**Aim**: Implementation of Data partitioning through Range and List partitioning.

**Objectives:** To understand concept of data fragmentation.

**Tools Used:** MySQL Workbench

**Concept:**

Data Partitioning: - Partitioning is the database process where very large tables are divided into multiple smaller parts. By splitting a large table into smaller, individual tables, queries that access only a fraction of the data can run faster because there is less data to scan. The main of goal of partitioning is to aid in maintenance of large tables and to reduce the overall response time to read and load data for particular SQL operations.

Range Partitioning: - Range partitioning is a type of relational database partitioning where in the partition is based on a predefined range for a specific data field such as uniquely numbered IDs, dates or simple values like currency. A partitioning key column is assigned with a specific range, and when a data entry fits this range, it is assigned to this partition; otherwise, it is placed in another partition where it fits.

List Partitioning: - Unlike range partitioning with list partitioning there is no apparent sense of order between partitions. You can also specify a default partition into which rows that do not map to any other partition are mapped.

**Problem Statement:**

1. Range Partitioning
2. List Partitioning

**Solution:**

1. Range Partitioning

CREATE database lab\_9;

USE lab\_9;

-- 1)

CREATE TABLE salesrange\_atharva (

salesman\_id INT(5),

salesman\_name VARCHAR(20),

sales\_amount DECIMAL(10, 2),

sales\_date DATE

)

PARTITION BY RANGE (YEAR(sales\_date)) (

PARTITION sales\_jan2000 VALUES LESS THAN (2000),

PARTITION sales\_feb2000 VALUES LESS THAN (2001),

PARTITION sales\_mar2000 VALUES LESS THAN (2002),

PARTITION sales\_apr2000 VALUES LESS THAN (2003),

PARTITION sales\_may2000 VALUES LESS THAN (2004)

);

INSERT INTO salesrange\_atharva VALUES

(1, 'Atharva Angre', 20000, '2000-01-10'),

(2, 'Adam Ansari', 40000, '2000-02-10'),

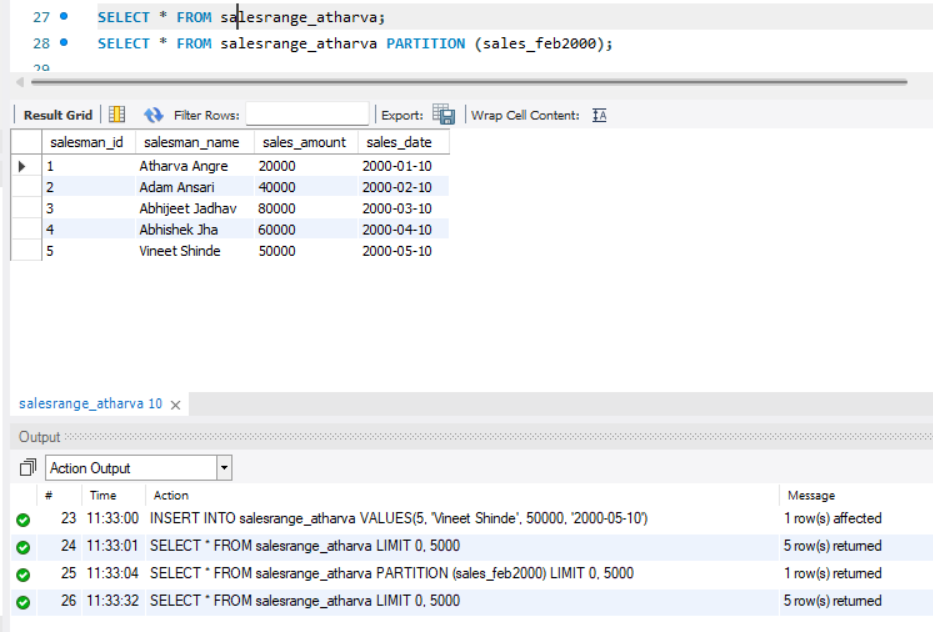
(3, 'Abhijeet Jadha', 80000, '2000-03-10'),

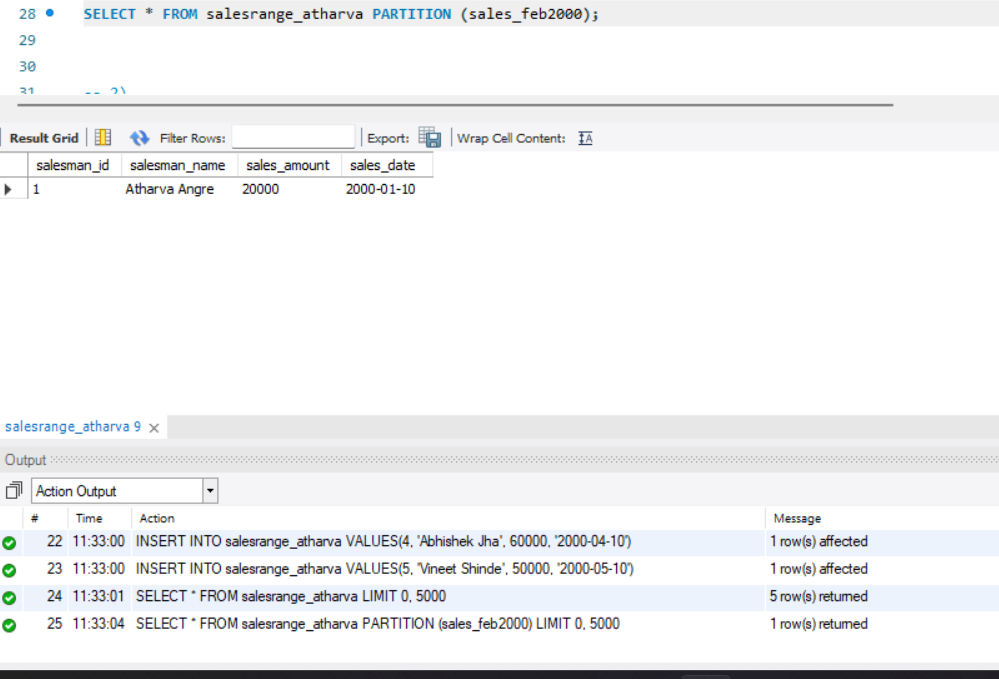
(4, 'Abhishek Jha', 60000, '2000-04-10'),

(5, 'Vineet Shinde', 50000, '2000-05-10');

SELECT \* FROM salesrange\_atharva;

SELECT \* FROM salesrange\_atharva PARTITION (sales\_feb2000);





1. List Partitioning

CREATE TABLE saleslist\_atharva (

salesman\_id INT(5),

salesman\_name VARCHAR(20),

sales\_city VARCHAR(15),

sales\_amount DECIMAL(10,2),

sales\_date DATE

)

PARTITION BY LIST COLUMNS(sales\_city) (

PARTITION sales\_west VALUES IN ('mumbai','pune'),

PARTITION sales\_east VALUES IN ('kolkata'),

PARTITION sales\_north VALUES IN ('chennai'),

PARTITION sales\_south VALUES IN ('delhi'),

PARTITION sales\_default VALUES IN ('default')

);

INSERT INTO saleslist\_atharva VALUES

(1, 'Atharva Angre', 'mumbai', 20000, '2000-01-10'),

(2, 'Adam Ansari', 'delhi', 40000, '2000-02-10'),

(3, 'Abhijeet Jadhav', 'kolkata', 80000, '2000-03-10'),

(4, 'Abhishek Jha', 'chennai', 60000, '2000-04-10'),

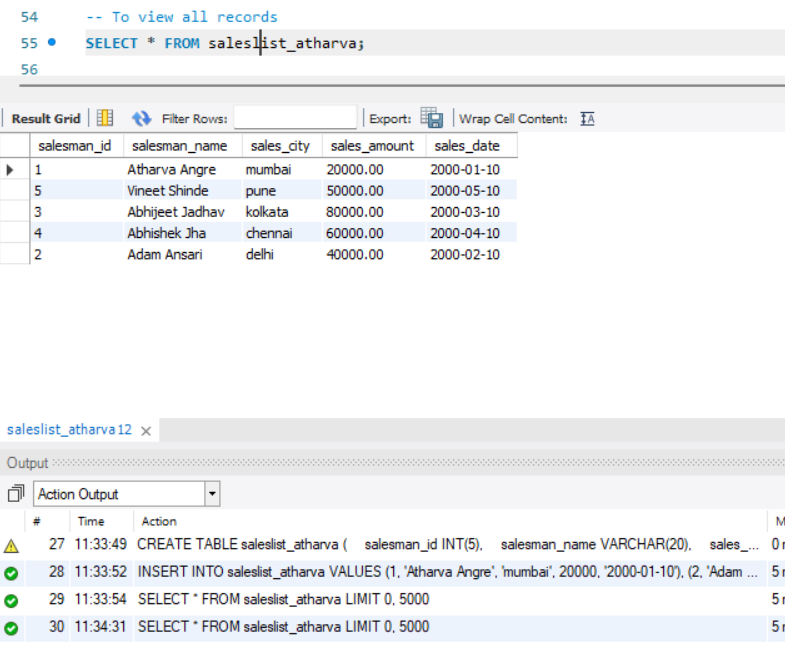
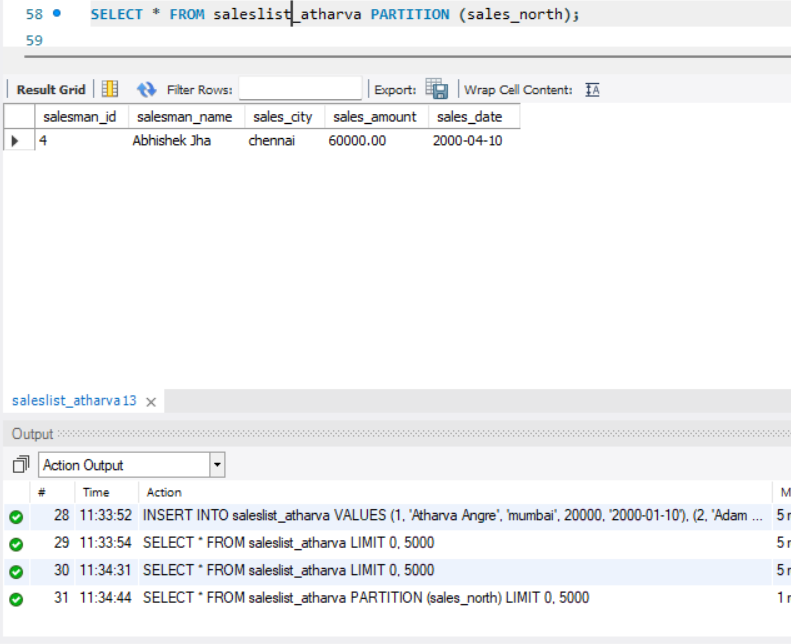
(5, 'Vineet Shinde', 'pune', 50000, '2000-05-10');

-- To view all records

SELECT \* FROM saleslist\_atharva;

-- To view records from north partition

SELECT \* FROM saleslist\_atharva PARTITION (sales\_north);

**Observation**:

To understand and implement the concept of data partitioning in MySQL using range and list partitioning techniques, aiming to reduce query performance degradation caused by fragmentation. This includes learning how partitioning can optimize disk I/O, enhance storage efficiency, and improve query response times by organizing data into smaller, more manageable partitions. Additionally, to explore how MySQL's features like the OPTIMIZE TABLE command and proper index management contribute to reducing fragmentation and maintaining database performance.