B2-25 Practical N0 : 03

. Write a query to create range portioned table:

- Creates a table named- Sales consisting of four partitions, one for each quarter of sales. The columns sale_year, sale_month, and sale_day 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, quantify sold, amount_sold – all in number format and time_id.

Result:

```
SQL> CREATE TABLESPACE tsa DATAFILE 'G:/DBA/tsa.dbf' SIZE 10M;
Tablespace created.

SQL> CREATE TABLESPACE tsb DATAFILE 'G:/DBA/tsb.dbf' SIZE 10M;
Tablespace created.

SQL> CREATE TABLESPACE tsc DATAFILE 'G:/DBA/tsc.dbf' SIZE 10M;
Tablespace created.

SQL> CREATE TABLESPACE tsd DATAFILE 'G:/DBA/tsd.dbf' SIZE 10M;
Tablespace created.



/* QUERY 01*/
SQL> create table sales(
  2  prod_id number(5),
  3  cust_id number(5),
  4  promo_id number(5),
  5  quantity_sold number(6),
  6  amount_sold number(5),
  7  time_id DATE)
  8  partition by range(time_id)
  9  (partition tsa_q1 values less than(TO_DATE('01-APR-2018','dd-MON-
yyyy'))
 10  TABLESPACE tsa,
 11  partition tsa_q2 values less than(TO_DATE('01-JUL-2018','dd-MON-
yyyy'))
 12  TABLESPACE tsb,
 13  partition tsa_q3 values less than(TO_DATE('01-SEP-2018','dd-MON-
yyyy'))
 14  TABLESPACE tsc,
 15  partition tsa_q4 values less than(TO_DATE('01-JAN-2019','dd-MON-
yyyy'))
 16  TABLESPACE tsa
 17  );
Table created.

SQL> INSERT INTO sales VALUES(101,201,301,4500,8500,'05-FEB-2018');
1 row created.

SQL> INSERT INTO sales VALUES(102,202,302,4740,9650,'09-MAY-2018');
1 row created.
```

```
SQL> INSERT INTO sales VALUES(103,203,303,4512,8547,'08-AUG-2018');
1 row created.

SQL> INSERT INTO sales VALUES(104,204,304,4500,8500,'12-NOV-2018');
1 row created.

SQL> INSERT INTO sales VALUES(105,205,305,4500,8500,'06-JAN-2019');
INSERT INTO sales VALUES(105,205,305,4500,8500,'06-JAN-2019')
            *
ERROR at line 1:
ORA-14400: inserted partition key does not map to any partition

select partition_name,high_value from user_tab_partitions where
table_name='SALES';
PARTITION_NAME           HIGH_VALUE
--------------  ------------------------------------------------------------
-
TSA_Q1          TO_DATE(' 2018-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIA
TSA_Q2          TO_DATE(' 2018-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIA
TSA_Q3          TO_DATE(' 2018-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIA
TSA_Q4          TO_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIA
```

Q2. Create the same table as in Q1. With a different name with ENABLE ROW MOVEMENT

```
SQL> create table sales_ERM(
  2  prod_id number(5),
  3  cust_id number(5),
  4  promo_id number(5),
  5  quantity_sold number(6),
  6  amount_sold number(5),
  7  time_id DATE)
  8  partition by range(time_id)
  9  (partition tsa_q11 values less than(TO_DATE('01-APR-2018','dd-MON-
yyyy'))
 10  TABLESPACE tsa,
 11  partition tsa_q21 values less than(TO_DATE('01-JUL-2018','dd-MON-
yyyy'))
 12  TABLESPACE tsb,
 13  partition tsa_q31 values less than(TO_DATE('01-SEP-2018','dd-MON-
yyyy'))
 14  TABLESPACE tsc,
 15  partition tsa_q41 values less than(TO_DATE('01-JAN-2019','dd-MON-
yyyy'))
 16  TABLESPACE tsa
 17  )ENABLE ROW MOVEMENT;
Table created.

SQL> INSERT INTO sales_ERM VALUES(101,201,301,4500,8500,'05-FEB-2018');
1 row created.

SQL> INSERT INTO sales_ERM VALUES(102,202,302,4740,9650,'09-MAY-2018');
1 row created.
```

```
SQL> INSERT INTO sales_ERM VALUES(103,203,303,4512,8547,'08-AUG-2018');
1 row created.

SQL> INSERT INTO sales_ERM VALUES(104,204,304,4500,8500,'12-NOV-2018');
1 row created.

SQL> SELECT * FROM sales_ERM partition(tsa_q11);
   PROD_ID    CUST_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD TIME_ID
---------- ---------- ---------- ------------- ----------- ---------
       101        201        301          4500        8500 05-FEB-18

SQL> SELECT * FROM sales_ERM partition(tsa_q21);
   PROD_ID    CUST_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD TIME_ID
---------- ---------- ---------- ------------- ----------- ---------
       102        202        302          4740        9650 09-MAY-18

SQL> SELECT * FROM sales_ERM partition(tsa_q31);
   PROD_ID    CUST_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD TIME_ID
---------- ---------- ---------- ------------- ----------- ---------
       103        203        303          4512        8547 08-AUG-18

SQL> SELECT * FROM sales_ERM partition(tsa_q41);
   PROD_ID    CUST_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD TIME_ID
---------- ---------- ---------- ------------- ----------- ---------
       104        204        304          4500        8500 12-NOV-18

UPDATE SALES_ERM SET time_id = TO_DATE('05-JAN-2018','dd-MON-yyyy') WHERE
prod_id='104';
SQL> SELECT * FROM sales_ERM partition(tsa_q11);

   PROD_ID    CUST_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD TIME_ID
---------- ---------- ---------- ------------- ----------- ---------
       101        201        301          4500        8500 05-FEB-18
       104        204        304          4500        8500 05-JAN-18

SQL> SELECT * FROM sales_ERM partition(tsa_q41);
no rows selected

SQL> EXEC dbms_stats.gather_table_stats('ss','SALES_ERM');
PL/SQL procedure successfully completed.

SQL> SELECT PARTITION_NAME,NUM_ROWS from user_tab_partitions;
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
TSA_Q1
TSA_Q2
TSA_Q3
TSA_Q4
TSA_Q11                                 2
TSA_Q21                                 1
TSA_Q31                                 1
TSA_Q41                                 0
8 rows selected.
```

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 sales_state
  2  (deptno number,
  3   deptname varchar2(20),
  4   quarterly_sales number(10, 2),
  5   state varchar2(2))
  6   PARTITION BY LIST (state)
  7  (PARTITION northwest VALUES ('OR', 'WA') TABLESPACE tsa,
  8   PARTITION southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE tsb,
  9   PARTITION northeast VALUES  ('NY', 'VM', 'NJ') TABLESPACE tsc,
 10   PARTITION southeast VALUES ('FL', 'GA') TABLESPACE tsd,
 11   PARTITION northcentral VALUES ('SD', 'WI') TABLESPACE tsa,
 12   PARTITION southcentral VALUES ('OK', 'TX') TABLESPACE tsa);
Table created.

SQL> INSERT INTO sales_state VALUES(10, 'accounting', 100, 'WA');
1 row created.

SQL> INSERT INTO sales_state VALUES(20, 'RANDD', 150, 'OR');
1 row created.

SQL> INSERT INTO sales_state VALUES(30, 'sales', 100, 'FL');
1 row created.

SQL> INSERT INTO sales_state VALUES(40, 'HR', 10, 'TX');
1 row created.

SQL> INSERT INTO sales_state VALUES(50, 'systems engineering', 10, 'CA');
INSERT INTO sales_state VALUES(50, 'systems engineering', 10, 'CA')
            *
ERROR at line 1:
ORA-14400: inserted partition key does not map to any partition

SQL> SELECT * FROM sales_state partition(northwest);
    DEPTNO DEPTNAME             QUARTERLY_SALES ST
---------- -------------------- --------------- --
        10 accounting                      100 WA
        20 RANDD                           150 OR

SQL> SELECT * FROM sales_state partition(southwest);
no rows selected

SQL> SELECT * FROM sales_state partition(northeast);
no rows selected
```

```
SQL> SELECT * FROM sales_state partition(southeast);
    DEPTNO DEPTNAME             QUARTERLY_SALES ST
---------- -------------------- --------------- --
        30 sales                            100 FL

SQL> SELECT * FROM sales_state partition(northcentral);
no rows selected

SQL> SELECT * FROM sales_state partition(southcentral);
    DEPTNO DEPTNAME             QUARTERLY_SALES ST
---------- -------------------- --------------- --
        40 HR                                10 TX

SQL> Alter table sales_state add partition new_def values(default);
Table altered.

SQL> INSERT INTO sales_state VALUES(50, 'systems engineering', 10, 'CA');
1 row created.

SQL> select partition_name,high_value from user_tab_partitions where
table_name='SALES_STATE';
PARTITION_NAME                 HIGH_VALUE
------------------------------ ----------------------------------------
--------------------------------------
NORTHWEST                      'OR', 'WA'
SOUTHWEST                      'AZ', 'UT', 'NM'
NORTHEAST                      'NY', 'VM', 'NJ'
SOUTHEAST                      'FL', 'GA'
NORTHCENTRAL                   'SD', 'WI'
SOUTHCENTRAL                   'OK', 'TX'
NEW_DEF                        default
7 rows selected.
```

Q4. Create a table with hash partition as follows:

• Create table Emp with attributes empno, job, sal, deptno and perform hash partitioning on empno.

Number of Partitions should be Demonstarte using system defined and user defined partition concepts.

```
SQL> Create table Emp_usr
  2  (emp_no number(2),
  3  job varchar(8),
  4  sal number(6),
  5  deptno number(3))
  6  partition by hash(emp_no)
  7  partitions 5;
Table created.

SQL> INSERT INTO Emp_Usr VALUES(01,'CA',12500,3);
1 row created.

SQL> INSERT INTO Emp_Usr VALUES(12,'AB',12500,7);
1 row created.

SQL> INSERT INTO Emp_Usr VALUES(28,'CV',12500,4);
1 row created.

SQL> INSERT INTO Emp_Usr VALUES(99,'CM',12500,2);
```

```
1 row created.

SQL> select * from emp_usr;
    EMP_NO JOB             SAL     DEPTNO
---------- -------- ---------- ----------
        12 AB            12500          7
        28 CV            12500          4
        99 CM            12500          2
         1 CA            12500          3

SQL> EXEC dbms_stats.gather_table_stats('ss','EMP_USR');
PL/SQL procedure successfully completed.

SQL> select partition_name,num_rows from user_tab_partitions where
table_name='EMP_USR';
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
SYS_P26                                 0
SYS_P27                                 3
SYS_P28                                 0
SYS_P29                                 1
SYS_P30                                 0

SQL> Create table Emp_usr1
  2  (emp_no number(2),
  3  job varchar(8),
  4  sal number(6),
  5  deptno number(3))
  6  partition by hash(emp_no)
  7  (partition p1,
  8  partition p2,
  9  partition p3,
 10  partition p4);
Table created.

SQL> INSERT INTO Emp_Usr1 VALUES(01,'CA',12500,3);
1 row created.

SQL> INSERT INTO Emp_Usr1 VALUES(12,'AB',12500,7);
1 row created.

SQL> INSERT INTO Emp_Usr1 VALUES(28,'CV',12500,4);
1 row created.

SQL> INSERT INTO Emp_Usr1 VALUES(99,'CM',12500,2);
1 row created.

SQL> select * from emp_usr1;
    EMP_NO JOB             SAL     DEPTNO
---------- -------- ---------- ----------
        12 AB            12500          7
        28 CV            12500          4
        99 CM            12500          2
         1 CA            12500          3

SQL> EXEC dbms_stats.gather_table_stats('ss','EMP_USR1');
PL/SQL procedure successfully completed.
```

```
SQL> select partition_name,num_rows from user_tab_partitions where
table_name='EMP_USR1';
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
P1                                      0
P2                                      3
P3                                      0
P4                                      1
```

Q5. 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
  - Les 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);

  Make conclusion for each result.

```
SQL> CREATE TABLE DATE_INFO(
  2  year number(4),
  3  month number(2),
  4  day number(2),
  5  amount_sold number(6))
  6  partition by range(year,month)
  7  (partition v11 values less than(2001,1),
  8  partition v12 values less than(2001,4),
  9  partition v13 values less than(2001,7),
 10  partition v14 values less than(2001,10),
 11  partition v15 values less than(2002,1),
 12  partition v16 values less than(maxvalue,maxvalue));
Table created.

SQL> INSERT INTO DATE_INFO VALUES(2001,3,17, 2000);
1 row created.

SQL> INSERT INTO DATE_INFO VALUES(2001,11,1, 5000);
1 row created.

SQL> INSERT INTO DATE_INFO VALUES(2002,1,1, 4000);
1 row created.

SQL> SELECT * FROM DATE_INFO PARTITION(V12);
      YEAR      MONTH        DAY AMOUNT_SOLD
---------- ---------- ---------- -----------
      2001          3         17        2000

SQL> SELECT * FROM DATE_INFO PARTITION(V15);
      YEAR      MONTH        DAY AMOUNT_SOLD
---------- ---------- ---------- -----------
      2001         11          1        5000
```

```
SQL> SELECT * FROM DATE_INFO PARTITION(V16);
      YEAR      MONTH        DAY AMOUNT_SOLD
---------- ---------- ---------- -----------
      2002          1          1        4000

SQL> EXEC dbms_stats.gather_table_stats('ss','DATE_INFO');
PL/SQL procedure successfully completed.

SQL> select partition_name,num_rows from user_tab_partitions where
table_name='DATE_INFO';
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
V11                                     0
V12                                     1
V13                                     0
V14                                     0
V15                                     1
V16                                     1
6 rows selected.
```

Q6. 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(
  2  SUPPLIER_ID NUMBER(2),
  3  PART_NUM NUMBER(4),
  4  AMOUNT_SOLD NUMBER(4)
  5  )
  6  PARTITION BY RANGE(SUPPLIER_ID, PART_NUM)
  7  (
  8  PARTITION UPTO_50 VALUES LESS THAN (5, 50),
  9  PARTITION UPTO_100 VALUES LESS THAN (5, 100),
 10  PARTITION UPTO_150 VALUES LESS THAN (5, 150),
 11  PARTITION UPTO_200 VALUES LESS THAN (5, 200),
 12  PARTITION MAX_VAL VALUES LESS THAN (MAXVALUE, MAXVALUE)
 13  );
Table created.

SQL> INSERT INTO SUPPLIER_PARTS VALUES (5,5, 1000);
1 row created.

SQL> INSERT INTO SUPPLIER_PARTS VALUES (5,150, 1000);
1 row created.

SQL> INSERT INTO SUPPLIER_PARTS VALUES (10,100, 1000);
1 row created.
```

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('ss', 'SUPPLIER_PARTS');
PL/SQL procedure successfully completed.

SQL> SELECT TABLE_NAME, PARTITION_NAME, NUM_ROWS FROM USER_TAB_PARTITIONS
WHERE TABLE_NAME = 'SUPPLIER_PARTS';
TABLE_NAME                     PARTITION_NAME                   NUM_ROWS
------------------------------ ------------------------------ ----------
SUPPLIER_PARTS                 UPTO_50                                 1
SUPPLIER_PARTS                 UPTO_100                                0
SUPPLIER_PARTS                 UPTO_150                                0
SUPPLIER_PARTS                 UPTO_200                                1
SUPPLIER_PARTS                 MAX_VAL                                 1

SQL> SELECT * FROM SUPPLIER_PARTS PARTITION(UPTO_50);
SUPPLIER_ID   PART_NUM AMOUNT_SOLD
----------- ---------- -----------
          5          5        1000

SQL> SELECT * FROM SUPPLIER_PARTS PARTITION(UPTO_100);
no rows selected

SQL> SELECT * FROM SUPPLIER_PARTS PARTITION(UPTO_150);
no rows selected

SQL> SELECT * FROM SUPPLIER_PARTS PARTITION(UPTO_200);
SUPPLIER_ID   PART_NUM AMOUNT_SOLD
----------- ---------- -----------
          5        150        1000

SQL> SELECT * FROM SUPPLIER_PARTS PARTITION(MAX_VAL);
SUPPLIER_ID   PART_NUM AMOUNT_SOLD
----------- ---------- -----------
         10        100        1000
```

Q7. 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 month in number format
- Perform interval partitioning on month and take interval of 01 months.

```
SQL> CREATE TABLE SALES_INT(
  2  PROMO_ID VARCHAR2(5),
  3  YEAR NUMBER(4),
  4  MONTH NUMBER(2),
  5  DAY NUMBER(2),
  6  QUANTITY_SOLD NUMBER(3),
  7  AMOUNT_SOLD NUMBER(5)
  8  )
  9  PARTITION BY RANGE(MONTH)
 10  INTERVAL(01)
 11  (
 12  PARTITION P1 VALUES LESS THAN(03),
 13  PARTITION P2 VALUES LESS THAN(06)
```

```
 14  );
Table created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2000, 4, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2001, 5, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2000, 6, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2000, 7, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2000, 8, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM102', 2000, 9, 27, 543, 888);
1 row created.

SQL> INSERT INTO SALES_INT VALUES('PM103', 2000, 7, 26, 543, 888);
1 row created.

SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('ss', 'SALES_INT');
PL/SQL procedure successfully completed.

SQL> SELECT TABLE_NAME, PARTITION_NAME, HIGH_VALUE, NUM_ROWS FROM
USER_TAB_PARTITIONS WHERE TABLE_NAME = 'SALES_INT';
TABLE_NAME                     PARTITION_NAME                 HIGH_VALUE
NUM_ROWS
------------------------------ ------------------------------ -----------
---------------------------------------------------------------------- ---
-------
SALES_INT                      P1                             03
0
SALES_INT                      P2                             06
2
SALES_INT                      SYS_P31                        7
1
SALES_INT                      SYS_P32                        8
2
SALES_INT                      SYS_P33                        9
1
SALES_INT                      SYS_P34                        10
1
6 rows selected.
```

Q8. 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
- Create child table order_items with attributes order_id, product_id, price and quantity.
- Perform Reference partitioning on child table.
- Delete the created partitions.

```
SQL> CREATE TABLE Orders(
  2  order_id number(4) not null,
  3  order_date date,
  4  cust_id number(4),
  5  shipper_id number(4),
  6  constraint order_id_pk primary key(order_id)
  7  )
  8  partition by range(order_date)
  9  (partition tsa_q1 values less than(TO_DATE('01-APR-2018','dd-MON-
yyyy')),
 10   partition tsa_q2 values less than(TO_DATE('01-JUL-2018','dd-MON-
yyyy')),
 11   partition tsa_q3 values less than(TO_DATE('01-SEP-2018','dd-MON-
yyyy')),
 12   partition tsa_q4 values less than(TO_DATE('01-JAN-2019','dd-MON-
yyyy'))
 13  );
Table created.

SQL> insert into orders values(101,'01-JAN-2018',201,301);
1 row created.

SQL> CREATE TABLE orders_items(
  2  order_id number(4) not null,
  3  product_id number(4) not null,
  4  quantity number(4),
  5  constraint order_id_fk Foreign Key(order_id) REFERENCES
Orders(order_id)
  6  )
  7  PARTITION BY REFERENCE (order_id_fk);
Table created.

SQL> alter table orders drop partition tsa_q1;
Table altered.

SQL> select * from orders_items PARTITION (tsa_q1);
select * from orders_items PARTITION (tsa_q1)
                                      *
ERROR at line 1:
ORA-02149: Specified partition does not exist
```

Q9. 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 colum.
- 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 (
  2    Emp_id NUMBER,
  3    emp_name VARCHAR2(20),
  4    fixed_sal NUMBER(6),
  5    variable_sal NUMBER(6),
  6    total_sal number(6)
```

```
  7    GENERATED ALWAYS AS
  8    (
  9           fixed_sal + variable_sal
 10    ) VIRTUAL
 11  )
 12  PARTITION BY RANGE (total_sal)
 13  (
 14    PARTITION p1 VALUES less than(25000),
 15    PARTITION p2 VALUES less than(50000),
 16    PARTITION p3 VALUES less than(75000),
 17    PARTITION p4 VALUES less than(maxvalue)
 18  );
Table created.

SQL> INSERT INTO employee (emp_id,emp_name,fixed_sal,variable_sal) VALUES
(101,'Andy Pandy',20000,12300);
1 row created.

SQL> INSERT INTO employee (emp_id,emp_name,fixed_sal,variable_sal) VALUES
(102,'Andy',30000,12300);
1 row created.

SQL> INSERT INTO employee (emp_id,emp_name,fixed_sal,variable_sal) VALUES
(103,'Pandy',20000,50000);
1 row created.

SQL> INSERT INTO employee (emp_id,emp_name,fixed_sal,variable_sal) VALUES
(104,'Ana',25000,12300);
1 row created.

SQL> INSERT INTO employee (emp_id,emp_name,fixed_sal,variable_sal) VALUES
(105,'ramesh',75000,12300);
1 row created.
SQL> commit;
Commit complete.

SQL> select * from employee PARTITION (p1);
no rows selected

SQL> select * from employee PARTITION (p2);
    EMP_ID EMP_NAME              FIXED_SAL VARIABLE_SAL  TOTAL_SAL
---------- -------------------- ---------- ------------ ----------
       101 Andy Pandy                20000        12300      32300
       102 Andy                      30000        12300      42300
       104 Ana                       25000        12300      37300

SQL> select * from employee PARTITION (p3);
    EMP_ID EMP_NAME              FIXED_SAL VARIABLE_SAL  TOTAL_SAL
---------- -------------------- ---------- ------------ ----------
       103 Pandy                     20000        50000      70000

SQL> select * from employee PARTITION (p4);
    EMP_ID EMP_NAME              FIXED_SAL VARIABLE_SAL  TOTAL_SAL
---------- -------------------- ---------- ------------ ----------
       105 ramesh                    75000        12300      87300

SQL> EXEC dbms_stats.gather_table_stats('ss','employee');
PL/SQL procedure successfully completed.
```

```
SQL> select partition_name,num_rows from user_tab_partitions where
table_name='EMPLOYEE';
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
P1                                      0
P2                                      3
P3                                      1
P4                                      1
```

Q10. 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 CUSTOMERS(
  2  cust_id NUMBER(3) PRIMARY KEY,
  3  cust_name VARCHAR(10),
  4  cust_state VARCHAR(4),
  5  time_id DATE )
  6  PARTITION BY RANGE(time_id)
  7  SUBPARTITION BY LIST(cust_state)
  8  SUBPARTITION TEMPLATE(
  9  SUBPARTITION west VALUES ('MH','GJ'),
 10  SUBPARTITION south VALUES ('TN','AP'),
 11  SUBPARTITION north VALUES ('UP','HP'),
 12  SUBPARTITION other VALUES (default))
 13  (
 14  PARTITION P11 VALUES LESS THAN(TO_DATE('01-JAN-2005','DD-MON-
YYYY')),
 15  PARTITION P12 VALUES LESS THAN(TO_DATE('01-JAN-2010','DD-MON-
YYYY')),
 16  PARTITION P13 VALUES LESS THAN(TO_DATE('01-JAN-2015','DD-MON-
YYYY')),
 17  PARTITION P14 VALUES LESS THAN(MAXVALUE)
 18  );
Table created.

SQL> INSERT INTO CUSTOMERS VALUES(101,'Rakesh','MH','01-JAN-2009');
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(102,'Ramesh','GJ','01-JAN-2011');
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(103,'Ekta','MP','01-JAN-2015');
```

```
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(104,'Swati','AP','01-JAN-2002');
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(105,'Fatema','UP','01-JAN-2019');
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(106,'Anika','TN','01-JAN-2006');
1 row created.

SQL> INSERT INTO CUSTOMERS VALUES(107,'Toshika','HP','01-JAN-2001');
1 row created.

SQL> select * from customers;
   CUST_ID CUST_NAME  CUST TIME_ID
---------- ---------- ---- ---------
       104 Swati      AP   01-JAN-02
       107 Toshika    HP   01-JAN-01
       101 Rakesh     MH   01-JAN-09
       106 Anika      TN   01-JAN-06
       102 Ramesh     GJ   01-JAN-11
       105 Fatema     UP   01-JAN-19
       103 Ekta       MP   01-JAN-15
7 rows selected.

SQL> select * from customers partition(p12);
   CUST_ID CUST_NAME  CUST TIME_ID
---------- ---------- ---- ---------
       101 Rakesh     MH   01-JAN-09
       106 Anika      TN   01-JAN-06

SQL> select * from customers partition(p13);
   CUST_ID CUST_NAME  CUST TIME_ID
---------- ---------- ---- ---------
       102 Ramesh     GJ   01-JAN-11

SQL> select * from customers partition(p14);
   CUST_ID CUST_NAME  CUST TIME_ID
---------- ---------- ---- ---------
       105 Fatema     UP   01-JAN-19
       103 Ekta       MP   01-JAN-15

SQL> exec dbms_stats.gather_table_stats('ss','CUSTOMERS');
PL/SQL procedure successfully completed.

SQL> select partition_name,num_rows from user_tab_partitions where
table_name='CUSTOMERS';
PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
P11                                     2
P12                                     2
P13                                     1
P14                                     2

SQL> select * from subpartition(p11_north);
select * from subpartition(p11_north)
                           *
```

```
ERROR at line 1:
ORA-00933: SQL command not properly ended

SQL> SELECT partition_name, subpartition_name, num_rows
  2  FROM user_tab_subpartitions where table_name='CUSTOMERS';
PARTITION_NAME                 SUBPARTITION_NAME                NUM_ROWS
------------------------------ ------------------------------ ----------
P11                            P11_WEST                                0
P11                            P11_SOUTH                               1
P11                            P11_NORTH                               1
P11                            P11_OTHER                               0
P12                            P12_WEST                                1
P12                            P12_SOUTH                               1
P12                            P12_NORTH                               0
P12                            P12_OTHER                               0
P13                            P13_WEST                                1
P13                            P13_SOUTH                               0
P13                            P13_NORTH                               0

PARTITION_NAME                 SUBPARTITION_NAME                NUM_ROWS
------------------------------ ------------------------------ ----------
P13                            P13_OTHER                               0
P14                            P14_WEST                                0
P14                            P14_SOUTH                               0
P14                            P14_NORTH                               1
P14                            P14_OTHER                               1

16 rows selected.




SQL> CREATE TABLE composite_rng_hash(
    2 cust_id NUMBER(10),
    3 cust_name VARCHAR2(25),
    4 cust_state VARCHAR2(2),
    5 amt_sold VARCHAR2(2),
    6 time_id DATE)
    7 PARTITION BY RANGE(time_id)
    8 SUBPARTITION BY HASH(cust_id)
    9 SUBPARTITION TEMPLATE(
    10 SUBPARTITION h1,
    11 SUBPARTITION h2,
    12 SUBPARTITION h3)
    13 (PARTITION YEAR_2006 VALUES LESS THAN (TO_DATE('01-APR- 2006','DD-
MON-YYYY')),
    14 PARTITION YEAR_2007 VALUES LESS THAN(TO_DATE('01-APR-2007','DD-
MON-YYYY')),
    15 PARTITION YEAR_2008 VALUES LESS THAN(TO_DATE('01-APR-2008','DD-
MON-YYYY'))
    16 );

    Table created.

    SQL> DESC composite_rng_hash;
    Name Null? Type
----------------------------------------- -------- ----------------------
```

```
    CUST_ID NUMBER(10)
    CUST_NAME VARCHAR2(25)
    CUST_STATE VARCHAR2(2)
    AMT_SOLD VARCHAR2(2)
    TIME_ID DATE


    SQL> insert into composite_rng_hash values(11,'cse','lp',21,'11-feb-
2008');

    1 row created.

    SQL> SELECT partition_name, subpartition_name, num_rows
    2 FROM user_tab_subpartitions where table_name='COMPOSITE_RNG_HASH';

    PARTITION_NAME SUBPARTITION_NAME NUM_ROWS
    ------------------------------ ------------------------------ -------
--
    YEAR_2006 YEAR_2006_H1
    YEAR_2006 YEAR_2006_H2
    YEAR_2006 YEAR_2006_H3
    YEAR_2007 YEAR_2007_H1
    YEAR_2007 YEAR_2007_H2
    YEAR_2007 YEAR_2007_H3
    YEAR_2008 YEAR_2008_H1
    YEAR_2008 YEAR_2008_H2
    YEAR_2008 YEAR_2008_H3

    9 rows selected.

    select * from composite_rng_hash subpartition(YEAR_2008_h1);
    CUST_ID CUST_NAME CU AM TIME_ID
    ---------- ------------------------- -- -- --------
    11 cse lp 21 11-FEB-08

    SQL> select * from composite_rng_hash subpartition(YEAR_2008_h2);
    no rows selected

    SQL> select * from composite_rng_hash subpartition(YEAR_2008_h3); 598
    no rows selected


    12)RANGE-RANGE
    CREATE TABLE composite_rng_rng (
    cust_id NUMBER(10),
    cust_name VARCHAR2(25),
    cust_state VARCHAR2(2),
    amt_sold VARCHAR2(2),
    time_id DATE)
    PARTITION BY RANGE(time_id)
    SUBPARTITION BY RANGE (cust_id)
    SUBPARTITION TEMPLATE(
    SUBPARTITION original VALUES LESS THAN (1001),
    SUBPARTITION acquired VALUES LESS THAN (8001),
    SUBPARTITION recent VALUES LESS THAN (MAXVALUE))
    (PARTITION YEAR_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','DD-MON-
YYYY')),
```

```
    PARTITION YEAR_2007 VALUES LESS THAN(TO_DATE('01-APR-2007','DD-MON-
YYYY')),
    PARTITION YEAR_2008 VALUES LESS THAN(TO_DATE('01-APR-2008','DD-MON-
YYYY'))
    );
    SQL> desc composite_rng_rng;
    Name Null? Type
-----------------------------------------------------------------------
    CUST_ID NUMBER(10)
    CUST_NAME VARCHAR2(25)
    CUST_STATE VARCHAR2(2)
    AMT_SOLD VARCHAR2(2)
    TIME_ID DATE


    SQL> insert into composite_rng_rng values(11,'cse','OR',21,'11-feb-
2007');

    1 row created.

    SQL> insert into composite_rng_rng values(11,'cse','OR',21,'11-feb-
2008');

    1 row created.

    SQL> SELECT partition_name, subpartition_name, num_rows
    2 FROM user_tab_subpartitions where table_name='COMPOSITE_RNG_RNG';

    PARTITION_NAME SUBPARTITION_NAME NUM_ROWS
------------------------------ ------------------------------ ---------
    YEAR_2006 YEAR_2006_ORIGINAL
    YEAR_2006 YEAR_2006_ACQUIRED
    YEAR_2006 YEAR_2006_RECENT
    YEAR_2007 YEAR_2007_ORIGINAL
    YEAR_2007 YEAR_2007_ACQUIRED
    YEAR_2007 YEAR_2007_RECENT
    YEAR_2008 YEAR_2008_ORIGINAL
    YEAR_2008 YEAR_2008_ACQUIRED
    YEAR_2008 YEAR_2008_RECENT

    9 rows selected.

    select * from composite_rng_rng subpartition(YEAR_2008_original); 654
CUST_ID CUST_NAME CU AM TIME_ID
    ---------- ------------------------- -- -- --------
    11 cse OR 21 11-FEB-08

    13)LIST-HASH
    SQL> CREATE TABLE composite_list_hash (
    2 cust_id NUMBER(10),
    3 cust_name VARCHAR2(25),
    4 cust_state VARCHAR2(2),
    5 amt_sold VARCHAR2(2),
    6 time_id DATE)
    7 PARTITION BY LIST(cust_state)
    8 SUBPARTITION BY HASH (cust_id)
    9 SUBPARTITION TEMPLATE(
    10 SUBPARTITION h1,
```

```
    11 SUBPARTITION h2,
    12 SUBPARTITION h3)
    13 (PARTITION west VALUES ('OR', 'WA'),
    14 PARTITION east VALUES ('NY', 'CT'),
    15 PARTITION cent VALUES ('IL', 'MN')
    16 );

    Table created.

    SQL> desc composite_list_hash;
    Name Null? Type
----------------------------------------- -------- ----------------------
    CUST_ID NUMBER(10)
    CUST_NAME VARCHAR2(25)
    CUST_STATE VARCHAR2(2)
    AMT_SOLD VARCHAR2(2)
    TIME_ID DATE


    SQL> SELECT partition_name, subpartition_name, num_rows
    2 FROM user_tab_subpartitions where table_name='COMPOSITE_LIST_HASH';

    PARTITION_NAME SUBPARTITION_NAME NUM_ROWS
  ------------------------------ ------------------------------ ---------
    WEST WEST_H1
    WEST WEST_H2
    WEST WEST_H3
    EAST EAST_H1
    EAST EAST_H2
    EAST EAST_H3
    CENT CENT_H1
    CENT CENT_H2
    CENT CENT_H3

    9 rows selected.
    SQL> insert into composite_list_hash values(2,'MEC','NY',22,'10-feb-
2018');

    1 row created.
    SQL> insert into composite_list_hash values(2,'CSE','IL',22,'10-JAN-
2018');

    1 row created.

    select * from composite_list_hash subpartition(west_h1);

    SQL> select * from composite_list_hash subpartition(east_h1);

    no rows selected

    SQL> select * from composite_list_hash subpartition(east_h2);

    no rows selected

    SQL> select * from composite_list_hash subpartition(east_h3);

    CUST_ID CUST_NAME CU AM TIME_ID
```

```
---------- ------------------------ -- -- --------
2 MEC NY 22 10-FEB-18

14)LIST-LIST
SQL> CREATE TABLE composite_list_list(
2 cust_id NUMBER(10),
3 cust_name VARCHAR2(25),
4 cust_state VARCHAR2(2),
5 amt_sold VARCHAR2(2),
6 time_id DATE)
7 PARTITION BY LIST(cust_state)
8 SUBPARTITION BY LIST(cust_id)
9 SUBPARTITION TEMPLATE
10 (SUBPARTITION original VALUES(1001),
11 SUBPARTITION acquired VALUES (8001),
12 SUBPARTITION recent VALUES (default))
13 (PARTITION west VALUES ('OR', 'WA'),
14 PARTITION east VALUES ('NY', 'CT'),
15 PARTITION cent VALUES ('IL', 'MN')
16 );

Table created.

SQL> desc composite_list_list;
Name Null? Type
---------------------------------------- -------- -----------------
CUST_ID NUMBER(10)
CUST_NAME VARCHAR2(25)
CUST_STATE VARCHAR2(2)
AMT_SOLD VARCHAR2(2)
TIME_ID DATE
SQL> SELECT partition_name, subpartition_name, num_rows
2 FROM user_tab_subpartitions where table_name='COMPOSITE_LIST_LIST';

PARTITION_NAME SUBPARTITION_NAME NUM_ROWS
------------------------------ ------------------------------ ---------
WEST WEST_ORIGINAL
WEST WEST_ACQUIRED
WEST WEST_RECENT
EAST EAST_ORIGINAL
EAST EAST_ACQUIRED
EAST EAST_RECENT
CENT CENT_ORIGINAL
CENT CENT_ACQUIRED
CENT CENT_RECENT

9 rows selected.
SQL> insert into composite_list_list values (21,'IND','IL',22,'10-
feb-2019');

1 row created.

SQL> insert into composite_list_list values(21,'IND','IL',32,'10-APR-
2020');
1 row created.

SQL> select * from composite_list_list subpartition(cent_recent);
```

```
CUST_ID CUST_NAME CU AM TIME_ID
---------- ------------------------- -- -- --------
21 IND IL 22 10-FEB-19
21 IND IL 32 10-APR-20


15)LIST-RANGE
SQL> CREATE TABLE composite_list_rng (
2 cust_id NUMBER(10),
3 cust_name VARCHAR2(25),
4 cust_state VARCHAR2(2),
5 amt_sold VARCHAR2(2),
6 time_id DATE)
7 PARTITION BY LIST(cust_state)
8 SUBPARTITION BY RANGE (cust_id)
9 SUBPARTITION TEMPLATE(
10 SUBPARTITION original VALUES LESS THAN (1001),
11 SUBPARTITION acquired VALUES LESS THAN (8001),
12 SUBPARTITION recent VALUES LESS THAN (MAXVALUE))
13 (PARTITION west VALUES ('OR', 'WA'),
14 PARTITION east VALUES ('NY', 'CT'),
15 PARTITION cent VALUES ('IL', 'MN')
16 );

Table created.

SQL> desc composite_list_rng;
Name Null? Type
---------------------------------------- -------- ------------------
CUST_ID NUMBER(10)
CUST_NAME VARCHAR2(25)
CUST_STATE VARCHAR2(2)
AMT_SOLD VARCHAR2(2)
TIME_ID DATE


SQL> SELECT partition_name, subpartition_name, num_rows
FROM user_tab_subpartitions where table_name='COMPOSITE_LIST_RNG';
PARTITION_NAME SUBPARTITION_NAME NUM_ROWS
------------------------------ ------------------------------ -------
CENT CENT_ORIGINAL
CENT CENT_ACQUIRED
CENT CENT_RECENT
EAST EAST_ORIGINAL
EAST EAST_ACQUIRED
EAST EAST_RECENT
WEST WEST_ORIGINAL
WEST WEST_ACQUIRED
WEST WEST_RECENT

9 rows selected.
SQL> insert into composite_list_rng values(1,'cse','OR',2,'10-feb-
2018');
1 row created.

SQL> insert into composite_list_rng values(2,'MEC','OR',22,'10-feb-
2018');
1 row created.
```

```
SQL> select * from composite_list_rng partition(west);
CUST_ID CUST_NAME CU AM TIME_ID
---------- ------------------------ -- -- --------
1 cse OR 2 10-FEB-18
2 MEC OR 22 10-FEB-18

SQL> select * from composite_list_rng subpartition(west_original);
CUST_ID CUST_NAME CU AM TIME_ID
---------- ------------------------ -- -- --------
1 cse OR 2 10-FEB-18
2 MEC OR 22 10-FEB-18
```