path = 3<-- 0
```

3. Write a C program that implements Prim's algorithm to generate minimum cost spanning tree.

```
#include<stdio.h>
#include<stdlib.h>

int main()

{
    int n, ne = 1, i, j;

    int c[10][10], traversed[10] = {0};

    int minimum, minc = 0, a, b, x, y;

    printf("enter no. of vertices\n");
```

```
scanf("%d",&n);

printf("enter matrix\n");

for(i=1;i<=n;i++)
{
	for(j=1;j<=n;j++)
	{
		scanf("%d",&c[i][j]);

		if(c[i][j] == 0)
		{
			c[i][j] = 999;
		}
	}
}

traversed[1] = 1;

while(ne < n)
{
	minimum = 999;

	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			if(c[i][j] < minimum)
			{
				if(traversed[i] != 0)
				{
					minimum = c[i][j];

					a = x = i;

					b = y = j;
				}
			}
```

```
            }
        }

        if(traversed[x] == 0 || traversed[y] == 0)
        {
            printf("%d edge(%d, %d) = %d\n",ne++, a, b, minimum);

            minc = minc + minimum;

            traversed[b] = 1;
        }

        c[a][b] = c[b][a] = 999;

    }

    printf("\n total cost = %d",minc);

    return 0;
}
```

Output

```
enter the no. of vertices
4
enter the matrix
0
10
0
2
10
0
6
0
0
6
0
8
2
0
8
0
```

```
1 edge(1, 4) : 2
2 edge(4, 3) : 8
3 edge(3, 2) : 6

total cost = 16
```

4. **Write a C program to implement greedy algorithm for job sequencing with deadlines.**

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef struct {
   char id;
   int deadline;
   int profit;

}Job;

int compareJob(const void *a, const void *b){
   //Will return true if a's profit > b's profit
   //else will return false
   return ((Job*)a)->profit - ((Job*)a)->profit;
}

void bestJob(Job jobs[],int sizeOfJobs){
   char jobsToDo[5]= {'\0'}; //Assign every element of array to '\0'-Only works in
few compilers
   //If above line do not work use for loop to assign '\0' to every element
   int i, k;

   for(i=0; i< sizeOfJobs; i++){
         k= jobs[i].deadline-1;
      //Searching for empty date nearest to deadline backwards
      while(jobsToDo[k] != '\0' && k >= 0){
         k--;
      }
      if(k != -1)
         jobsToDo[k]= jobs[i].id;
   }
```

```
    printf("Best order and jobs to do is \n");
    k=0;
    while(jobsToDo[k] != '\0'){
          printf("%c ",jobsToDo[k]);
          k++;
       }
}

void display(Job jobs[],int n){
    int i;
    printf("Job Id: \t");
    for(i=0; i<n; i++){
       printf("%c \t",jobs[i].id);
    }
    printf("\n");

    printf("Job Deadline: \t");
    for(i=0; i<n; i++){
       printf("%d \t",jobs[i].deadline);
    }
    printf("\n");

    printf("Job Profit: \t");
    for(i=0; i<n; i++){
        printf("%d \t",jobs[i].profit);
    }
    printf("\n");
}
int main()
{
    Job jobs[]=  {{'w', 1, 19}, {'v', 2, 100}, {'x', 2, 27},
                {'y', 1, 25}, {'z', 3, 15}};
    display(jobs,5);

    //sorting jobs[] w.r.t profit
    qsort(jobs,5,sizeof(jobs[0]),compareJob);
```

```
    bestJob(jobs,5);
    return 0;
}
```

Output

```
Job Id:          w    v    x    y    z
Job Deadline:    1    2    2    1    3
Job Profit:      19   100  27   25   15

Best order and jobs to do is
x  v  z
```

5. Write a C program to implement Dynamic Programming algorithm for the 0/1 knapsack problem.

```
#include <stdio.h>


// A utility function that returns
// maximum of two integers
int max(int a, int b) { return (a > b) ? a : b; }


// Returns the maximum value that can be
// put in a knapsack of capacity W
int knapSack(int W, int wt[], int val[], int n)
{
   // Base Case
   if (n == 0 || W == 0)
      return 0;


   // If weight of the nth item is more than
   // Knapsack capacity W, then this item cannot
   // be included in the optimal solution
   if (wt[n - 1] > W)
      return knapSack(W, wt, val, n - 1);
```

```
    // Return the maximum of two cases:
    // (1) nth item included
    // (2) not included
    else
        return max(
            val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),
            knapSack(W, wt, val, n - 1));
}

// Driver program to test above function
int main()
{
    int val[] = { 60, 100, 120 };
    int wt[] = { 10, 20, 30 };
    int W = 50;
    int n = sizeof(val) / sizeof(val[0]);
    printf("%d", knapSack(W, wt, val, n));
    return 0;
}
```

Output:
220

6. Write a C program to implement Dynamic programming algorithm for the Optimal Binary search Tree problem.

```
#include <stdio.h>
#include <limits.h>

// A utility function to get sum of array elements  freq[i] to freq[j]
int sum(int freq[], int i, int j);

/* A Dynamic Programming based function that calculates
  minimum cost of a Binary Search Tree. */
int optimalSearchTree(int keys[], int freq[], int n)
{
   /* Create an auxiliary 2D matrix to store results   of subproblems */
   int cost[n][n];

   /* cost[i][j] = Optimal cost of binary search tree  that can be  formed from keys[i]
         to keys[j].  cost[0][n-1] will store the resultant cost */

   // For a single key, cost is equal to frequency of the key
   for (int i = 0; i < n; i++)
      cost[i][i] = freq[i];

   // Now we need to consider chains of length 2, 3, ... . L is chain length.
   for (int L=2; L<=n; L++)
   {
      // i is row number in cost[][]
      for (int i=0; i<=n-L+1; i++)
      {
         // Get column number j from row number i and    chain length L
         int j = i+L-1;
         cost[i][j] = INT_MAX;

         // Try making all keys in interval keys[i..j] as root
         for (int r=i; r<=j; r++)
         {
           // c = cost when keys[r] becomes root of this subtree
           int c = ((r > i)? cost[i][r-1]:0) +
                   ((r < j)? cost[r+1][j]:0) +
                   sum(freq, i, j);
           if (c < cost[i][j])
              cost[i][j] = c;
         }
      }
```

```
    }
    return cost[0][n-1];
}

// A utility function to get sum of array elements  freq[i] to freq[j]
int sum(int freq[], int i, int j)
{
    int s = 0;
    for (int k = i; k <=j; k++)
       s += freq[k];
    return s;
}

// Driver program to test above functions
int main()
{
    int keys[] = {10, 12, 20};
    int freq[] = {34, 8, 50};
    int n = sizeof(keys)/sizeof(keys[0]);
    printf("Cost of Optimal BST is %d ",
                optimalSearchTree(keys, freq, n));
    return 0;
          }
```

Output:

Cost of Optimal BST is 142

7. **Write a C program to implement backtracking algorithm for n-queens problems**.

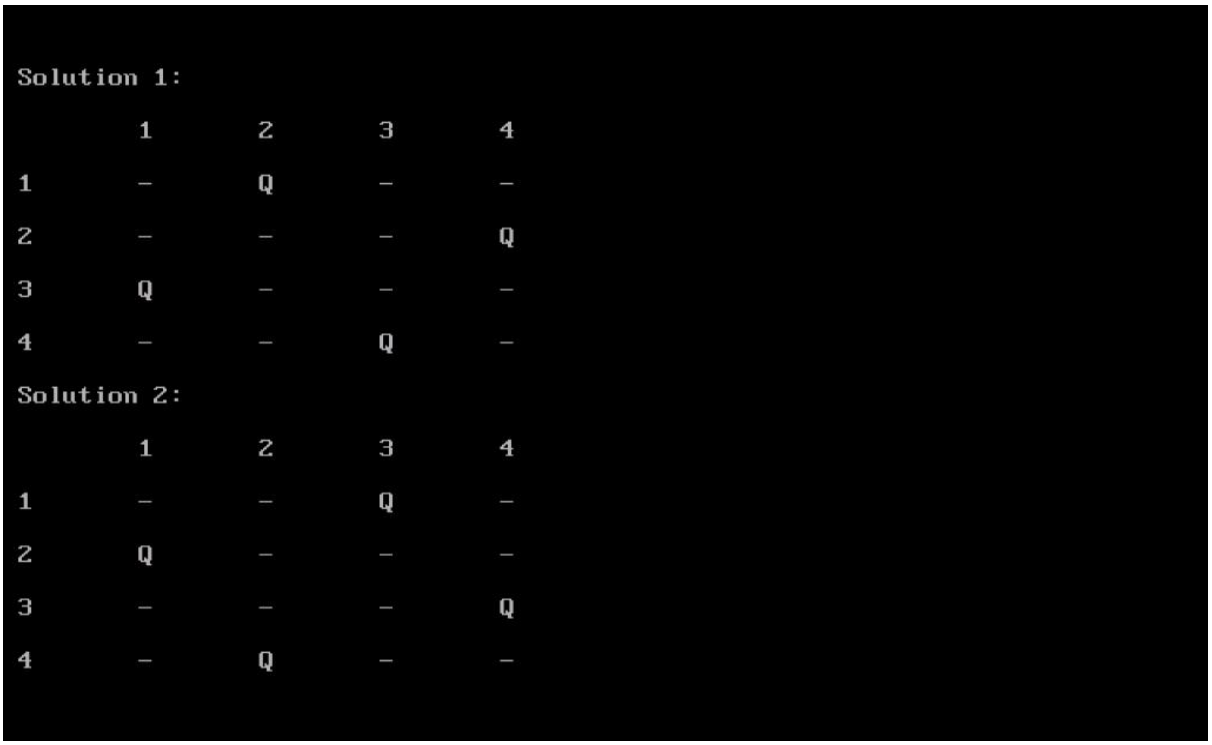
```
      #include<stdio.h>
#include<math.h>
int board[20],count;

int main()
{
 int n,i,j;
 void queen(int row,int n);
```

```
 printf(" - N Queens Problem Using Backtracking -");
 printf("\n\nEnter number of Queens:");
 scanf("%d",&n);
 queen(1,n);
 return 0;
}

//function for printing the solution
void print(int n)
{
 int i,j;
 printf("\n\nSolution %d:\n\n",++count);

        for(i=1;i<=n;++i)
                printf("\t%d",i);

        for(i=1;i<=n;++i)
        {
        printf("\n\n%d",i);
        for(j=1;j<=n;++j) //for nxn board
        {
        if(board[i]==j)
        printf("\tQ"); //queen at i,j position
        else
        printf("\t-"); //empty slot
        }
        }
   }

/*funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0*/
int place(int row,int column)
{
 int i;
         for(i=1;i<=row-1;++i)
        {
        //checking column and digonal conflicts
        if(board[i]==column)
                return 0;
        else
                if(abs(board[i]-column)==abs(i-row))
```

```
                    return 0;
        }
 return 1; //no conflicts
}
//function to check for proper positioning of queen
void queen(int row,int n)
{
 int column;
        for(column=1;column<=n;++column)
        {
        if(place(row,column))
                {
                board[row]=column; //no conflicts so place queen
                        if(row==n) //dead end
                                print(n); //printing the board configuration
                        else //try queen with next position
                                queen(row+1,n);
                }
        }
}
```

OUTPUT:

Enter number of Queens:4



Other Practice Programs:

1. **Write a C program to implement Quick Sort algorithm for sorting a list of integers in ascending order.**

```
/* This function takes last element as pivot, places
   the pivot element at its correct position in sorted
   array, and places all smaller (smaller than pivot)
   to left of pivot and all greater elements to right
   of pivot */
   int partition (int arr[], int low, int high)
   {
   int pivot = arr[high];    // pivot
   int i = (low - 1);  // Index of smaller element

   for (int j = low; j <= high- 1; j++)
   {
            // If current element is smaller than the pivot
            if (arr[j] < pivot)
            {
                  i++;    // increment index of smaller element
                  swap(&arr[i], &arr[j]);
```

```
            }
}
swap(&arr[i + 1], &arr[high]);
return (i + 1);
}

/* The main function that implements QuickSort
arr[] --> Array to be sorted,
low  --> Starting index,
high  --> Ending index */
void quickSort(int arr[], int low, int high)
{
        if (low < high)
        {
                /* pi is partitioning index, arr[p] is now
                at right place */
                int pi = partition(arr, low, high);

                // Separately sort elements before
                // partition and after partition
                quickSort(arr, low, pi - 1);
                quickSort(arr, pi + 1, high);
        }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
        int i;
        for (i=0; i < size; i++)
                printf("%d ", arr[i]);
        printf("n");
}

// Driver program to test above functions
int main()
{
        int arr[] = {10, 7, 8, 9, 1, 5};
        int n = sizeof(arr)/sizeof(arr[0]);
        quickSort(arr, 0, n-1);
        printf("Sorted array: n");
        printArray(arr, n);
        return 0;
}
```

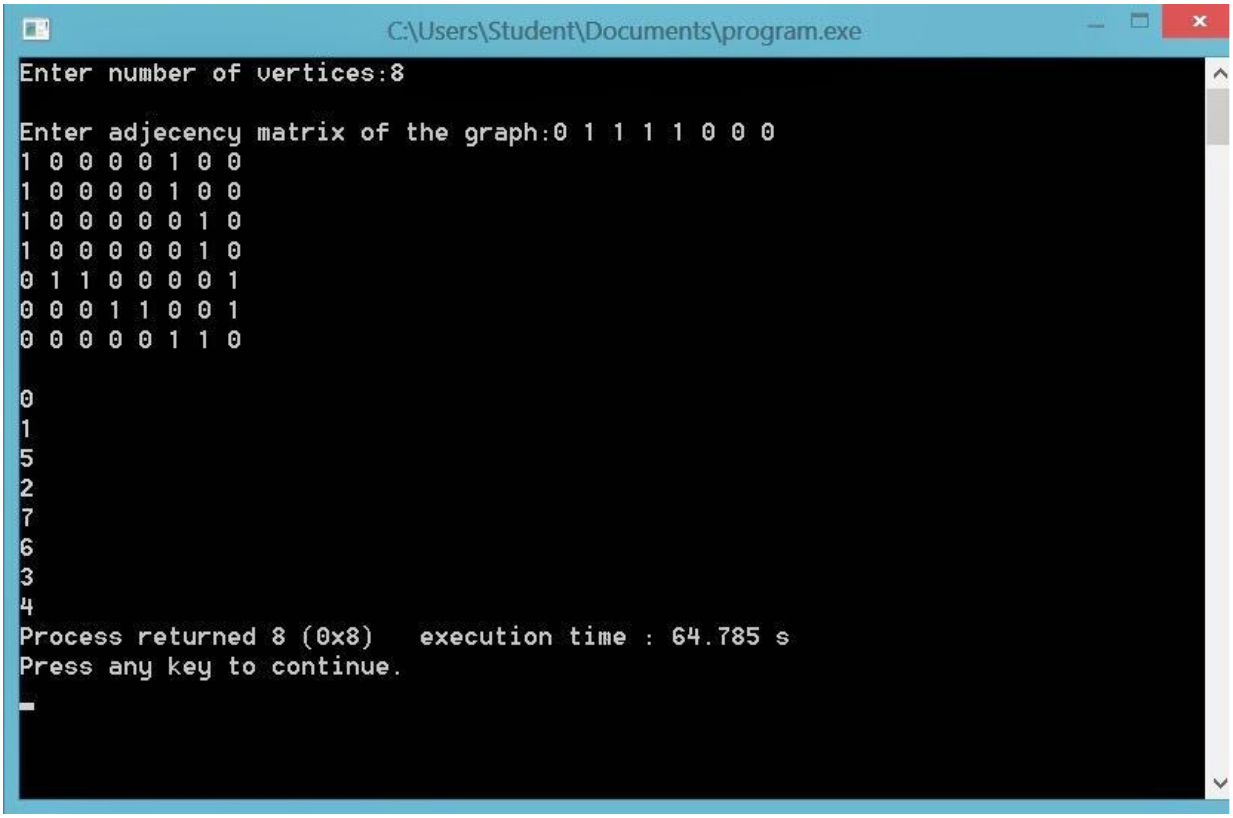
OUTPUT:

Sorted array:
1 5 7 8 9 10

2.Write a C program to implement the DFS algorithm for a Graph.

```
#include<stdio.h>

void DFS(int);
int G[10][10],visited[10],n;    //n is no of vertices and graph is sorted in array G[10][10]

void main()
{
int i,j;
printf("Enter number of vertices:");
scanf("%d",&n);
//read the adjecency matrix
printf("\nEnter adjecency matrix of the graph:");
    for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                scanf("%d",&G[i][j]);

//visited is initialized to zero
    for(i=0;i<n;i++)
            visited[i]=0;

    DFS(0);
}

void DFS(int i)
{
int j;
printf("\n%d",i);
visited[i]=1;
    for(j=0;j<n;j++)
            if(!visited[j]&&G[i][j]==1)
                    DFS(j);
  }
```

OUTPUT:



3. Write a C program to implement the BFS algorithm for a graph.

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v)
{
 for(i=1;i<=n;i++)
  if(a[v][i] && !visited[i])
     q[++r]=i;
 if(f<=r)
 {
  visited[q[f]]=1;
```

```
  bfs(q[f++]);
 }
}
void main()
{
 int v;
 clrscr();
 printf("\n Enter the number of vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  q[i]=0;
  visited[i]=0;
 }
 printf("\n Enter graph data in matrix
 form:\n"); for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  scanf("%d",&a[i][j]);
 printf("\n Enter the starting vertex:");
 scanf("%d",&v);
 bfs(v);
 printf("\n The node which are reachable are:\n");
 for(i=1;i<=n;i++)
 if(visited[i])
 printf("%d\t",i);
 getch();
}
```

Output:

```
Enter the number of vertices:4

Enter graph data in matrix form:
0       1       1       1
0       0       0       1
0       0       0       0
0       0       1       0

Enter the starting vertex:1

The node which are reachable are:
2       3       4       _
```

4.Write a C program that implements kruskal's algorithm to generate minimum cost spanning tree.

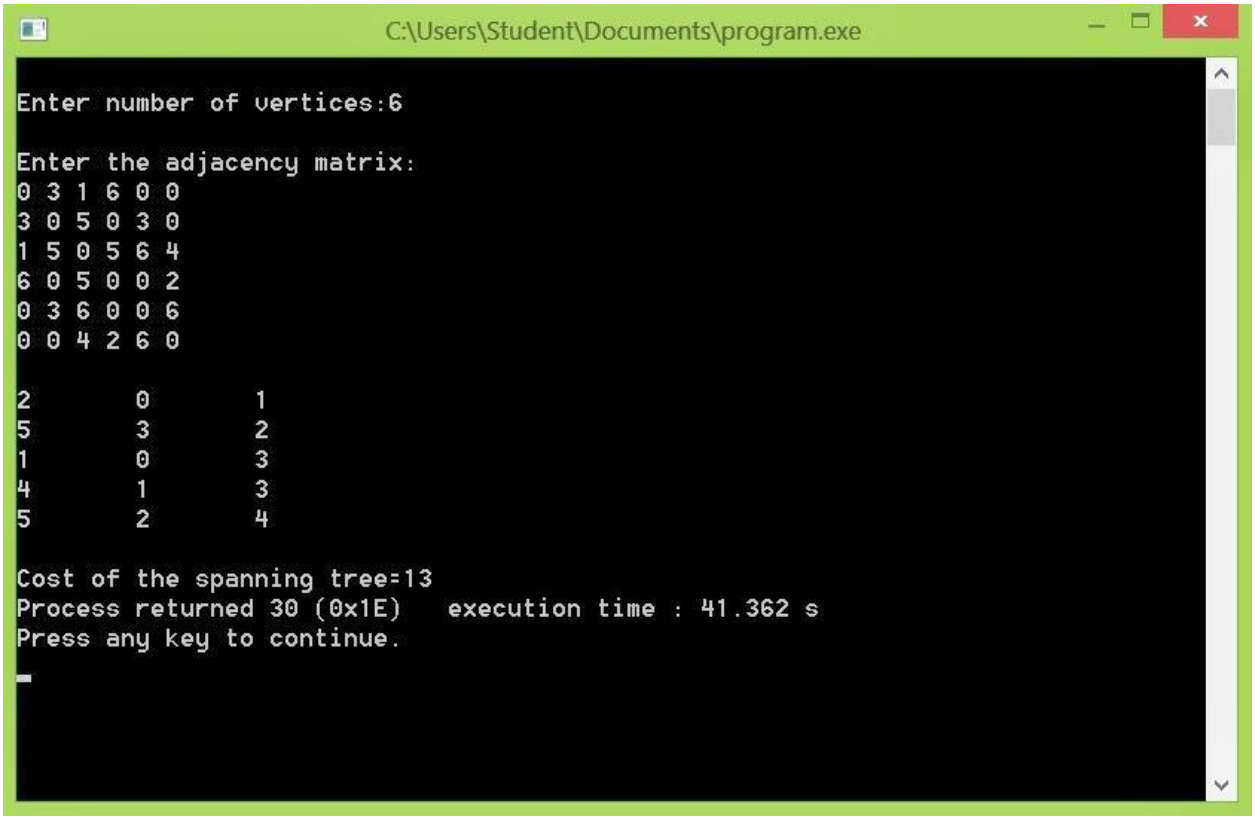
Ans:

```
#include<stdio.h>

#define MAX 30

typedef struct edge
{
int u,v,w;
}edge;

typedef struct edgelist
{
edge data[MAX];
int n;
}edgelist;

edgelist elist;

int G[MAX][MAX],n;
edgelist spanlist;

void kruskal();
int find(int belongs[],int vertexno);
void union1(int belongs[],int c1,int c2);
void sort();
void print();

void main()
{
int i,j,total_cost;

printf("\nEnter number of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

for(i=0;i<n;i++)
                    for(j=0;j<n;j++)
                        scanf("%d",&G[i][j]);
```

```
kruskal();
print();
}

void kruskal()
{
int belongs[MAX],i,j,cno1,cno2;
elist.n=0;

for(i=1;i<n;i++)
                for(j=0;j<i;j++)
                {
                        if(G[i][j]!=0)
                        {
                                elist.data[elist.n].u=i;
                                elist.data[elist.n].v=j;
                                elist.data[elist.n].w=G[i][j];
                                elist.n++;
                        }
                }

sort();

for(i=0;i<n;i++)
                belongs[i]=i;

spanlist.n=0;

for(i=0;i<elist.n;i++)
{
                cno1=find(belongs,elist.data[i].u);
                cno2=find(belongs,elist.data[i].v);

                if(cno1!=cno2)
                {
                        spanlist.data[spanlist.n]=elist.data[i];
                        spanlist.n=spanlist.n+1;
                        union1(belongs,cno1,cno2);
                }
}
}

int find(int belongs[],int vertexno)
{
return(belongs[vertexno]);
```

```
}

void union1(int belongs[],int c1,int c2)
{
int i;

for(i=0;i<n;i++)
                if(belongs[i]==c2)
                        belongs[i]=c1;
}

void sort()
{
int i,j;
edge temp;

for(i=1;i<elist.n;i++)
                for(j=0;j<elist.n-1;j++)
                        if(elist.data[j].w>elist.data[j+1].w)
                        {
                                temp=elist.data[j];
                                elist.data[j]=elist.data[j+1];
                                elist.data[j+1]=temp;
                        }
}

void print()
{
int i,cost=0;

for(i=0;i<spanlist.n;i++)
{
                printf("\n%d\t%d\t%d",spanlist.data[i].u,spanlist.data[i].v,spanlist.data[i].w)
                ;
                cost=cost+spanlist.data[i].w;
}

printf("\n\nCost of the spanning tree=%d",cost);
}
```

OUTPUT:



5. **Write a C program to implement Floyd;s algorithm for all pairs shortest path problem.**

```
// C Program for Floyd Warshall Algorithm
#include<stdio.h>

// Number of vertices in the graph
#define V 4

/* Define Infinite as a large enough value. This value will be used
for vertices not connected to each other */
#define INF 99999

// A function to print the solution matrix
void printSolution(int dist[][V]);

// Solves the all-pairs shortest path problem using Floyd Warshall algorithm
void floydWarshall (int graph[][V])
{
```

```
/* dist[][] will be the output matrix that will finally have the shortest
      distances between every pair of vertices */
int dist[V][V], i, j, k;

/* Initialize the solution matrix same as input graph matrix. Or
      we can say the initial values of shortest distances are based
      on shortest paths considering no intermediate vertex. */
for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
            dist[i][j] = graph[i][j];

/* Add all vertices one by one to the set of intermediate vertices.
      ---> Before start of an iteration, we have shortest distances between all
      pairs of vertices such that the shortest distances consider only the
      vertices in set {0, 1, 2, .. k-1} as intermediate vertices.
      ----> After the end of an iteration, vertex no. k is added to the set of
      intermediate vertices and the set becomes {0, 1, 2, .. k} */
for (k = 0; k < V; k++)
{
      // Pick all vertices as source one by one
      for (i = 0; i < V; i++)
      {
            // Pick all vertices as destination for the
            // above picked source
            for (j = 0; j < V; j++)
            {
            // If vertex k is on the shortest path from
            // i to j, then update the value of dist[i][j]
            if (dist[i][k] + dist[k][j] < dist[i][j])
            dist[i][j] = dist[i][k] + dist[k][j];
            }
      }
}

// Print the shortest distance matrix
printSolution(dist);
}

/* A utility function to print solution */
void printSolution(int dist[][V])
{
printf ("The following matrix shows the shortest distances"
            " between every pair of vertices \n");
for (int i = 0; i < V; i++)
{
      for (int j = 0; j < V; j++)
```

```
        {
                if (dist[i][j] == INF)
                printf("%7s", "INF");
                else
                printf ("%7d", dist[i][j]);
        }
        printf("\n");
}
}

// driver program to test above function
int main()
{
/* Let us create the following weighted graph
             10
        (0)------->(3)
        |          /|\
5       |           |
        |           | 1
        \|/         |
        (1)------->(2)
             3
*/
int graph[V][V] = { {0,   5,  INF, 10},
                    {INF, 0,   3, INF},
                    {INF, INF, 0,   1},
                    {INF, INF, INF, 0}
                    };

// Print the solution
floydWarshall(graph);
return 0;
}
```

OUTPUT:

```
Following matrix shows the shortest distances between every pair of vertices
    0     5     8     9
  INF     0     3     4
  INF   INF     0     1
  INF   INF   INF     0
```

6. Write a C program to implement the backtracking algorithm for the Hamiltonian circuit's problem.

```
/* C program for solution of Hamiltonian Cycle problem
  using backtracking */
    #include<stdio.h>

    // Number of vertices in the graph
    #define V 5

    void printSolution(int path[]);

    /* A utility function to check if the vertex v can be added at
      index 'pos' in the Hamiltonian Cycle constructed so far (stored
      in 'path[]') */
    bool isSafe(int v, bool graph[V][V], int path[], int pos)
    {
       /* Check if this vertex is an adjacent vertex of the previously
          added vertex. */
       if (graph [ path[pos-1] ][ v ] == 0)
          return false;

       /* Check if the vertex has already been included.
        This step can be optimized by creating an array of size V */
       for (int i = 0; i < pos; i++)
          if (path[i] == v)
             return false;

       return true;
    }

    /* A recursive utility function to solve hamiltonian cycle problem */
    bool hamCycleUtil(bool graph[V][V], int path[], int pos)
    {
       /* base case: If all vertices are included in HamiltonianCycle */

   if (pos == V)
   {
```

```
        // And if there is an edge from the last included vertex to the
        // first vertex
        if ( graph[ path[pos-1] ][ path[0] ] == 1 )
          return true;
        else
          return false;
    }

    // Try different vertices as a next candidate in Hamiltonian Cycle.
    // We don't try for 0 as we included 0 as starting point in hamCycle()
    for (int v = 1; v < V; v++)
    {
        /* Check if this vertex can be added to Hamiltonian Cycle */
        if (isSafe(v, graph, path, pos))
        {
            path[pos] = v;

            /* recur to construct rest of the path */
            if (hamCycleUtil (graph, path, pos+1) == true)
                return true;

            /* If adding vertex v doesn't lead to a solution,
               then remove it */
            path[pos] = -1;
        }
    }

    /* If no vertex can be added to Hamiltonian Cycle constructed so far,
       then return false */
    return false;
}

/* This function solves the Hamiltonian Cycle problem using Backtracking.
  It mainly uses hamCycleUtil() to solve the problem. It returns false
  if there is no Hamiltonian Cycle possible, otherwise return true and
  prints the path. Please note that there may be more than one solutions,
  this function prints one of the feasible solutions. */
bool hamCycle(bool graph[V][V])
{
    int *path = new int[V];
    for (int i = 0; i < V; i++)
        path[i] = -1;

    /* Let us put vertex 0 as the first vertex in the path. If there is
       a Hamiltonian Cycle, then the path can be started from any point
       of the cycle as the graph is undirected */
```

```
    path[0] = 0;
    if ( hamCycleUtil(graph, path, 1) == false )
    {
        printf("\nSolution does not exist");
        return false;
    }

    printSolution(path);
    return true;
}

/* A utility function to print solution */
void printSolution(int path[])
{
    printf ("Solution Exists:"
            " Following is one Hamiltonian Cycle \n");
    for (int i = 0; i < V; i++)
        printf(" %d ", path[i]);

    // Let us print the first vertex again to show the complete cycle
    printf(" %d ", path[0]);
    printf("\n");
}

// driver program to test above function
int main()
{
   /* Let us create the following graph
      (0)--(1)--(2)
       |   / \   |
       |  /   \  |
       | /     \ |
      (3)-------(4)    */
   bool graph1[V][V] = {{0, 1, 0, 1, 0},
                        {1, 0, 1, 1, 1},
                        {0, 1, 0, 0, 1},
                        {1, 1, 0, 0, 1},
                        {0, 1, 1, 1, 0},
                       };

    // Print the solution
    hamCycle(graph1);

   /* Let us create the following graph
      (0)--(1)--(2)
       |   / \   |
```

```
      | /   \ |
      | /     \ |
     (3)       (4)    */
    bool graph2[V][V] = {{0, 1, 0, 1, 0},
                         {1, 0, 1, 1, 1},
                         {0, 1, 0, 0, 1},
                         {1, 1, 0, 0, 0},
                         {0, 1, 1, 0, 0},
                        };

    // Print the solution
    hamCycle(graph2);

    return 0;
}
```

OUTPUT:

Solution Exists: Following is one Hamiltonian Cycle
0 1 2 4 3 0

Solution does not exist

7. Write a C program to implement backtracking algorithm for the sum of subsets problem.

```
#include <stdio.h>
#include <stdlib.h>
#define ARRAYSIZE(a) (sizeof(a)) / (sizeof(a[0]))
static int total_nodes;
// prints subset found
void printSubset(int A[], int size)
{
for (int i = 0; i < size; i++) {
printf("%*d", 5, A[i]);
}
printf("n");
}

// inputs
// s            - set vector
// t            - tuplet vector
// s_size       - set size
```

```
// t_size       - tuplet size so far
// sum          - sum so far
// ite          - nodes count
// target_sum   - sum to be found
void subset_sum(int s[], int t[],
          int s_size, int t_size,
          int sum, int ite,
          int const target_sum)
{
total_nodes++;
if (target_sum == sum) {
// We found subset
printSubset(t, t_size);
// Exclude previously added item and consider next candidate
subset_sum(s, t, s_size, t_size - 1, sum - s[ite], ite + 1, target_sum);
return;
}
else {
// generate nodes along the breadth
for (int i = ite; i < s_size; i++) {
t[t_size] = s[i];
// consider next level node (along depth)
subset_sum(s, t, s_size, t_size + 1, sum + s[i], i + 1, target_sum);
}
}
}

// Wrapper to print subsets that sum to target_sum
// input is weights vector and target_sum
void generateSubsets(int s[], int size, int target_sum)
{
int* tuplet_vector = (int*)malloc(size * sizeof(int));
subset_sum(s, tuplet_vector, size, 0, 0, target_sum);
free(tuplet_vector);
}
int main()
{
int weights[] = { 10, 7, 5, 18, 12, 20, 15 };
int size = ARRAYSIZE(weights);

generateSubsets(weights, size, 35);
printf("Nodes generated %dn", total_nodes);
return 0;
}
```

OUTPUT:

10 7 18n 10 5 20n 5 18 12n 20 15n 20n Nodes generated 126n