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:

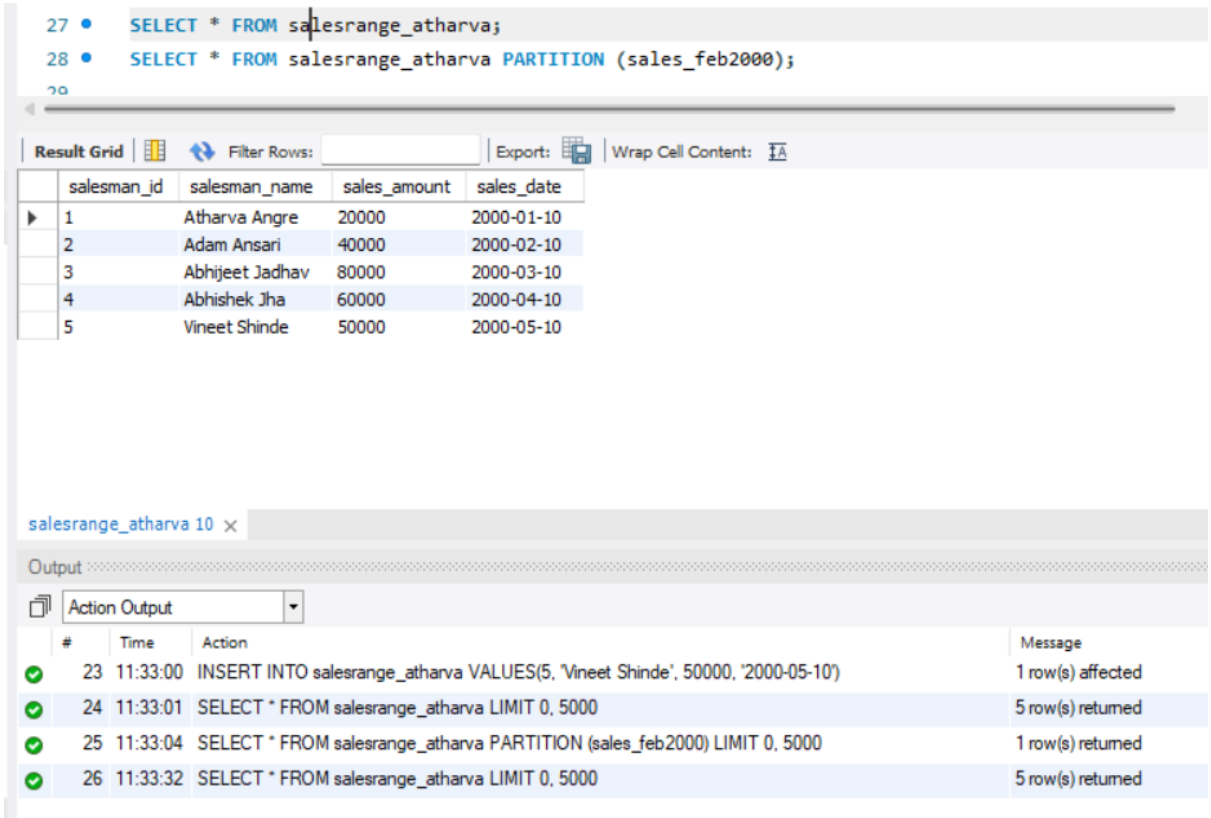
1. Range Partitioning
2. List Partitioning

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);
```



```
28 •  SELECT * FROM salesrange_atharva PARTITION (sales_feb2000);
29
30
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

salesman_id	salesman_name	sales_amount	sales_date
1	Atharva Angre	20000	2000-01-10

salesrange_atharva 9

Output

Action Output

#	Time	Action	Message
22	11:33:00	INSERT INTO salesrange_atharva VALUES(4, 'Abhishek Jha', 60000, '2000-04-10')	1 row(s) affected
23	11:33:00	INSERT INTO salesrange_atharva VALUES(5, 'Vineet Shinde', 50000, '2000-05-10')	1 row(s) affected
24	11:33:01	SELECT * FROM salesrange_atharva LIMIT 0, 5000	5 row(s) returned
25	11:33:04	SELECT * FROM salesrange_atharva PARTITION (sales_feb2000) LIMIT 0, 5000	1 row(s) returned

2. 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);
```

```
54      -- To view all records
55 •    SELECT * FROM saleslist_atharva;
56
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

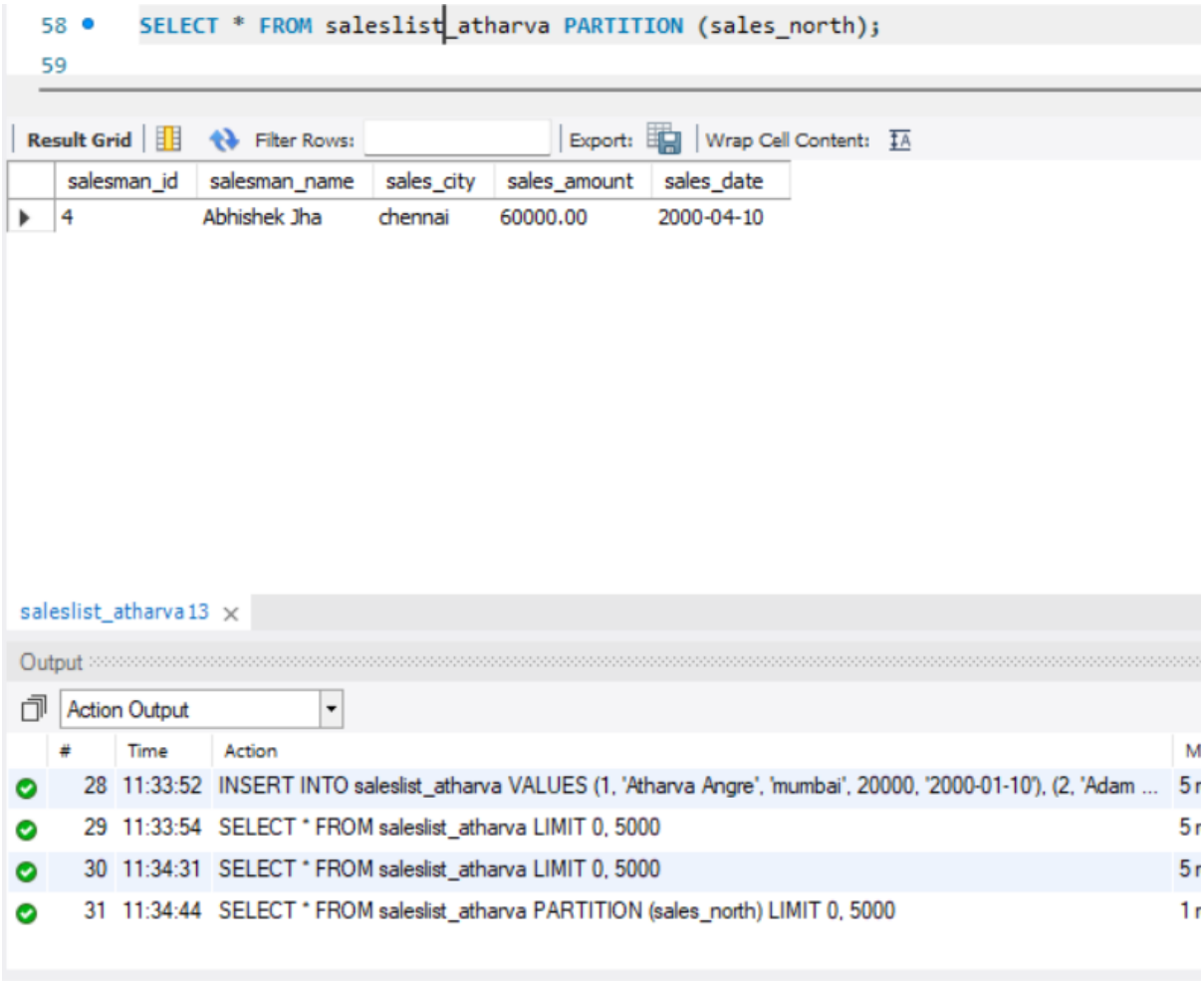
salesman_id	salesman_name	sales_city	sales_amount	sales_date
1	Atharva Angre	mumbai	20000.00	2000-01-10
5	Vineet Shinde	pune	50000.00	2000-05-10
3	Abhijeet Jadhav	kolkata	80000.00	2000-03-10
4	Abhishek Jha	chennai	60000.00	2000-04-10
2	Adam Ansari	delhi	40000.00	2000-02-10

saleslist_atharva 12

Output

Action Output

#	Time	Action	M
27	11:33:49	CREATE TABLE saleslist_atharva (salesman_id INT(5), salesman_name VARCHAR(20), sales_...	0 r
28	11:33:52	INSERT INTO saleslist_atharva VALUES (1, 'Atharva Angre', 'mumbai', 20000, '2000-01-10'), (2, 'Adam ...	5 r
29	11:33:54	SELECT * FROM saleslist_atharva LIMIT 0, 5000	5 r
30	11:34:31	SELECT * FROM saleslist_atharva LIMIT 0, 5000	5 r



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.