

Ex: No: 07 WRITE A PL/SQL BLOCK THAT HANDLES

Date:26.03.24 ALL TYPES OF EXCEPTIONS.

AIM:

To Write a PL/SQL block that handles all types of exceptions

PROCEDURE:

Program to handle divide by zero exception:

```
SQL> DECLARE
    a NUMBER;
    b NUMBER;
    c NUMBER;
BEGIN
    -- Prompt for user input
    a := &a;
    b := &b;

    -- Perform division
    c := a / b;

    -- Output the result
    dbms_output.put_line('Result of division: ' || to_char(c));

EXCEPTION
    WHEN ZERO_DIVIDE THEN
        dbms_output.put_line('Divisor cannot be zero');
    WHEN VALUE_ERROR THEN
        dbms_output.put_line('Invalid input, please enter valid numbers');
END;
/
Enter value for a: 30
old 7:  a := &a;
new 7:  a := 10;
Enter value for b: 0
old 8:  b := &b;
new 8:  b := 0;
```

PL/SQL procedure successfully completed.

```
SQL> SET SERVEROUTPUT ON
```

```
SQL> /
Enter value for a: 30
old 7: a := &a;
new 7: a := 10;
Enter value for b: 0
old 8: b := &b;
new 8: b := 0;
Divisor cannot be zero
```

PL/SQL procedure successfully completed.

```
SQL> /
Enter value for a: 100
old 7: a := &a;
new 7: a := 100;
Enter value for b: 10
old 8: b := &b;
new 8: b := 10;
Result of division: 10
```

PL/SQL procedure successfully completed.

Program to accept sno from supplier table and print name of supplier if the status is greater than 20 else raise exception:

```
SQL> CREATE TABLE supplier (
2   sno VARCHAR2(10),
3   sname VARCHAR2(20),
4   status NUMBER(3),
5   city VARCHAR2(20)
6 );
```

Table created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S101', 'Supplier 1',
10, 'New York');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S2', 'Supplier 2', 25,
'Los Angeles');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S3', 'Supplier 3', 30, 'Chicago');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S4', 'Supplier 4', 15, 'London');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S5', 'Supplier 5', 20, 'Paris');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S6', 'Supplier 6', 22, 'Berlin');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S7', 'Supplier 7', 18, 'Madrid');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S8', 'Supplier 8', 28, 'Rome');
```

1 row created.

```
SQL> INSERT INTO supplier (sno, sname, status, city) VALUES ('S9', 'Supplier 9', 12, 'Tokyo');
```

1 row created.

```
SQL> Select * from supplier;
```

| SNO | SNAME | STATUS CITY |
|-----|------------|----------------|
| S1 | Supplier 1 | 10 New York |
| S2 | Supplier 2 | 25 Los Angeles |
| S3 | Supplier 3 | 30 Chicago |
| S4 | Supplier 4 | 15 London |

| | | |
|----|------------|-----------|
| S5 | Supplier 5 | 20 Paris |
| S6 | Supplier 6 | 22 Berlin |
| S7 | Supplier 7 | 18 Madrid |
| S8 | Supplier 8 | 28 Rome |
| S9 | Supplier 9 | 12 Tokyo |

9 rows selected.

```
SQL> DECLARE
    sn supplier.sno%TYPE;
    snam supplier.sname%TYPE;
    stat supplier.status%TYPE;
    e1 EXCEPTION;
BEGIN
    -- Prompt for user input
    sn := '&serialno';

    -- Select data from supplier table
    SELECT sname, status INTO snam, stat FROM supplier WHERE sno = sn;

    -- Check status and output name if greater than 20
    IF stat > 20 THEN
        dbms_output.put_line(snam);
    ELSE
        RAISE e1;
    END IF;

EXCEPTION
    WHEN e1 THEN
        dbms_output.put_line('Status <= 20');
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('No data found for the given serial number');
END;
/
```

Enter value for serialno: S4

old 8: sn := '&serialno';

new 8: sn := 'S4';

Status <= 20

PL/SQL procedure successfully completed.

```
SQL> /
Enter value for serialno: S2
old 8:  sn := '&serialno';
new 8:  sn := 'S2';
Supplier 2
```

Program to accept sno from supplier table and print name of supplier if he resides in LONDON else raise exception:

```
SQL> DECLARE
2  sn supplier.sno%TYPE;
3  snam supplier.sname%TYPE;
4  stat supplier.status%TYPE;
5  cit supplier.city%TYPE;
6  e1 EXCEPTION;
7  BEGIN
8  -- Prompt for user input
9  sn := '&sn';
10
11  -- Select data from supplier table
12  SELECT sname, status, city INTO snam, stat, cit FROM supplier WHERE sno =
sn;
13
14  -- Debug statement to check retrieved values
15  dbms_output.put_line('Supplier Name: ' || snam);
16  dbms_output.put_line('City: ' || cit);
17
18  -- Check city and output name if it's 'LONDON'
19  IF UPPER(cit) = 'LONDON' THEN
20      dbms_output.put_line(snam);
21  ELSE
22      dbms_output.put_line('Supplier not in London');
23  END IF;
24
25  EXCEPTION
26  WHEN NO_DATA_FOUND THEN
27      dbms_output.put_line('No data found for the given serial number');
28  WHEN OTHERS THEN
29      RAISE e1;
```

30 END;

Enter value for sn: S4

old 9: sn := '&sn';

new 9: sn := 'S4';

Supplier Name: Supplier 4

City: London

Supplier 4

PL/SQL procedure successfully completed.

SQL> /

Enter value for sn: S3

old 9: sn := '&sn';

new 9: sn := 'S3';

Supplier Name: Supplier 3

City: Chicago

Supplier not in London

PL/SQL procedure successfully completed.

RESULT:

Thus the PL/SQL block that handles all types of exceptions has been verified and executed successfully

Ex: No: 08 CREATION OF PROCEDURES.

Date:02.04.24

AIM:

To work with PL / SQL Procedures

PROCEDURE

SQL> SET SERVEROUTPUT ON

SQL> CREATE OR REPLACE PROCEDURE pro

```
2 IS
3   a NUMBER;
4   b NUMBER;
5   c NUMBER;
6   d NUMBER;
7 BEGIN
8   -- Prompt for user input
9   a := &a;
10  b := &b;
11
12  IF a > b THEN
13      c := MOD(a, b);
14      IF c = 0 THEN
15          dbms_output.put_line('GCD is');
16          dbms_output.put_line(b);
17      ELSE
18          dbms_output.put_line('GCD is');
19          dbms_output.put_line(c);
20      END IF;
21  ELSE
22      d := MOD(b, a);
23      IF d = 0 THEN
24          dbms_output.put_line('GCD is');
25          dbms_output.put_line(a);
26      ELSE
```

```
27      dbms_output.put_line('GCD is');
28      dbms_output.put_line(d);
29  END IF;
30  END IF;
31  END;
32  /
```

SQL> SET SERVEROUTPUT ON;

SQL> /

Enter value for a: 212

old 9: a := &a;

new 9: a := 212;

Enter value for b: 12

old 10: b := &b;

new 10: b := 12;

Procedure created.

SQL> execute pro;

GCD is

8

PL/SQL procedure successfully completed.

RESULT:

Thus the implementation of PL/SQL procedure has been verified and executed successfully.

Ex: No: 09 Creation of Database Triggers and functions
Date:16.04.24

AIM:

To work with PL/SQL Triggers for the purpose of monitor the database object(table..etc) and functions .

PROCEDURE:

Program to create a DB trigger before insert for each row on the spj table not allowing insertion for sno 's3' and pno 'p4'

```
SQL> CREATE TABLE spj (  
2   sno VARCHAR2(10),  
3   pno VARCHAR2(10),  
4   jno VARCHAR2(10)  
5 );
```

Table created.

```
SQL> CREATE OR REPLACE TRIGGER t1  
2 BEFORE INSERT ON spj  
3 FOR EACH ROW  
4 BEGIN  
5   IF (:new.sno = 's3' AND :new.pno = 'p4') THEN  
6     raise_application_error(-20000, 'Cannot insert s3, p4');  
7   END IF; -- End IF statement  
8 END; -- End trigger body  
9 /
```

```
SQL> @p1001.sql  
SP2-0310: unable to open file "p1001.sql"  
SQL> INSERT INTO spj VALUES ('s3', 'p4', 'j2', 30);  
INSERT INTO spj VALUES ('s3', 'p4', 'j2', 30)  
*
```

ERROR at line 1:
ORA-00913: too many values

Program to create a DB trigger to update the qty if qty is greater than existing qty

```
SQL> CREATE OR REPLACE TRIGGER t2
  2 BEFORE UPDATE ON spj
  3 FOR EACH ROW
  4 BEGIN
  5   IF (:new.qty < :old.qty) THEN
  6     raise_application_error(-20001, 'Cannot Update');
  7   END IF;
  8 END;
  9 /
```

SQL>@p1002.sql

Trigger created

SQL>update spj set qty=10 where sno='s1';

update spj set qty=10 where sno='s1'

*

ERROR at line 1:

ORA-20001: Cannot update

ORA-06512: at "309038.T2",line 4

ORA-04088: error during execution of trigger '309038.T2'

Program to create a DB trigger not allowing deletion in supplier table

```
SQL> CREATE OR REPLACE TRIGGER t3
  2 BEFORE DELETE ON supplier
  3 FOR EACH ROW
  4 BEGIN
  5   raise_application_error(-20002, 'Deletion Not allowed');
  6 END;
  7 /
```

Trigger created.

SQL> DELETE FROM supplier WHERE sno = 'S1';

DELETE FROM supplier WHERE sno = 'S1'

*

ERROR at line 1:

ORA-20002: Deletion Not allowed

ORA-06512: at "SCOTT.T3", line 2

ORA-04088: error during execution of trigger 'SCOTT.T3'

Create a PL/SQL trigger which prevents the insertion of new record into the employee table.

```
SQL> create or replace trigger trig before insert on emp for each row
  2 begin
  3 raise_application_error (-20998, 'insertion not allowed');
  4 end;
  5 /
```

Trigger created.

```
SQL> INSERT INTO emp (empno, ename) VALUES (123, 'John');
INSERT INTO emp (empno, ename) VALUES (123, 'John')
      *
```

ERROR at line 1:

ORA-20998: insertion not allowed

ORA-06512: at "SCOTT.TRIG", line 2

ORA-04088: error during execution of trigger 'SCOTT.TRIG'

Create a PL/SQL trigger which prevents all DML operations on the table account.

```
SQL> CREATE OR REPLACE TRIGGER kkks
  2 BEFORE INSERT OR DELETE OR UPDATE ON accounts
  3 FOR EACH ROW
  4 BEGIN
  5   raise_application_error(-04098, 'Changes not allowed');
  6 END;
  7 /
```

Trigger created.

```
SQL> INSERT INTO accounts (account_id, account_name) VALUES (6, 'Mortgage');
INSERT INTO accounts (account_id, account_name) VALUES (6, 'Mortgage')
      *
```

ERROR at line 1:

ORA-21000: error number argument to raise_application_error of -4098 is out of range

ORA-06512: at "SCOTT.KKKS", line 2

ORA-04088: error during execution of trigger 'SCOTT.KKKS'

Create a PL/SQL trigger to update the salary of employee if the salary is greater than the existing salary.

```
SQL> CREATE OR REPLACE TRIGGER kkt
  2 BEFORE UPDATE ON emp
  3 FOR EACH ROW
  4 BEGIN
  5   IF :new.sal < :old.sal THEN
  6     raise_application_error(-20002, 'Salary cannot be reduced');
  7   END IF;
  8 END;
  9 /
```

Trigger created.

```
SQL> UPDATE emp SET sal = 3000 WHERE empno = 7844;
```

1 row updated.

```
SQL> UPDATE emp SET sal = 2000 WHERE empno = 7844;
UPDATE emp SET sal = 2000 WHERE empno = 7844
      *
```

ERROR at line 1:

ORA-20002: Salary cannot be reduced

ORA-06512: at "SCOTT.KKT", line 3

ORA-04088: error during execution of trigger 'SCOTT.KKT'

FUNCTIONS

To write a PL/SQL block to find factorial of given number using function

```
SQL> CREATE OR REPLACE FUNCTION fact(limit NUMBER) RETURN
NUMBER IS
  2   ans NUMBER(3);
  3 BEGIN
  4   ans := 1;
  5   FOR i IN 1..limit LOOP
  6     ans := ans * i;
  7   END LOOP;
  8   RETURN ans;
  9 END;
```

10 /

Function created.

```
SQL>
SQL> DECLARE
2   n NUMBER(3);
3   f NUMBER(3);
4 BEGIN
5   n := &limit;
6   f := fact(n);
7   dbms_output.put_line(n || '! = ' || f);
8 END;
9 /
```

Enter value for limit: 5

old 5: n := &limit;

new 5: n := 5;

5! = 120

PL/SQL procedure successfully completed.

Create a function which count total no.of employees having salary less than 6000.

```
SQL> CREATE OR REPLACE FUNCTION count_emp(esal NUMBER) RETURN
NUMBER AS
2   CURSOR vin_cur IS SELECT empno, sal FROM emp;
3   Xno emp.empno%TYPE;
4   Xsal emp.sal%TYPE;
5   C NUMBER := 0;
6 BEGIN
7   OPEN vin_cur;
8
9   LOOP
10      FETCH vin_cur INTO Xno, Xsal;
11      EXIT WHEN vin_cur%NOTFOUND;
12
13      IF Xsal < esal THEN
14         C := C + 1;
15      END IF;
16   END LOOP;
17
18   CLOSE vin_cur;
19
20   RETURN C;
```

```
21 END;
22 /
```

Function created.

```
SQL> /* Function specification */
SQL> DECLARE
  2   Ne NUMBER;
  3   Xsal NUMBER := 3000; -- Example salary value
  4 BEGIN
  5   Ne := count_emp(Xsal);
  6   DBMS_OUTPUT.PUT_LINE('Number of employees with salary less than ' ||
Xsal || ': ' || Ne);
  7 END;
  8 /
```

Number of employees with salary less than 3000: 9

PL/SQL procedure successfully completed.

Program to accept pno from parts table and print name of parts (using function):

```
SQL> -- Create the parts table
SQL> CREATE TABLE parts (
  2   pno VARCHAR2(10),
  3   pname VARCHAR2(100),
  4   CONSTRAINT pk_parts PRIMARY KEY (pno)
  5 );
```

Table created.

```
SQL>
SQL> -- Insert sample values into the parts table
SQL> INSERT INTO parts (pno, pname) VALUES ('P1', 'Engine Assembly');
```

1 row created.

```
SQL> INSERT INTO parts (pno, pname) VALUES ('P2', 'Chassis Frame');
```

1 row created.

```
SQL> INSERT INTO parts (pno, pname) VALUES ('P3', 'Brake System');
```

1 row created.

```
SQL> INSERT INTO parts (pno, pname) VALUES ('P4', 'Transmission Unit');
```

1 row created.

```
SQL> INSERT INTO parts (pno, pname) VALUES ('P5', 'Suspension Kit');
```

1 row created.

```
SQL> CREATE OR REPLACE FUNCTION findname(p parts.pno%TYPE) RETURN  
VARCHAR2 IS  
2   a parts.pname%TYPE;  
3 BEGIN  
4   SELECT pname INTO a FROM parts WHERE pno = p;  
5   RETURN a;  
6 END;  
7 /
```

Function created.

```
SQL> SELECT findname('P1') AS part_name FROM DUAL;
```

PART_NAME

Engine Assembly

RESULT:

Thus the implementation of functions and database triggers has been executed successfully.

Ex: No: 10 Database Connectivity with Front End Tools

Date:23.04.24

EB BILL PREPARATION

AIM:

To prepare a form in VB to generate EB bill and connect it SQL back end.

PROCEDURE:

```
Dim su As Integer
Dim eu As Integer
Dim consumed As Integer
Dim amount As Integer
Dim var1 As Integer
Dim cn As New ADODB.Connection
Dim rs As New ADODB.Recordset
```

```
Private Sub clear_Click()
```

```
    ' Clear all text boxes
```

```
    Text1.Text = ""
```

```
    Text2.Text = ""
```

```
    Text3.Text = ""
```

```
    Text4.Text = ""
```

```
    Text5.Text = ""
```

```
    Text6.Text = ""
```

```
    Text7.Text = ""
```

```
    Text8.Text = ""
```

```
    Text9.Text = ""
```

```
End Sub
```

```
Private Sub Delete_Click()
```

```
    ' Delete the current record in the recordset
```

```
    rs.Delete
```

```
    ' Clear all text boxes after deleting
```

```
    Text1.Text = ""
```

```
    Text2.Text = ""
```

```
    Text3.Text = ""
```

```
    Text4.Text = ""
```

```
    Text5.Text = ""
```



```
' Update the recordset
rs.Update
' Display a message box indicating the record is deleted
MsgBox "Record Deleted"
End Sub
```

```
Private Sub eb_Click()
' Show the DataReport1
DataReport1.Show
End Sub
```

```
Private Sub first_Click()
' Move to the first record in the recordset
rs.MoveFirst
' Update Text1 to Text5 with the corresponding field values
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
' Display a message box indicating this is the first record
MsgBox "This is the first record"
End Sub
```

```
Private Sub gm_Click()
' Calculate consumed and amount based on Text6 and Text7 values
su = Text6.Text
eu = Text7.Text
consumed = eu - su
Text8.Text = consumed
If (consumed > 400) Then
    amount = consumed * 2
Else
    amount = consumed * 1.5
End If
Text9.Text = amount
End Sub
```

```
Private Sub insert_Click()
' Add a new record to the recordset based on Text1 to Text5 values
rs.AddNew
rs.Fields(0) = Text1.Text
rs.Fields(1) = Text2.Text
rs.Fields(2) = Text3.Text
rs.Fields(3) = Text4.Text
```

```

rs.Fields(4) = Text5.Text
' Update the recordset
rs.Update
' Display a message box indicating data was successfully added
MsgBox "Data was successfully added"
End Sub

```

```

Private Sub last_Click()
' Move to the last record in the recordset
rs.MoveLast
' Update Text1 to Text5 with the corresponding field values
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
' Display a message box indicating this is the last record
MsgBox "This is the last record"
End Sub

```

```

Private Sub next_Click()
' Move to the next record in the recordset
rs.MoveNext
' If at the end of the recordset, display a message box
If (rs.EOF) Then
MsgBox "This is the last record"
Else
' Update Text1 to Text5 with the corresponding field values
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
End If
End Sub

```

```

Private Sub previous_Click()
' Move to the previous record in the recordset
rs.MovePrevious
' If at the beginning of the recordset, display a message box
If (rs.BOF) Then
MsgBox "This is the first record"
Else
' Update Text1 to Text5 with the corresponding field values
Text1.Text = rs.Fields(0)

```

```

        Text2.Text = rs.Fields(1)
        Text3.Text = rs.Fields(2)
        Text4.Text = rs.Fields(3)
        Text5.Text = rs.Fields(4)
    End If
End Sub

Private Sub Form_Load()
    ' Establish connection to the database
    Set cn = New ADODB.Connection
    Set rs = New ADODB.Recordset
    cn.Open "dsn=eb;UserId=96017;Password=96017;"
    rs.Open "eb", cn, adOpenDynamic, adLockOptimistic
End Sub

Private Sub report_Click()
    ' Show the DataReport1
    DataReport1.Show
End Sub

Private Sub update_Click()
    ' Update the current record in the recordset with Text1 to Text5 values
    rs.Fields(0) = Text1.Text
    rs.Fields(1) = Text2.Text
    rs.Fields(2) = Text3.Text
    rs.Fields(3) = Text4.Text
    rs.Fields(4) = Text5.Text
    ' Update the recordset
    rs.Update
    ' Display a message box indicating data was updated
    MsgBox "Data Updated"
End Sub

```

Form1

ELECTRIC BILL PREPARATION

| | | | |
|-------------------|------------|-------------|------------|
| Customer Number | 189 | Insert New | Find First |
| Customer Name | Dwight | | previous |
| Door number | 79 | | Next |
| Street | Vallen | Delete This | Find Last |
| City | chengalpat | | Clear |
| Start Unit | | | |
| End Unit | | Update | |
| Unit Consumed | | | |
| Amount To Be Paid | | | |

Copy of bharamit
Data was successfully added in last
OK

Form1

ELECTRIC BILL PREPARATION

| | | | |
|-------------------|-------------|-------------|------------|
| Customer Number | 183 | Insert New | Find First |
| Customer Name | Dwight | | previous |
| Door number | 79 | | Next |
| Street | Vallen | Delete This | Find Last |
| City | chengalpat1 | | Clear |
| Start Unit | 120 | | |
| End Unit | 220 | Update | |
| Unit Consumed | 100 | | |
| Amount To Be Paid | 190 | | |

Generate Amount

Form1

ELECTRIC BILL PREPARATION

Customer Number:

Customer Name:

Door number:

Street:

City:

Start Unit:

End Unit:

Unit Consumed:

Amount To Be Paid:

Insert New

Delete This

Update

Find First

previous

Next

Find Last

Clear

Generate Amount

Form1

ELECTRIC BILL PREPARATION

Customer Number:

Customer Name:

Door number:

Street:

City:

Start Unit:

End Unit:

Unit Consumed:

Amount To Be Paid:

Insert New

Delete This

Update

Find First

previous

Next

Find Last

Clear

Generate Amount

Copy of bhairani1
this is last record
OK

RESULT:

Thus the mini project for eb bill with sql back end was created Successfully

Exp No. 11
Date:30.04.24

Implementation of Decision tree

AIM:

To create a program to implement Decision Tree in Python sklearn

PROCEDURE:

```
import pandas as pd import numpy as np

Import matplotlib.pyplot as plt

from sklearn import metrics import seaborn as sns

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn import tree

iris = load_iris()

data = pd.DataFrame(data = iris.data, columns iris.feature_names)
data['Species'] = iris.target

target = np.unique(iris.target)
target_n = np.unique(iris.target_names) target_dict = dict(zip(target,
target_n))

data['Species']=data['Species'].replace(target_dict)

x =data.drop(columns = "Species")

y = data["Species"]

names_features = x.columns target_labels = y.unique()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
random_state =93)
```

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(max_depth = 3, random_state = 93)
dtc.fit(x_train, y_train)

plt.figure(figsize = (30, 10), facecolor = 'b')

Tree = tree.plot_tree(dtc, feature_names = names_features, class_names =
target_labels, rounded = True, filled = True, fontsize = 14)

plt.show()

y_pred = dtc.predict(x_test)

confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
matrix = pd.DataFrame(confusion_matrix)
axis = plt.axes() sns.set(font_scale = 1.3)

plt.figure(figsize = (10, 7))

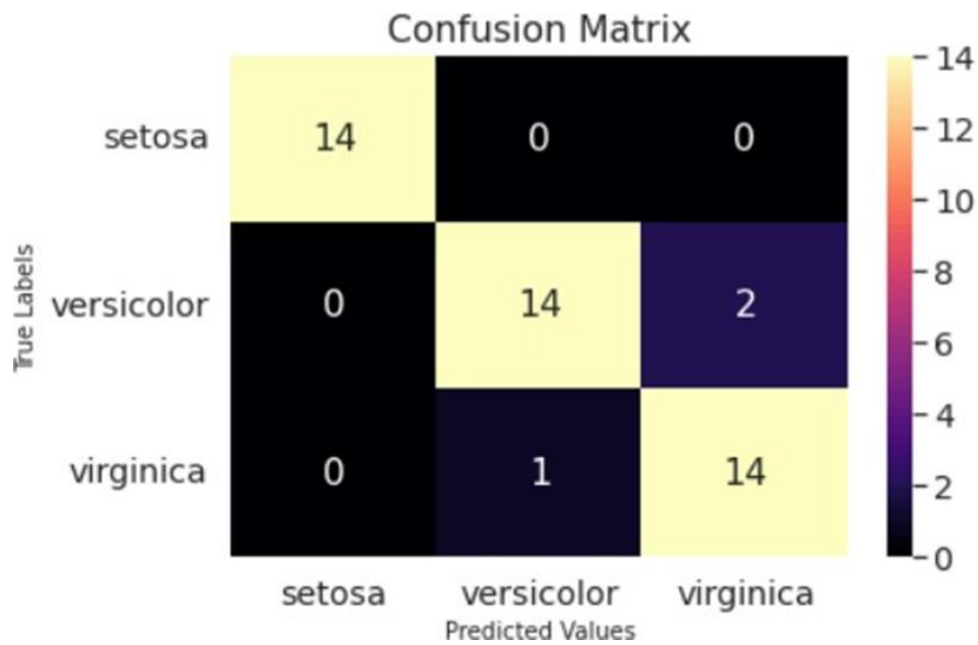
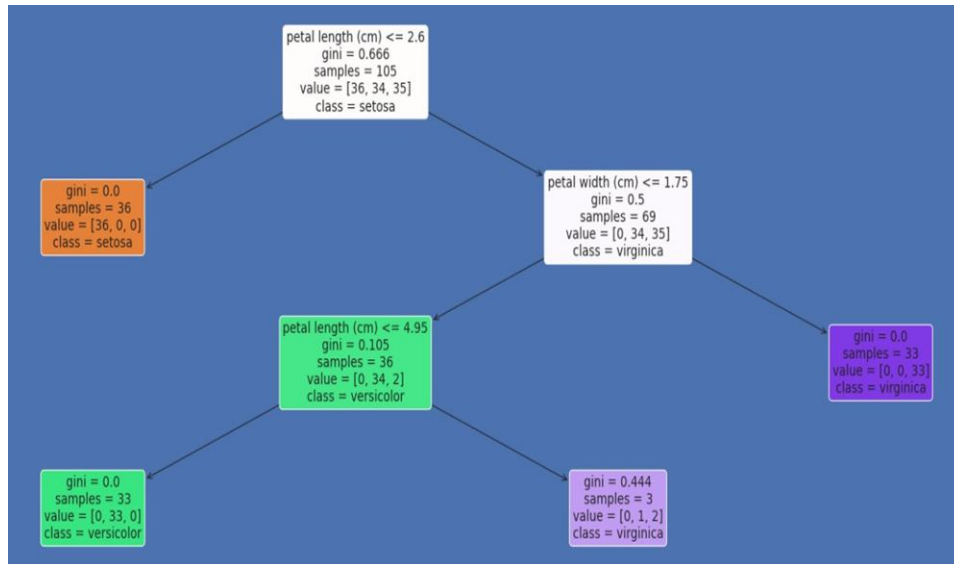
sns.heatmap(matrix, annot = True, fmt = "g", ax = axis, cmap = "magma")
axis.set_title('Confusion Matrix')

axis.set_xlabel("Predicted Values", fontsize = 10)
axis.set_xticklabels([""] + target_labels)

axis.set_ylabel("True Labels", fontsize = 10)
axis.set_yticklabels(list(target_labels), rotation = 0)

plt.show()
```

OUTPUT



RESULT:

Thus the program to implement the decision tree is implemented and the output is obtained.

EX 12 Implementation of Apriori Algorithm

Date: 07.05.24

Aim:

To implement the apriori algorithm in python

Procedure:

```
import numpy as np
```

```
import pandas as pd
```

```
from mlxtend.frequent_patterns import apriori, association_rules
```

```
# Now, we will load the Data
```

```
data1 = pnd.read_excel('Online_Retail.xlsx') data1.head()
```

Output:

| | Invoice No | Stock Code | Description | Q ua nti ty | InvoiceDa te | Uni tP rice | Custo m erID | Country |
|---|---------------|---------------|------------------|----------------------|----------------------------|-------------------|-----------------|----------------|
| 0 | 536365 | 85123A | WHITE HANGING | 6 | 2010-12- 01 08:26:00 | 2.55 | 17850.0 | United Kingdom |

| | | | | | | | | |
|---|--------|--------|--|---|------------------------|------|---------|----------------|
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 | 3.39 | 17850.0 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 2010-12-01 08:26:00 | 2.75 | 17850.0 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 2010-12-01 08:26:00 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 2010-12-01 08:26:00 | 3.39 | 17850.0 | United Kingdom |

Input:

here, we will explore the

columns of the data

data1.columns

Output:

```
Index(['InvoiceNo', 'StockCode', 'Description',  
      'Quantity', 'InvoiceDate', 'UnitPrice',  
      'CustomerID', 'Country'],  
      Dtype = 'object')
```

Input:

```
# Now, we will explore the different  
  
regions of transactions  
  
data1.Country.unique()
```

Output:

```
array(['United Kingdom', 'France', 'Australia',  
      'Netherlands', 'Germany', 'Norway', 'EIRE',  
      'Switzerland', 'Spain', 'Poland', 'Portugal',  
  
      'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',  
  
      'Channel Islands', 'Denmark', 'Cyprus',  
      'Sweden', 'Austria', 'Israel', 'Finland',  
      'Bahrain', 'Greece', 'Hong Kong',  
      'Singapore', 'Lebanon', 'United Arab  
      Emirates', 'Saudi Arabia',  
      'Czech Republic', 'Canada',  
      'Unspecified', 'Brazil', 'USA',  
      'European Community', 'Malta',  
      'RSA'], dtype = object)
```

```
# here, we will strip the extra spaces in the description data1['Description']
```

```
=data1['Description'].str.strip()
```

```
# Now, drop the rows which does not have any invoice number
```

```

data1.dropna(axis = 0, subset = ['InvoiceNo'], inplace = True)

data1['InvoiceNo']=data1['InvoiceNo'].astype('str')

# Now, we will drop all transactions which were done

on

credit_data1=data1[~data1['InvoiceNo'].str.contains('C')]

# Transactions done in France

basket1_France=(data1[data1['Country']=="France"].groupby(['InvoiceNo',
'Description']))['Quantity'].sum().unstack().reset_index().fillna(0)

.set_index('InvoiceNo'))

# Transactions done in the United Kingdom

basket1_UK = (data1[data1['Country'] == "UnitedKingdom"].groupby(['InvoiceNo',
'Description']))['Quantity'].sum().unstack().reset_index().fillna(0).set_index('Invoice
No'))

# Transactions done in Portugal

basket1_Por = (data1[data1['Country'] == "Portugal"].groupby(['InvoiceNo',
'Description']))['Quantity'].sum().unstack().reset_index().fillna(0).set_index('In
voiceNo'))

basket1_Sweden=(data1[data1['Country']=="Sweden"].groupby(['InvoiceN
o',
'Description']))['Quantity'].sum().unstack().reset_index().fillna(0).set_index(
'InvoiceNo'))

# Here, we will define the hot encoding function # for making the data

suitable encode the datasets

```

```
basket1_encoded = basket1_France.applymap(hot_encode1) basket1_France =
```

```
basket1_encoded# for the concerned libraries
```

```
def hot_encode1(P):
```

```
    if(P<= 0):
```

```
        return 0
```

```
    if(P>= 1):
```

```
        return 1
```

```
# Here, we will
```

```
basket1_encoded = basket1_UK.applymap(hot_encode1)
```

```
basket1_UK = basket1_encoded
```

```
basket1_encoded = basket1_Por.applymap(hot_encode1)
```

```
basket1_Por = basket1_encoded
```

```
basket1_encoded = basket1_Sweden.applymap(hot_encode1)
```

```
basket1_Sweden = basket1_encoded
```

```
France:
```

```
    # Build the model
```

```
    frq_items1 = AP(basket1_France, min_support = 0.05,
```

```
    use_colnames = True) # Collect the inferred rules in a dataframe
```

```
    rules1 = AR(frq_items1, metric = "lift", min_threshold = 1)
```

```
rules1 = rules1.sort_values(['confidence', 'lift'], ascending = [False, False])
```

```
print(rules1.head())
```

Output:

| |
|---|
| antecedents \ |
| 45 (JUMBO BAG WOODLAND ANIMALS) |
| 260 (PLASTERS IN TIN CIRCUS PARADE, RED TOADSTOOL ... |

| |
|--|
| 272 (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI... |
| 302 (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO... |
| 301 (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET... |
| |
| consequents antecedent support consequent support \ |
| 45 (POSTAGE) 0.076531 0.765306 |
| 260 (POSTAGE) 0.051020 0.765306 |
| 272 (POSTAGE) 0.053571 0.765306 |
| 302 (SET/6 RED SPOTTY PAPER PLATES) 0.102041s 0.127551 |
| 301 (SET/6 RED SPOTTY PAPER CUPS) 0.102041 0.137755 |
| |

| support | confidence | lift | leverage | conviction | |
|---------|------------|-------|----------|------------|-----------|
| 45 | 0.076531 | 1.000 | 1.306667 | 0.017961 | inf |
| 260 | 0.051020 | 1.000 | 1.306667 | 0.011974 | inf |
| 272 | 0.053571 | 1.000 | 1.306667 | 0.012573 | inf |
| 302 | 0.099490 | 0.975 | 7.644000 | 0.086474 | 34.897959 |
| 301 | 0.099490 | 0.975 | 7.077778 | 0.085433 | 34.489796 |

It can be seen that paper cups, paper and plates are brought together in France.

RESULT:

Thus the python program to implement the apriori algorithm is executed and the output is obtained successfully.