ADS Experiment 5

Name: Digvijay Patil
PRN:2122000814
Roll No: B53

Objective

To implement range and hash partitioning on tables, populate data, and perform SQL queries to retrieve and manipulate partitioned data.

Part 1: Range Partitioning

Schema for Table `employees` with Range Partitioning

sql
Copy code

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    fname VARCHAR(25) NOT NULL,
    lname VARCHAR(25) NOT NULL,
    store_id INT NOT NULL,
    department_id INT NOT NULL
)
PARTITION BY RANGE (id) (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (10),
    PARTITION p2 VALUES LESS THAN (15),
    PARTITION p3 VALUES LESS THAN (20),
    PARTITION p4 VALUES LESS THAN (MAXVALUE)
);
```

Inserting Data into `employees`

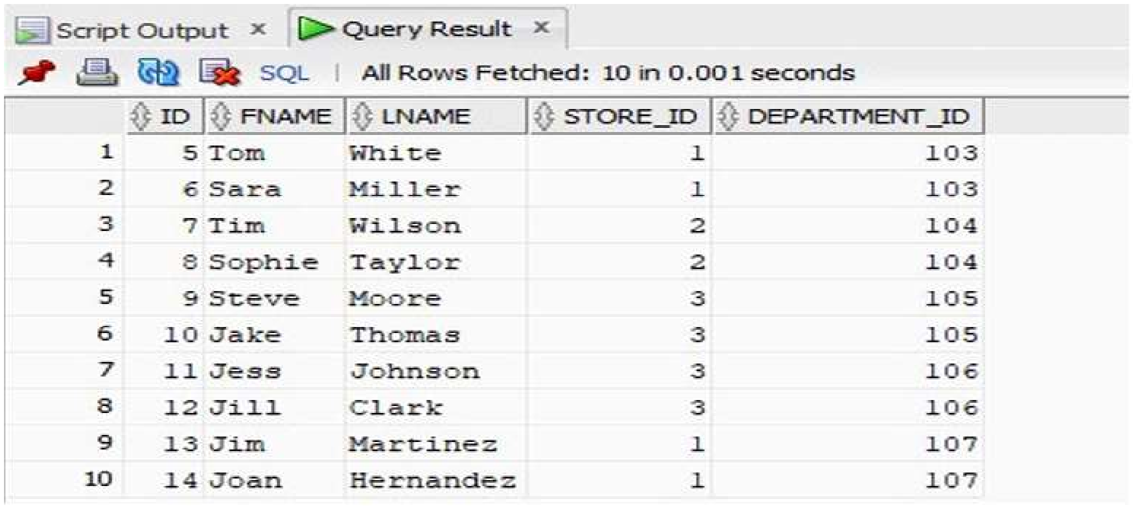
```
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (2, 'Jane', 'Smith', 1, 101);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (3, 'Sam', 'Brown', 2, 102);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (4, 'Sue', 'Davis', 2, 102);
```

```
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (5, 'Tom', 'White', 1, 103);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (6, 'Sara', 'Miller', 1, 103);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (7, 'Tim', 'Wilson', 2, 104);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (8, 'Sophie', 'Taylor', 2, 104);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (9, 'Steve', 'Moore', 3, 105);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (10, 'Jake', 'Thomas', 3, 105);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (11, 'Jess', 'Johnson', 3, 106);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (12, 'Jill', 'Clark', 3, 106);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (13, 'Jim', 'Martinez', 1, 107);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (14, 'Joan', 'Hernandez', 1, 107);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (15, 'Jack', 'Lopez', 2, 108);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (16, 'Jason', 'Gonzalez', 2, 108);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (17, 'Julia', 'Perez', 3, 109);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (18, 'Javier', 'Martinez', 3, 109);
INSERT INTO employees (id, fname, lname, store_id, department_id)
VALUES (19, 'Joseph', 'Ramirez', 1, 110);
```

Queries on Range Partitioned Table

1. Retrieve Employee Details from Partitions P1 and P2

```
SELECT * FROM employees
WHERE id >= 5 AND id < 15;
```

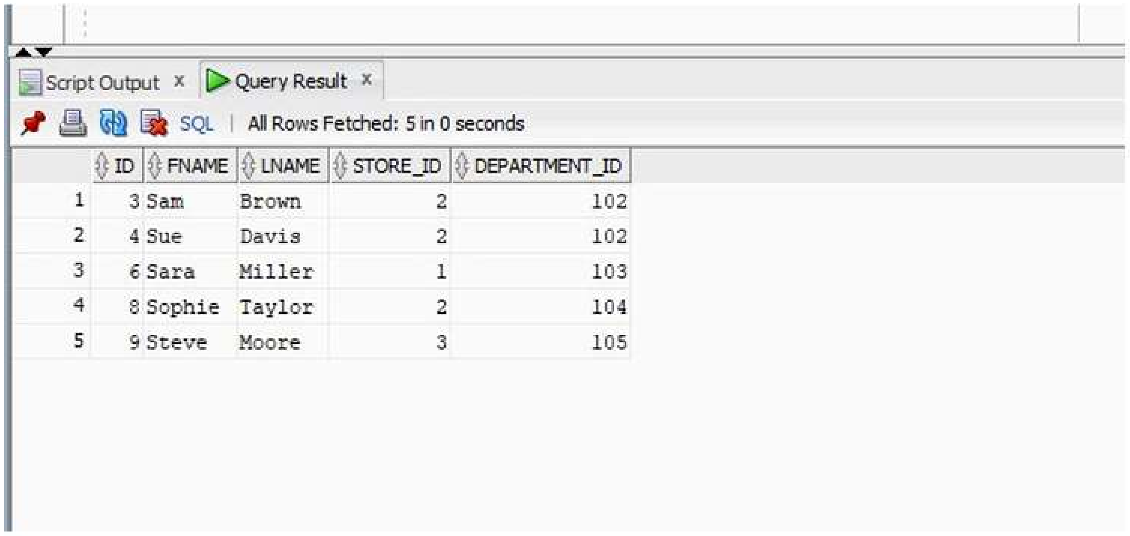

Script Output × | Query Result ×

SQL | All Rows Fetched: 10 in 0.001 seconds

	ID	FNAME	LNAME	STORE_ID	DEPARTMENT_ID
1	5	Tom	White	1	103
2	6	Sara	Miller	1	103
3	7	Tim	Wilson	2	104
4	8	Sophie	Taylor	2	104
5	9	Steve	Moore	3	105
6	10	Jake	Thomas	3	105
7	11	Jess	Johnson	3	106
8	12	Jill	Clark	3	106
9	13	Jim	Martinez	1	107
10	14	Joan	Hernandez	1	107

2. Retrieve Employee Details from Partitions P0 and P1 Where First Name Begins with 'S'

```
SELECT * FROM employees
WHERE id < 10
AND fname LIKE 'S%';
```

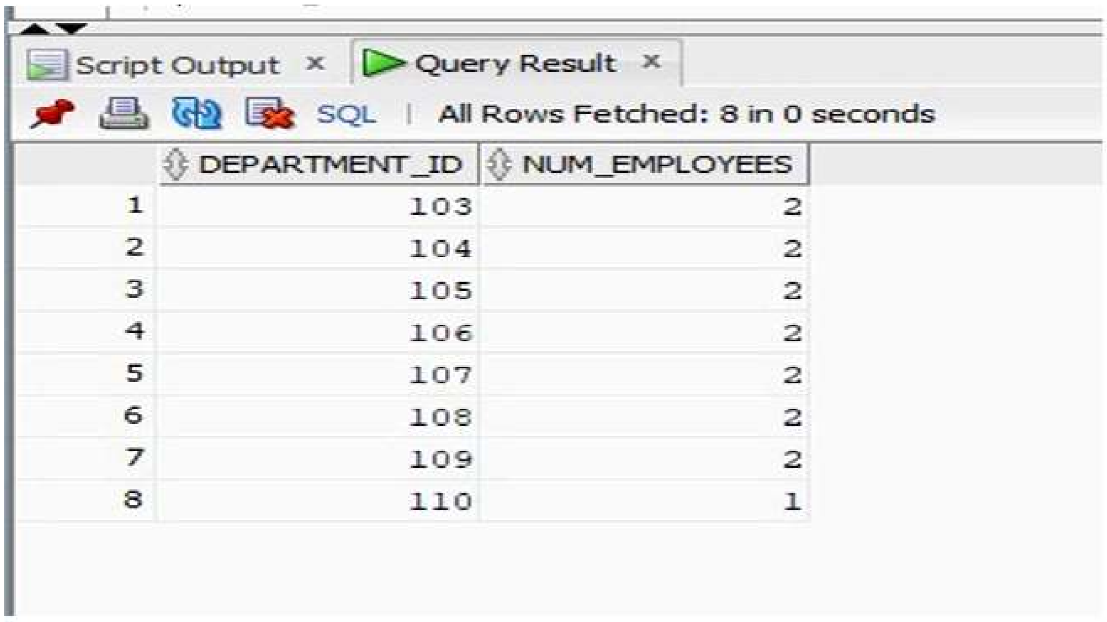

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0 seconds

	ID	FNAME	LNAME	STORE_ID	DEPARTMENT_ID
1	3	Sam	Brown	2	102
2	4	Sue	Davis	2	102
3	6	Sara	Miller	1	103
4	8	Sophie	Taylor	2	104
5	9	Steve	Moore	3	105

3. Count Number of Employees in Each Department (P1, P2, P3)

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
WHERE id >= 5 AND id < 20
GROUP BY department_id;
```



All Rows Fetched: 8 in 0 seconds

```
   DEPARTMENT_ID  NUM_EMPLOYEES
1            103              2
2            104              2
3            105              2
4            106              2
5            107              2
6            108              2
7            109              2
8            110              1
```

Part 2: Hash Partitioning

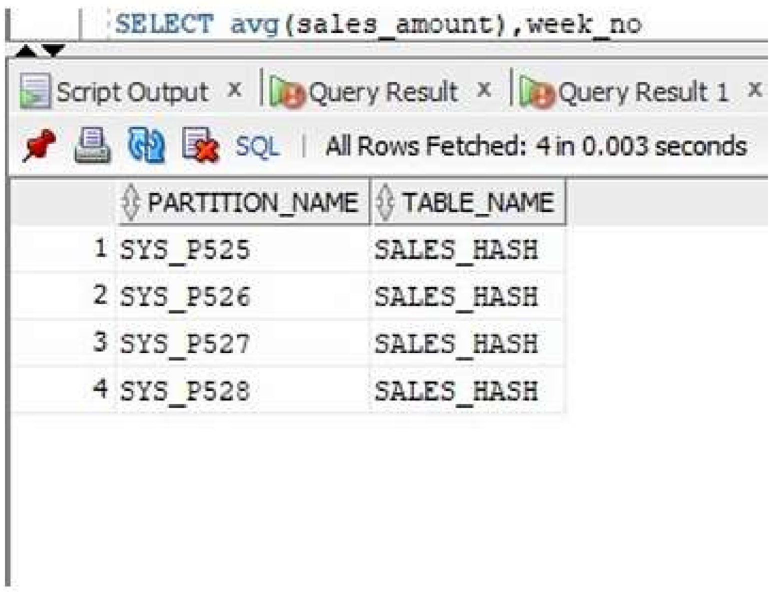
Schema for Table `sales_hash` with Hash Partitioning

```
CREATE TABLE sales_hash (
    salesman_id NUMBER(5) PRIMARY KEY,
    salesman_name VARCHAR2(30),
    sales_amount NUMBER(10),
    week_no NUMBER(2)
)
PARTITION BY HASH (salesman_id)
PARTITIONS 4;
```

Inserting Data into `sales_hash`

```
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (1, 'Arjun Rao', 1500, 1);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (2, 'Priya Sharma', 2000, 2);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (3, 'Ravi Kumar', 3000, 3);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (4, 'Anita Verma', 4000, 4);
```

```
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (5, 'Sandeep Patel', 2500, 5);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (6, 'Neha Yadav', 3500, 6);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (7, 'Rajesh Gupta', 2200, 7);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (8, 'Priyanka Mehta', 2700, 8);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (9, 'Amit Singh', 5000, 9);
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount,
week_no) VALUES (10, 'Rohit Kapoor', 1800, 10);
```



```
SELECT avg(sales_amount),week_no
```

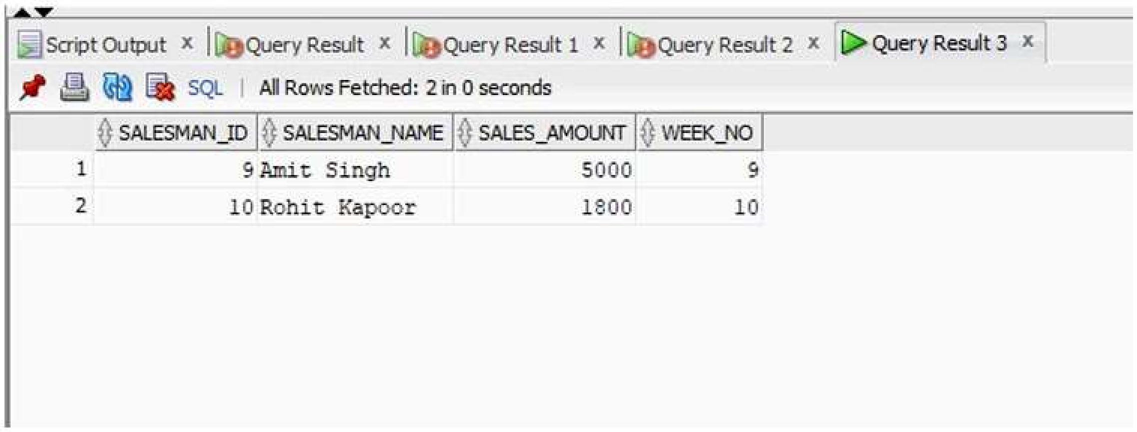
All Rows Fetched: 4 in 0.003 seconds

	PARTITION_NAME	TABLE_NAME
1	SYS_P525	SALES_HASH
2	SYS_P526	SALES_HASH
3	SYS_P527	SALES_HASH
4	SYS_P528	SALES_HASH

Queries on Hash Partitioned Table

1. Retrieve Sales Details from 2nd Partition

```
SELECT * FROM sales_hash PARTITION (SYS_P526);
```

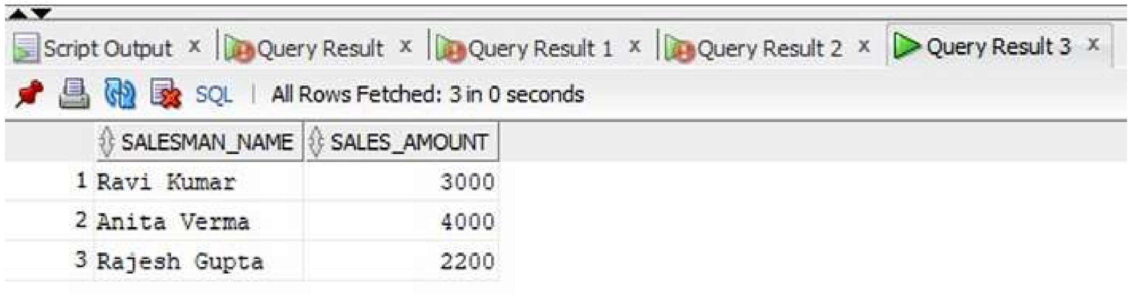


All Rows Fetched: 2 in 0 seconds

	SALESMAN_ID	SALESMAN_NAME	SALES_AMOUNT	WEEK_NO
1	9	Amit Singh	5000	9
2	10	Rohit Kapoor	1800	10

2. Retrieve Salesman Names and Amount from 4th Partition Where Sales Amount is Between 2000 and 5000

```
SELECT salesman_name, sales_amount
FROM sales_hash PARTITION (SYS_P528)
WHERE sales_amount BETWEEN 2000 AND 5000;
```

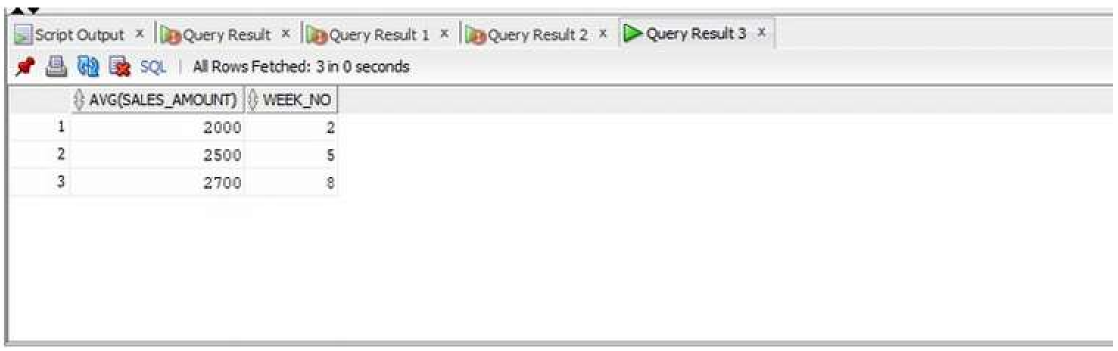


All Rows Fetched: 3 in 0 seconds

	SALESMAN_NAME	SALES_AMOUNT
1	Ravi Kumar	3000
2	Anita Verma	4000
3	Rajesh Gupta	2200

3. Find Average Sales Amount Per Week from 3rd Partition

```
SELECT AVG(sales_amount) AS avg_sales, week_no
FROM sales_hash PARTITION (SYS_P527)
GROUP BY week_no
ORDER BY week_no;
```



Script Output × | Query Result × | Query Result 1 × | Query Result 2 × | Query Result 3 ×

SQL | All Rows Fetched: 3 in 0 seconds

	AVG(SALES_AMOUNT)	WEEK_NO
1	2000	2
2	2500	5
3	2700	8

Conclusion

This experiment demonstrated the creation of range and hash partitions, insertion of data into partitioned tables, and performing queries to retrieve and analyze data effectively within partitions.