

Assignment no.5

Name:Aryan Mangrule

Roll no:46

Prn:2122000220

Range Partition:

Consider a table named `employees` with schema `emp (id int, fname varchar(25) not null, lname varchar(25) not null, store_id int not null, department_id int not null)` with `id` as a primary key and insert 20 records with `id` ranges from 1 to 20.

Make 4 partitions by range:

P0: `id < 5`

P1: `id < 10`

P2: `id < 15`

P3: `id < 20` or `Maxvalue`.

```
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)
```

```
);
```

```
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);
```

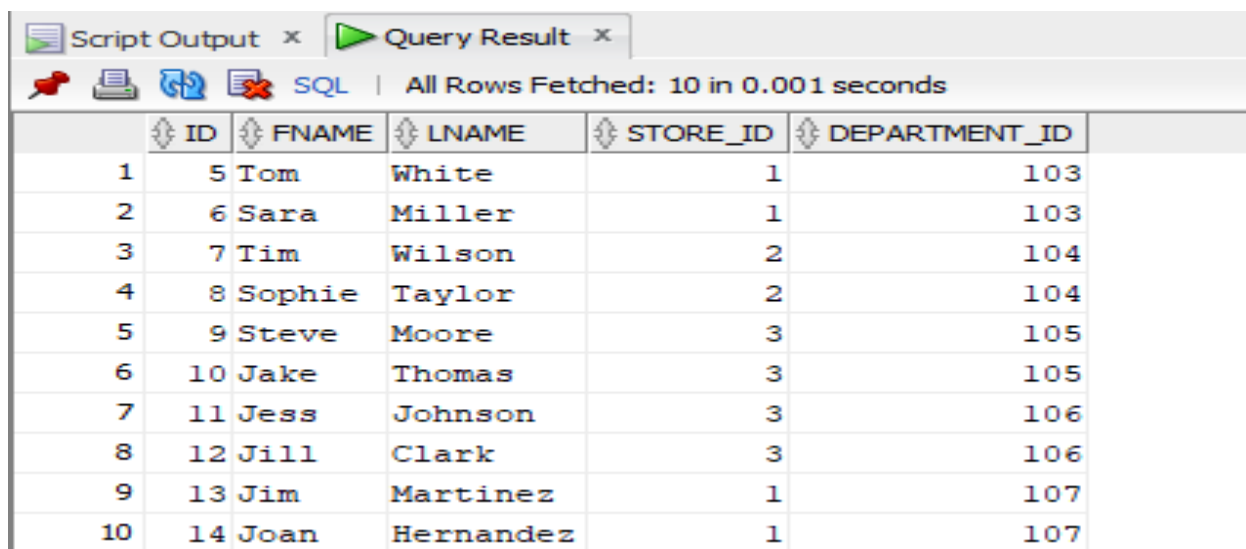
```
INSERT INTO employees (id, fname, lname, store_id, department_id) VALUES (5, 'Tom',  
'White', 1, 103);
```

Assignment no.5

```
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);
```

1. Retrieve employee details from partition P1 and P2.

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



The screenshot shows a database query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the results of the SQL query. The window title is 'SQL | All Rows Fetched: 10 in 0.001 seconds'. The results are shown in a table with 6 columns: ID, FNAME, LNAME, STORE_ID, and DEPARTMENT_ID. The table contains 10 rows of data, corresponding to the employees with IDs 5 through 14.

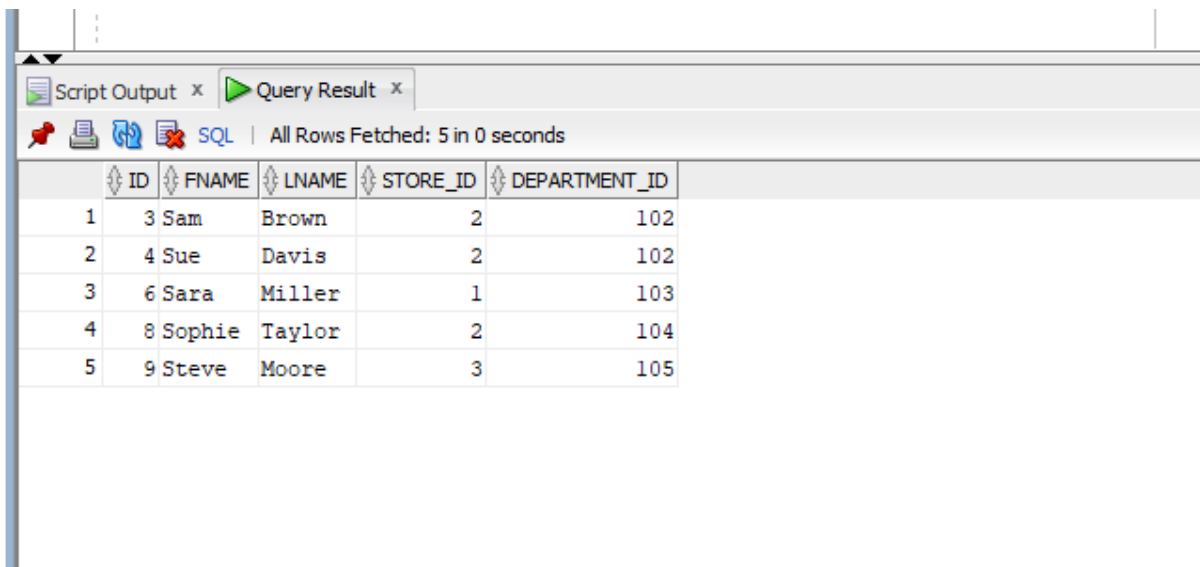
	ID	FNAME	LNAME	STORE_ID	DEPARTMENT_ID
1	5	Tom	White	1	103
2	6	Sara	Miller	1	103
3	7	Tim	Wilson	2	104
4	8	Sophie	Taylor	2	104
5	9	Steve	Moore	3	105
6	10	Jake	Thomas	3	105
7	11	Jess	Johnson	3	106
8	12	Jill	Clark	3	106
9	13	Jim	Martinez	1	107
10	14	Joan	Hernandez	1	107

Assignment no.5

2.Retrieve employee details from partition P0 and P1 where fname begin with 'S'.

Ans:

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



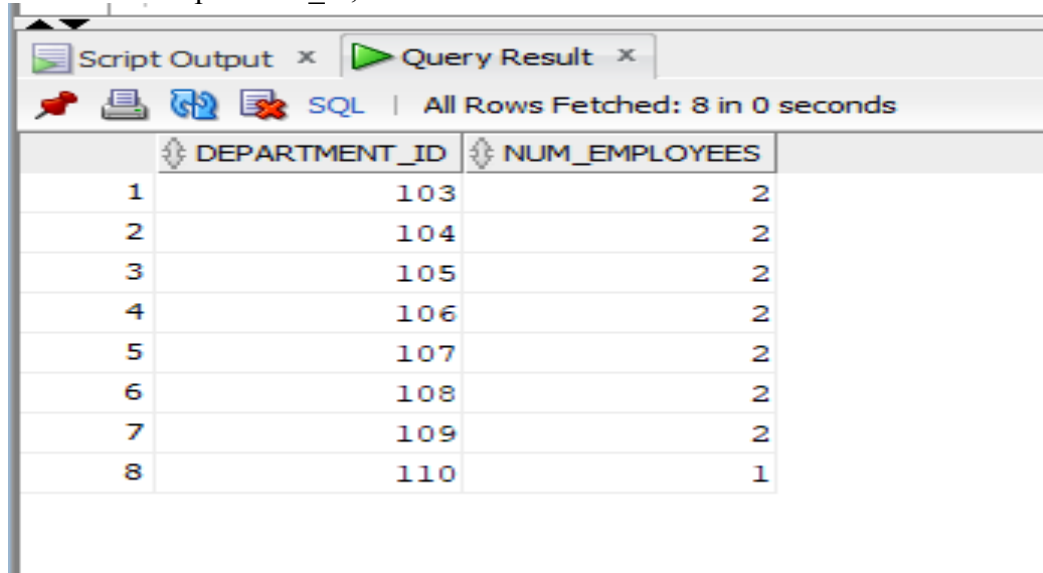
The screenshot shows a database query result window with a tab labeled 'Query Result'. The window displays the results of a SQL query. The status bar indicates 'All Rows Fetched: 5 in 0 seconds'. The table has 5 columns: ID, FNAME, LNAME, STORE_ID, and DEPARTMENT_ID. The data is as follows:

	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 from each department from p1, p2 and p3.)

Ans:

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



The screenshot shows a database query result window with a tab labeled 'Query Result'. The window displays the results of a SQL query. The status bar indicates 'All Rows Fetched: 8 in 0 seconds'. The table has 2 columns: DEPARTMENT_ID and NUM_EMPLOYEES. The data is as follows:

DEPARTMENT_ID	NUM_EMPLOYEES
103	2
104	2
105	2
106	2
107	2
108	2
109	2
110	1

Assignment no.5

Hash Partition:

Consider a table named sales_hash with schema (salesman_id number(5), salesman_name varchar2(30), sales_amount number(10), week_no number(2)) with salesman_id as primary key and insert at least 10 records.

Create 4 partitions using 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;
```

```
drop table 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);
```

Assignment no.5

```
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);
```

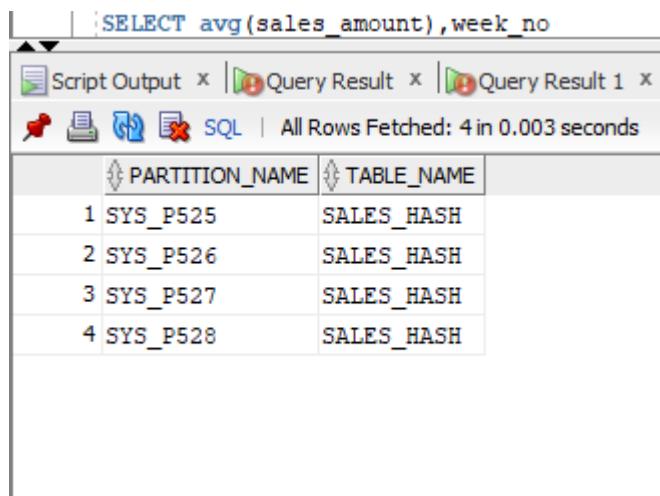
```
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount, week_no) VALUES  
(9, 'Amit Singh', 5000, 9);
```

```
INSERT INTO sales_hash (salesman_id, salesman_name, sales_amount, week_no) VALUES  
(10, 'Rohit Kapoor', 1800, 10);
```

```
SELECT partition_name, table_name
```

```
FROM all_tab_partitions
```

```
WHERE table_name = 'SALES_HASH';
```



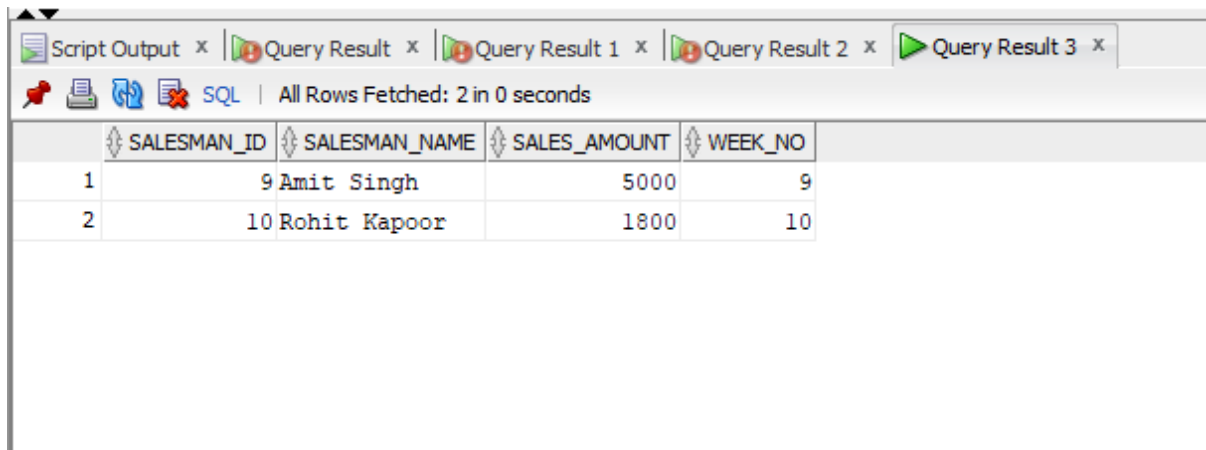
The screenshot shows a SQL query window with the query: `SELECT avg(sales_amount), week_no`. Below the query, there are tabs for 'Script Output', 'Query Result', and 'Query Result 1'. The 'Query Result' tab is active, displaying a table with 4 rows and 2 columns: 'PARTITION_NAME' and 'TABLE_NAME'. The rows are numbered 1 to 4, showing partitions SYS_P525, SYS_P526, SYS_P527, and SYS_P528, all belonging to the table SALES_HASH. The status bar indicates '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

1.Retrieve sales details from 2nd partition.

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

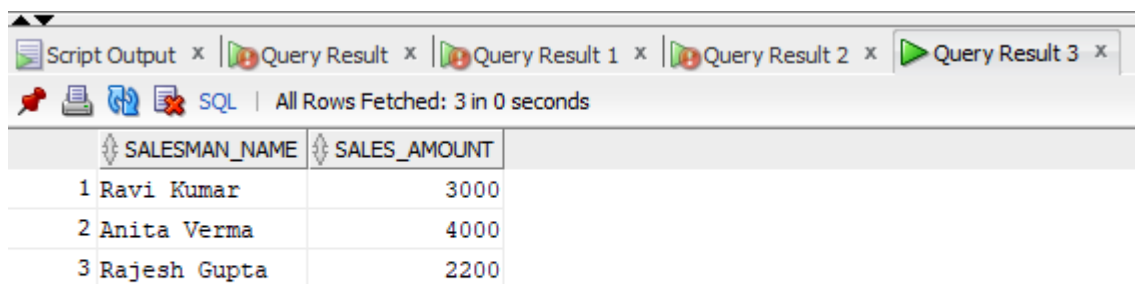
Assignment no.5



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

2.Retrieve name of sales mans and amount from 4th partition where sale amount between 2000 and 5000.

```
SELECT salesman_name, sales_amount  
FROM sales_hash PARTITION (sys_p528)  
WHERE sales_amount BETWEEN 2000 AND 5000;
```



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

3.Find average sale amount per week from 3rd partition.

```
SELECT avg(sales_amount), week_no  
from sales_hash PARTITION (sys_p527)  
group by week_no order by week_no;
```

Assignment no.5

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x		
SQL All Rows Fetched: 3 in 0 seconds		
	AVG(SALES_AMOUNT)	WEEK_NO
1	2000	2
2	2500	5
3	2700	8