Practical 3

Name: Heet Dhanuka
Roll No.: 34
Batch: B2

Aim: To execute following data partitioning technique in data warehouse. Operations can be demonstrated on any schema.

a. Range Partitioning
b. List Partitioning
c. Multicolumn Partitioning
d. Interval Partitioning
e. Reference Partitioning
f. Virtual Column based partitioning
g. Composite Partitioning

Q1. Write a query to create range partitioned table:

➢ Creates a table named- Sales consisting of four partitions, one for each quarter of sales. The column sale_date are the partitioning columns, while their values constitute the partitioning key of a specific row.

➢ Each partition is given a name (sales_q1, sales_q2, ...), and each partition is contained in a separate tablespace (tsa, tsb, ...)

➢ The columns for table must be prod_id, cust_id, promo_id, quantity sold, amount_sold all in number format and sale_date.

```
SQL> create tablespace tsa_1 datafile 'tsa.dbf' size 10m;

Tablespace created.

SQL> create tablespace tsb_1 datafile 'tsb.dbf' size 10m;

Tablespace created.

SQL> create tablespace tsc_1 datafile 'tsc.dbf' size 10m;

Tablespace created.

SQL> create tablespace tsd_1 datafile 'tsd.dbf' size 10m;

Tablespace created.

SQL> create table sales_1
  2  (prod_id number(2),
  3  cust_id number(3),
  4  promo_id number(2),
  5  quantity number(3),
  6  amount number(3),
  7  sale_date date)
  8  partition by range(sale_date)
  9  ( partition sales_q1 values less than (to_date(' 01-APR-2022','dd-mon-yyyy')) TABLESPACE tsa_1,
 10  partition sales_q2 values less than (to_date(' 01-JUL-2022','dd-mon-yyyy')) TABLESPACE tsb_1,
 11  partition sales_q3 values less than (to_date(' 01-OCT-2022','dd-mon-yyyy')) TABLESPACE tsc_1,
 12  partition sales_q4 values less than (to_date(' 01-JAN-2023','dd-mon-yyyy')) TABLESPACE tsd_1);

Table created.
```

Q2. Create the same table as in Q1. With a different name with ENABLE ROW MOVEMENT. Bring out the difference in these two tables.

```
SQL> alter table sales_1 enable row movement ;

Table altered.
```

Q3. Create a table with list partition as follows:
Table having columns deptno, deptname, quarterly_sales and state.
Create partition on state:

➢ Northwest on OR and WA
➢ Southwest on AZ, UT and NM
➢ northeast on NY, VM and NJ
➢ southeast on FL and GA
➢ northcentral on SD and WI
➢ southcentral on OK and TX

Add the following entries into the table and make conclusion to which partition the entry maps:

➢ (10, 'accounting', 100, 'WA')
➢ (20, 'R&D', 150, 'OR')

➢ (30, 'sales', 100, 'FL')
➢ (40, 'HR', 10, 'TX')
➢ (50, 'systems engineering', 10, 'CA')

```
SQL> CREATE TABLE dept_sales1(
  2  deptno number(10),
  3  deptname varchar(10),
  4  quarterly_sales number(10),
  5  state varchar(10)
  6  )
  7  partition by list(state)
  8  (
  9  partition northwest values ('OR','WA'),
 10  partition southwest values('AZ','UT','NM'),
 11  partition northeast values ('NY','VM','NJ'),
 12  partition southeast values ('FL','GA'),
 13  partition northcentral values ('SD','WI'),
 14  partition southcentral values ('OK','TX')
 15  );

Table created.
```

```
SQL> INSERT INTO dept_sales1 VALUES (10, 'accounting', 100, 'WA');

1 row created.
```

```
SQL> INSERT INTO dept_sales1 VALUES (20, 'R&D', 150, 'OR');
Enter value for d: &D
old   1: INSERT INTO dept_sales1 VALUES (20, 'R&D', 150, 'OR')
new   1: INSERT INTO dept_sales1 VALUES (20, 'R&D', 150, 'OR')

1 row created.
```

```
SQL> INSERT INTO dept_sales1 VALUES (30, 'sales', 100, 'FL');

1 row created.
```

```
SQL> INSERT INTO dept_sales1 VALUES (40, 'HR', 10, 'TX');

1 row created.
```

Q4. Create a multi-column range partitioned table as directed:
Create a table with the actual DATE information in three separate columns: year, month, and day. Also amount_ sold.
Create following partitions:
Before 2001: Less than jan 2001
Less than april 2001
Less than july 2001
Less than oct 2001
Less than jan 2002
Future with max incoming value
Insert values into table and show to which partition does the value belong.
(2001,3,17, 2000);
(2001,11,1, 5000);
(2002,1,1, 4000);

```
SQL> CREATE TABLE sales4_1 (
  2  year INT,
  3  month INT,
  4  day INT,
  5  amount_sold INT
  6  )
  7  PARTITION BY RANGE (year, month, day) (
  8  PARTITION p1 VALUES LESS THAN (2001, 1, 1),
  9  PARTITION p2 VALUES LESS THAN (2001, 4, 1),
 10  PARTITION p3 VALUES LESS THAN (2001, 7, 1),
 11  PARTITION p4 VALUES LESS THAN (2001, 10, 1),
 12  PARTITION p5 VALUES LESS THAN (2002, 1, 1),
 13  PARTITION p_future VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE)
 14  );

Table created.
```

```
SQL> INSERT INTO sales4_1 (year, month, day, amount_sold)
  2  VALUES (2001, 3, 17, 2000);

1 row created.

SQL> INSERT INTO sales4_1 (year, month, day, amount_sold)
  2  VALUES (2001, 11, 1, 5000);

1 row created.

SQL> INSERT INTO sales4_1 (year, month, day, amount_sold)
  2  VALUES (2002, 1, 1, 4000);

1 row created.
```

Q5. Create a multicolumn partitioned table as directed:
➢ Table supplier_parts, storing the information about which suppliers deliver which parts.To distribute the data in equal-sized partitions, it is not sufficient to partition the table based on the supplier_id, because some suppliers might provide hundreds of thousands of parts, while others provide only a few specialty parts. Instead, you
partition the table on (supplier_id, partnum) to manually enforce equal-sized partitions.

➢ Insert the following values
(5,5, 1000); (5,150, 1000); (10,100, 1000);

```
SQL> CREATE TABLE supplier_parts_1 (
  2  supplier_id INT,
  3  partnum INT,
  4  quantity INT
  5  )
  6  PARTITION BY RANGE (supplier_id, partnum) (
  7  PARTITION p1 VALUES LESS THAN (6, 0),
  8  PARTITION p2 VALUES LESS THAN (11, 0),
  9  PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE)
 10  );

Table created.
```

```
SQL> INSERT INTO supplier_parts_1(supplier_id, partnum, quantity)
  2  VALUES (5, 5, 1000);

1 row created.

SQL> INSERT INTO supplier_parts_1(supplier_id, partnum, quantity)
  2  VALUES (5, 150, 1000);

1 row created.

SQL> INSERT INTO supplier_parts_1(supplier_id, partnum, quantity)
  2  VALUES (10, 100, 1000);

1 row created.
```

Q6. Create interval partitioned table as directed:Creates a table named- Sales consisting of four partitions, one for each quarter of sales. Each partition is given a name (sales_q1, sales_q2, ...) The columns for table must be prod_id, cust_id, promo_id, quantify sold, amount_sold – all in number format and time_id in date format
Perform internal partitioning on time_id and take interval of 01 months.

```
SQL> CREATE TABLE Sales3_1 (
  2  prod_id NUMBER,
  3  cust_id NUMBER,
  4  promo_id NUMBER,
  5  quantity_sold NUMBER,
  6  amount_sold NUMBER,
  7  time_id DATE
  8  )
  9  PARTITION BY RANGE (time_id)
 10  INTERVAL (NUMTOYMINTERVAL(1, 'MONTH'))
 11  (
 12  PARTITION sales_q1 VALUES LESS THAN (TO_DATE('01-APR-2022', 'DD-MON-YYYY')),
 13  PARTITION sales_q2 VALUES LESS THAN (TO_DATE('01-JUL-2022', 'DD-MON-YYYY')),
 14  PARTITION sales_q3 VALUES LESS THAN (TO_DATE('01-OCT-2022', 'DD-MON-YYYY')),
 15  PARTITION sales_q4 VALUES LESS THAN (TO_DATE('01-JAN-2023', 'DD-MON-YYYY'))
 16  );

Table created.
```

Q7. Demonstrate reference partitioning as directed:
➢ Create parent table Orders with the attributes order_id, order_date, customer_id, shipper_id.
➢ Perform Range partitioning on Order Date. Take Range of 03 Months i.e. 01 quarter

```
SQL> CREATE TABLE Orders_1 (
  2      order_id INT PRIMARY KEY,
  3      order_date DATE,
  4      customer_id INT,
  5      shipper_id INT
  6  )
  7  PARTITION BY RANGE (order_date) (
  8      PARTITION q1 VALUES LESS THAN (TO_DATE('2023-04-01', 'YYYY-MM-DD')),
  9      PARTITION q2 VALUES LESS THAN (TO_DATE('2023-07-01', 'YYYY-MM-DD')),
 10      PARTITION q3 VALUES LESS THAN (TO_DATE('2023-10-01', 'YYYY-MM-DD')),
 11      PARTITION q4 VALUES LESS THAN (MAXVALUE)
 12  );

Table created.
```

```
SQL> ALTER TABLE Orders_1 DROP PARTITION q4;

Table altered.
```

Q8. Implement virtual column based partitioning as below:

➢ Create table employee with attributes Emp_id, emp_name, fixed_salary, variable_salary. Generate Total salary as virtual column.

➢ Perform range partitioning on Total Salary with four partitions as below:

○ Partition P1 stores salary less than 25000

○ Partition P2 stores salary less than 50000

○ Partition P3 stores salary less than 75000

○ Partition P4 stores any salary above and equal to than 75000

```
SQL> CREATE TABLE employee_1 (
  2      emp_id INT PRIMARY KEY,
  3      emp_name VARCHAR(100),
  4      fixed_salary DECIMAL(10, 2),
  5      variable_salary DECIMAL(10, 2),
  6      total_salary DECIMAL(10, 2),  -- physical column
  7      CONSTRAINT salary_check CHECK (total_salary = fixed_salary + variable_salary)
  8  )
  9  PARTITION BY RANGE (total_salary) (
 10      PARTITION p1 VALUES LESS THAN (25000),
 11      PARTITION p2 VALUES LESS THAN (50000),
 12      PARTITION p3 VALUES LESS THAN (75000),
 13      PARTITION p4 VALUES LESS THAN (MAXVALUE)
 14  );

Table created.
```

```
SQL> INSERT INTO employee_1 (emp_id, emp_name, fixed_salary, variable_salary)
  2  VALUES (1, 'John Doe', 20000, 5000);

1 row created.

SQL> INSERT INTO employee_1 (emp_id, emp_name, fixed_salary, variable_salary)
  2  VALUES (2, 'Jane Smith', 35000, 12000);

1 row created.

SQL> INSERT INTO employee_1 (emp_id, emp_name, fixed_salary, variable_salary)
  2  VALUES (3, 'Michael Johnson', 60000, 15000);

1 row created.

SQL> INSERT INTO employee_1 (emp_id, emp_name, fixed_salary, variable_salary)
  2  VALUES (4, 'Emily Davis', 80000, 20000);

1 row created.
```

```
SQL> SELECT partition_name, high_value
  2  FROM user_tab_partitions
  3  WHERE table_name = 'EMPLOYEE_1';

PARTITION_NAME
------------------------------
HIGH_VALUE
------------------------------------------
P1
25000

P2
50000

P3
75000


PARTITION_NAME
------------------------------
HIGH_VALUE
------------------------------------------
P4
MAXVALUE
```

Q9. Demonstrate Composite partitioning technique as directed. Implement range list partitioning for customer table having attributes cust_id, cust_name, cust_state, and time_id
Perform range partitioning on time-id and list partitioning on state attributes.
Also create maxvalue and default partition for range and list partition
respectively.
Partition definitions for range are as below:

Partition old should accept values less than 01-Jan-2005
Partition acquired should accept values less than 01-Jan-2010
Partition recent should accept values less than 01-Jan-2015
Partition unknown should accept values greater than 01-Jan-2015
Partition definitions for list are as below:
Partition west should accept values (‘MH’, ‘GJ’)
Partition south should accept values (‘TN’, ‘AP’)
Partition north should accept values (‘UP’, ‘HP’)
Partition unknown should accept any other state.

```
SQL> CREATE TABLE customer (
  2      cust_id INT,
  3      cust_name VARCHAR(100),
  4      cust_state VARCHAR(2),
  5      time_id DATE
  6  )
  7  PARTITION BY RANGE (time_id)
  8  SUBPARTITION BY LIST (cust_state)
  9  (
 10      PARTITION old VALUES LESS THAN (TO_DATE('2005-01-01', 'YYYY-MM-DD'))
 11      (
 12          SUBPARTITION west_1 VALUES ('MH', 'GJ'),
 13          SUBPARTITION south_1 VALUES ('TN', 'AP'),
 14          SUBPARTITION north_1 VALUES ('UP', 'HP')
 15      ),
 16      PARTITION acquired VALUES LESS THAN (TO_DATE('2010-01-01', 'YYYY-MM-DD'))
 17      (
 18          SUBPARTITION west_2 VALUES ('MH', 'GJ'),
 19          SUBPARTITION south_2 VALUES ('TN', 'AP'),
 20          SUBPARTITION north_2 VALUES ('UP', 'HP')
 21      ),
 22      PARTITION recent VALUES LESS THAN (TO_DATE('2015-01-01', 'YYYY-MM-DD'))
 23      (
 24          SUBPARTITION west_3 VALUES ('MH', 'GJ'),
 25          SUBPARTITION south_3 VALUES ('TN', 'AP'),
 26          SUBPARTITION north_3 VALUES ('UP', 'HP')
 27      ),
 28      PARTITION unknown VALUES LESS THAN (MAXVALUE)
 29      (
 30          SUBPARTITION west_4 VALUES ('MH', 'GJ'),
 31          SUBPARTITION south_4 VALUES ('TN', 'AP'),
 32          SUBPARTITION north_4 VALUES ('UP', 'HP'),
 33          SUBPARTITION unknown_4 VALUES (DEFAULT)
 34      )
 35  );

Table created.
```