\*\*\* EXPERIMENT NO: 03 \*\*\*

-------------------------------------------------------------------------------

Author : Pradyot Patil

Roll No : 53 [7B]

Date : 31-July-2018

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Query 1: 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLESPACE tsa DATAFILE '/temp/abc\_tbs.dbf' SIZE 10M;**

Tablespace created.

**CREATE TABLESPACE tsb DATAFILE '/temp/tsb\_tbs.dbf' SIZE 10M;**

Tablespace created.

**CREATE TABLESPACE tsc DATAFILE '/temp/tsc\_tbs.dbf' SIZE 10M;**

Tablespace created.

**CREATE TABLESPACE tsd DATAFILE '/temp/tsd\_tbs.dbf' SIZE 10M;**

Tablespace created.

**create table sales (**

**prod\_id number primary key,**

**time\_id date not null,**

**cust\_id number not null,**

**promo\_id number not null,**

**qty number not null,**

**sale\_total number not null**

**)**

**partition by range (time\_id)**

**(**

**PARTITION sales\_q1 VALUES LESS THAN (TO\_DATE('01-APR-2011', 'DD-MON-YYYY')) TABLESPACE tsa,**

**PARTITION sales\_q2 VALUES LESS THAN (TO\_DATE('01-JUL-2011', 'DD-MON-YYYY')) TABLESPACE tsb,**

**PARTITION sales\_q3 VALUES LESS THAN (TO\_DATE('01-OCT-2011', 'DD-MON-YYYY')) TABLESPACE tsc,**

**PARTITION sales\_q4 VALUES LESS THAN (TO\_DATE('01-JAN-2012', 'DD-MON-YYYY')) TABLESPACE tsd );**

Table created.

**INSERT INTO sales VALUES(1,'02-JAN-2011',1,1,5,500);**

1 row created.

**INSERT INTO sales VALUES(2,'02-APR-2011',1,1,5,500);** **Table created**

1 row created.

**INSERT INTO sales VALUES(3,'02-JUL-2011',1,1,5,500);**

1 row created.

**INSERT INTO sales VALUES(4,'02-OCT-2011',1,1,5,500);**

1 row created.

**INSERT INTO sales VALUES(5,'02-JAN-2012',1,1,5,500);**

ERROR at line 1:

ORA-14400: inserted partition key does not map to any partition

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Query 2: Create the same table as in Q1. With a different name with ENABLE ROW MOVEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**create table sales (**

**prod\_id number primary key,**

**time\_id date not null,**

**cust\_id number not null,**

**promo\_id number not null,**

**qty number not null,**

**sale\_total number not null**

**)**

**partition by range (time\_id)**

**(**

**PARTITION sales\_q1 VALUES LESS THAN (TO\_DATE('01-APR-2011', 'DD-MON-YYYY')) TABLESPACE tsa,**

**PARTITION sales\_q2 VALUES LESS THAN (TO\_DATE('01-JUL-2011', 'DD-MON-YYYY')) TABLESPACE tsb,**

**PARTITION sales\_q3 VALUES LESS THAN (TO\_DATE('01-OCT-2011', 'DD-MON-YYYY')) TABLESPACE tsc,**

**PARTITION sales\_q4 VALUES LESS THAN (TO\_DATE('01-JAN-2012', 'DD-MON-YYYY')) TABLESPACE tsd )**

**enable row movement ;**

Table created.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Query 3: 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')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLE sales\_by\_region**

**(deptno number,**

**deptname varchar2(20),**

**quarterly\_sales number(10, 2),**

**state varchar2(2))**

**PARTITION BY LIST (state)**

**(PARTITION q1\_north VALUES ('OR', 'WA'),**

**PARTITION q1\_south VALUES ('AZ', 'UT', 'NM'),**

**PARTITION q1\_east VALUES ('NY', 'VM', 'NJ'),**

**PARTITION q1\_southeast VALUES ('FL', 'GA'),**

**PARTITION q1\_morthcentral VALUES ('SD', 'WI'),**

**PARTITION q1\_west VALUES ('OK', 'TX'));**

**insert into sales\_by\_region values (10, 'accounting', 100, 'WA');**

**insert into sales\_by\_region values(20, 'RD', 150, 'OR');**

**insert into sales\_by\_region values(30, 'sales', 100, 'FL') ;**

**insert into sales\_by\_region values(40, 'HR', 10, 'TX');**

**insert into sales\_by\_region values (50, 'system engg', 10, 'CA');**

ERROR at line 1:

ORA-14400: inserted partition key does not map to any partition

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Query 4: 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 5. Demonstarte using system defined and user defined partition concepts.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**Create table emp**

**(emp\_no number(2),**

**job number not null,**

**sal number not null,**

**dept number not null**

**)**

**partition by hash(emp\_no)**

**partitions 5;**

**Create table emp2**

**(emp\_no number(2),**

**job number not null,**

**sal number not null,**

**dept number not null**

**)**

**partition by hash(emp\_no)**

**(partition h1,**

**partition h2,**

**partition h3,**

**partition h4,**

**partition h5);**

INSERT INTO emp values(1,1,1000,1);

INSERT INTO emp values(5,1,1000,1);

INSERT INTO emp values(90,1,1000,1);

INSERT INTO emp2 values(1,1,1000,1);

INSERT INTO emp2 values(5,1,1000,1);

INSERT INTO emp2 values(90,1,1000,1);

INSERT INTO emp2 values(52,1,1000,1);

**SELECT \* from emp2 partition(h1);**

no rows selected

**SELECT \* from emp2 partition(h2);**

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

90 1 1000 1

52 1 1000 1

2 rows selected.

**SELECT \* from emp2 partition(h3);**

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

5 1 1000 1

1 row selected.

**SELECT \* from emp2 partition(h4);**

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

1 1 1000 1

1 row selected.

**SELECT \* from emp2 partition(h5);**

no rows selected

**SELECT \* from emp partition(SYS\_P21);**

**SELECT \* from emp partition(SYS\_P22);**

**SELECT \* from emp partition(SYS\_P23);**

**SELECT \* from emp partition(SYS\_P24);**

**SELECT \* from emp partition(SYS\_P25);**

no rows selected

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

90 1 1000 1

1 row selected.

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

5 1 1000 1

1 row selected.

EMP\_NO JOB SAL DEPT

---------- ---------- ---------- ----------

1 1 1000 1

1 row selected.

no rows selected

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Query 5: 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**create table epoch (**

**year\_e number not null,**

**mon\_e number not null,**

**day\_e number not null,**

**amt number not null**

**)**

**partition by range (year\_e,mon\_e)(**

**partition p1 values less than (2001,1),**

**partition p2 values less than (2001,4),**

**partition p3 values less than (2001,7),**

**partition p4 values less than (2001,10),**

**partition p5 values less than (2002,1),**

**partition future values less than (MAXVALUE,MAXVALUE)**

**);**

**SQL> INSERT INTO epoch VALUES(2001,3,17, 2000);**

1 row created.

**SQL> INSERT INTO epoch VALUES(2001,11,1, 5000);**

1 row created.

**SQL> INSERT INTO epoch VALUES(2002,1,1, 4000);**

1 row created.

**SQL> SELECT \* from epoch partition(p1);**

no rows selected

**SQL> SELECT \* from epoch partition(p2);**

YEAR\_E MON\_E DAY\_E AMT

---------- ---------- ---------- ----------

2001 3 17 2000

1 row selected.

**SQL> SELECT \* from epoch partition(p3);**

no rows selected

**SQL> SELECT \* from epoch partition(p4);**

no rows selected

**SQL> SELECT \* from epoch partition(p5);**

YEAR\_E MON\_E DAY\_E AMT

---------- ---------- ---------- ----------

2001 11 1 5000

1 row selected.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 6**: 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);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**create table epoch2**

**(seller\_id number(6),**

**prod\_id number(6),**

**amt number(6))**

**partition by range(seller\_id,prod\_id)**

**(**

**partition p1 values less than (5,100),**

**partition p2 values less than (5,200),**

**partition p3 values less than (10,100),**

**partition p4 values less than (10,200)**

**);**

**\*/**

**INSERT INTO epoch2 VALUES(5,5, 1000);**

**INSERT INTO epoch2 VALUES(5,150, 1000);**

**INSERT INTO epoch2 VALUES(10,100, 1000);**

**SQL> SELECT \* from epoch2 partition(p1);**

**SELLER\_ID PROD\_ID AMT**

**---------- ---------- ----------**

**5 5 1000**

**1 row selected.**

**SQL> SELECT \* from epoch2 partition(p2);**

**SELLER\_ID PROD\_ID AMT**

**---------- ---------- ----------**

**5 150 1000**

1 row selected.

**SQL> SELECT \* from epoch2 partition(p3);**

no rows selected

**SQL> SELECT \* from epoch2 partition(p4);**

**SELLER\_ID PROD\_ID AMT**

**---------- ---------- ----------**

**10 100 1000**

1 row selected.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Query 7: 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**create table test**

**(prod\_id number(6),**

**cust\_id number(6),**

**promo\_id number(6),**

**qty number(6),**

**amt number(6),**

**mon number(6))**

**partition by range(mon)**

**interval(1)**

**(**

**partition p1 values less than (2),**

**partition p2 values less than (3),**

**partition p3 values less than (4),**

**partition p4 values less than (5)**

**);**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Query 8: 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**create table orders**

**(order\_id number(6) primary key,**

**order\_date date,**

**cust\_id number(6),**

**shipper\_id number(6))**

**partition by range (order\_date)**

**(**

**PARTITION P201101 VALUES LESS THAN (TO\_DATE('01-APR-2011', 'DD-MON-YYYY')),**

**PARTITION P201102 VALUES LESS THAN (TO\_DATE('01-JUL-2011', 'DD-MON-YYYY')),**

**PARTITION P201103 VALUES LESS THAN (TO\_DATE('01-OCT-2011', 'DD-MON-YYYY')),**

**PARTITION P201104 VALUES LESS THAN (TO\_DATE('01-JAN-2012', 'DD-MON-YYYY')));**

**create table orders2**

**(order\_id number(6) not null,**

**cust\_id number(6),**

**price number(6),**

**qty number(6),**

**constraint orders\_fk foreign key (order\_id) references orders)**

**PARTITION BY REFERENCE (orders\_fk);**

**SELECT TABLE\_NAME, PARTITION\_NAME FROM USER\_TAB\_PARTITIONS WHERE TABLE\_NAME IN ('ORDERS');**

TABLE\_NAME PARTITION\_NAME

------------------------------ ------------------------------

ORDERS P201101

ORDERS P201102

ORDERS P201103

3 rows selected.

**ALTER TABLE orders DROP PARTITION P201103;**

Table altered.

**SELECT TABLE\_NAME, PARTITION\_NAME FROM USER\_TAB\_PARTITIONS WHERE TABLE\_NAME IN ('ORDERS','ORDERS2');**

TABLE\_NAME PARTITION\_NAME

------------------------------ ------------------------------

ORDERS P201101

ORDERS P201102

2 rows selected.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Query 9: 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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLE emp (**

**id NUMBER,**

**username VARCHAR2(20),**

**fsalary NUMBER(6),**

**vsalary NUMBER(6),**

**tot NUMBER(6)**

**GENERATED ALWAYS AS**

**(**

**vsalary+fsalary**

**) VIRTUAL**

**)**

**PARTITION BY RANGE (tot)**

**(**

**partition p1 values less than (25000),**

**partition p2 values less than (50000),**

**partition p3 values less than (75000),**

**partition future values less than (MAXVALUE)**

**);**

**INSERT INTO emp(id,username,fsalary,vsalary) values(1,'paddi',20000,10000);**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Query 10: 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLE composite (**

**cust\_id NUMBER(10),**

**cust\_name VARCHAR2(25),**

**cust\_state VARCHAR2(2),**

**time\_id DATE)**

**PARTITION BY LIST(cust\_state)**

**SUBPARTITION BY RANGE (time\_id)**

**SUBPARTITION TEMPLATE(**

**SUBPARTITION original VALUES LESS THAN (TO\_DATE('01-JAN-2005', 'DD-MON-YYYY')),**

**SUBPARTITION acquired VALUES LESS THAN (TO\_DATE('01-JAN-2010', 'DD-MON-YYYY')),**

**SUBPARTITION recent VALUES LESS THAN (MAXVALUE))**

**(**

**PARTITION west VALUES ('OR', 'WA'),**

**PARTITION east VALUES ('NY', 'CT'),**

**PARTITION cent VALUES ('IL', 'MN'),**

**partition p4 values(default)**

**);**

**INSERT INTO composite VALUES(1,'pradyot','OR','25-APR-2004');**

1 row created.

**SELECT partition\_name, composite, high\_value**

**FROM user\_tab\_partitions where table\_name='COMPOSITE';**

**select \* from composite SUBPARTITION(west\_original);**

CUST\_ID CUST\_NAME CU TIME\_ID

---------- ------------------------- -- ---------

1 pradyot OR 25-APR-04

1 row selected.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*