University of Mumbai

Center for Distance & Online Education (CDOE)
Dr. Shankardayal Sharma bhavan, Vidyanagari, Santacruz(E)

PCP CENTER:

Vidyavardhini's College of Engineering and Technology, Vasai (West)

Certificate

This is to certify that Mr/Ms. _____
of MCA Semester _____ has completed the specified Practical in the subject of _____
satisfactorily within this institute as laid down by University of Mumbai during the academic year 20___ to 20___.

Faculty In-charge PCP Coordinator

Examiner

INDEX

Sr. No.	Practical Name	Remark
1	Implementation of Data partitioning through Range.	
2	Implementation of Analytical queries like Roll_UP, CUBE, First, Last	
3	Implementation of Abstract Data Type & Reference	
4	Installation of Pentaho Software.	
5	Introduction to R.	
6	Linear Regression	
7	Analysis of Regression	
8	Logistic Regression	
9	SVM	
10	Varied Algorithms	

Practical No. 1

Aim: Implementation of Data partitioning through Range

Objective: A table that is partitioned by range is partitioned in such a way that each partition contains rows for which the partitioning expression value lies within a given range. Ranges should be contiguous but not overlapping, and are defined using the VALUES LESS THAN operator. The example below demonstrates range partitioning in a MySQL database.

Program:

```
CREATE DATABASE mca;
USE mca;

CREATE TABLE tr (
	id INT,
	name VARCHAR(50),
	purchased DATE
)
PARTITION BY RANGE( YEAR(purchased) ) (
	PARTITION p0 VALUES LESS THAN (1990),
	PARTITION p1 VALUES LESS THAN (1995),
	PARTITION p2 VALUES LESS THAN (2000),
	PARTITION p3 VALUES LESS THAN (2005),
	PARTITION p4 VALUES LESS THAN (2010),
	PARTITION p5 VALUES LESS THAN (2015)
);

INSERT INTO tr VALUES
(1, 'desk organiser', '2003-10-15'),
(2, 'alarm clock', '1997-11-05'),
(3, 'chair', '2009-03-10'),
(4, 'bookcase', '1989-01-10'),
(5, 'exercise bike', '2014-05-09'),
(6, 'sofa', '1987-06-05'),
(7, 'espresso maker', '2011-11-22'),
(8, 'aquarium', '1992-08-04'),
(9, 'study desk', '2006-09-16'),
(10, 'lava lamp', '1998-12-25');

SELECT * FROM tr;
```

```
mysql> show tables;
+---------------+
| Tables_in_mca |
+---------------+
| tr            |
+---------------+
1 row in set (0.01 sec)
```

```
mysql> select * from tr;
+----+----------------+------------+
| id | name           | purchased  |
+----+----------------+------------+
|  4 | bookcase       | 1989-01-10 |
|  6 | sofa           | 1987-06-05 |
|  8 | aquarium       | 1992-08-04 |
|  2 | alarm clock    | 1997-11-05 |
| 10 | lava lamp      | 1998-12-25 |
|  1 | desk organiser | 2003-10-15 |
|  3 | chair          | 2009-03-10 |
|  9 | study desk     | 2006-09-16 |
|  5 | exercise bike  | 2014-05-09 |
|  7 | espresso maker | 2011-11-22 |
+----+----------------+------------+
10 rows in set (00.00 sec)
```

SELECT* FROM tr PARTITION (p2);

```
mysql> SELECT * FROM tr PARTITION
(p2);
+----+------------+------------+
| id | name       | purchased  |
+----+------------+------------+
|  2 | alarm clock | 1997-11-05 |
| 10 | lava lamp   | 1998-12-25 |
+----+------------+------------+
2 rows in set (0.00 sec)
```

SELECT * FROM tr PARTITION (p5);

```
mysql> SELECT * FROM tr PARTITION
(p5);
+----+---------------+------------+
| id | name          | purchased  |
+----+---------------+------------+
|  5 | exercise bike  | 2014-05-09 |
|  7 | espresso maker | 2011-11-22 |
+----+---------------+------------+
2 rows in set (0.00 sec)
```

SELECT* FROM tr PARTITION (p4);

```
mysql> SELECT * FROM tr PARTITION
(p4);
+----+-----------+------------+
| id | name      | purchased  |
+----+-----------+------------+
|  3 | chair     | 2009-03-10 |
|  9 | study desk | 2006-09-16 |
+----+-----------+------------+
2 rows in set (0.00 sec)
```

SELECT PARTITION_NAME, TABLE_ROWS FROM NFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME='tr';

```
mysql> SELECT PARTITION_NAME, TABLE_ROWS FROM
    -> INFORMATION_SCHEMA.PARTITIONS WHERE
    -> TABLE_NAME='tr';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          2 |
| p1             |          1 |
| p2             |          2 |
| p3             |          1 |
| p4             |          2 |
| p5             |          2 |
+----------------+------------+
6 rows in set (0.01 sec)
```

Practical No. 2

Aim: Implementation of Analytical queries like Roll_UP, CUBE, First, Last.

Concept: The last decade has seen a tremendous increase in the use of query, reporting, and on-line analytical processing (OLAP) tools, often in conjunction with data warehouses and data marts. Enterprises exploring new markets and facing greater competition expect these tools to provide the maximum possible decision-making value from their data resources.

Oracle expands its long-standing support for analytical applications in Oracle8i release 8.1.5 with the CUBE and ROLLUP extensions to SQL. Oracle also provides optimized performance and simplified syntax for Top-N queries. These enhancements make important calculations significantly easier and more efficient, enhancing database performance, scalability, and simplicity.

1. **ROLLUP:** Aggregates data hierarchically.

SELECT Region, Product, SUM(SalesAmount) AS TotalSales
FROM Sales
GROUP BY Region, Product WITH ROLLUP;

```
mysql> SELECT Region, Product,
SUM(SalesAmount) AS TotalSales
    -> FROM Sales
    -> GROUP BY Region, Product WITH ROLLUP;
+--------+---------+------------+
| Region | Product | TotalSales |
+--------+---------+------------+
| North  | Laptop  |       2500 |
| North  | Phone   |       4500 |
| North  | NULL    |       7000 |
| South  | Laptop  |       3000 |
| South  | Phone   |       4900 |
| South  | NULL    |       7900 |
| NULL   | NULL    |      14900 |
+--------+---------+------------+
7 rows in set (0.02 sec)
```

2. **CUBE:** Generates all possible subtotal combinations.

SELECT Region, Product, SUM(SalesAmount) AS TotalSales FROM Sales GROUP BY Region, Product
UNION ALL
SELECT Region, NULL AS Product, SUM(SalesAmount) AS TotalSales FROM Sales GROUP BY Region
UNION ALL
SELECT NULL AS Region, Product, SUM(SalesAmount) AS TotalSales FROM Sales GROUP BY Product
UNION ALL

SELECT NULL AS Region, NULL AS Product, SUM(SalesAmount) AS TotalSales FROM Sales;

```
+--------+---------+------------+
| Region | Product | TotalSales |
+--------+---------+------------+
| North  | Laptop  |       2500 |
| North  | Phone   |       4500 |
| South  | Laptop  |       3000 |
| South  | Phone   |       4900 |
| North  | NULL    |       7000 |
| South  | NULL    |       7900 |
| NULL   | Laptop  |       5500 |
| NULL   | Phone   |       9400 |
| NULL   | NULL    |      14900 |
+--------+---------+------------+
9 rows in set (0.01 sec)
```

3. **FIRST_VALUE and LAST_VALUE:** Retrieve the first or last value in a sorted partition.

```
+--------+---------+----------+-------------+-----------+----------+
| Region | Product | Month    | SalesAmount | FirstSale | LastSale |
+--------+---------+----------+-------------+-----------+----------+
| North  | Laptop  | February |        1500 |      1500 |     1000 |
| North  | Laptop  | January  |        1000 |      1500 |     1000 |
| North  | Phone   | February |        2500 |      2500 |     2000 |
| North  | Phone   | January  |        2000 |      2500 |     2000 |
| South  | Laptop  | February |        1800 |      1800 |     1200 |
| South  | Laptop  | January  |        1200 |      1800 |     1200 |
| South  | Phone   | February |        2700 |      2700 |     2200 |
| South  | Phone   | January  |        2200 |      2700 |     2200 |
+--------+---------+----------+-------------+-----------+----------+
8 rows in set (0.01 sec)
```

Practical No. 3

Aim: Implementation of Abstract Data Type & Reference

Objective: The relational approach normalizes everything into tables. The table names are Customer_reltab, Purchase Order_reltab, and Stock_reltab. Each part of an address becomes a column in the Customer_reltab table. Structuring telephone numbers as columns sets an arbitrary limit on the number of telephone numbers a customer can have. The relational approach separates line items from their purchase orders and puts each into its own table, named Purchase Order_reltab and Lineltems_reltab. The relational approach results in the tables describe in the following sections.

Syntax:

1. Customer Table

```
CREATE TABLE Customer_reltab (
        CustNo INT NOT NULL,
        CustName VARCHAR(200) NOT NULL,
        Street VARCHAR(200) NOT NULL,
        City VARCHAR(200) NOT NULL,
        State CHAR(2) NOT NULL,
        Zip VARCHAR(20) NOT NULL,
        Phone1 VARCHAR(20),
        Phone2 VARCHAR(20),
        Phone3 VARCHAR(20),
        PRIMARY KEY (CustNo)
);
```

2. Purchase Order Table

```
CREATE TABLE PurchaseOrder_reltab (
        PONo INT,
        Custno INT,
        OrderDate DATE,
        ShipDate DATE,
        ToStreet VARCHAR(200),
        ToCity VARCHAR(200),
        ToState CHAR(2),
        ToZip VARCHAR(20),
        PRIMARY KEY (PONo),
        FOREIGN KEY (Custno) REFERENCES Customer_reltab(CustNo)
);
```

3. Stock Table

```
CREATE TABLE Stock_reltab (
        StockNo INT PRIMARY KEY,
        Price DECIMAL(10, 2),
        TaxRate DECIMAL(5, 2)
);
```

4. Line Items Table

```
CREATE TABLE LineItems_reltab (
        LineItemNo INT,
        PONo INT,
        StockNo INT,
        Quantity INT,
        Discount DECIMAL(5, 2),
        PRIMARY KEY (PONo, LineItemNo),
        FOREIGN KEY (PONo) REFERENCES PurchaseOrder_reltab(PONo),
        FOREIGN KEY (StockNo) REFERENCES Stock_reltab(StockNo)
);
```

Practical No. 4

Aim: Installation of Pentaho Software.

Objective:

1. Download the Pentaho Data Integration software.
2. Install JRE and JDK. 3. Set up JRE and JDK environment variables for Pentaho Data Integration.

Theory: Pentaho Data Integration - Kettle ETL tool

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

- ✓ Data extraction from source databases
- ✓ Transport of the data
- ✓ Data transformation
- ✓ Loading of data into a data warehouse

Kettle is a set of tools and applications which allows data manipulations across multiple sources. The main components of Pentaho Data Integration are:

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

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

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

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

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

Practical No. 5

Aim: Introduction to R.

Objective: To learn basics of R Programming. How to download and install R.

Theory: What is R Programming

R is an interpreted computer programming language developed by Ross Ihaka and Robert Gentleman in 1993. It is a software environment used to analyze statistical information, graphical representation and reporting.

In the current era, R is one of the most important tools used by researchers, data analyst, statisticians, and marketers for retrieving, cleaning, analyzing, visualizing, and presenting data. R allows integration with the procedures written in the C, C++, .Net, Python, and FORTRAN languages to improve efficiency.

Installation of R

R programming is a very popular language and to work on it we must install two things, i.e., R and R Studio.

R and R Studio works together to create a project on R. Installing R to the local computer is easy. First, we must know which operating system we are using so that we can download the setup accordingly. The official site https://cloud.r-project.org provides binary files for major operating systems including Windows, Linux, and Mac OS. In some Linux distributions, R is installed by default, which we can verify from the console by entering R

Practical No. 6

Aim: Linear Regression.

Description: Linear Regression is a supervised machine learning algorithm used to find a linear relationship between an independent variable (X) and a dependent variable (y).
In this practical, Linear Regression is implemented from scratch without using any machine learning library. The slope (b1) and intercept (b0) are calculated using mean values.
After training the model, it is used to predict the output for a given input.

Code:

```
import numpy as np

class LinearRegression:
    def __init__(self):
        self.b0 = 0
        self.b1 = 0

    def fit(self, X, y):
        X_mean = np.mean(X)
        y_mean = np.mean(y)
        numerator, denomintor = 0, 0
        for i in range(len(X)):
            numerator += (X[i] - X_mean) * (y[i] - y_mean)
            denomintor += (X[i] - X_mean)**2
        self.b1 = numerator / denomintor
        self.b0 = y_mean - (X_mean * self.b1)
        return self.b0, self.b1

    def predict(self, X):
        y_hat = self.b0 + (self.b1 * X)
        return y_hat

if __name__ == '__main__':
    X = np.array([173, 160, 154, 188, 168])
    X = X.reshape(5, 1)
    y = np.array([73, 65, 54, 80, 70])
    model = LinearRegression()
    b0, b1 = model.fit(X, y)
    print(b0, b1)
    y_pred = model.predict([165])
    print(y_pred)
```

Output:

```
[-50.99209602] [0.70813817]
[65.85070258]
```

Practical No. 7

Aim: Analysis of Regression

Description: This practical focuses on analyzing a Linear Regression model using the scikit-learn library.
The model is trained using given data and predictions are generated.
The performance of the regression model is evaluated using Mean Squared Error (MSE) and R^2 score, which indicate the accuracy and goodness of fit of the model.

Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

if __name__ == '__main__':
    X = np.array([173, 160, 154, 188, 168], ndmin=2)
    X = X.reshape(5, 1)
    y = np.array([73, 65, 54, 80, 70])

    model = LinearRegression()
    model.fit(X, y)

    y_hat = model.predict(X)
    print(y_hat)

    mse = mean_squared_error(y_true=y, y_pred=y_hat)
    print(f'Loss Calculated : {mse}')

    r2 = r2_score(y_true=y, y_pred=y_hat)
    print(f'Goodness of Fit : {r2}')
```

Output:

```
[71.51580796 62.31081171 58.06118267 82.13788056 67.9751171 ]
Loss Calculated : 6.920550351288062
Goodness of Fit : 0.9082641788005293
```

Practical No. 8

Aim: Logistic Regression

Description:
Logistic Regression is a classification algorithm used for binary classification problems.
In this practical, the model is trained on sample data and predictions are made. The performance of the model is evaluated using Log Loss, Confusion Matrix, Precision, Recall, and F1-Score to measure classification accuracy.

Code:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix, precision_score, recall_score,
f1_score

if __name__ == '__main__':
    # Input features (reshaped to 2D array)
    X = np.array([6, 2, 5, 9, 1], ndmin=2)
    X = X.reshape(5, 1)

    # Target binary labels
    y = np.array([1, 0, 1, 1, 0])

    # Initialize and train the model
    model = LogisticRegression()
    model.fit(X, y)

    # Predict outcomes
    y_hat = model.predict(X)
    print(y_hat)

    # Calculate performance metrics
    loss = log_loss(y_true=y, y_pred=y_hat)
    print(f'Logarithimic Log : {loss}')

    cm = confusion_matrix(y_true=y, y_pred=y_hat)
    precision = precision_score(y_true=y, y_pred=y_hat)
    recall = recall_score(y_true=y, y_pred=y_hat)
    f1 = f1_score(y_true=y, y_pred=y_hat)

    # Display results
    print(f'Confusion Matrix : {cm}')
```

```
print(f'Precision : {precision}')
print(f'Recall : {recall}')
print(f'F1-Score : {f1}')
```

Output:

```
[1 0 1 1 0]
Logarithimic Log :
2.220446049250313e-16
Confusion Matrix : [[2 0]
 [0 3]]
Precision : 1.0
Recall : 1.0
F1-Score : 1.0
```

Practical No. 9

Aim: SVM

Description:
Support Vector Machine (SVM) is a supervised learning algorithm used for classification tasks.
In this practical, the Iris dataset is used to train the SVM model.
The dataset is split into training and testing sets, and model performance is evaluated using Precision, Recall, and F1-Score.

Code:

```
from sklearn.datasets import load_iris
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score

if __name__ == '__main__':
    # Load the iris dataset
    iris = load_iris()
    X = np.array(iris.data)
    y = np.array(iris.target)

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True, test_size=0.1)

    # Initialize and train the Support Vector Classifier (SVC)
    model = SVC()
    model.fit(X_train, y_train)

    # Predict labels for the test set
    y_hat = model.predict(X_test)

    # Calculate performance metrics using micro-averaging
    precision = precision_score(y_true=y_test, y_pred=y_hat, average='micro')
    recall = recall_score(y_true=y_test, y_pred=y_hat, average='micro')
    f1 = f1_score(y_true=y_test, y_pred=y_hat, average='micro')

    # Print the results
    print(f'Precision : {precision}')
    print(f'Recall : {recall}')
    print(f'F1-Score : {f1}')
```

Output:

```
Precision : 0.9333333333333333
Recall : 0.9333333333333333
F1-Score : 0.9333333333333333
```

Practical No. 10

Aim: Varied Algorithms.

Description:
This practical demonstrates the use of multiple machine learning algorithms on the same dataset.
Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and Naive Bayes classifiers are applied to the Iris dataset.
The output of each algorithm is compared to understand their behavior and prediction results.

Code:

```
from sklearn.datasets import load_iris
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

if __name__ == '__main__':
    # Load the iris dataset
    iris = load_iris()
    X = np.array(iris.data)
    y = np.array(iris.target)

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True,
test_size=0.1)

    # 1. Decision Tree Classifier
    dt = DecisionTreeClassifier()
    dt.fit(X_train, y_train)
    dt_pred = dt.predict([[1, 2, 3, 4]])
    print(f'Decision Tree : {dt_pred}')

    # 2. Random Forest Classifier
    rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
    rf_pred = rf.predict([[1, 2, 3, 4]])
    print(f'Random Forest : {rf_pred}')

    # 3. K-Neighbors Classifier
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
```

```
knn_pred = knn.predict([[1, 2, 3, 4]])
print(f'KNN : {knn_pred}')

# 4. Naive Bayes (GaussianNB)
nb = GaussianNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict([[1, 2, 3, 4]])
print(f'Naive Bayes : {nb_pred}')
```

Output:

```
Decision Tree : [2]
Random Forest : [2]
KNN : [1]
Naive Bayes : [2]
```