

NATIONAL INSTITUTE OF TECHNOLOGY, RAIPUR

**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING**



**DATA WAREHOUSING AND DATA MINING & AI LAB
[LAB-FILE]**

Name: Jonnadula Venkata Sai Tanish

Roll No: 19115038

Branch: Computer Science and Engineering

Semester: 6th semester (Sixth)

S. No	Name of Experiment	Page No.	Remarks
1	Perform an experiment on data cleaning.	4	
2	Perform an experiment on binning equal width and binning equal depth.	10	
3	Perform an experiment on normalization.	12	
4	Perform an experiment on data pre-processing.	15	
5	Write a program to perform decision tree classification.	18	
6	Write a program to perform Support Vector Machine Classification.	20	
7	Write a program to perform K-means Clustering.	23	
8	Write a program to perform Fuzzy c-means Clustering.	26	
9	Write a program to perform DBSCAN clustering.	28	
10	Write a program for performing Linear Regression.	30	
11	Perform an experiment for finding frequent item sets, confidence and support using association rules of mining.	34	
12	Implement Apriori algorithm for association rules of mining.	38	
13	Implement Snow-flake schema in Data warehousing	41	
14	Implement Star schema in Data warehousing	42	
15	Write a prolog program to find the rules for father, mother, child, son, daughter, brother, sister, uncle, aunt, ancestor given the facts about parent, male and female.	43	
16	Write a prolog program to find the length of a given list.	46	
17	Write a prolog program to find the last element of a given list.	47	
18	Write a prolog program to delete the first occurrence and also all occurrences of a particular element in a given list.	48	
19	Write a prolog program to find union of two given sets represented as lists.	50	
20	Write a prolog program to reverse a given list of values.	51	
21	Write a prolog program given the knowledge base, If x is on the top of y, y supports x. If x is above y and they are touching each other, x is on top of y. A cup is above a book. The cup is touching that book. Convert the following into wffs, clausal form; Is it possible to deduce	52	

	that 'The book supports the cup'.		
22	<p>Write a prolog program given the knowledge base, If Town x is connected to Town y by highway z and bikes are allowed on z, you can get to y from x by bike. If Town x is connected to y by z then y is also connected to x by z. If you can get to town q from p and also to town r from town q, you can get to town r from town p.</p> <p>a. Town A is connected to Town B by Road1. b. Town B is connected to Town C by Road2. c. Town A is connected to Town C by Road3. d. Town D is connected to Town E by Road4. e. Town D is connected to Town B by Road5. f. Bikes are allowed on roads 3, 4, 5.</p> <p>Bikes are only either allowed on Road1 or on Road2 every day. Convert the following into WFF's, clausal form and deduce that 'One can get to town B from town D'.</p>	53	
23	Solve the classical Monkey Banana problem of AI using prolog.	55	
24	Define a LISP function to compute sum of squares.	58	
25	Define a LISP function to compute difference of squares. (if $x > y$ return $x^2 - y^2$, otherwise return $y^2 - x^2$).	59	
26	Define a LISP function to compute the factorial of a given number.	60	
27	Define a LISP function to reverse the number entered as parameter in function call.	61	
28	Write a LISP program containing two functions: one to read input values and one to display them.	62	
29	Write a LISP program to compute factorial of a given number using recursion.	63	
30	Write a LISP Program using recursion to perform GCD of two numbers entered by user.	64	

Experiment 1

Aim: Perform an experiment on data cleaning.

Theory:

Data cleaning is also known as data scrubbing. Data cleaning is a process which ensures the set of data is correct and accurate. Data accuracy and consistency, data integration is checked during data cleaning. Data cleaning can be applied for a set of records or multiple sets of data which need to be merged.

Data cleaning is fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct, so it is important to do this process.

General steps for data cleaning:

- i) Remove duplicate or irrelevant observations.
- ii) Fix structural errors.
- iii) Filter unwanted outliers.
- iv) Handle missing data.

Some methods to clean data:

- i) You can ignore the tuple. This is done when the class label is missing. This method is not very effective unless the tuple contains several attributes with missing values.
- ii) You can fill in the missing value manually. This approach is effective on small data sets with some missing values.
- iii) You can replace all missing attribute values with global constants, such as a label like “Unknown” or zero or minus infinity.
- iv) You can use the attribute mean to fill in the missing value. For example, customer's average income is 25000 then you can use this value to replace the missing value for income.
- v) Use the most probable value to fill in the missing value and backfill method.

Before Cleaning operation

Code:

```
import pandas as pd

df = pd.read_csv('missing.csv')

print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

print(df)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Sr. No.  Age
0         1  32.0
1         2  37.0
2         3  55.0
3         4  53.0
4         5  54.0
5         6  60.0
6         7  49.0
7         8  56.0
8         9  59.0
9        10   NaN
10       11  62.0
11       12  55.0
12       13  46.0
13       14   NaN
14       15  63.0
15       16  62.0
16       17   NaN
17       18  56.0
18       19  62.0
19       20  49.0
```

After the Cleaning Process (By filling with zero)

Code:

```
import pandas as pd

df = pd.read_csv("missing.csv")
new_df = df.fillna(0)

print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

print(new_df)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Sr. No.  Age
0         1  32.0
1         2  37.0
2         3  55.0
3         4  53.0
4         5  54.0
5         6  60.0
6         7  49.0
7         8  56.0
8         9  59.0
9        10   0.0
10       11  62.0
11       12  55.0
12       13  46.0
13       14   0.0
14       15  63.0
15       16  62.0
16       17   0.0
17       18  56.0
18       19  62.0
19       20  49.0
```

After the Cleaning Process (By filling with mean)

Code:

```
import pandas as pd
import numpy as np

df = pd.read_csv("missing.csv")
new_df = df.fillna(np.mean(df))

print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

print(new_df)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Sr. No.  Age
0         1  32.000000
1         2  37.000000
2         3  55.000000
3         4  53.000000
4         5  54.000000
5         6  60.000000
6         7  49.000000
7         8  56.000000
8         9  59.000000
9        10  53.529412
10       11  62.000000
11       12  55.000000
12       13  46.000000
13       14  53.529412
14       15  63.000000
15       16  62.000000
16       17  53.529412
17       18  56.000000
18       19  62.000000
19       20  49.000000
```

After the Cleaning Process (By filling with median)

Code:

```
import pandas as pd
import numpy as np

df = pd.read_csv("missing.csv")
new_df = df.fillna(df.median())

print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

print(new_df)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Sr. No.  Age
0         1  32.0
1         2  37.0
2         3  55.0
3         4  53.0
4         5  54.0
5         6  60.0
6         7  49.0
7         8  56.0
8         9  59.0
9        10  55.0
10       11  62.0
11       12  55.0
12       13  46.0
13       14  55.0
14       15  63.0
15       16  62.0
16       17  55.0
17       18  56.0
18       19  62.0
19       20  49.0
```


After the Cleaning Process (By Using Backfill method)

Code:

```
import pandas as pd

df = pd.read_csv('missing.csv')
new_df=df.fillna(method='backfill')

print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

print(new_df)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Sr. No.  Age
0         1  32.0
1         2  37.0
2         3  55.0
3         4  53.0
4         5  54.0
5         6  60.0
6         7  49.0
7         8  56.0
8         9  59.0
9        10  62.0
10       11  62.0
11       12  55.0
12       13  46.0
13       14  63.0
14       15  63.0
15       16  62.0
16       17  56.0
17       18  56.0
18       19  62.0
19       20  49.0
```

Conclusion:

Data cleaning activity performed on data where some missing values were present which was replaced by random, mean or median value for that attribute.

Experiment 2

Aim: Perform an experiment on binning equal width and binning equal depth.

Theory:

Noise is a random error or variance in a measured variable. Noisy Data may be due to faulty data collection instruments, data entry problems and technology limitation. It is handled by binning methods followed by smoothing techniques.

Binning methods are:

1. Equi-depth binning method: Equal number of elements are present in each bucket.
2. Equi-width binning method: Data is divided into fixed set of intervals and corresponding frequency is assigned to it.

Smoothing techniques are:

1. Smoothing by bin means
2. Smoothing by bin median
3. Smoothing by bin boundaries

Bining Equal Frequency

Code:

```
def equifreq(arr1, m):
    a = len(arr1)
    n = int(a / m)
    for i in range(0, m):
        arr = []
        for j in range(i * n, (i + 1) * n):
            if j >= a:
                break
            arr = arr + [arr1[j]]
        print(arr)

data = [5, 8, 12, 16, 24, 32, 51, 59, 79, 92, 200, 211]
# no of bins
nb = 3
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
print("Equal frequency binning")
equifreq(data, nb)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Equal frequency binning
[5, 8, 12, 16]
[24, 32, 51, 59]
[79, 92, 200, 211]
```

Bining Equal Width

Code:

```
def equiwidth(arr1, m):
    a = len(arr1)
    w = int((max(arr1) - min(arr1)) / m)
    min1 = min(arr1)
    arr = []
    for i in range(0, m + 1):
        arr = arr + [min1 + w * i]
    arri = []
    for i in range(0, m):
        temp = []
        for j in arr1:
            if j >= arr[i] and j <= arr[i + 1]:
                temp += [j]
        arri += [temp]
    print(arri)

data = [5, 8, 12, 16, 24, 32, 51, 59, 79, 92, 200, 211]
# no of bins
nb = 3
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
print("Equal frequency binning")
equiwidth(data, nb)
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Equal frequency binning
[[5, 8, 12, 16, 24, 32, 51, 59], [79, 92], [200]]
```

Conclusion:

The data is first sorted and converted to buckets through two binning methods. In each binning methods all three smoothing techniques are applied.

Experiment 3

Aim: Perform an experiment on normalization.

Theory: Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. It is generally useful for classification algorithms.

Following normalization techniques are used:

1. Decimal Scaling.
2. Min-Max Normalization.
3. Z-Score Normalization.

Code (Normalization of Random Data):

```
from sklearn import preprocessing
import numpy as np
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

a = np.random.random((1, 4))
a = a*15
print("Data = ", a)

# normalize the data attributes
normalized = preprocessing.normalize(a)
print("Normalized Data = ", normalized)
```

Output (Normalization of Random Data):

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Data = [[ 1.06108945  7.10987986  6.60423262 11.38203443]]
Normalized Data = [[0.07076401 0.47415755 0.44043596 0.75906734]]
```

Code (Decimal Scaling):

```
import pandas as pd
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
df = pd.DataFrame(
    [
        [160000, 120, 18.9, 1400],
        [320000, 905, 23.4, 1800],
        [210000, 230, 14.0, 1300],
        [55000, 450, 13.5, 1500],
    ]
)
```

```

],
columns=["Col A", "Col B", "Col C", "Col D"],
)

df_1_scale = df.copy()

def Dec_scale(df_1_scale):
    for x in df_1_scale:
        p = df_1_scale[x].max()
        q = len(str(abs(p)))
        df_1_scale[x] = df_1_scale[x] / 10 ** q

Dec_scale(df_1_scale)
print(df_1_scale)

```

Output (Decimal Scaling):

```

Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
   Col A  Col B   Col C  Col D
0  0.160  0.120  0.00189  0.14
1  0.320  0.905  0.00234  0.18
2  0.210  0.230  0.00140  0.13
3  0.055  0.450  0.00135  0.15

```

Code (Min-Max Scaling):

```

import pandas as pd
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
df = pd.DataFrame(
    [
        [160000, 120, 18.9, 1400],
        [320000, 905, 23.4, 1800],
        [210000, 230, 14.0, 1300],
        [55000, 450, 13.5, 1500],
    ],
    columns=["Col A", "Col B", "Col C", "Col D"],
)

df_scale = df.copy()
# apply normalization techniques
for column in df_scale.columns:
    df_scale[column] = (df_scale[column] - df_scale[column].min()) /
    (
        df_scale[column].max() - df_scale[column].min()
    )
print(df_scale)

```

Output (*Min-Max Scaling*):

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
      Col A      Col B      Col C      Col D
0  0.396226  0.000000  0.545455  0.2
1  1.000000  1.000000  1.000000  1.0
2  0.584906  0.140127  0.050505  0.0
3  0.000000  0.420382  0.000000  0.4
```

Code (*Z-Score Method*):

```
import pandas as pd
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
df = pd.DataFrame(
    [
        [160000, 120, 18.9, 1400],
        [320000, 905, 23.4, 1800],
        [210000, 230, 14.0, 1300],
        [55000, 450, 13.5, 1500],
    ],
    columns=["Col A", "Col B", "Col C", "Col D"],
)

df_z_scaled = df.copy()
# apply normalization techniques
for column in df_z_scaled.columns:
    df_z_scaled[column] = (
        df_z_scaled[column] - df_z_scaled[column].mean()
    ) / df_z_scaled[column].std()
# view normalized data
print(df_z_scaled)
```

Output (*Z-Score Method*):

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
      Col A      Col B      Col C      Col D
0 -0.238411 -0.881538  0.311486 -0.46291
1  1.214759  1.378078  1.278167  1.38873
2  0.215705 -0.564904 -0.741122 -0.92582
3 -1.192054  0.068364 -0.848531  0.00000
```

Conclusion:

The data is normalized using the techniques like Decimal Scaling, Min-Max Normalization and Z-Score Normalization.

Experiment 4

Aim: Perform an experiment on data pre-processing.

Theory: Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data pre-processing:

Data Cleaning: Remove wrong or invalid entries.

1. Missing Data: This situation arises when some data is missing in the data.

a. Ignore the tuples:

b. Fill the Missing values

2. Noisy Data: Noisy data is a meaningless data that can't be interpreted by machines.

a. Binning Method:

This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task.

b. Regression:

Here data can be made smooth by fitting it to a regression function.

c. Clustering:

This approach groups the similar data in a cluster. The outliers may be undetected, or it will fall outside the clusters.

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

1. Normalization: It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. Attribute Selection: In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. Discretization: This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

4. Concept Hierarchy Generation: Here attributes are converted from level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

Data Reduction: Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

Data before Pre-processing

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
data = pd.read_csv('Data.csv')
data
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Data before Pre-processing

Code:

```
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
from sklearn.impute import SimpleImputer
```



```
# 'np.nan' signifies that we are targeting missing values
# and the strategy we are choosing is replacing it with 'mean'
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

imputer.fit(data.iloc[:, 1:3])
data.iloc[:, 1:3] = imputer.transform(data.iloc[:, 1:3])

# print the dataset
data
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

Conclusion:

We have performed the data pre-processing on the assumed data.

Experiment 5

Aim: Write a program to perform decision tree classification.

Theory:

Decision Tree Mining is a type of data mining technique that is used to build Classification Models. It builds classification models in the form of a tree-like structure, just like its name. This type of mining belongs to supervised class learning.

Code:

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi',
'pedigree', 'age', 'label']
# load dataset
pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

pima.head()

#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.3, random_state=1) # 70% training and 30% test

# Create Decision Tree classifier object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.683982683982684

from sklearn.tree import export_graphviz
import six
import sys
```

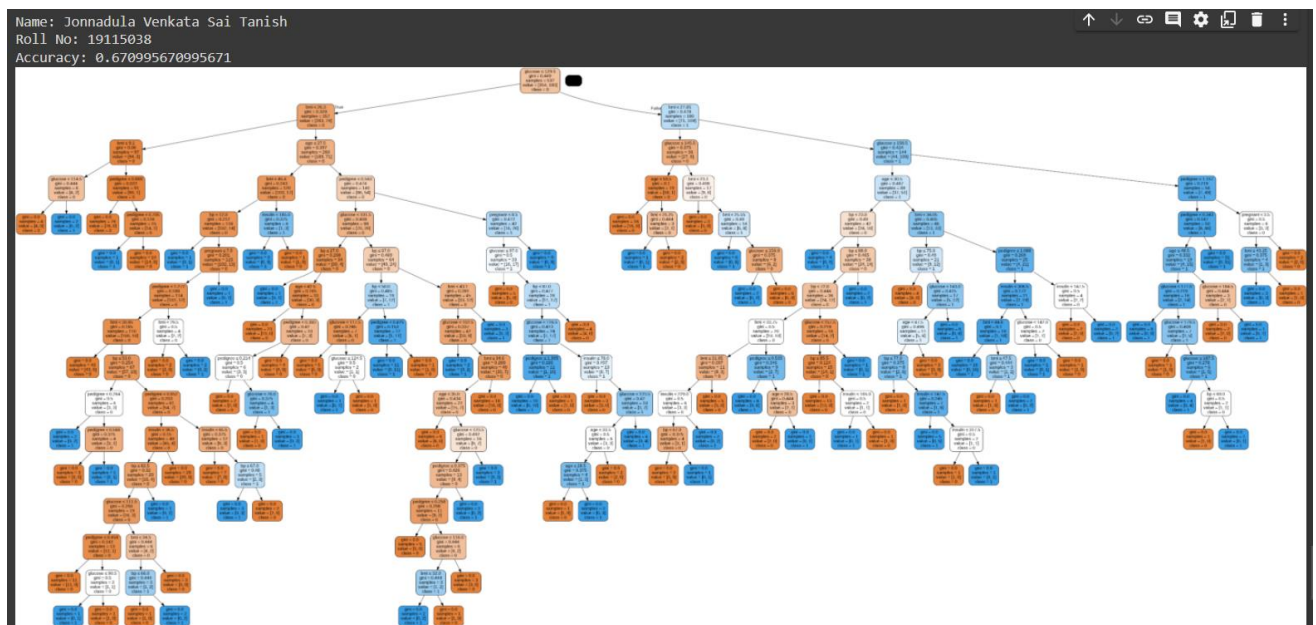
```

sys.modules['sklearn.externals.six'] = six
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols
, class_names=['0', '1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())

```

Output:



Conclusion:

Therefore, we perform the decision tree classification on a given data set.

Experiment 6

Aim: Write a program to perform Support Vector Machine Classification.

Theory: Support Vector Machine (SVM) model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

Code:

```
from sklearn import datasets
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
cancer_data = datasets.load_breast_cancer()
print(cancer_data.data[5])
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
[1.245e+01 1.570e+01 8.257e+01 4.771e+02 1.278e-01 1.700e-01 1.578e-01
 8.089e-02 2.087e-01 7.613e-02 3.345e-01 8.902e-01 2.217e+00 2.719e+01
 7.510e-03 3.345e-02 3.672e-02 1.137e-02 2.165e-02 5.082e-03 1.547e+01
 2.375e+01 1.034e+02 7.416e+02 1.791e-01 5.249e-01 5.355e-01 1.741e-01
 3.985e-01 1.244e-01]
```

Code:

```
print(cancer_data.data.shape)
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
#target set
print(cancer_data.target)
```

Output:

```
(569, 30)
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1
 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1]
```

Code:

```
from sklearn.model_selection import train_test_split
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
cancer_data = datasets.load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer_data.data,
                                                    cancer_data.target, test_size=0.4, random_state=109)

from sklearn import svm
#create a classifier
cls = svm.SVC(kernel="linear")
#train the model
cls.fit(X_train, y_train)
#predict the response
pred = cls.predict(X_test)

from sklearn import metrics
#accuracy
print("accuracy:", metrics.accuracy_score(y_test, y_pred=pred))
#precision score
print("precision:", metrics.precision_score(y_test, y_pred=pred))
#recall score
print("recall" , metrics.recall_score(y_test, y_pred=pred))
print(metrics.classification_report(y_test, y_pred=pred))
```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
accuracy: 0.9649122807017544
precision: 0.9642857142857143
recall 0.9782608695652174
```

	precision	recall	f1-score	support
0	0.97	0.94	0.96	90
1	0.96	0.98	0.97	138
accuracy			0.96	228
macro avg	0.97	0.96	0.96	228
weighted avg	0.96	0.96	0.96	228

Code:

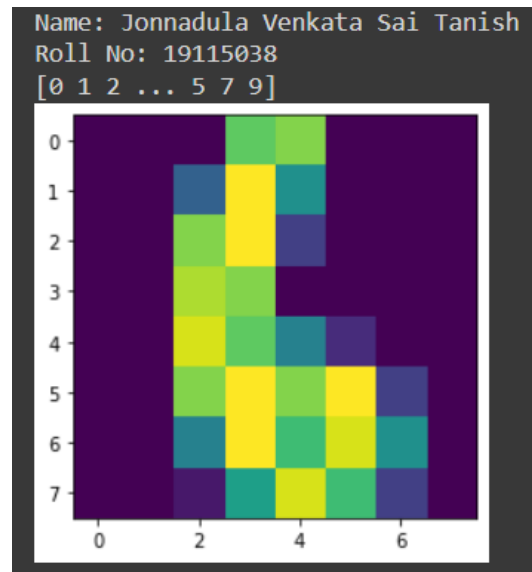
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
#loading the dataset
letters = datasets.load_digits()
```

```

#generating the classifier
clf = svm.SVC(gamma=0.001, C=100)
#training the classifier
X,y = letters.data[:10], letters.target[:10]
clf.fit(X,y)
#predicting the output
print(clf.predict(letters.data[:10]))
plt.imshow(letters.images[6], interpolation='nearest')
plt.show()

```

Output:



Conclusion: Therefore, we perform the support vector machine classification on a given data set.

Experiment 7

Aim: Write a program to perform K-means Clustering.

Theory:

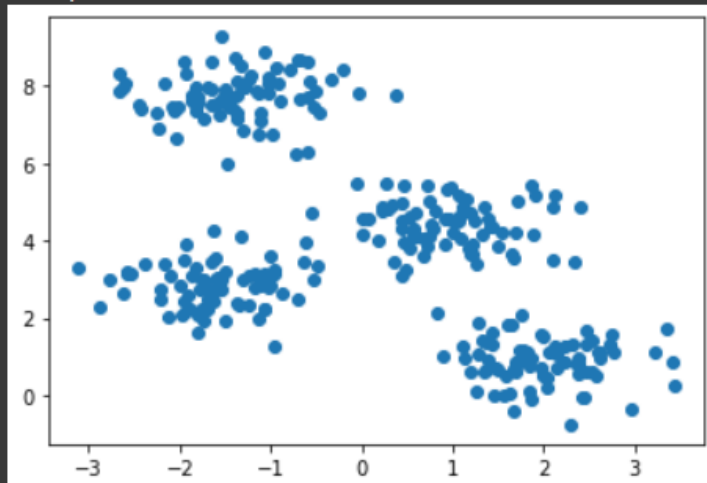
Clustering: k-means algorithm is an iterative algorithm that tries to partition the dataset into k pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic means of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

Code:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
plt.scatter(X[:,0], X[:,1])
```

Output:

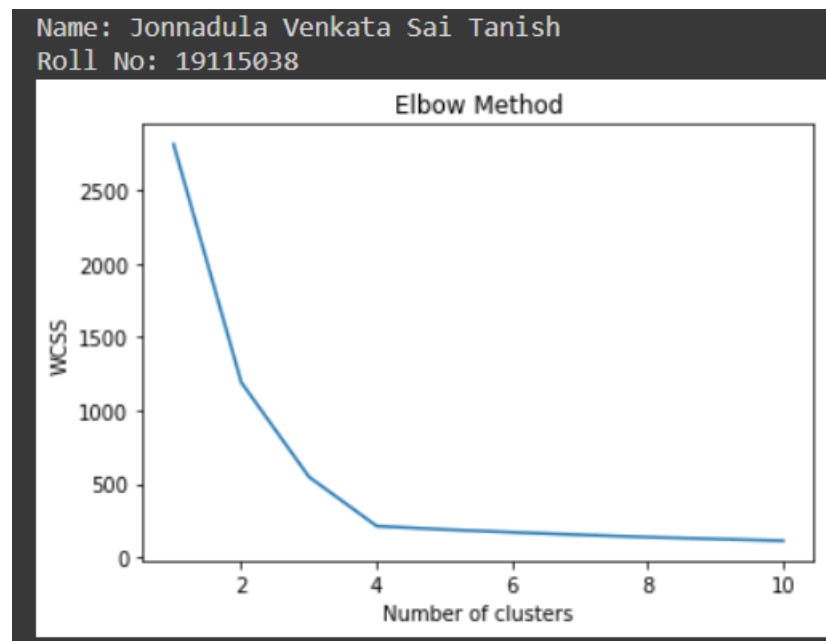
```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
<matplotlib.collections.PathCollection at 0x7fe155e19850>
```



Code:

```
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-
means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Output:

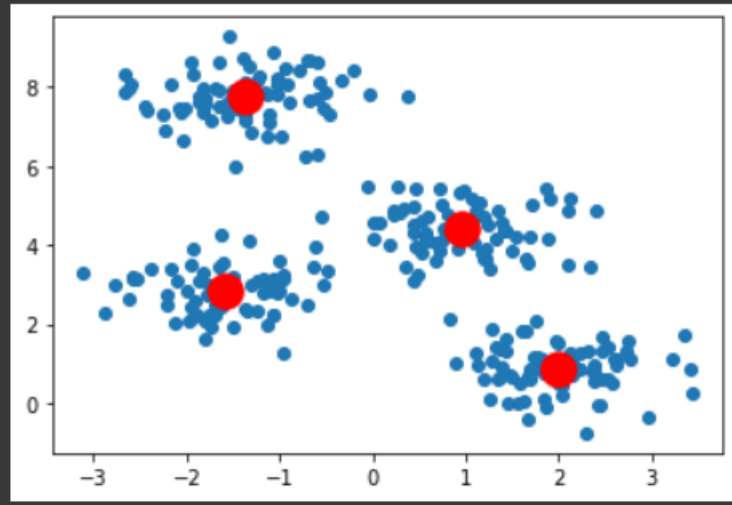


Code:

```
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
kmeans = KMeans(n_clusters=4, init='k-
means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X)
plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s=300, c='red')
plt.show()
```


Output:

Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038



Conclusion: With the help of assumed data, we have implemented the k-means clustering.

Experiment 8

Aim: Write a program to perform Fuzzy c-means Clustering.

Theory:

Fuzzy logic is a generalization of standard logic, in which a concept can possess a degree of truth anywhere between 0.0 and 1.0. Standard logic applies only to concepts that are completely true (having degree of truth 1.0) or false (having degree of truth 0.0). Fuzzy logic is supposed to be used for reasoning about inherently vague concepts, such as 'tallness.' For example, we might say that 'President Kovind is tall,' with degree of truth of 0.9.

Clustering can be performed using fuzzy logic by determining the fuzzy set to which it can belong with certain degree as discussed in example of president above.

Intuition: Fuzzy clustering generalizes partition clustering methods (such as k-means and medoid) by allowing an individual to be partially classified into more than one cluster. In regular clustering, everyone is a member of only one cluster. Suppose we have k clusters and we define a set of variables mi_1, mi_2, mi_k , that represent the probability that object i is classified into cluster k . In partition clustering algorithms, one of these values will be one and the rest will be zero. This represents the fact that these algorithms classify an individual into one and only one cluster.

Advantage: In fuzzy clustering, the membership is spread among all clusters. The mi_k can now be between zero and one, with the stipulation that the sum of their values is one. We call this a fuzzification of the cluster configuration. It has the advantage that it does not force every object into a specific cluster. It has the disadvantage that there is much more information to be interpreted.

Code:

```
import numpy as np
from fcmeans import FCM
from matplotlib import pyplot as plt

n_samples = 5000

X = np.concatenate((
    np.random.normal((-2, -2), size=(n_samples, 2)),
    np.random.normal((2, 2), size=(n_samples, 2))
))

fcm = FCM(n_clusters=2)
```

```

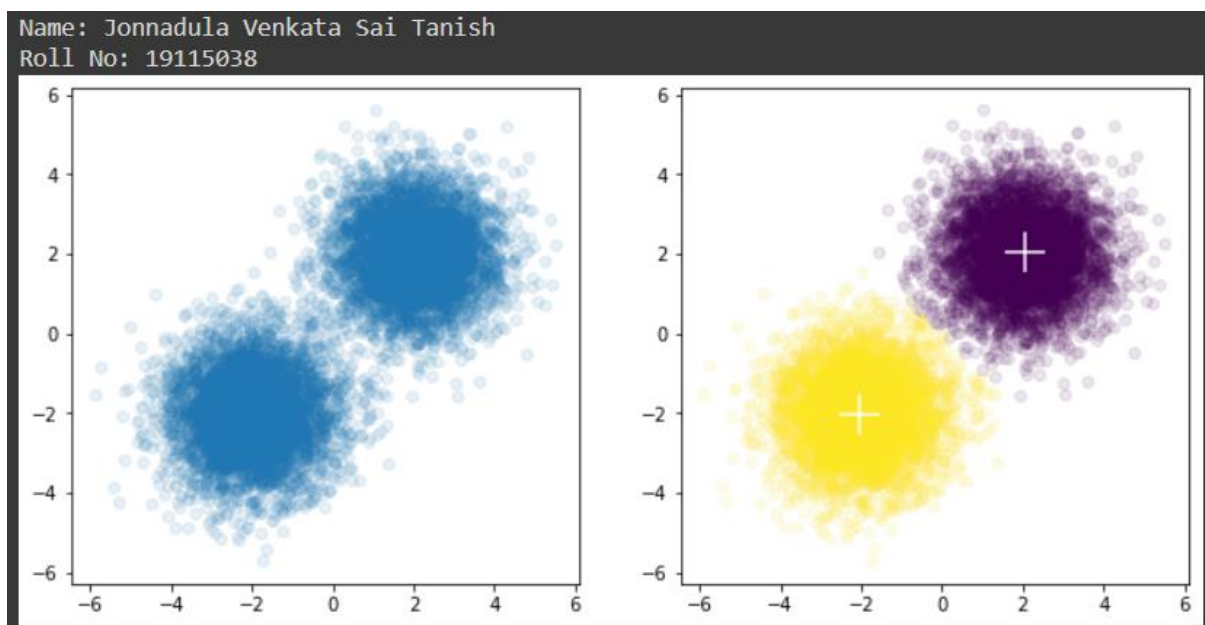
fcm.fit(X)

# outputs
fcm_centers = fcm.centers
fcm_labels = fcm.predict(X)

# plot result
f, axes = plt.subplots(1, 2, figsize=(11,5))
axes[0].scatter(X[:,0], X[:,1], alpha=.1)
axes[1].scatter(X[:,0], X[:,1], c=fcm_labels, alpha=.1)
axes[1].scatter(fcm_centers[:,0], fcm_centers[:,1], marker="+", s=500, c='w')
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
plt.show()

```

Output:



Conclusion: Fuzzy c-means clustering is performed for the Iris dataset to determine the set to which the petals belong with standard deviation determine the probability of its membership.

Experiment 9

Aim: Write a program to perform DBSCAN clustering.

Theory: Clusters are dense regions in the data space, separated by regions of the lower density of points. The DBSCAN algorithm is based on this intuitive notion of clusters and noise. The key idea is that for each point of a cluster, the neighborhood of a given radius must contain at least a minimum number of points.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")

# Load data in X
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.50, random_state=0)
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print(labels)

# Plot result

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = ["y", "b", "g", "r"]
print(colors)
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = "k"

    class_member_mask = labels == k

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(
```

```

        xy[:, 0], xy[:, 1], "o", markerfacecolor=col, markeredgecolor="
k", markersize=6
    )

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(
        xy[:, 0], xy[:, 1], "o", markerfacecolor=col, markeredgecolor="
k", markersize=6
    )

plt.title("number of clusters: %d" % n_clusters_)
plt.show()

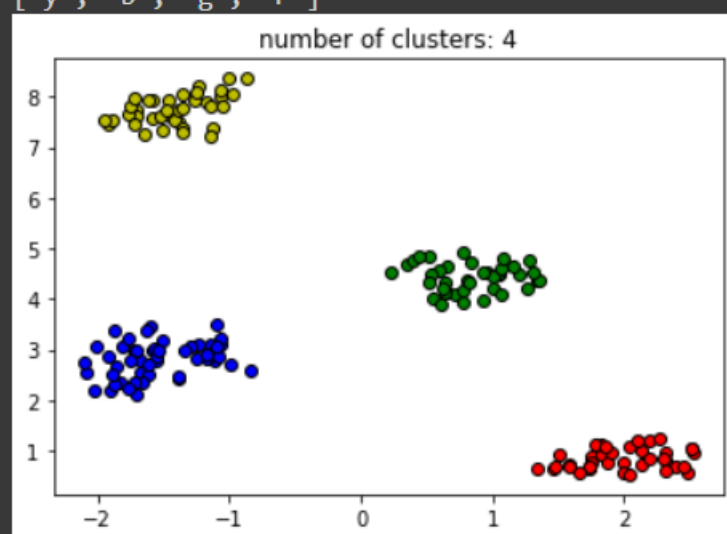
```

Output:

```

Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
[-1  0 -1  0 -1 -1 -1  2 -1 -1  1 -1  2 -1 -1  2  2  3  1 -1 -1  3  2  1
 1 -1  3  2 -1 -1 -1  0 -1 -1  0 -1  0 -1  1  3 -1  1 -1 -1  1  1  0  1
 0 -1  1  3 -1  3 -1  1 -1 -1  0  3 -1  2 -1  1  1  1 -1  3 -1  1  2 -1
 0  1 -1  0  1  2  3  0 -1  2 -1  3  0 -1  3  2 -1  0  2  3 -1  1  1 -1
-1  3 -1 -1 -1 -1 -1  3  2  3 -1  2 -1 -1 -1  1  3 -1  3 -1  0 -1 -1 -1
-1  3  1  3 -1  3  3  1 -1  1 -1  1  1 -1  0 -1 -1  0 -1 -1 -1  1 -1 -1
-1  1  0 -1  0  0  0  2 -1  2 -1  1  0  1  3  2 -1  2  2 -1  2 -1 -1 -1
-1 -1  2  0  3 -1 -1  0 -1  3  2  1  3 -1  1  1  2  2 -1  2 -1  0 -1  1
 2  2  1  1 -1 -1  1  0 -1  1 -1  1 -1 -1  1 -1  2 -1  2  1 -1 -1  0  1
 1  3 -1  2  0  3  3 -1 -1 -1 -1 -1  0 -1  2 -1  2 -1  1  2  3  1 -1  1
-1  2 -1  0  0  0  0  1  1 -1 -1  1  3  2  1 -1 -1 -1  3  0  2  2 -1  3
-1  1 -1 -1 -1  3  3 -1  1 -1 -1 -1 -1 -1  0  0 -1  3 -1  3  3 -1  0 -1
-1  2 -1  3  0 -1  0 -1 -1  2 -1  1]
['y', 'b', 'g', 'r']

```



Conclusion: Clustering formed using density-based clustering for Iris dataset.

Experiment 10

Aim: Write a program for performing Linear Regression.

Theory: Linear Regression is a machine learning algorithm based on supervised learning. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on the kind of relationship between dependent and independent variables, they are considering, and the number of independent variables being used. The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.

Simple linear regression is an approach for predicting a response using a single feature.

Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to the observed data.

Code (Simple Linear Regression):

```
import numpy as np
import matplotlib.pyplot as plt
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
               marker = "o", s = 30)
```

```

# predicted response vector
y_pred = b[0] + b[1]*x

# plotting the regression line
plt.plot(x, y_pred, color = "g")

# putting labels
plt.xlabel('x')
plt.ylabel('y')

# function to show plot
plt.show()

def main():
    # observations / data
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \
        \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)

if __name__ == "__main__":
    main()

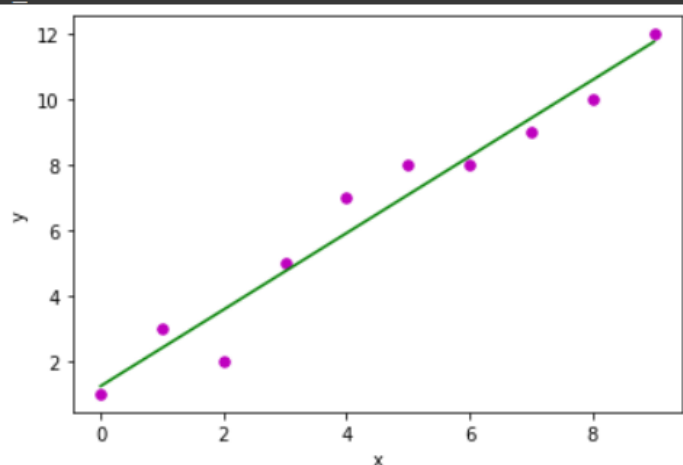
```

Output (*Simple Linear Regression*):

```

Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Estimated coefficients:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697

```



Code (*Multiple Linear Regression*):

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
print("Name: Jonnadula Venkata Sai Tanish")
print("Roll No: 19115038")
# load the boston dataset
boston = datasets.load_boston(return_X_y=False)

# defining feature matrix(X) and response vector(y)
X = boston.data
y = boston.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.4,
                                                    random_state=1)

# create linear regression object
reg = linear_model.LinearRegression()

# train the model using the training sets
reg.fit(X_train, y_train)

# regression coefficients
print('Coefficients: ', reg.coef_)

# variance score: 1 means perfect prediction
print('Variance score: {}'.format(reg.score(X_test, y_test)))

# plot for residual error

## setting plot style
plt.style.use('fivethirtyeight')

## plotting residual errors in training data
plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color = "green", s = 10, label = 'Train data')

## plotting residual errors in test data
plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color = "blue", s = 10, label = 'Test data')

## plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)

## plotting legend
```



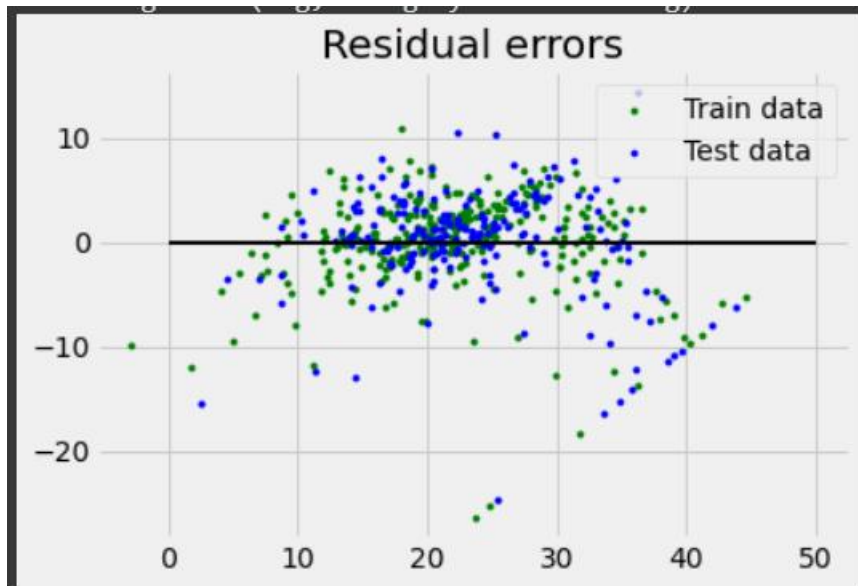
```
plt.legend(loc = 'upper right')

## plot title
plt.title("Residual errors")

## method call for showing the plot
plt.show()
```

Output (*Multiple Linear Regression*):

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Coefficients: [-8.95714048e-02  6.73132853e-02  5.04649248e-02  2.18579583e+00
 -1.72053975e+01  3.63606995e+00  2.05579939e-03 -1.36602886e+00
 2.89576718e-01 -1.22700072e-02 -8.34881849e-01  9.40360790e-03
 -5.04008320e-01]
```



Conclusion: Linear regression with types such as simple linear regression and multiple linear regression are performed over sample data.

Experiment 11

Aim: Perform an experiment for finding frequent item sets, confidence and support using association rules of mining.

Theory:

The terms can be explained as follows:

- Frequent item sets are the maximum items in set that satisfy the minimum support value.
- Confidence determines the threshold value for an item to belong a set.
- Association rules determines the relation between the sets.

Following program calculates the most frequent item sets based on the minimum support. All the possible sub-set are considered and then those set with equal or greater minimum support it filtered. Following it, the maximum element in such a set is taken.

Note: The dataset is converted in binary format such that if an attribute is present in a transaction will be 1 otherwise 0.

Code:

```
public class Main {
    static void itemSet(int setLen, int trasNum, int db[][], int dbInt[],
int minsup) {
        int powSetLen = (int) Math.pow(2, setLen);
        int powSetInt [] = new int[powSetLen];
        int cnt [] = new int[powSetLen];
        int qt = 0, maxQt = 0, idx = 0;
        int [][] flag = new int[powSetLen][setLen];
        // counter from 000...0 to 111...1
        for(int i = 0; i < powSetLen; i++) {
            for(int j = 0; j < setLen; j++) {
                if((i & (1 << j)) > 0) {
                    flag[i][j] = 1;
                }
                else
                    flag[i][j] = 0;
            }
        }
        System.out.println("All combinations are: ");
        for(int i = 0; i < powSetLen; i++) {
            for(int j = setLen - 1; j >= 0; j--) {
                System.out.print(flag[i][j] + " ");
                powSetInt[i] = powSetInt[i] + flag[i][j] * (int)
```

```

        Math.pow(2, setLen - 1 - j);
    }
    System.out.println();
}
System.out.println("All combinations in numeric: ");
for(int i = 0; i < powSetLen; i++) {
    System.out.print(powSetInt[i] + " ");
}
for(int i = 0; i < powSetInt.length; i++)
    cnt[i] = 0;
for(int i = 1; i < powSetInt.length; i++){
    for(int j = 0; j < dbInt.length; j++){
        if((powSetInt[i] & dbInt[j]) == powSetInt[i])
            cnt[i] = cnt[i] + 1;
    }
}
System.out.println();
System.out.println("Probable combinations with minimum support
combination: ");

for(int i = 0; i < powSetInt.length; i++) {
    qt = 0;
    if(cnt[i] >= minsup) {
        for(int j = 0; j < setLen; j++) {
            System.out.print(flag[i][j] + " ");
            if(flag[i][j] == 1)
                qt = qt + 1;
        }
        if(qt > maxQt) {
            maxQt = qt;
            idx = i;
        }
        System.out.println();
    }
}
System.out.println("Frequent set is: ");
for(int j = 0; j < setLen; j++)
    System.out.print(flag[idx][j] + " ");
}

public static void main (String[] args) {
    System.out.println("Name: Jonnadula Venkata Sai Tanish");
    System.out.println("Roll No: 19115038");
    int trasNum = 4, len = 3, minsup = 2;
    int db [][] = {{1, 0, 1},
    {0, 0, 0},
    {0, 0, 0},
    {1, 1, 1}};
    int dbInt [] = new int[trasNum];

```

```

        System.out.println("Databse is: ");
        for(int i = 0; i < trasNum ; i++) {
            for(int j = len - 1; j >= 0; j--) {
                System.out.print(db[i][j] + " ");
                dbInt[i] = dbInt[i] + db[i][j] * (int) Math.pow(2, len - 1
- j);

            }
            System.out.println();
        }
        System.out.println("Databse in numeric: ");

        for(int i = 0; i < trasNum; i++) {
            System.out.print(dbInt[i] + " ");
        }
        System.out.println();
        itemSet(len, trasNum, db, dbInt, minsup);
    }
}

```

Output:

```

Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Databse is:
1 0 1
0 0 0
0 0 0
1 1 1
Databse in numeric:
5 0 0 7
All combinations are:
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
All combinations in numeric:
0 4 2 6 1 5 3 7
Probable combinations with minimum support combination:
1 0 0
0 0 1
1 0 1
Frequent set is:
1 0 1

```

Conclusion:

The frequent set is $\{1, 0, 1\}$ states that attribute A and C are present while B is present. So, the set is $\{A, C\}$. It should be noted that it is the maximum set size with most frequent occurrence for the dataset given.

Experiment 12

Aim: Implement Apriori algorithm for association rules of mining.

Theory:

The steps for Apriori algorithm are:

- In the first iteration of the algorithm, each item is a member of the set of candidate1-itemsets, C1. The algorithm simply scans all the transactions to count the number of occurrences of each item. Suppose that the minimum support count required is k, that is, $\min \text{sup} = k$.
- The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. Here, Li shows the database with i elements in each set.
- To discover the set of frequent 2-itemsets, L2, the algorithm uses the join $L1 * L1$ to generate a candidate set of 2-itemsets.
- The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate2-itemsets in C2 having minimum support.
- The generation of the set of the candidate3-itemsets, C3, is then calculated. It is based on the Apriori property that all subsets of a frequent itemset must also be frequent.
- Further, higher order candidate sets are generated and finally determine the maximum of such set that also satisfies the minimum support.

Code:

```
public class Main {
public static void main(String[] args) {
    System.out.println("Name: Jonnadula Venkata Sai Tanish");
    System.out.println("Roll No: 19115038");
    int totalAttrs = 3, numberTransac = 4, c1 = 0, c2 = 0;
    double minsup = 0.5, sup = 0.0;
    // a → b Shows the association rule between two sets.
    int a[] = {1, 1, 0};
    int b[] = {0, 0, 0};
    int aorb[] = new int[totalAttrs];
    int aInt = 0;
    int aorbInt = 0;
    int dbInt[] = new int[numberTransac];
    int db[][] = {{1, 0, 1},{0, 0, 0},{0, 0, 0},{1, 1, 1}};
    // show data set
    System.out.println("The database is: ");
    for(int i = 0; i < numberTransac; i++) {
        for(int j = 0; j < totalAttrs; j++) {
            System.out.print(db[i][j] + " ");
        }
    }
}
```

```

    }
    System.out.println();
}
// show item set.
System.out.println("The relation is: ");
for(int i = 0; i < totalAttrs; i++)
    System.out.print(a[i] + " ");
System.out.print("-> ");
for(int i = 0; i < totalAttrs; i++)
    System.out.print(b[i] + " ");
for(int i = 0; i < totalAttrs; i++) {
    aorb[i] = a[i] | b[i];
}
// convert the values to integers.
for(int i = totalAttrs - 1; i >= 0; i--)
    aInt = aInt + a[i] * (int) Math.pow(2, totalAttrs - 1 - i);

for(int i = totalAttrs - 1; i >= 0; i--)
    aorbInt = aorbInt + aorb[i] * (int) Math.pow(2, totalAttrs - 1 - i);

for(int i = 0; i < numberTransac; i++)
    for(int j = totalAttrs - 1; j >= 0; j--)
        dbInt[i] = dbInt[i] + db[i][j] * (int) Math.pow(2, totalAttrs - 1 - j);
//101 & 111 => 101 i.e a.b = a means the attributes of first are
present in second.

for(int i = 0; i < numberTransac; i++)
    if((aInt & dbInt[i]) == aInt)
        c1 = c1 + 1;
//calculate confidence.
for(int i = 0; i < numberTransac; i++)
    if((aorbInt & dbInt[i]) == aorbInt)
        c2 = c2 + 1;

sup = c2 / c1;
System.out.println();
//print output.
if(sup > minsup)
    System.out.println("It will be included in the itemset with
confidence: " + sup);
else
    System.out.println("It won't be included in itemset with
confidence: " + sup);
}
}

```

Output:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
The database is:
1 0 1
0 0 0
0 0 0
1 1 1
The relation is:
1 1 0 -> 0 0 0
It will be included in the itemset with confidence: 1.0
```

Conclusion:

The program shows whether the $a \rightarrow b$ present or not using the apriori algorithm and the association rules. It uses the formula:

$$confidence(A \rightarrow B) = \frac{supportcount(A \cup B)}{supportcount(A)}$$

Experiment 13

Aim: Implement Snow-flake schema in Data warehousing.

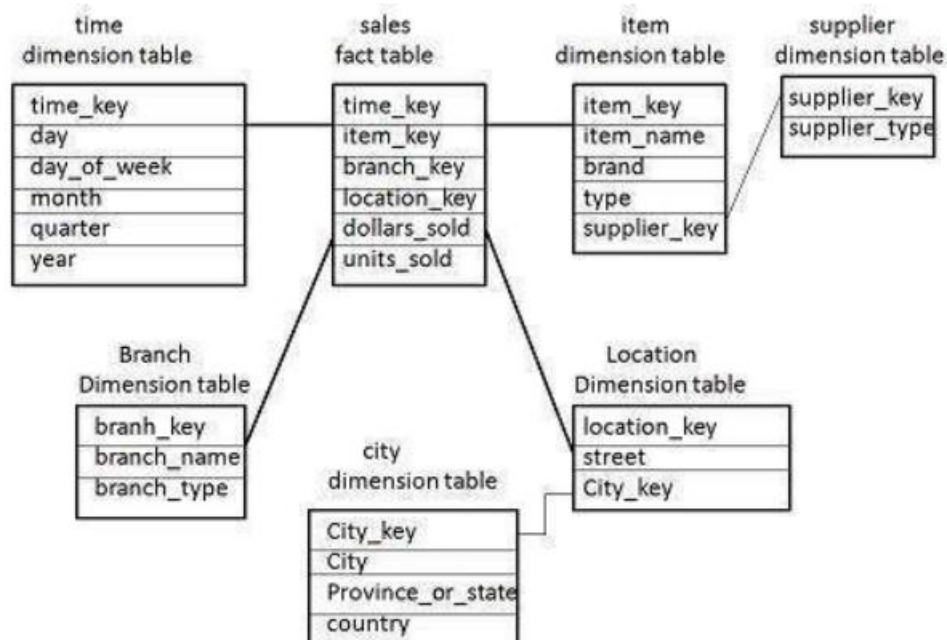
Theory:

A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape like snowflake.

DMQL Code:

```
define cube sales snowflake [time, item, branch, location]:  
dollars sold = sum(sales in dollars), units sold = count(*)  
  
define dimension time as (time key, day, day of week, month, quarter,  
year)  
  
define dimension item as (item key, item name, brand, type, supplier  
(supplier key, supplier type))  
  
define dimension branch as (branch key, branch name, branch type)  
  
define dimension location as (location key, street, city (city key, city,  
province or state, country))
```

Schema:



Experiment 14

Aim: Implement Star schema in Data warehousing.

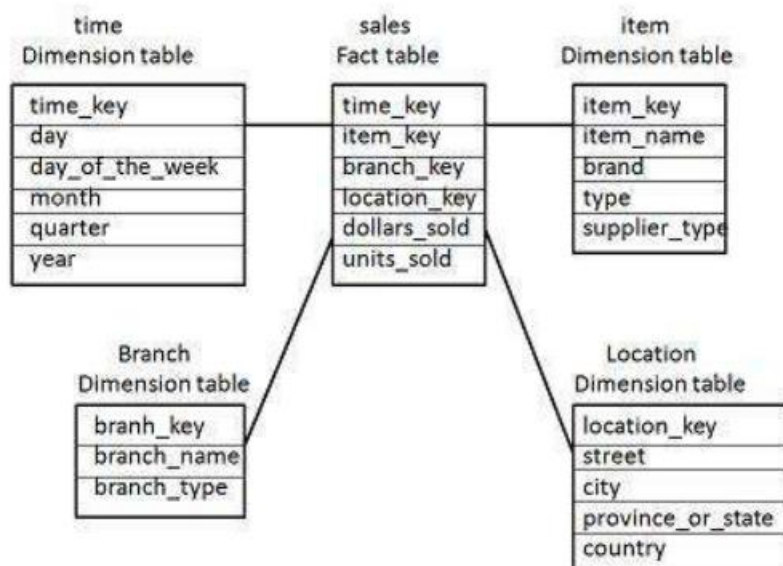
Theory:

Star schema is the fundamental schema among the data mart schema and it is simplest. This schema is widely used to develop or build a data warehouse and dimensional data marts. It includes one or more fact tables indexing any number of dimensional tables.

DMQL Code:

```
define cube sales star [time, item, branch, location]:  
dollars sold = sum(sales in dollars), units sold = count(*)  
  
define dimension time as (time key, day, day of week, month, quarter,  
year)  
  
define dimension item as (item key, item name, brand, type, supplier type)  
  
define dimension branch as (branch key, branch name, branch type)  
  
define dimension location as (location key, street, city, province or  
state, country)
```

Schema:



Experiment 15

Aim: Write a prolog program to find the rules for father, mother, child, son, daughter, brother, sister, uncle, aunt, ancestor given the facts about parent, male and female.

Theory:

PROLOG stands for Programming in Logic and has a very important role in artificial intelligence. It expresses any objects in the form of relations which are described using facts and rules.

Here we will see the family relationship. This is an example of complex relationship that can be formed using Prolog. We want to make a family tree, and that will be mapped into facts and rules, then we can run some queries on them.

- We have defined parent relation by stating the n-tuples of objects based on the given info in the family tree.
- The user can easily query the Prolog system about relations defined in the program.
- A Prolog program consists of clauses terminated by a full stop.
- The arguments of relations can (among other things) be: concrete objects, or constants or general objects such as X and Y. Objects of the first kind in our program are called atoms. Objects of the second kind are called variables.
- Questions to the system consist of one or more goals.

Some facts can be written in two different ways, like gender of family members can be written in either of the forms:

female(rosy)

Now if we want to make mother and sister relationship, then we can write as given below:

mother(X,Y) :- parent(X,Y), female(X).

sister(X,Y) :- parent(Z,X), parent(Z,Y), female(X), X \== Y.

Clause:

female(mary).

female(rosy).

female(sindhu).

```

female(swetha).
female(megha).
male(tanish).
male(sid).
male(vikram).
male(raj).
male(john).
parent(tanish,swetha).
parent(megha,swetha).
parent(john,sid).
parent(raj,sindhu).
parent(sid,vikram).
parent(sindhu,vikram).
parent(sid,rosy).
parent(john,mary).
parent(raj,sid).
mother(X,Y):- parent(X,Y),female(X).
father(X,Y):- parent(X,Y),male(X).
haschild(X):- parent(X,_).
sister(X,Y):- parent(Z,X),parent(Z,Y),female(X),X\==Y.
brother(X,Y):-parent(Z,X),parent(Z,Y),male(X),X\==Y.
son(X,Y,Z) :- male(X),father(Y,X),mother(Z,X).
daughter(X,Y,Z) :- female(X),father(Y,X),mother(Z,X).
uncle(X,Y) :- male(X),brother(X,Z),father(Z,Y).
aunt(X,Y) :- female(X),sister(X,Z),father(Z,Y).

```

Predicates:

```

write('Roll No: 19115038'),nl,
write('Name: Jonnadula Venkata Sai Tanish'),nl,
sister(tanish,swetha).

```

```

write('Roll No: 19115038'),nl,
write('Name: Jonnadula Venkata Sai Tanish'),nl,

```

```
sister(swetha,rosy).
```

```
write('Roll No: 19115038'),nl,  
write('Name: Jonnadula Venkata Sai Tanish'),nl,  
sister(rosy,vikram).
```

Output:



The image displays three sequential screenshots of a code execution window, each showing a Prolog query and its output. Each window has a title bar with a gear icon, a download icon, a close icon, and a maximize icon.

First Screenshot:
Query: `write("Roll No: 19115038"),nl, write("Name: Jonnadula Venkata Sai Tanish"),nl, father(tanish,swetha).`
Output:
Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
true

Second Screenshot:
Query: `write("Roll No: 19115038"),nl, write("Name: Jonnadula Venkata Sai Tanish"),nl, mother(swetha,rosy).`
Output:
Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
false

Third Screenshot:
Query: `write("Roll No: 19115038"),nl, write("Name: Jonnadula Venkata Sai Tanish"),nl, sister(rosy,vikram).`
Output:
Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
true

Experiment 16

Aim: Write a prolog program to find the length of a given list.

Theory:

A list in Prolog is a collection of terms, which is useful for grouping items together, or for dealing with large volumes of related data, etc.

- [red, white, black, yellow] Lists are enclosed by square brackets, and items are separated by commas. The length of a list is the number of items it contains. The length of this list is 4.
- We can think of a list as being made up of two parts: the first element, known as the Head, and everything else, called the Tail.
- Prolog uses a built-in operator, the pipe (|) in order to give us this split for a list. If we use unification with the first example list above and this new operator, as follows:

Syntax for declaring a list:

List_name = [item1, item2, item3,.....]

Clause:

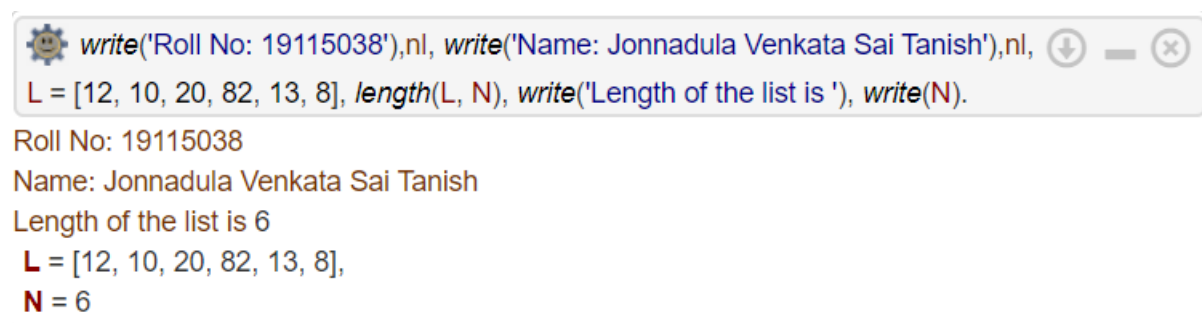
```
len([],0).
```

```
len([_|T],N) :- len(T,X), N is X+1.
```

Predicate:

```
write('Roll No: 19115038'),nl,  
write('Name: Jonnadula Venkata Sai Tanish'),nl,  
L = [12, 10, 20, 82, 13, 8],  
length(L, N),  
write('Length of the list is '), write(N).
```

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,  
L = [12, 10, 20, 82, 13, 8], length(L, N), write('Length of the list is '), write(N).
```

Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
Length of the list is 6
L = [12, 10, 20, 82, 13, 8],
N = 6

Experiment 17

Aim: Write a prolog program to find the last element of a given list.

Theory:

We can think of a list as being made up of two parts: the first element, known as the Head, and everything else, called the Tail.

Prolog uses a built-in operator, the pipe (|) in order to give us this split for a list. If we use unification with the first example list above and this new operator, as follows:

- [Head|Tail] = [red, white, black, yellow]

We will get this output: Head = red Tail = [white, black, yellow]

Clause:

```
last([X]):-
```

```
    write('Last element is '),
```

```
    write(X).
```

```
last([_|Tail]):-
```

```
    last(Tail).
```

Predicate:

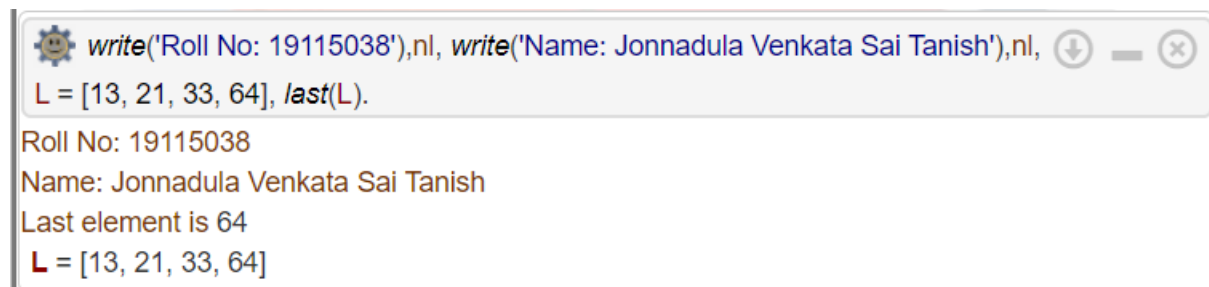
```
write('Roll No: 19115038'),nl,
```

```
write('Name: Jonnadula Venkata Sai Tanish'),nl,
```

```
L = [13, 21, 33, 64],
```

```
last(L).
```

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,  
L = [13, 21, 33, 64], last(L).  
Roll No: 19115038  
Name: Jonnadula Venkata Sai Tanish  
Last element is 64  
L = [13, 21, 33, 64]
```

Experiment 18

Aim: Write a prolog program to delete the first occurrence and also all occurrences of a particular element in a given list.

Theory:

- When the “input” list is empty, then the output list is “empty”.

```
delete_all(_Head, [], []).
```

- When the heads of the “input” and “output” lists don’t match the element being deleted:

```
delete_all(Item, [Head|Tail], [Head|New_Tail]) :-
```

```
Item \= Head, delete_all(Item, Tail, New_Tail).
```

- When the heads of the “input” list matches the element being deleted:

```
delete_all(Item, [Item|Tail], New_Tail) :-
```

```
delete_all(Item, Tail, New_Tail).
```

Deletion of first occurrence

Clause:

```
delete_one(_, [], []).
```

```
Delete_one(Term, [Term|Tail], Tail).
```

```
Delete_one(Term, [Head|Tail], [Head|Result]) :-
```

```
delete_one(Term, Tail, Result).
```

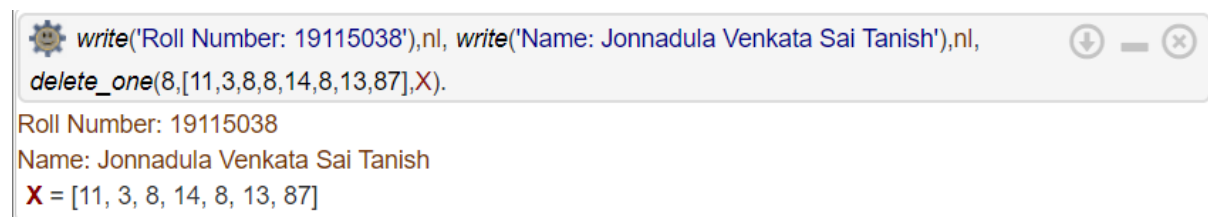
Predicate:

```
write('Roll Number: 19115038'),nl,
```

```
write('Name: Jonnadula Venkata Sai Tanish'),nl,
```

```
delete_one(8,[11,3,8,8,14,8,13,87],X).
```

Output:



```
write('Roll Number: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,  
delete_one(8,[11,3,8,8,14,8,13,87],X).  
Roll Number: 19115038  
Name: Jonnadula Venkata Sai Tanish  
X = [11, 3, 8, 14, 8, 13, 87]
```


Deletion of all occurrences

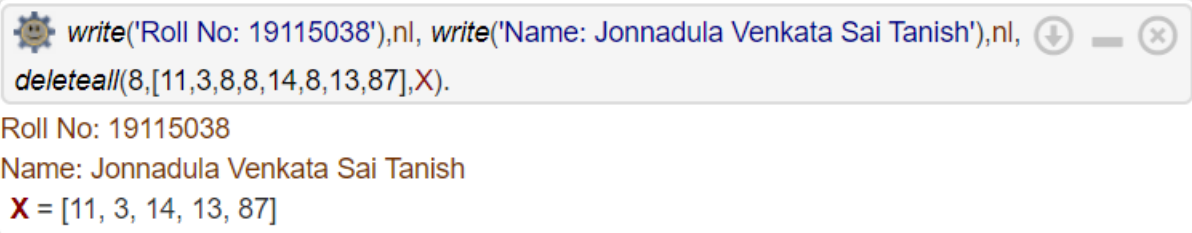
Clause:

```
deleteall(_,[],[]).  
deleteall(X,[H|T],[H|NT]):-  
    H\=X,deleteall(X,T,NT).  
deleteall(X,[X|T],NT):-  
    deleteall(X,T,NT).
```

Predicate:

```
write('Roll No: 19115038'),nl,  
write('Name: Jonnadula Venkata Sai Tanish'),nl,  
deleteall(8,[11,3,8,8,14,8,13,87],X).
```

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,  
deleteall(8,[11,3,8,8,14,8,13,87],X).  
Roll No: 19115038  
Name: Jonnadula Venkata Sai Tanish  
X = [11, 3, 14, 13, 87]
```

Experiment 19

Aim: Write a prolog program to find union of two given sets represented as lists.

Theory:

(Y is list1, Z is list2 and W is new list)

- For each element of list1 which is already present in list2 :

`union([X|Y],Z,W) :- member(X,Z), union(Y,Z,W).`

- For each element of list1 which is not present in list2:

`union([X|Y],Z,[X|W]) :- \+ member(X,Z), union(Y,Z,W).`

- Copy all the values of list2 into new list.

`union([],Z,Z).`

Clause:

`union([X|Y],Z,W) :- member(X,Z), union(Y,Z,W).`

`union([X|Y],Z,[X|W]) :- \+ member(X,Z), union(Y,Z,W).`

`union([],Z,Z).`

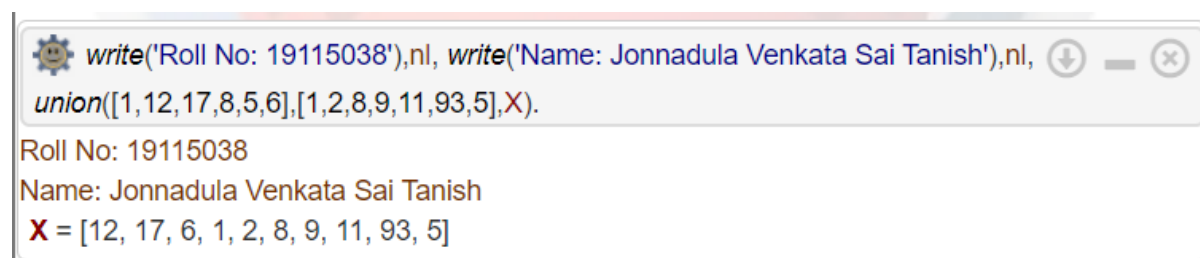
Predicate:

`write('Roll No: 19115038'),nl,`

`write('Name: Jonnadula Venkata Sai Tanish'),nl,`

`union([1,12,17,8,5,6],[1,2,8,9,11,93,5],X).`

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,
union([1,12,17,8,5,6],[1,2,8,9,11,93,5],X).
```

Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
X = [12, 17, 6, 1, 2, 8, 9, 11, 93, 5]

Experiment 20

Aim: Write a prolog program to reverse a given list of values.

Theory:

Reversing a list can be done with

```
reverse([X|Y],Z,W) :- reverse(Y,[X|Z],W).
```

```
reverse([],X,X).
```

This program illustrates Prolog's approach to an important strategy - using an accumulating parameter (the middle variable) - to accumulate a list answer until the computation is finished.

Clause:

```
reverse([X|Y],Z,W) :- reverse(Y,[X|Z],W).
```

```
reverse([],X,X).
```

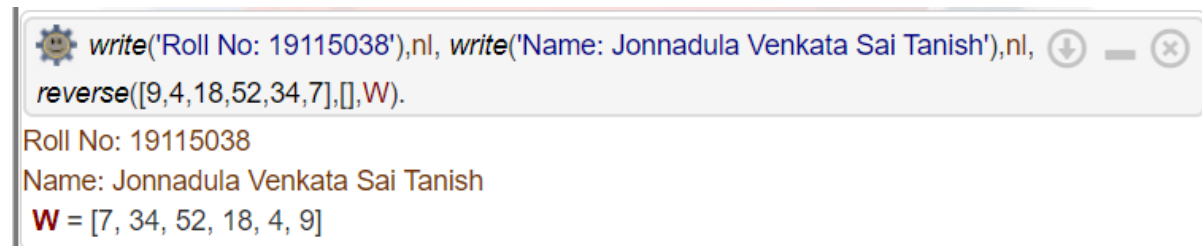
Predicate:

```
write('Roll No: 19115038'),nl,
```

```
write('Name: Jonnadula Venkata Sai Tanish'),nl,
```

```
reverse([9,4,18,52,34,7],[],W).
```

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,  
reverse([9,4,18,52,34,7],[],W).
```

Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
W = [7, 34, 52, 18, 4, 9]

Experiment 21

Aim: Write a prolog program given the knowledge base, If, x is on the top of y, y supports x. If x is above y and they are touching each other, x is on top of y. A cup is above a book. The cup is touching that book. Convert the following into wff's, clausal form; Is it possible to deduce that 'The book supports the cup'.

Theory:

- A cup is above a book.

above(cup,book).

- The cup is touching that book.

touch(cup,book).

- If x is on the top of y, y supports x. If x is above y and they are touching each other, x is on top of y.

support(Y,X) :-above(X,Y), touch(X,Y).

Clause:

above(cup,book).

touch(cup,book).

support(Y,X) :-above(X,Y), touch(X,Y).

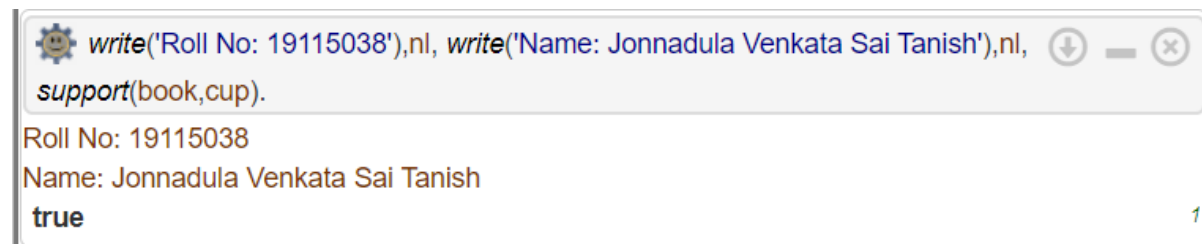
Predicate:

write('Roll No: 19115038'),nl,

write('Name: Jonnadula Venkata Sai Tanish'),nl,

support(book,cup).

Output:



```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl,
support(book,cup).
Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
true
```

Experiment 22

Aim: Write a prolog program given the knowledge base, If Town x is connected to Town y by highway z and bikes are allowed on z, you can get to y from x by bike. If Town x is connected to y by z then y is also connected to x by z. If you can get to town q from p and also to town r from town q, you can get to town r from town p.

- a. Town A is connected to Town B by Road1.
- b. Town B is connected to Town C by Road2.
- c. Town A is connected to Town C by Road3.
- d. Town D is connected to Town E by Road4.
- e. Town D is connected to Town B by Road5.
- f. Bikes are allowed on roads 3, 4, 5.

Bikes are only either allowed on Road1 or on Road2 every day. Convert the following into WFF's, clausal form and deduce that 'One can get to town B from town D'.

Theory:

- If you can get to town q from p and also to town r from town q, you can get to town r from town p.

`connect(P,R,_) :- connect(P,Q,_), connect(Q,R,_).`

- If Town x is connected to Town y by highway z and bikes are allowed on z, you can get to y from x by bike. If Town x is connected to y by z then y is also connected to x by z.

`canget(X,Y,Z) :- (connect(X,Y,Z);connect(Y,X,Z)), bikeallowed(Z).`

Clause:

`connect(A,B,1).`

`connect(B,C,2).`

`connect(A,C,3).`

`connect(D,E,4).`

`connect(D,B,5).`

`connect(P,R,_) :- connect(P,Q,_), connect(Q,R,_).`

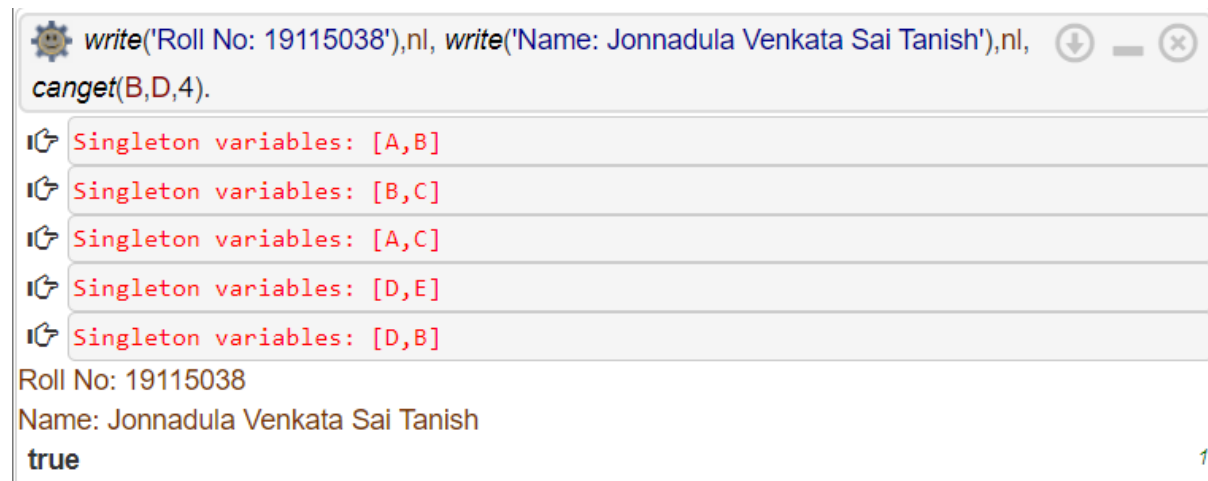
`bikeallowed(3).`

```
bikeallowed(4).  
bikeallowed(5).  
bikeallowed(1) :- not(bikeallowed(2)).  
bikeallowed(2) :- not(bikeallowed(1)).  
canget(X,Y,Z) :- (connect(X,Y,Z);connect(Y,X,Z)), bikeallowed(Z).
```

Predicate:

```
write('Roll No: 19115038'),nl,  
write('Name: Jonnadula Venkata Sai Tanish'),nl,  
canget(B,D,4).
```

Output:



```
write("Roll No: 19115038"),nl, write("Name: Jonnadula Venkata Sai Tanish"),nl,  
canget(B,D,4).
```

Singleton variables: [A,B]
Singleton variables: [B,C]
Singleton variables: [A,C]
Singleton variables: [D,E]
Singleton variables: [D,B]

Roll No: 19115038
Name: Jonnadula Venkata Sai Tanish
true

Experiment 23

Aim: Solve the classical Monkey Banana problem of AI using prolog.

Theory:

The problem is as given below:

- A hungry monkey is in a room, and he is near the door.
- The monkey is on the floor.
- Bananas have been hung from the center of the ceiling of the room.
- There is a block (or chair) present in the room near the window.
- The monkey wants the banana, but cannot reach it. So, if the monkey is clever enough, he can come to the block, drag the block to the center, climb on it, and get the banana. Below are few observations in this case:
 - Monkey can reach the block, if both of them are at the same level. From the above image, we can see that both the monkey and the block are on the floor.
 - If the block position is not at the center, then monkey can drag it to the center.
 - If monkey and the block both are on the floor, and block is at the center, then the monkey can climb up on the block. So the vertical position of the monkey will be changed.
 - When the monkey is on the block, and block is at the center, then the monkey can get the bananas.

We have some predicates that will move from one state to another state, by performing action.

- When the block is at the middle, and monkey is on top of the block, and monkey does not have the banana (i.e., has not state), then using the grasp action, it will change from has not state to have state.
- From the floor, it can move to the top of the block (i.e., on top state), by performing the action climb.
- The push or drag operation moves the block from one place to another.
- Monkey can move from one place to another using walk or move clauses.

Another predicate will be canget(). Here we pass a state, so this will perform move predicate from one state to another using different actions, then perform

canget() on state 2. When we have reached to the state 'has>', this indicates 'has banana'. We will stop the execution.

Clause:

```
move(state(middle,onbox,middle,hasnot),
    grasp,
    state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H),
    climb,
    state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H),
    drag(P1,P2),
    state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H),
    walk(P1,P2),
    state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :-
    move(State1,_,State2),
    canget(State2).
```

Predicate:

```
write('Roll No: 19115038'),nl,
write('Name: Jonnadula Venkata Sai Tanish'),nl,
trace,
canget(state(atdoor, onfloor, atwindow, hasnot)).
```


Output:

```
write('Roll No: 19115038'),nl, write('Name: Jonnadula Venkata Sai Tanish'),nl, trace,  
canget(state(atdoor, onfloor, atwindow, hasnot)).  
Roll No: 19115038  
Name: Jonnadula Venkata Sai Tanish  
Call: canget(state(atdoor,onfloor,atwindow,hasnot))  
Call: move(state(atdoor,onfloor,atwindow,hasnot),_864,_734)  
Exit: move(state(atdoor,onfloor,atwindow,hasnot),walk(atdoor,_740),state(_740,onfloor,atwindow,hasnot))  
Call: canget(state(_740,onfloor,atwindow,hasnot))  
Call: move(state(_736,onfloor,atwindow,hasnot),_880,_748)  
Exit: move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,onbox,atwindow,hasnot))  
Call: canget(state(atwindow,onbox,atwindow,hasnot))  
Call: move(state(atwindow,onbox,atwindow,hasnot),_894,_760)  
Fail: move(state(atwindow,onbox,atwindow,hasnot),_894,_760)  
Fail: canget(state(atwindow,onbox,atwindow,hasnot))  
Redo: move(state(_736,onfloor,atwindow,hasnot),_886,_748)  
Exit: move(state(atwindow,onfloor,atwindow,hasnot),drag(atwindow,_754),state(_754,onfloor,_754,hasnot))  
Call: canget(state(_754,onfloor,_754,hasnot))  
Call: move(state(_750,onfloor,_750,hasnot),_896,_762)  
Exit: move(state(_750,onfloor,_750,hasnot),climb,state(_750,onbox,_750,hasnot))  
Call: canget(state(_750,onbox,_750,hasnot))  
Call: move(state(_750,onbox,_750,hasnot),_910,_774)  
Exit: move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,middle,has))  
Call: canget(state(middle,onbox,middle,has))  
Exit: canget(state(middle,onbox,middle,has))  
Exit: canget(state(middle,onbox,middle,hasnot))  
Exit: canget(state(middle,onfloor,middle,hasnot))  
Exit: canget(state(atwindow,onfloor,atwindow,hasnot))  
Exit: canget(state(atdoor,onfloor,atwindow,hasnot))  
true
```

1

Experiment 24

Aim: Define a LISP function to compute sum of squares.

Theory:

Sum of square means by adding the squares of two numbers.

The mathematical equations in the LISP are not in the standard in which we evaluate the functions, they are in prefix form.

Example: 1. $(a + b)$ is written as $(+ a b)$

2. $(a * b)$ is written as $(* a b)$

Code:

```
(defun sumsqr(x y)
  (format t "Sum of squares is : ~d" (+(* x x)(* y y)))
)
(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(sumsqr 8 14)
```

Output:

Result

executed in 0.832 sec(s)

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Sum of squares is : 260
|
```

Experiment 25

Aim: Define a LISP function to compute difference of squares. (if $x > y$ return $x^2 - y^2$, otherwise return $y^2 - x^2$).

Theory:

If: The if construct has various forms. In the simplest form, it is followed by a test clause, a test action, and some other consequent action(s). If the test clause evaluates to true, then the test action is executed otherwise, the consequent clause is evaluated.

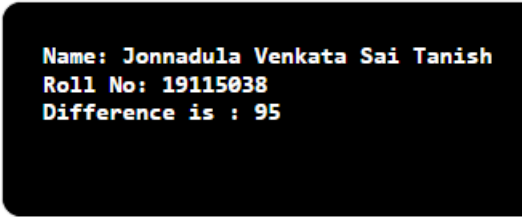
Code:

```
(defun diff(x y)
  (if (> x y)
      (setq result (- (* x x) (* y y)))
      (setq result (- (* y y) (* x x))))
  )
(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(format t "Difference is : ~d" result)
)
(diff 12 7)
```

Output:

Result

executed in 0.847 sec(s)



```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Difference is : 95
```

Experiment 26

Aim: Define a LISP function to compute the factorial of a given number.

Theory:

The factorial of any non-negative number n , (denoted by $n!$), is the product of all positive integers less than or equal to n .

Note: The value of $0!$ is 1.

For example, $3! = 3 \times 2 \times 1$.

Code:

```
(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(defun fact(x)
  (setq result 1)
  (loop
    (setq result (* result x))
    (setq x (- x 1))
    (when (= x 1)(return))
  )
  (format t "Factorial is : ~d " result)
)
(fact 7)
```

Output:

Result

executed in 0.805 sec(s)

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Factorial is : 5040
|
```

Experiment 27

Aim: Define a LISP function to reverse the number entered as parameter in function call.

Theory:

defun: The defun construct is used for defining a function, we will look into it in the Functions chapter.

loop: The loop construct is the simplest form of iteration provided by LISP. In its simplest form, it allows you to execute some statement(s) repeatedly until it finds a return statement.

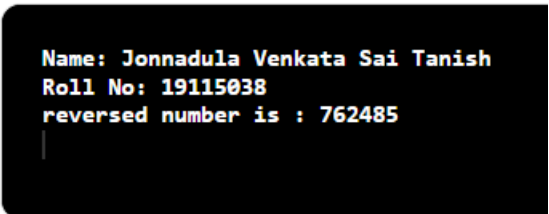
Code:

```
(defun rev(x)
  (setq result 0)
  (loop
    (setq rem (mod x 10))
    (setq result (+ (* result 10) rem))
    (setq x (floor x 10))
    (when (= x 0)(return))
  )
  (write-line "Name: Jonnadula Venkata Sai Tanish")
  (write-line "Roll No: 19115038")
  (format t "reversed number is : ~d" result)
)
(rev 584267)
```

Output:

Result

executed in 0.8 sec(s)



```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
reversed number is : 762485
|
```

Experiment 28

Aim: Write a LISP program containing two functions: one to read input values and one to display them.

Theory:

In LISP, we can read input from STDIN using 'read' and output values into STDOUT using 'write'. This has been demonstrated in the following program

Code:

```
(defun reading()
(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(write-line "enter value of A")
(setq a (read))
(write-line "enter value of B")
(setq b (read))
)

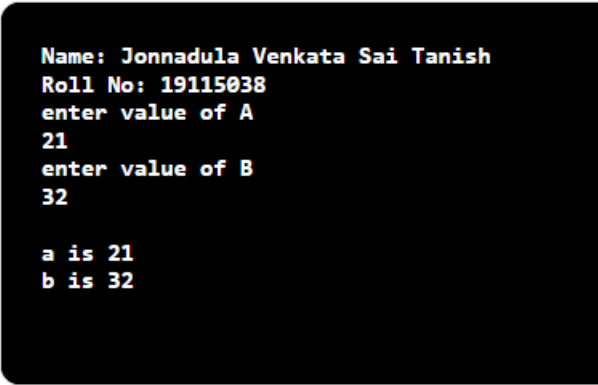
(defun writing()
(format t "~%a is ~d" a)
(format t "~%b is ~d" b)
)

(reading)
(writing)
```

Output:

Result

executed in 9.384 sec(s)

A screenshot of a terminal window with a black background and white text. The output of the LISP program is displayed as follows:

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
enter value of A
21
enter value of B
32

a is 21
b is 32
```

Experiment 29

Aim: Write a LISP program to compute factorial of a given number using recursion.

Theory:

The factorial of a non-negative integer n (denoted by $n!$), is the product of all positive integers less than or equal to n .

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n * (n - 1)! & \text{if } n > 0 \end{cases}$$

The program code employs recursion to calculate the factorial, by using conditional statements to check the value of n . The function is recalled when n is greater than 0.

Code:

```
(defun factorial (N)
  (if (= N 1)
      1
      (* N (factorial (- N 1)))))

(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(format t "Factorial is ~d" (factorial 8))
```

Output:

Result

executed in 0.816 sec(s)

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Factorial is 40320
```

Experiment 30

Aim: Write a LISP Program using recursion to perform GCD of two numbers entered by user.

Theory:

Euclidean algorithm for computing the greatest common divisor.

Originally, the Euclidean algorithm was formulated as follows: subtract number from the larger one until one of the numbers is zero.

Indeed, if g divides a and b , it also divides $a - b$. On the other hand, if g divides $a - b$, it also divides $a - b$. On the other hand, if g divides $a - b$ and b , then it also divides $a = b + (a - b)$, which means that the sets on the common divisors of $\{a, b\}$ and $\{b, a - b\}$ coincide.

Note that a remains the larger number until b is subtracted from it at least $\lfloor a/b \rfloor$ times. Therefore, to speed things up, $a - b$ is substituted with $a - \lfloor a/b \rfloor b = a \bmod b$. Then the algorithm is formulated in an extremely simple way:

$$gcd(a, b) = \begin{cases} a, & \text{if } b = 0 \\ gcd(b, a \bmod b), & \text{otherwise} \end{cases}$$

Code:

```
(write-line "Name: Jonnadula Venkata Sai Tanish")
(write-line "Roll No: 19115038")
(write-line "Enter value of a")
(setq a (read))
(write-line "Enter value of b")
(setq b (read))
(defun gcd1(a b)
  (if (= b 0) a (gcd1 b (mod a b))))

(format t "gcd is : ~d " (gcd1 a b))
```


Output:

Result

executed in 5.326 sec(s)

```
Name: Jonnadula Venkata Sai Tanish
Roll No: 19115038
Enter value of a
16
Enter value of b
12
gcd is : 4
|
```

THE END