TASK 6 REPORT

1. HANDS-ON TASK PARTITIONS

CREATE TABLE WITH PARTITIONS

```
CREATE TABLE BL_3NF.CE_SALES(
  SALE_ID                 NUMBER(38)          NOT NULL,
  SALE_SOURCE_SYSTEM      VARCHAR2(50 CHAR)   NOT NULL,
  SALE_SOURCE_ENTITY      VARCHAR2(50 CHAR)   NOT NULL,
  PRODUCT_ID              NUMBER(38)          NOT NULL,
  DATE_ID                 DATE                NOT NULL,
  PROMOTION_ID            NUMBER(38)          NOT NULL,
  CHANNEL_ID              NUMBER(38)          NOT NULL,
  STORE_ID                NUMBER(38)          NOT NULL,
  EMPLOYEE_ID             NUMBER(38)          NOT NULL,
  CUSTOMER_ID             NUMBER(38)          NOT NULL,
  SALE_COST               NUMBER(15,3)        ,
  SALE_PRICE              NUMBER(15,3)        ,
  SALE_QUANTITY           NUMBER(38)          ,
  TA_UPDATE_DT            DATE                NOT NULL,
  TA_INSERT_DT            DATE                NOT NULL

)
PARTITION BY RANGE (DATE_ID)
(
PARTITION PART_1 VALUES LESS THAN (TO_DATE ('01/02/2020','DD/MM/YYYY')),
PARTITION PART_2 VALUES LESS THAN (TO_DATE ('01/03/2020','DD/MM/YYYY')),
PARTITION PART_3 VALUES LESS THAN (TO_DATE ('01/04/2020','DD/MM/YYYY')),
PARTITION PART_4 VALUES LESS THAN (TO_DATE ('01/05/2020','DD/MM/YYYY')),
PARTITION PART_5 VALUES LESS THAN (TO_DATE ('01/06/2020','DD/MM/YYYY')),
PARTITION PART_6 VALUES LESS THAN (TO_DATE ('01/07/2020','DD/MM/YYYY')),
PARTITION PART_7 VALUES LESS THAN (TO_DATE ('01/08/2020','DD/MM/YYYY')),
PARTITION PART_8 VALUES LESS THAN (TO_DATE ('01/09/2020','DD/MM/YYYY')),
PARTITION PART_9 VALUES LESS THAN (TO_DATE ('01/10/2020','DD/MM/YYYY')),
PARTITION PART_10 VALUES LESS THAN (TO_DATE ('01/11/2020','DD/MM/YYYY')),
PARTITION PART_11 VALUES LESS THAN (TO_DATE ('01/12/2020','DD/MM/YYYY')),
PARTITION PART_12 VALUES LESS THAN (TO_DATE ('01/01/2021','DD/MM/YYYY')),
PARTITION PART_13 VALUES LESS THAN (TO_DATE ('01/02/2021','DD/MM/YYYY')),
PARTITION PART_14 VALUES LESS THAN (TO_DATE ('01/03/2021','DD/MM/YYYY')),
PARTITION PART_15 VALUES LESS THAN (TO_DATE ('01/04/2021','DD/MM/YYYY')),
PARTITION PART_16 VALUES LESS THAN (TO_DATE ('01/05/2021','DD/MM/YYYY')),
PARTITION PART_17 VALUES LESS THAN (TO_DATE ('01/06/2021','DD/MM/YYYY')),
PARTITION PART_18 VALUES LESS THAN (TO_DATE ('01/07/2021','DD/MM/YYYY')),
PARTITION PART_19 VALUES LESS THAN (TO_DATE ('01/08/2021','DD/MM/YYYY')),
PARTITION PART_20 VALUES LESS THAN (TO_DATE ('01/09/2021','DD/MM/YYYY'))
```

);

CHECK PARTITIONS

*SELECT **

FROM ALL_TAB_PARTITIONS

WHERE TABLE_NAME = 'CE_SALES';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE
1	BL_3NF	CE_SALES	NO	PART_1	0	TO_DATE(' 2020-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
2	BL_3NF	CE_SALES	NO	PART_2	0	TO_DATE(' 2020-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
3	BL_3NF	CE_SALES	NO	PART_3	0	TO_DATE(' 2020-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
4	BL_3NF	CE_SALES	NO	PART_4	0	TO_DATE(' 2020-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
5	BL_3NF	CE_SALES	NO	PART_5	0	TO_DATE(' 2020-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
6	BL_3NF	CE_SALES	NO	PART_6	0	TO_DATE(' 2020-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
7	BL_3NF	CE_SALES	NO	PART_7	0	TO_DATE(' 2020-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
8	BL_3NF	CE_SALES	NO	PART_8	0	TO_DATE(' 2020-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
9	BL_3NF	CE_SALES	NO	PART_9	0	TO_DATE(' 2020-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
10	BL_3NF	CE_SALES	NO	PART_10	0	TO_DATE(' 2020-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
11	BL_3NF	CE_SALES	NO	PART_11	0	TO_DATE(' 2020-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
12	BL_3NF	CE_SALES	NO	PART_12	0	TO_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
13	BL_3NF	CE_SALES	NO	PART_13	0	TO_DATE(' 2021-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
14	BL_3NF	CE_SALES	NO	PART_14	0	TO_DATE(' 2021-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
15	BL_3NF	CE_SALES	NO	PART_15	0	TO_DATE(' 2021-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
16	BL_3NF	CE_SALES	NO	PART_16	0	TO_DATE(' 2021-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
17	BL_3NF	CE_SALES	NO	PART_17	0	TO_DATE(' 2021-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
18	BL_3NF	CE_SALES	NO	PART_18	0	TO_DATE(' 2021-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
19	BL_3NF	CE_SALES	NO	PART_19	0	TO_DATE(' 2021-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
20	BL_3NF	CE_SALES	NO	PART_20	0	TO_DATE(' 2021-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

ADDING PARTITION

ADD PARTITION

ALTER TABLE CE_SALES

ADD PARTITION PART_21 VALUES LESS THAN (TO_DATE ('01/10/2021','DD/MM/YYYY'));

CHECK PARTITIONS

*SELECT **

FROM ALL_TAB_PARTITIONS

WHERE TABLE_NAME = 'CE_SALES';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE
1	BL_3NF	CE_SALES	NO	PART_1	0	TO_DATE(' 2020-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
2	BL_3NF	CE_SALES	NO	PART_2	0	TO_DATE(' 2020-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
3	BL_3NF	CE_SALES	NO	PART_3	0	TO_DATE(' 2020-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
4	BL_3NF	CE_SALES	NO	PART_4	0	TO_DATE(' 2020-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
5	BL_3NF	CE_SALES	NO	PART_5	0	TO_DATE(' 2020-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
6	BL_3NF	CE_SALES	NO	PART_6	0	TO_DATE(' 2020-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
7	BL_3NF	CE_SALES	NO	PART_7	0	TO_DATE(' 2020-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
8	BL_3NF	CE_SALES	NO	PART_8	0	TO_DATE(' 2020-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
9	BL_3NF	CE_SALES	NO	PART_9	0	TO_DATE(' 2020-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
10	BL_3NF	CE_SALES	NO	PART_10	0	TO_DATE(' 2020-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
11	BL_3NF	CE_SALES	NO	PART_11	0	TO_DATE(' 2020-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
12	BL_3NF	CE_SALES	NO	PART_12	0	TO_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
13	BL_3NF	CE_SALES	NO	PART_13	0	TO_DATE(' 2021-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
14	BL_3NF	CE_SALES	NO	PART_14	0	TO_DATE(' 2021-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
15	BL_3NF	CE_SALES	NO	PART_15	0	TO_DATE(' 2021-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
16	BL_3NF	CE_SALES	NO	PART_16	0	TO_DATE(' 2021-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
17	BL_3NF	CE_SALES	NO	PART_17	0	TO_DATE(' 2021-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
18	BL_3NF	CE_SALES	NO	PART_18	0	TO_DATE(' 2021-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
19	BL_3NF	CE_SALES	NO	PART_19	0	TO_DATE(' 2021-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
20	BL_3NF	CE_SALES	NO	PART_20	0	TO_DATE(' 2021-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
21	BL_3NF	CE_SALES	NO	PART_21	0	TO_DATE(' 2021-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

COALESCING PARTITION

CREATE TABLE WITH HASH PARTITION:

CREATE TABLE HASH_EXAMPLE

(HASH_KEY_COL DATE,

DATA VARCHAR2(20))

PARTITION BY HASH (HASH_KEY_COL)

(

PARTITION PART_1,

PARTITION PART_2,

PARTITION PART_3

);

CHECK PARTITIONS

*SELECT **

FROM ALL_TAB_PARTITIONS

WHERE TABLE_NAME = 'HASH_EXAMPLE';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE	HIGH_VALUE_LENGTH	PARTITION_POSITION
1	BL_3NF	HASH_EXAMPLE	NO	PART_1	0	(null)	0	1
2	BL_3NF	HASH_EXAMPLE	NO	PART_2	0	(null)	0	2
3	BL_3NF	HASH_EXAMPLE	NO	PART_3	0	(null)	0	3

COALESCE PARTITION

ALTER TABLE HASH_EXAMPLE COALESCE PARTITION;

CHECK PARTITIONS

*SELECT **

FROM ALL_TAB_PARTITIONS

WHERE TABLE_NAME = 'HASH_EXAMPLE';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE	HIGH_VALUE_LENGTH	PARTITION_POSITION
1	BL_3NF	HASH_EXAMPLE	NO	PART_1	0	(null)	0	1
2	BL_3NF	HASH_EXAMPLE	NO	PART_2	0	(null)	0	2

DROPPING PARTITION

DROP PARTITION

ALTER TABLE CE_SALES DROP PARTITION PART_21 UPDATE INDEXES;

CHECK PARTITIONS

*SELECT **

FROM ALL_TAB_PARTITIONS

WHERE TABLE_NAME = 'CE_SALES';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE
1	BL_3NF	CE_SALES	NO	PART_1	0	TO_DATE(' 2020-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
2	BL_3NF	CE_SALES	NO	PART_2	0	TO_DATE(' 2020-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
3	BL_3NF	CE_SALES	NO	PART_3	0	TO_DATE(' 2020-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
4	BL_3NF	CE_SALES	NO	PART_4	0	TO_DATE(' 2020-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
5	BL_3NF	CE_SALES	NO	PART_5	0	TO_DATE(' 2020-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
6	BL_3NF	CE_SALES	NO	PART_6	0	TO_DATE(' 2020-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
7	BL_3NF	CE_SALES	NO	PART_7	0	TO_DATE(' 2020-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
8	BL_3NF	CE_SALES	NO	PART_8	0	TO_DATE(' 2020-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
9	BL_3NF	CE_SALES	NO	PART_9	0	TO_DATE(' 2020-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
10	BL_3NF	CE_SALES	NO	PART_10	0	TO_DATE(' 2020-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
11	BL_3NF	CE_SALES	NO	PART_11	0	TO_DATE(' 2020-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
12	BL_3NF	CE_SALES	NO	PART_12	0	TO_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
13	BL_3NF	CE_SALES	NO	PART_13	0	TO_DATE(' 2021-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
14	BL_3NF	CE_SALES	NO	PART_14	0	TO_DATE(' 2021-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
15	BL_3NF	CE_SALES	NO	PART_15	0	TO_DATE(' 2021-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
16	BL_3NF	CE_SALES	NO	PART_16	0	TO_DATE(' 2021-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
17	BL_3NF	CE_SALES	NO	PART_17	0	TO_DATE(' 2021-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
18	BL_3NF	CE_SALES	NO	PART_18	0	TO_DATE(' 2021-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
19	BL_3NF	CE_SALES	NO	PART_19	0	TO_DATE(' 2021-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
20	BL_3NF	CE_SALES	NO	PART_20	0	TO_DATE(' 2021-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

MERGING PARTITION

MERGE PARTITIONS

ALTER TABLE CE_SALES MERGE PARTITIONS PART_1,PART_2, PART_3 INTO PARTITION QUATER_1_2020;

ALTER TABLE CE_SALES MERGE PARTITIONS PART_4,PART_5, PART_6 INTO PARTITION QUATER_2_2020;

ALTER TABLE CE_SALES MERGE PARTITIONS PART_7,PART_8, PART_9 INTO PARTITION QUATER_3_2020;
ALTER TABLE CE_SALES MERGE PARTITIONS PART_10,PART_11, PART_12 INTO PARTITION QUATER_4_2020;
ALTER TABLE CE_SALES MERGE PARTITIONS PART_13,PART_14, PART_15 INTO PARTITION QUATER_1_2021;
ALTER TABLE CE_SALES MERGE PARTITIONS PART_16,PART_17, PART_18 INTO PARTITION QUATER_2_2021;
ALTER TABLE CE_SALES MERGE PARTITIONS PART_19,PART_20 INTO PARTITION QUATER_3_2021;

CHECK PARTITIONS

```
SELECT *
FROM ALL_TAB_PARTITIONS
WHERE TABLE_NAME = 'CE_SALES';
```

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE	HIG
1	BL_3NF	CE_SALES	NO	QUATER_1_2020	0	TO_DATE(' 2020-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
2	BL_3NF	CE_SALES	NO	QUATER_2_2020	0	TO_DATE(' 2020-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
3	BL_3NF	CE_SALES	NO	QUATER_3_2020	0	TO_DATE(' 2020-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
4	BL_3NF	CE_SALES	NO	QUATER_4_2020	0	TO_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
5	BL_3NF	CE_SALES	NO	QUATER_1_2021	0	TO_DATE(' 2021-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
6	BL_3NF	CE_SALES	NO	QUATER_2_2021	0	TO_DATE(' 2021-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	
7	BL_3NF	CE_SALES	NO	QUATER_3_2021	0	TO_DATE(' 2021-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')	

MOVING PARTITION

CREATE TABLESPASE

```
CREATE TABLESPACE TBS_01
  DATAFILE 'TBS_F2.DAT' SIZE 100M
  ONLINE;
```

CHECK TABLESPASE

```
SELECT TABLESPACE_NAME, FILE_NAME,BYTES FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME='TBS_01';
```

	TABLESPACE_NAME	FILE_NAME	BYTES
1	TBS_01	C:\APP\ANASTASIYA_VIKTAROVI\PRODUCT\21C\DBHOMEXE\DATABASE\TBS_F2.DAT	104857600

CHECK CURRENT PARTITIONS TABLESPASE

```
SELECT TABLE_NAME, PARTITION_NAME, TABLESPACE_NAME FROM USER_TAB_PARTITIONS WHERE
TABLE_NAME='CE_SALES'
```

	TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
1	CE_SALES	QUATER_1_2020	USERS
2	CE_SALES	QUATER_1_2021	USERS
3	CE_SALES	QUATER_2_2020	USERS
4	CE_SALES	QUATER_2_2021	USERS
5	CE_SALES	QUATER_3_2020	USERS
6	CE_SALES	QUATER_3_2021	USERS
7	CE_SALES	QUATER_4_2020	USERS

MOVE PARTITITON

ALTER TABLE CE_SALES MOVE PARTITION QUATER_3_2021
TABLESPACE TBS_01 NOLOGGING COMPRESS;

CHECK NEW PARTITIONS TABLESPASE

SELECT TABLE_NAME, PARTITION_NAME, TABLESPACE_NAME FROM USER_TAB_PARTITIONS WHERE TABLE_NAME='CE_SALES'

	TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
1	CE_SALES	QUATER_1_2020	USERS
2	CE_SALES	QUATER_1_2021	USERS
3	CE_SALES	QUATER_2_2020	USERS
4	CE_SALES	QUATER_2_2021	USERS
5	CE_SALES	QUATER_3_2020	USERS
6	CE_SALES	QUATER_4_2020	USERS
7	CE_SALES	QUATER_3_2021	TBS_01

SPLITTING PARTITION

SPLIT PARTITITON

ALTER TABLE CE_SALES SPLIT PARTITION QUATER_1_2020 INTO
(PARTITION PART_1 VALUES LESS THAN (TO_DATE ('01/02/2020','DD/MM/YYYY')),
PARTITION PART_2 VALUES LESS THAN (TO_DATE ('01/03/2020','DD/MM/YYYY')),
PARTITION PART_3);

CHECK PARTITIONS

*SELECT **
FROM ALL_TAB_PARTITIONS
WHERE TABLE_NAME = 'CE_SALES';

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE
1	BL_3NF	CE_SALES	NO	PART_1	0	TO_DATE(' 2020-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
2	BL_3NF	CE_SALES	NO	PART_2	0	TO_DATE(' 2020-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
3	BL_3NF	CE_SALES	NO	PART_3	0	TO_DATE(' 2020-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
4	BL_3NF	CE_SALES	NO	QUATER_2_2020	0	TO_DATE(' 2020-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
5	BL_3NF	CE_SALES	NO	QUATER_3_2020	0	TO_DATE(' 2020-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
6	BL_3NF	CE_SALES	NO	QUATER_4_2020	0	TO_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
7	BL_3NF	CE_SALES	NO	QUATER_1_2021	0	TO_DATE(' 2021-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
8	BL_3NF	CE_SALES	NO	QUATER_2_2021	0	TO_DATE(' 2021-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
9	BL_3NF	CE_SALES	NO	QUATER_3_2021	0	TO_DATE(' 2021-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

TRUNCATING PARTITION

CHECK PARTITION

*SELECT **
FROM CE_SALES
PARTITION (PART_21);

	SALE_ID	SALE_SOURCE_SYSTEM	SALE_SOURCE_ENTITY	PRODUCT_ID	DATE_ID	PROMOTION_ID	CHANNEL_ID	STORE_ID	EMPLOYEE_ID	CUSTOMER_ID	SALE_COST	SALE_PRICE	SALE_QUANTITY	TA_UPDATE_DT	TA_INSERT_DT
1	17498725	pesonnel_sales	src_sales	163899	02-SEP-21	121	187	668	1222	538791	974.23	1217.79	1	10-MAR-22	10-MAR-22
2	17498726	pesonnel_sales	src_sales	163855	02-SEP-21	121	187	666	1223	504172	848.64	1060.8	1	10-MAR-22	10-MAR-22
3	17498727	pesonnel_sales	src_sales	169638	02-SEP-21	121	188	703	1225	540631	93.74	117.18	1	10-MAR-22	10-MAR-22
4	17499314	pesonnel_sales	src_sales	161835	03-SEP-21	121	186	664	1243	519737	1630.76	2038.45	1	10-MAR-22	10-MAR-22
5	17499315	pesonnel_sales	src_sales	161940	03-SEP-21	121	186	695	1244	527011	1858.93	2323.66	1	10-MAR-22	10-MAR-22
6	17499316	pesonnel_sales	src_sales	171936	03-SEP-21	121	187	696	1245	529694	1510.6	1888.25	1	10-MAR-22	10-MAR-22
7	17499317	pesonnel_sales	src_sales	167086	03-SEP-21	121	187	688	1246	546504	657.74	822.18	1	10-MAR-22	10-MAR-22
8	17499318	pesonnel_sales	src_sales	170344	03-SEP-21	121	187	683	1248	528344	1772.77	2215.96	1	10-MAR-22	10-MAR-22
9	17499319	pesonnel_sales	src_sales	172161	03-SEP-21	121	187	665	1249	518936	1012.92	1266.15	1	10-MAR-22	10-MAR-22
10	17499320	pesonnel_sales	src_sales	168175	03-SEP-21	121	187	702	1250	504572	557.61	697.01	1	10-MAR-22	10-MAR-22
11	17499321	pesonnel_sales	src_sales	169872	03-SEP-21	121	187	669	1251	536798	1802.41	2253.01	1	10-MAR-22	10-MAR-22
12	17499322	pesonnel_sales	src_sales	171924	03-SEP-21	121	187	686	1253	541210	1843.55	2304.44	1	10-MAR-22	10-MAR-22
13	17499323	pesonnel_sales	src_sales	168555	03-SEP-21	121	188	666	1255	540612	1024.96	1281.2	1	10-MAR-22	10-MAR-22
14	17499324	pesonnel_sales	src_sales	169902	03-SEP-21	121	188	703	1257	523867	1522.1	1902.63	1	10-MAR-22	10-MAR-22
15	17499325	pesonnel_sales	src_sales	170318	03-SEP-21	121	188	704	1258	531876	434.54	543.17	1	10-MAR-22	10-MAR-22
16	17499399	pesonnel_sales	src_sales	166556	01-SEP-21	121	186	670	1284	517447	487.78	609.72	1	10-MAR-22	10-MAR-22
17	17499400	pesonnel_sales	src_sales	167376	01-SEP-21	121	188	686	1285	547425	1634.6	2043.25	1	10-MAR-22	10-MAR-22

Trancate partition

ALTER TABLE CE_SALES TRUNCATE PARTITION PART_21 UPDATE INDEXES;

Check partition

*select **

from CE_SALES

Partition (PART_21);

SALE_ID	SALE_S...	SALE_S...	PRODUC...	DATE_ID	PROMO...	CHANNE...	STORE_ID	EMPLOY...	CUSTO...	SALE_C...	SALE_P...	SALE_Q...	TA_UPD...	TA_INS...

Exchange with

Before truncate in previous point create table:

Create table temp_part as

*select **

from CE_SALES Where date_id > to_date ('01/09/2021','dd/mm/yyyy')

And check count of rows in ce_sales

Select count(*) from ce_sales;

Script Output × | Query Result × | Query Result 1 × | C

SQL | All Rows Fetched: 1 in 0.055 seconds

	COUNT(*)
1	1025193

After truncate check number of rows

Select count(*) from ce_sales;

Script Output × | Query Result × | Query Result

SQL | All Rows Fetched: 1 in 0.038 seco

	COUNT(*)
1	1010699

And partition state

*select **

from CE_SALES

Partition (PART_21);

SALE_ID	SALE_S...	SALE_S...	PRODUC...	DATE_ID	PROMO...	CHANNE...	STORE_ID	EMPLOY...	CUSTO...	SALE_C...	SALE_P...	SALE_Q...	TA_UPD...	TA_INS...

EXCHANGE

ALTER TABLE CE_SALES

EXCHANGE PARTITION PART_21

WITH TABLE TEMP_PART;

CHECK STATE OF PARTITION AND NUMBER OF ROWS:

*SELECT **

FROM CE_SALES

PARTITION (PART_21);

	SALE_ID	SALE_SOURCE_SYSTEM	SALE_SOURCE_ENTITY	PRODUCT_ID	DATE_ID	PROMOTION_ID	CHANNEL_ID	STORE_ID	EMPLOYEE_ID	CUSTOMER_ID	SALE_COST	SALE_PRICE	SALE_QUANTITY	TA_UPDATE_DT	TA_INSERT_DT
1	19550161	pesonnel_sales	src_sales	163899	02-SEP-21	121	187	668	1222	538791	974.23	1217.79	1	10-MAR-22	10-MAR-22
2	19550162	pesonnel_sales	src_sales	163855	02-SEP-21	121	187	666	1223	504172	848.64	1060.8	1	10-MAR-22	10-MAR-22
3	19550163	pesonnel_sales	src_sales	169638	02-SEP-21	121	188	703	1225	540631	93.74	117.18	1	10-MAR-22	10-MAR-22
4	19550750	pesonnel_sales	src_sales	161835	03-SEP-21	121	186	664	1243	519737	1630.76	2038.45	1	10-MAR-22	10-MAR-22
5	19550751	pesonnel_sales	src_sales	161940	03-SEP-21	121	186	695	1244	527011	1858.93	2323.66	1	10-MAR-22	10-MAR-22
6	19550752	pesonnel_sales	src_sales	171936	03-SEP-21	121	187	696	1245	529694	1510.6	1888.25	1	10-MAR-22	10-MAR-22
7	19550753	pesonnel_sales	src_sales	167086	03-SEP-21	121	187	688	1246	546504	657.74	822.18	1	10-MAR-22	10-MAR-22
8	19550754	pesonnel_sales	src_sales	170344	03-SEP-21	121	187	683	1248	528344	1772.77	2215.96	1	10-MAR-22	10-MAR-22
9	19550755	pesonnel_sales	src_sales	172161	03-SEP-21	121	187	665	1249	518936	1012.92	1266.15	1	10-MAR-22	10-MAR-22
10	19550756	pesonnel_sales	src_sales	168175	03-SEP-21	121	187	702	1250	504572	557.61	697.01	1	10-MAR-22	10-MAR-22
11	19550757	pesonnel_sales	src_sales	169872	03-SEP-21	121	187	669	1251	536798	1802.41	2253.01	1	10-MAR-22	10-MAR-22
12	19550758	pesonnel_sales	src_sales	171924	03-SEP-21	121	187	686	1253	541210	1843.55	2304.44	1	10-MAR-22	10-MAR-22
13	19550759	pesonnel_sales	src_sales	168555	03-SEP-21	121	188	666	1255	540612	1024.96	1281.2	1	10-MAR-22	10-MAR-22
14	19550760	pesonnel_sales	src_sales	169902	03-SEP-21	121	188	703	1257	523867	1522.1	1902.63	1	10-MAR-22	10-MAR-22
15	19550761	pesonnel_sales	src_sales	170318	03-SEP-21	121	188	704	1258	531876	434.54	543.17	1	10-MAR-22	10-MAR-22
16	19550835	pesonnel_sales	src_sales	166556	01-SEP-21	121	186	670	1284	517447	487.78	609.72	1	10-MAR-22	10-MAR-22

SELECT COUNT(*) FROM CE_SALES;

Script Output × | Query Result × | Query Result 1 × | C

SQL | All Rows Fetched: 1 in 0.055 seconds

	COUNT(*)
1	1025193

2. PARALL EXECUTION

PARALLEL EXECUTION DRAMATICALLY REDUCES RESPONSE TIME FOR DATA-INTENSIVE OPERATIONS ON LARGE DATABASES TYPICALLY ASSOCIATED WITH DECISION SUPPORT SYSTEMS (DSS) AND DATA WAREHOUSES. WE CAN ALSO IMPLEMENT **PARALLEL EXECUTION** ON CERTAIN TYPES OF ONLINE TRANSACTION PROCESSING (OLTP) AND HYBRID SYSTEMS. SIMPLY EXPRESSED, PARALLELISM IS THE IDEA OF BREAKING DOWN A TASK SO THAT, INSTEAD OF ONE PROCESS DOING ALL OF THE WORK IN A QUERY, MANY PROCESSES DO PART OF THE WORK AT THE SAME TIME. AN EXAMPLE OF THIS IS WHEN FOUR PROCESSES HANDLE FOUR DIFFERENT QUARTERS IN A YEAR INSTEAD OF ONE PROCESS HANDLING ALL FOUR QUARTERS BY ITSELF. THE IMPROVEMENT IN PERFORMANCE CAN BE QUITE HIGH. IN THIS CASE, EACH QUARTER WILL BE A **PARTITION**, A SMALLER AND MORE MANAGEABLE UNIT OF AN INDEX OR TABLE. PARALLEL EXECUTION IMPROVES PROCESSING FOR:

- QUERIES REQUIRING LARGE TABLE SCANS, JOINS, OR PARTITIONED INDEX SCANS
- CREATION OF LARGE INDEXES
- CREATION OF LARGE TABLES (INCLUDING MATERIALIZED VIEWS)
- BULK INSERTS, UPDATES, MERGES, AND DELETES

WE CAN ALSO USE PARALLEL EXECUTION TO ACCESS OBJECT TYPES WITHIN AN ORACLE DATABASE.

When to Implement Parallel Execution

The benefits of parallel execution can be seen in data warehousing environments. OLTP systems can also benefit from parallel execution during batch processing and during schema maintenance operations such as creation of indexes. The average simple DML or SELECT statements that characterize OLTP applications would not see any benefit from being executed in parallel.

When Not to Implement Parallel Execution

Parallel execution is not normally useful for:

- Environments in which the typical query or transaction is very short (a few seconds or less). This includes most online transaction systems. Parallel execution is not useful in these environments because there is a cost associated with coordinating the parallel execution servers; for short transactions, the cost of this coordination may outweigh the benefits of parallelism.
- Environments in which the CPU, memory, or I/O resources are already heavily utilized. Parallel execution is designed to exploit additional available hardware resources; if no such resources are available, then parallel execution will not yield any benefits and indeed may be detrimental to performance.

Operations That Can Be Parallelized

- Access methods
  Some examples are table scans, index full scans, and partitioned index range scans.
- Join methods
  Some examples are nested loop, sort merge, hash, and star transformation.
- DDL statements
  Some examples are CREATE TABLE AS SELECT, CREATE INDEX, REBUILD INDEX, REBUILD INDEX PARTITION, and MOVE/SPLIT/COALESCE PARTITION.
- DML statements
  Some examples are INSERT AS SELECT, updates, deletes, and MERGE operations.
- Miscellaneous SQL operations
  Some examples are GROUP BY, NOT IN, SELECT DISTINCT, UNION, UNION ALL, CUBE, and ROLLUP, as well as aggregate and table functions.

- Parallel query

  You can parallelize queries and subqueries in SELECT statements, as well as the query portions of DDL statements and DML statements (INSERT, UPDATE, DELETE, and MERGE).

How Parallel Execution Works

Parallel execution divides the task of executing a SQL statement into multiple small units, each of which is executed by a separate process. Also the incoming data (tables, indexes, partitions) can be divided into parts called granules. The user shadow process that wants to execute a query in parallel takes on the role as parallel execution coordinator or query coordinator. The query coordinator does the following:

- Parses the query and determines the degree of parallelism
- Allocates one or two set of slaves (threads or processes)
- Controls the query and sends instructions to the PQ slaves
- Determines which tables or indexes need to be scanned by the PQ slaves
- Produces the final output to the user

3.ANALYTICAL TASK

Partitioning is done to enhance performance and facilitate easy management of data. Partitioning also helps in balancing the various requirements of the system. It optimizes the hardware performance and simplifies the management of data warehouse by partitioning each fact table into multiple separate partitions.

historical data (DATE_ID) will be partitioning key for range partitions For 3NF layer and for star schema layer

DDL for ce_sales

```
CREATE TABLE BL_3NF.CE_SALES(
  SALE_ID                 NUMBER(38)          NOT NULL,
  SALE_SOURCE_SYSTEM      VARCHAR2(50 CHAR)   NOT NULL,
  SALE_SOURCE_ENTITY      VARCHAR2(50 CHAR)   NOT NULL,
  PRODUCT_ID              NUMBER(38)          NOT NULL,
  DATE_ID                 DATE                NOT NULL,
  PROMOTION_ID            NUMBER(38)          NOT NULL,
  CHANNEL_ID              NUMBER(38)          NOT NULL,
  STORE_ID                NUMBER(38)          NOT NULL,
  EMPLOYEE_ID             NUMBER(38)          NOT NULL,
  CUSTOMER_ID             NUMBER(38)          NOT NULL,
  SALE_COST               NUMBER(15,3)        ,
  SALE_PRICE              NUMBER(15,3)        ,
```

```
  SALE_QUANTITY                    NUMBER(38)          ,
  TA_UPDATE_DT                     DATE                NOT NULL,
  TA_INSERT_DT              DATE                  NOT NULL

)
PARTITION BY RANGE (DATE_ID)
(
PARTITION PART_1 VALUES LESS THAN (TO_DATE ('01/02/2020','DD/MM/YYYY')),
PARTITION PART_2 VALUES LESS THAN (TO_DATE ('01/03/2020','DD/MM/YYYY')),
PARTITION PART_3 VALUES LESS THAN (TO_DATE ('01/04/2020','DD/MM/YYYY')),
PARTITION PART_4 VALUES LESS THAN (TO_DATE ('01/05/2020','DD/MM/YYYY')),
PARTITION PART_5 VALUES LESS THAN (TO_DATE ('01/06/2020','DD/MM/YYYY')),
PARTITION PART_6 VALUES LESS THAN (TO_DATE ('01/07/2020','DD/MM/YYYY')),
PARTITION PART_7 VALUES LESS THAN (TO_DATE ('01/08/2020','DD/MM/YYYY')),
PARTITION PART_8 VALUES LESS THAN (TO_DATE ('01/09/2020','DD/MM/YYYY')),
PARTITION PART_9 VALUES LESS THAN (TO_DATE ('01/10/2020','DD/MM/YYYY')),
PARTITION PART_10 VALUES LESS THAN (TO_DATE ('01/11/2020','DD/MM/YYYY')),
PARTITION PART_11 VALUES LESS THAN (TO_DATE ('01/12/2020','DD/MM/YYYY')),
PARTITION PART_12 VALUES LESS THAN (TO_DATE ('01/01/2021','DD/MM/YYYY')),
PARTITION PART_13 VALUES LESS THAN (TO_DATE ('01/02/2021','DD/MM/YYYY')),
PARTITION PART_14 VALUES LESS THAN (TO_DATE ('01/03/2021','DD/MM/YYYY')),
PARTITION PART_15 VALUES LESS THAN (TO_DATE ('01/04/2021','DD/MM/YYYY')),
PARTITION PART_16 VALUES LESS THAN (TO_DATE ('01/05/2021','DD/MM/YYYY')),
PARTITION PART_17 VALUES LESS THAN (TO_DATE ('01/06/2021','DD/MM/YYYY')),
PARTITION PART_18 VALUES LESS THAN (TO_DATE ('01/07/2021','DD/MM/YYYY')),
PARTITION PART_19 VALUES LESS THAN (TO_DATE ('01/08/2021','DD/MM/YYYY')),
PARTITION PART_20 VALUES LESS THAN (TO_DATE ('01/09/2021','DD/MM/YYYY')),
PARTITION PART_21 VALUES LESS THAN (TO_DATE ('01/10/2021','DD/MM/YYYY')),
PARTITION PART_22 VALUES LESS THAN (TO_DATE ('01/11/2021','DD/MM/YYYY')),
PARTITION PART_23 VALUES LESS THAN (TO_DATE ('01/12/2021','DD/MM/YYYY')),
PARTITION PART_24 VALUES LESS THAN (TO_DATE ('01/01/2022','DD/MM/YYYY'))
);
```

DDL FOR DIM_SALES

```
CREATE TABLE FCT_SALES (
```

```
  SOURCE_SYSTEM     VARCHAR2(50) NOT NULL,
  SOUTCE_ENTITY     VARCHAR2(50) NOT NULL,
  PRODUCT_SURR_ID   NUMBER(38) NOT NULL,
  PROMOTION_SURR_ID NUMBER(38) NOT NULL,
  CHANNEL_SURR_ID   NUMBER(38) NOT NULL,
  STORE_SURR_ID     NUMBER(38) NOT NULL,
  EMPLOYEE_SURR_ID  NUMBER(38) NOT NULL,
  CUSTOMER_SURR_ID  NUMBER(38) NOT NULL,
  DATE_ID           DATE NOT NULL,
  UNIT_COST         NUMBER(15, 2),
  UNIT_PRICE        NUMBER(15, 2),
  SALES_QUANTITY    NUMBER(38),
  TA_UPDATE_DT      DATE  NOT NULL,
  TA_INSERT_DT      DATE  NOT NULL
)PARTITION BY RANGE (DATE_ID)
(
PARTITION PART_1 VALUES LESS THAN (TO_DATE ('01/02/2020','DD/MM/YYYY')),
PARTITION PART_2 VALUES LESS THAN (TO_DATE ('01/03/2020','DD/MM/YYYY')),
PARTITION PART_3 VALUES LESS THAN (TO_DATE ('01/04/2020','DD/MM/YYYY')),
PARTITION PART_4 VALUES LESS THAN (TO_DATE ('01/05/2020','DD/MM/YYYY')),
PARTITION PART_5 VALUES LESS THAN (TO_DATE ('01/06/2020','DD/MM/YYYY')),
PARTITION PART_6 VALUES LESS THAN (TO_DATE ('01/07/2020','DD/MM/YYYY')),
PARTITION PART_7 VALUES LESS THAN (TO_DATE ('01/08/2020','DD/MM/YYYY')),
PARTITION PART_8 VALUES LESS THAN (TO_DATE ('01/09/2020','DD/MM/YYYY')),
PARTITION PART_9 VALUES LESS THAN (TO_DATE ('01/10/2020','DD/MM/YYYY')),
PARTITION PART_10 VALUES LESS THAN (TO_DATE ('01/11/2020','DD/MM/YYYY')),
PARTITION PART_11 VALUES LESS THAN (TO_DATE ('01/12/2020','DD/MM/YYYY')),
PARTITION PART_12 VALUES LESS THAN (TO_DATE ('01/01/2021','DD/MM/YYYY')),
PARTITION PART_13 VALUES LESS THAN (TO_DATE ('01/02/2021','DD/MM/YYYY')),
PARTITION PART_14 VALUES LESS THAN (TO_DATE ('01/03/2021','DD/MM/YYYY')),
PARTITION PART_15 VALUES LESS THAN (TO_DATE ('01/04/2021','DD/MM/YYYY')),
PARTITION PART_16 VALUES LESS THAN (TO_DATE ('01/05/2021','DD/MM/YYYY')),
```

```
PARTITION PART_17 VALUES LESS THAN (TO_DATE ('01/06/2021','DD/MM/YYYY')),
PARTITION PART_18 VALUES LESS THAN (TO_DATE ('01/07/2021','DD/MM/YYYY')),
PARTITION PART_19 VALUES LESS THAN (TO_DATE ('01/08/2021','DD/MM/YYYY')),
PARTITION PART_20 VALUES LESS THAN (TO_DATE ('01/09/2021','DD/MM/YYYY')),
PARTITION PART_21 VALUES LESS THAN (TO_DATE ('01/10/2021','DD/MM/YYYY')),
PARTITION PART_22 VALUES LESS THAN (TO_DATE ('01/11/2021','DD/MM/YYYY')),
PARTITION PART_23 VALUES LESS THAN (TO_DATE ('01/12/2021','DD/MM/YYYY')),
PARTITION PART_24 VALUES LESS THAN (TO_DATE ('01/01/2022','DD/MM/YYYY'))
);
```