Database Design and Implementation of a Secure Online Shopping System

Brandon Russell, Joshua Oluwadamilare, Roha Seyoum, Tinuola Okusanya

University of Maryland University College

Group 3 Project Part 2: DBST670

Professor: Selwyn Igwe

April 21, 2019

Table of Contents

[Checklist Item 1 - ALTER SQL Command 4](#_Toc6251602)

[**Add Column** 4](#_Toc6251603)

[**Lengthen Column** 4](#_Toc6251604)

[**Shorten Column** 4](#_Toc6251605)

[**Change Data Type from NUMBER to CHAR** 5](#_Toc6251606)

[**Change Data Type from CHAR to NUMBER** 5](#_Toc6251607)

[**Validation** 6](#_Toc6251608)

[Checklist Item 2 - CREATE SQL Command 6](#_Toc6251609)

[**Create New Table** 7](#_Toc6251610)

[**Populate Data into Newly Created Table** 7](#_Toc6251611)

[**Validation** 8](#_Toc6251612)

[**Reestablish Referential Integrity** 8](#_Toc6251613)

[**Drop Existing Table** 9](#_Toc6251614)

[**Rename Table** 9](#_Toc6251615)

[Checklist Item 3 - Design for a Test of Restructuring Existing Objects 10](#_Toc6251616)

[**Step 1) Create Backup** 10](#_Toc6251617)

[**Step 2) Create New Table** 10](#_Toc6251618)

[**Step 3) Populate Data into the Newly Created Table from the Existing Table** 11](#_Toc6251619)

[**Step 4) Alter the Newly Created Table** 11](#_Toc6251620)

[**Step 5) Drop Existing Table** 12](#_Toc6251621)

[**Step 6) Rename Newly Created Table** 12](#_Toc6251622)

[**Step 7) Confirm the Removal of the Existing Table** 12](#_Toc6251623)

[Checklist Items 4-5 – Static Performance Test 13](#_Toc6251624)

[**Check V$SYSSTAT** 13](#_Toc6251625)

[**Read Load Results** 15](#_Toc6251626)

[**Write Load Results** 17](#_Toc6251627)

[Checklist Item 6 – Dynamic Performance Test 23](#_Toc6251628)

[**Read Load Results** 23](#_Toc6251629)

[**Write Load Results** 27](#_Toc6251630)

[Checklist Item 7 – Performance Test Matrix 38](#_Toc6251631)

[Checklist Items 8-12 - Backup and Recovery 38](#_Toc6251632)

[**Create Recovery Table – Checklist 8** 38](#_Toc6251633)

[**Export/Import – Checklist 9 & 10** 41](#_Toc6251634)

[**RMAN – Checklist 9 & 10** 48](#_Toc6251635)

[**RMAN Versus Import/Export – Checklist 11** 52](#_Toc6251636)

[**Flashback – Checklist 12** 52](#_Toc6251637)

[Checklist Items 13-19 – Monitor Health, Create Users & Synonyms, & Reverse Engineer 55](#_Toc6251638)

[**Monitor Health of Database – Items 13-15** 55](#_Toc6251639)

[**Create Users – Item 16** 58](#_Toc6251640)

[**Create Synonyms – Item 17** 64](#_Toc6251641)

[**Create reverse engineer on all\_objects views – Item 18** 69](#_Toc6251642)

[**Create script to detect end-user and what they are running – Item 19** 92](#_Toc6251643)

[References 93](#_Toc6251644)

# Checklist Item 1 - ALTER SQL Command

Below are SQL commands that are applied to alter a table by adding a column, lengthening a column, or shortening a column and changing the data type of an existing column from NUMBER to CHAR and then CHAR to NUMBER.

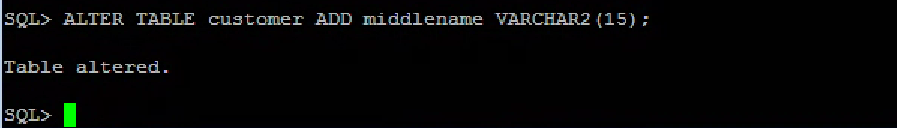
## **Add Column**

SQL> connect dbauser/abc123;

Connected.

SQL> ALTER TABLE customer ADD middlename VARCHAR2(15);

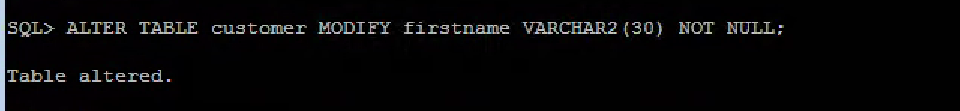
Table altered.



## **Lengthen Column**

SQL> ALTER TABLE customer MODIFY firstname VARCHAR2(30) NOT NULL;

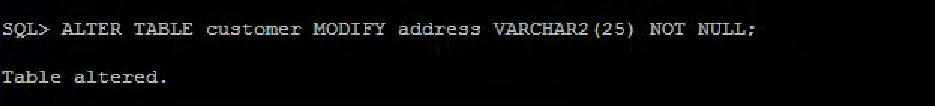
Table altered.



## **Shorten Column**

SQL> ALTER TABLE customer MODIFY address VARCHAR2(25) NOT NULL;

Table altered.

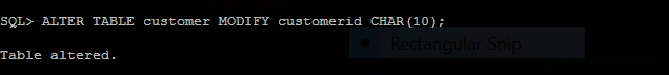


## **Change Data Type from NUMBER to CHAR**

A data type for a customerid column changed from NUMBER TO CHAR as follows:

SQL> ALTER TABLE customerid MODIFY customerid CHAR (10);

Table altered.

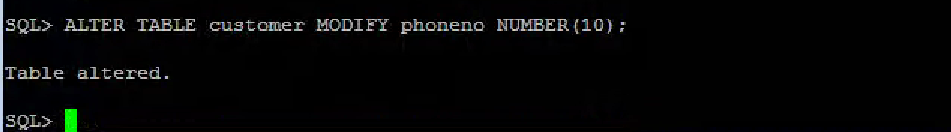


## **Change Data Type from CHAR to NUMBER**

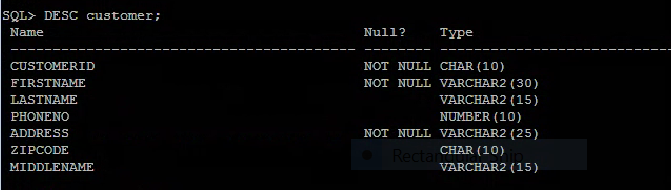
A data type for phoneno column changed from NUMBER TO CHAR as follows:

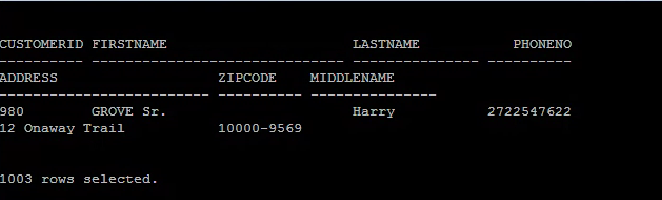
SQL> ALTER TABLE customer MODIFY phoneno NUMBER (10);

Table altered.



## **Validation**

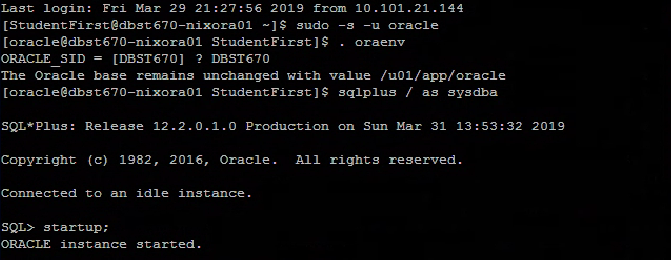




# Checklist Item 2 - CREATE SQL Command

Below are SQL commands that are applied to create another table like an existing table without one column. Then, populate the data from the previously existing table to the newly created table. Thereafter, reestablished referential integrity with the primary key on the newly created table. Finally, dropped the previously existing table and renamed the newly created table.

## **Create New Table**



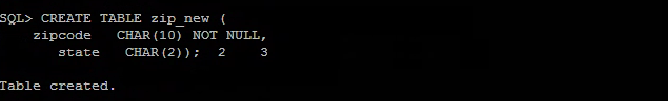
The new table zip\_new like an existing table without the column “city” is created.

SQL> CREATE TABLE zip\_new (

zipcode CHAR (10) NOT NULL,

state CHAR (2)); 2 3

Table created.

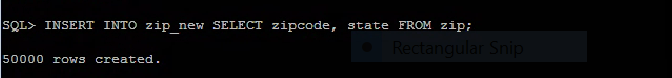


## **Populate Data into Newly Created Table**

Loaded data into the newly created table from previously existing table “zip” as follows:

SQL> INSERT INTO zip\_new SELECT zipcode, state FROM zip;

50000 rows created.

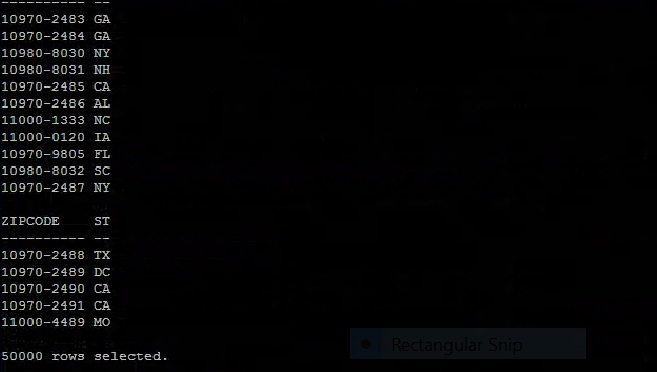


## **Validation**

Verified the values are inserted into the new table:

SQL> select \* from zip\_new;



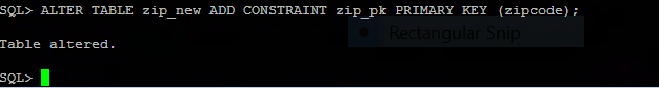


## **Reestablish Referential Integrity**

Referential integrity with primary reestablished for the newly created table as follows:

SQL> ALTER TABLE zip\_new ADD CONSTRAINT zip\_pk PRIMARY KEY (zipcode);

Table altered.



## **Drop Existing Table**

The previously existing table “zip” dropped as follows:

SQL> DROP TABLE zip CASCADE CONSTRAINTS;

Table dropped.



## **Rename Table**

The newly created table “zip\_new” renamed as follows:

SQL> RENAME zip\_new TO zip;

Table renamed.



# Checklist Item 3 - Design for a Test of Restructuring Existing Objects

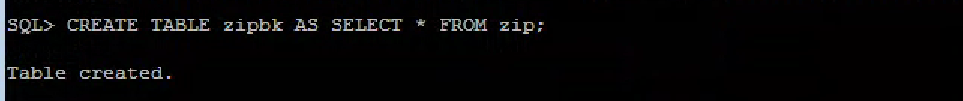
Below are important steps and SQL commands implemented to test the restructuring of existing objects in the above section.

## **Step 1) Create Backup**

Backup the existing table (create back up) as follows:

SQL> CREATE TABLE zipbk AS SELECT \* FROM zip;

Table created.



## **Step 2) Create New Table**

Create the new table zip\_new:

SQL> CREATE TABLE zip\_new(

2 zipcode CHAR (10) NOT NULL,

3 state CHAR (2));

Table created.

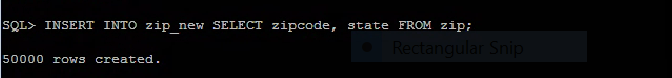


## **Step 3) Populate Data into the Newly Created Table from the Existing Table**

Loaded data into the newly created table from a previously existing table as follows:

SQL> INSERT INTO zip\_new SELECT zipcode, state FROM zip;

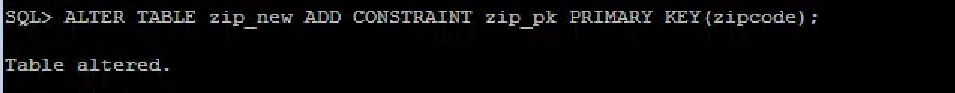
50000 rows created.



## **Step 4) Alter the Newly Created Table**

SQL> ALTER TABLE zip\_new ADD CONSTRAINT zip\_pk PRIMARY KEY (zipcode);

Table altered.

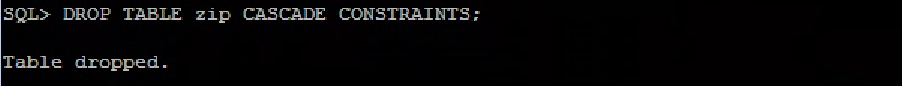


## **Step 5) Drop Existing Table**

The previously existing table is dropped.

SQL> DROP TABLE zip CASCADE CONSTRAINTS;

Table dropped.



## **Step 6) Rename Newly Created Table**

Renamed the newly created table to the original one as follows:

SQL> RENAME zip\_new TO zip;

Table renamed.



## **Step 7) Confirm the Removal of the Existing Table**

SQL> DESC zip;

Name Null? Type

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

ZIPCODE NOT NULL CHAR (10)

STATE CHAR (2)



# Checklist Items 4-5 – Static Performance Test

## **Check V$SYSSTAT**

SQL> SET ECHO ON;

SQL>

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> /\*---------------------------------------------------\*/

SQL> /\* Part 2 Checklist Items 4-5 \*/

SQL> /\*---------------------------------------------------\*/

SQL>

SQL> /\*Check v$sysstat\*/

SQL> DESC v$sysstat;

Name Null? Type

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

STATISTIC# NUMBER

NAME VARCHAR2(64)

CLASS NUMBER

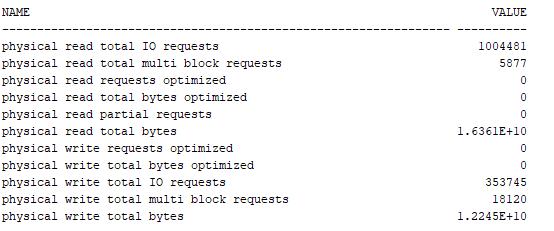
VALUE NUMBER

STAT\_ID NUMBER

CON\_ID NUMBER

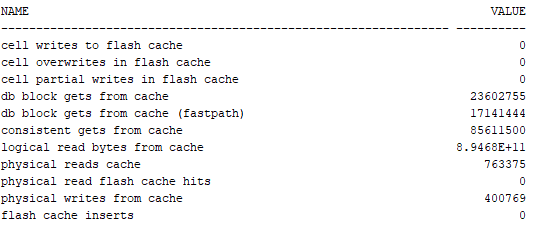
SQL>

SQL> SELECT name,value FROM v$sysstat WHERE name like 'physical%';



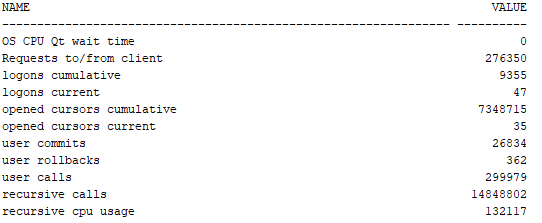
SQL>

SQL> SELECT name,value FROM v$sysstat WHERE name like '%cache%';



SQL>

SQL> SELECT name,value FROM v$sysstat;



## **Read Load Results**

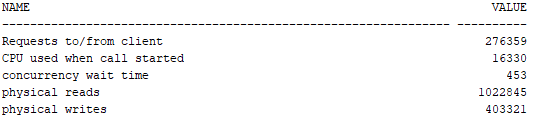
SQL>

SQL>

SQL> /\*Run five stats, then read load, then five stats again\*/

SQL> /\*Check Stats\*/

SQL> SELECT name, value FROM V$sysstat where name IN ('CPU used when call started', 'physical reads', 'physical writes', 'concurrency wait time', 'Requests to/from client');



SQL> /\*Read Load - Same Command is ran 200 times to generate a intense load that takes at least 60 seconds.\*/

SQL>

SQL> SELECT z1.zipcode,

2 (SELECT count(c1.zipcode) FROM DataDesignLeadUser.CUSTOMER c1 WHERE c1.zipcode = z1.zipcode AND c1.zipcode =

3 (SELECT c0.zipcode FROM DataDesignLeadUser.CUSTOMER c0 WHERE c0.customerid = c1.customerid)) cust\_zip\_usage,

4 (SELECT count(e0.zipcode) FROM DataDesignLeadUser.EMPLOYEE e0 WHERE e0.zipcode = z1.zipcode AND e0.zipcode =

5 (SELECT e2.zipcode FROM DataDesignLeadUser.EMPLOYEE e2 WHERE e2.sales\_emp\_ssn = e0.sales\_emp\_ssn)) emp\_zip\_usage,

6 (SELECT avg(p1.payment\_amount) FROM DataDesignLeadUser.PAYMENT p1 WHERE p1.orderid = o1.orderid) payment\_avg,

7 (SELECT sum(p0.prod\_unit\_price) FROM DataDesignLeadUser.PRODUCT p0 WHERE p0.productid = pr1.productid) unit\_cost\_sum,

8 (SELECT count(z0.zipcode) FROM DataDesignLeadUser.ZIP z0 WHERE z0.zipcode = z1.zipcode) random\_zip,

9 (SELECT count(o0.orderid) FROM DataDesignLeadUser.ORDERS o0 WHERE o0.orderid = o1.orderid) order\_count,

10 (SELECT count(s0.storeid) FROM DataDesignLeadUser.STORES s0 WHERE s0.storeid = s1.storeid) store\_count,

11 (SELECT count(r0.regionid) FROM DataDesignLeadUser.REGION r0 WHERE r0.regionid = r1.regionid) region\_count,

12 (SELECT count(oi0.orderid) FROM DataDesignLeadUser.ORDER\_ITEM oi0 WHERE oi0.orderid = o1.orderid) order\_item\_count,

13 (SELECT count(pc0.prodcatid) FROM DataDesignLeadUser.PRODUCT\_CAT pc0 WHERE pc0.prodcatid = pc1.prodcatid) prod\_cat\_count

14 FROM DataDesignLeadUser.ZIP z1

15 FULL JOIN DataDesignLeadUser.CUSTOMER c2 ON c2.zipcode = z1.zipcode

16 FULL JOIN DataDesignLeadUser.ORDERS o1 ON o1.customerid = c2.customerid

17 FULL JOIN DataDesignLeadUser.PAYMENT p2 ON p2.orderid = o1.orderid

18 FULL JOIN DataDesignLeadUser.ORDER\_ITEM oi1 ON oi1.orderid = o1.orderid

19 FULL JOIN DataDesignLeadUser.PRODUCT pr1 ON pr1.productid = oi1.productid

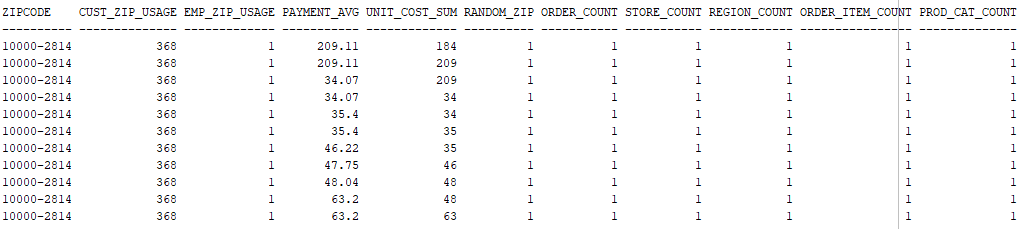
20 FULL JOIN DataDesignLeadUser.PRODUCT\_CAT pc1 ON pc1.prodcatid = pr1.prodcatid

21 FULL JOIN DataDesignLeadUser.EMPLOYEE e1 ON e1.sales\_emp\_ssn = o1.sales\_emp\_ssn

22 FULL JOIN DataDesignLeadUser.STORES s1 ON s1.storeid = e1.storeid

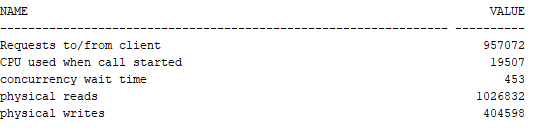
23 FULL JOIN DataDesignLeadUser.REGION r1 ON r1.regionid = s1.regionid

24 ORDER BY cust\_zip\_usage DESC;



SQL> /\*Check Stats\*/

SQL> SELECT name, value FROM V$sysstat where name IN ('CPU used when call started', 'physical reads', 'physical writes', 'concurrency wait time', 'Requests to/from client');



SQL> /\*Flush shared pool before write load\*/

SQL> ALTER SYSTEM flush shared\_pool;

System FLUSH altered.

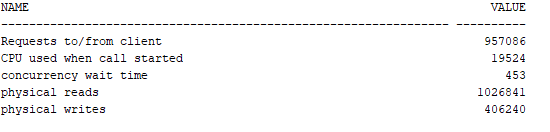
## **Write Load Results**

SQL> /\*Run five stats, then write load, then five stats again\*/

SQL>

SQL> /\*Check Stats\*/

SQL> SELECT name, value FROM V$sysstat where name IN ('CPU used when call started', 'physical reads', 'physical writes', 'concurrency wait time', 'Requests to/from client');



SQL>

SQL> /\*Drop/Create region backup table\*/

SQL> DROP TABLE regionbackup;

Error starting at line : 4,646 in command -

DROP TABLE regionbackup

Error report -

ORA-00942: table or view does not exist

00942. 00000 - "table or view does not exist"

\*Cause:

\*Action:

SQL>

SQL> CREATE TABLE regionbackup (

2 regionid NUMBER(10) NOT NULL,

3 regionname VARCHAR(25),

4 description VARCHAR(150),

5 CONSTRAINT pk\_region PRIMARY KEY ( regionid )

6 USING INDEX TABLESPACE user\_indx1

7 STORAGE ( INITIAL 50 K NEXT 10 K )

8 )

9 PCTFREE 5 PCTUSED 90 TABLESPACE user\_data1

10 STORAGE ( INITIAL 192 k NEXT 20 k MAXEXTENTS UNLIMITED PCTINCREASE 0 );

Table REGIONBACKUP created.

SQL>

SQL> /\*Insert current data\*/

SQL> INSERT INTO regionbackup SELECT \* FROM DataDesignLeadUser.region;

1,000 rows inserted.

SQL>

SQL>

SQL> /\*Create loop to insert from 1,001 to 1,000,000 additional rows\*/

SQL> BEGIN

2 FOR v\_LoopCounter IN 1001..1000000 LOOP

3 INSERT INTO REGIONBACKUP (regionid,regionname, description)

4 VALUES (v\_LoopCounter, 'name', 'description');

5 END LOOP;

6 END;

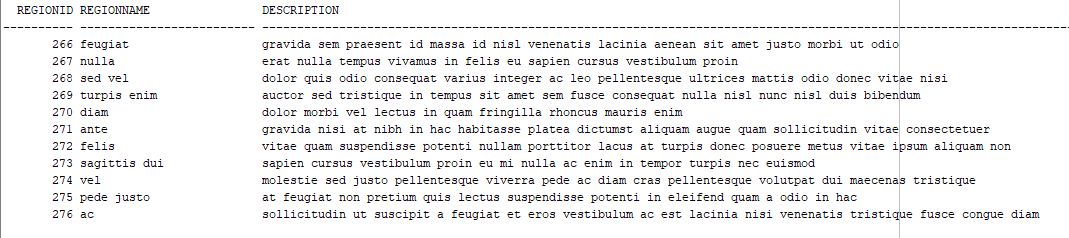
7 /

PL/SQL procedure successfully completed.

SQL> /\*Verify data is there\*/

SQL>

SQL> SELECT \* FROM REGIONBACKUP;



SQL>

SQL> /\*Run loop to delete each record individually\*/

SQL>

SQL> declare

2 i number := 1;

3 cursor s1 is SELECT rowid, r.\* FROM REGIONBACKUP r;

4 begin

5 for c1 in s1 loop

6 DELETE FROM REGIONBACKUP

7 WHERE regionid = i;

8 i := i + 1;

9 end loop;

10 commit;

11 end;

12 /

PL/SQL procedure successfully completed.

SQL>

SQL> /\*Verify data is Gone\*/

SQL> SELECT \* FROM REGIONBACKUP;

no rows selected

SQL>

SQL> /\*Do it again!\*/

SQL>

SQL> /\*Drop/Create region backup table\*/

SQL> DROP TABLE regionbackup;

Table REGIONBACKUP dropped.

SQL>

SQL> CREATE TABLE regionbackup (

2 regionid NUMBER(10) NOT NULL,

3 regionname VARCHAR(25),

4 description VARCHAR(150),

5 CONSTRAINT pk\_region PRIMARY KEY ( regionid )

6 USING INDEX TABLESPACE user\_indx1

7 STORAGE ( INITIAL 50 K NEXT 10 K )

8 )

9 PCTFREE 5 PCTUSED 90 TABLESPACE user\_data1

10 STORAGE ( INITIAL 192 k NEXT 20 k MAXEXTENTS UNLIMITED PCTINCREASE 0 );

Table REGIONBACKUP created.

SQL>

SQL> /\*Insert current data\*/

SQL> INSERT INTO regionbackup SELECT \* FROM DataDesignLeadUser.region;

1,000 rows inserted.

SQL>

SQL> /\*Create loop to insert from 1,001 to 1,000,000 additional rows\*/

SQL> BEGIN

2 FOR v\_LoopCounter IN 1001..1000000 LOOP

3 INSERT INTO REGIONBACKUP (regionid,regionname, description)

4 VALUES (v\_LoopCounter, 'name', 'description');

5 END LOOP;

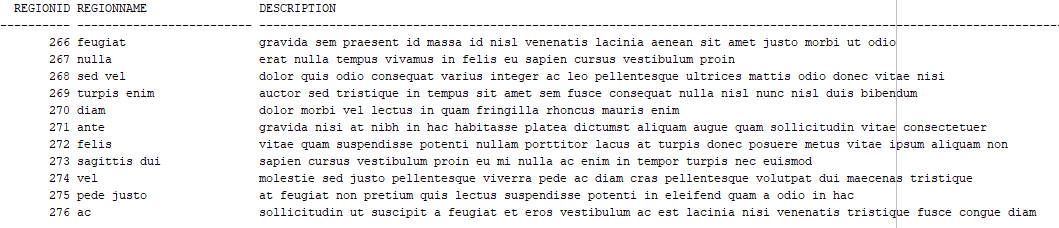
6 END;

7 /

PL/SQL procedure successfully completed.

SQL> /\*Verify data is there\*/

SQL> SELECT \* FROM REGIONBACKUP;



SQL>

SQL> /\*Run loop to delete each record individually\*/

SQL>

SQL> declare

2 i number := 1;

3 cursor s1 is SELECT rowid, r.\* FROM REGIONBACKUP r;

4 begin

5 for c1 in s1 loop

6 DELETE FROM REGIONBACKUP

7 WHERE regionid = i;

8 i := i + 1;

9 end loop;

10 commit;

11 end;

12 /

PL/SQL procedure successfully completed.

SQL>

SQL> /\*Verify data is Gone\*/

SQL> SELECT \* FROM REGIONBACKUP;

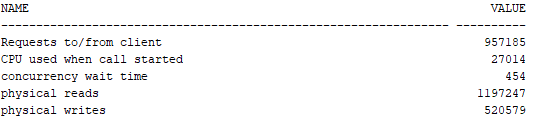
no rows selected

SQL>

SQL>

SQL> /\*Check Stats\*/

SQL> SELECT name, value FROM V$sysstat where name IN ('CPU used when call started', 'physical reads', 'physical writes', 'concurrency wait time', 'Requests to/from client');



# Checklist Item 6 – Dynamic Performance Test

## **Read Load Results**

This section follows lab 11, using the Automatic Database Diagnostic Monitor (ADDM) utility to run dynamic performance tests. A read and write loads were created to demonstrate the capabilities of this tool. First, we log in as an SYSDBA and create an initial report, see figure 1.

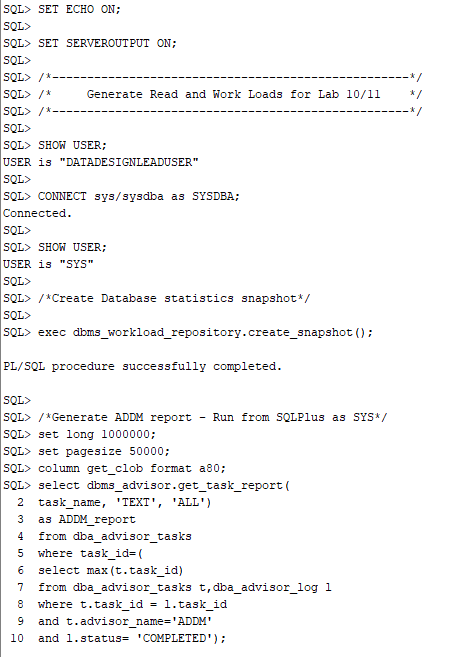


Figure 1. The ADDM report code execution.

We can see from the report, see figure 2, there are currently no findings.

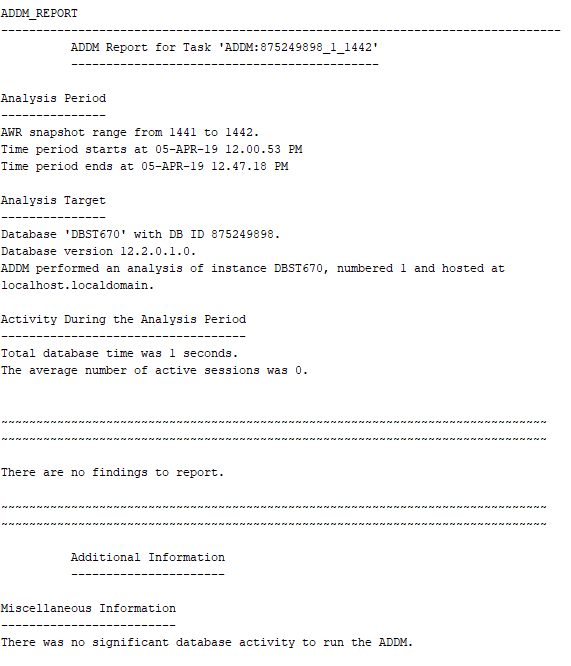


Figure 2. Initial ADDM report.

We then run an overly complex SELECT statement with multiple layers of correlated subqueries and full joins on the entire database. This statement is then repeated 200 times to create at least a 60-second database read load, see figure 3.

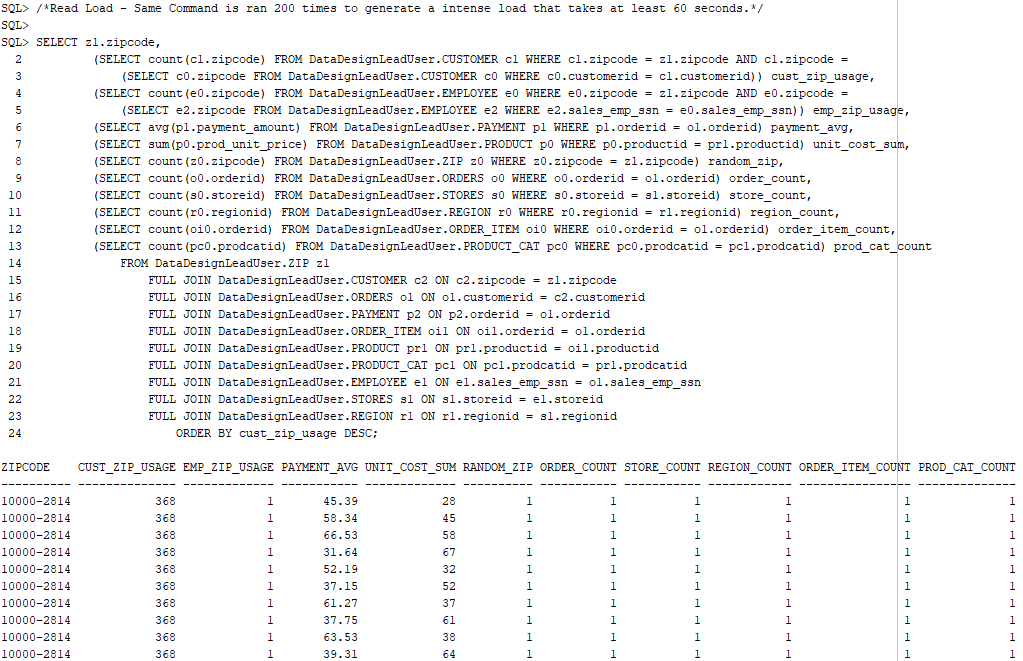


Figure 3. Read load SQL SELECT statement.

We then take another snapshot and run our ADDM report again see figure 4.

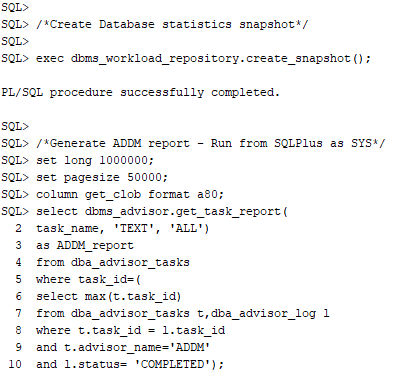


Figure 4. The ADDM report code execution.

The report, see figure 5, shows our database ran for 62 seconds and there were still no findings or recommended actions to improve performance.

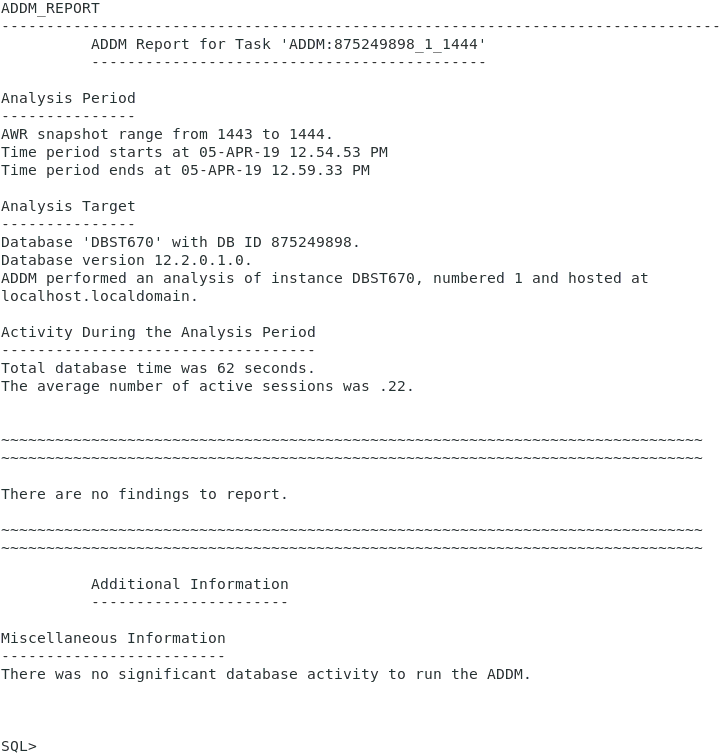


Figure 5. The ADDM report with no findings from the read load.

## **Write Load Results**

The next test runs the same ADDM reports, but this time with a write load. First, we generate the report, see figure 6.

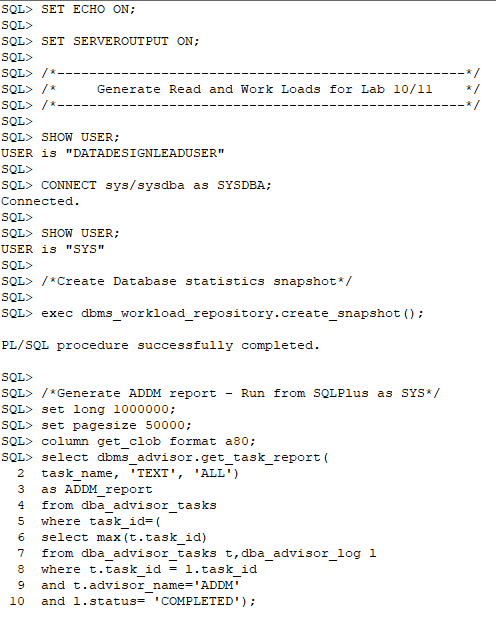


Figure 6. The ADDM report code execution.

The ADDM report, see figure 7, shows no current findings or recommendations.

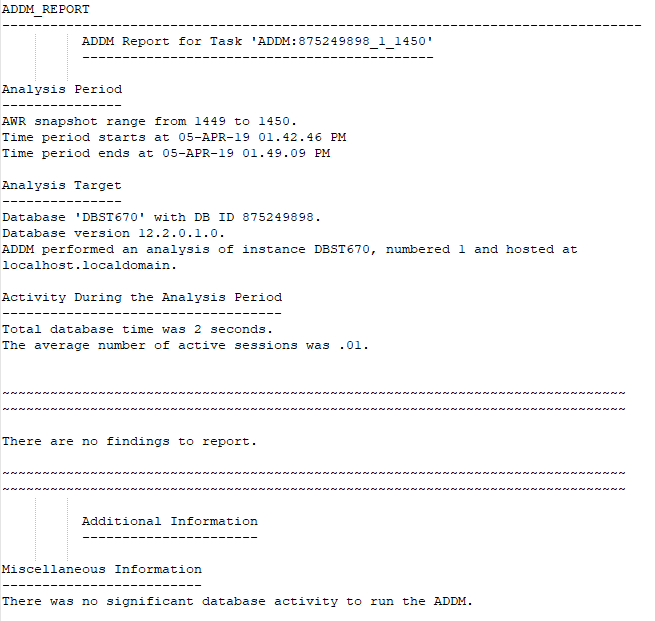


Figure 7. The ADDM report.

We can now apply the write load. For this test, we create a new table and populate it with data from the REGION table. We then run a loop to add additional rows of data to the table with 1 million rows in total. After verifying the data is there, we purposely create an inefficient DELETE loop to remove each row once per loop cycle. After this entire process is complete, we run through the entire process a second time to further increase the load. See figures 8,9 and 10.

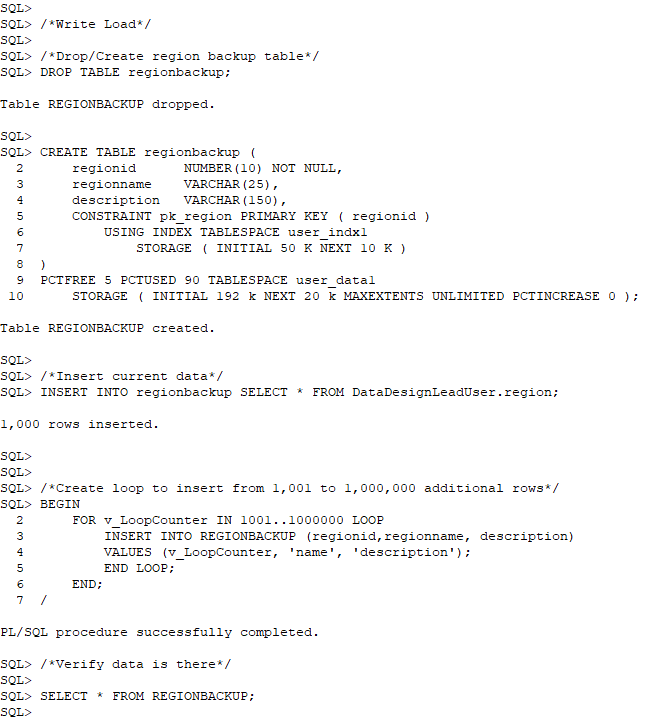


Figure 8. Create the write load table and insert 1 million rows of data.

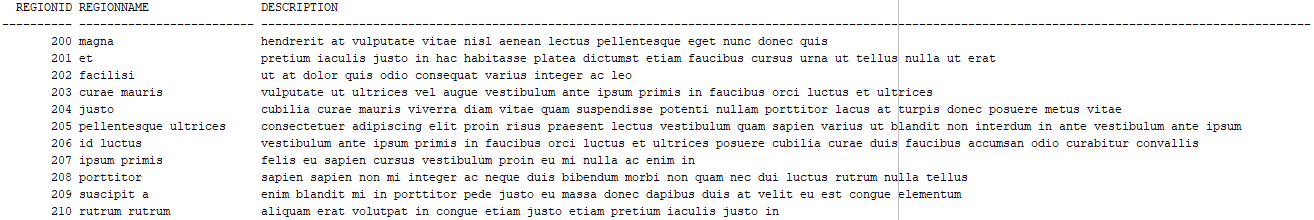


Figure 9. Verify the data is in the table.

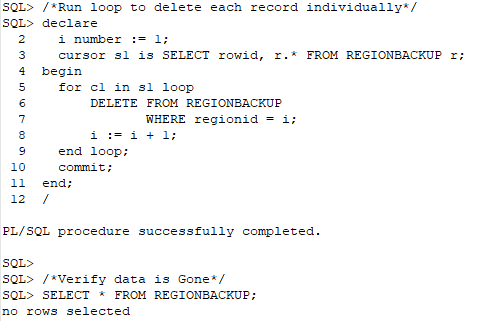


Figure 10. Run expensive loop operation to delete each record.

After this is complete, we run our ADDM report again, see figure 11.

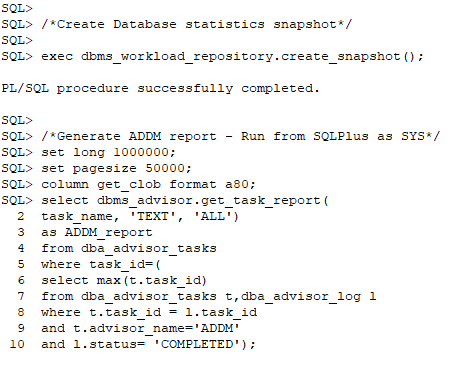


Figure 11. The ADDM report code execution.

The ADDM report, see figures 12 through 15, highlights five findings. It found an issue with the sequence cache, inefficient SQL statements, slow archivers, and some unusual wait events.

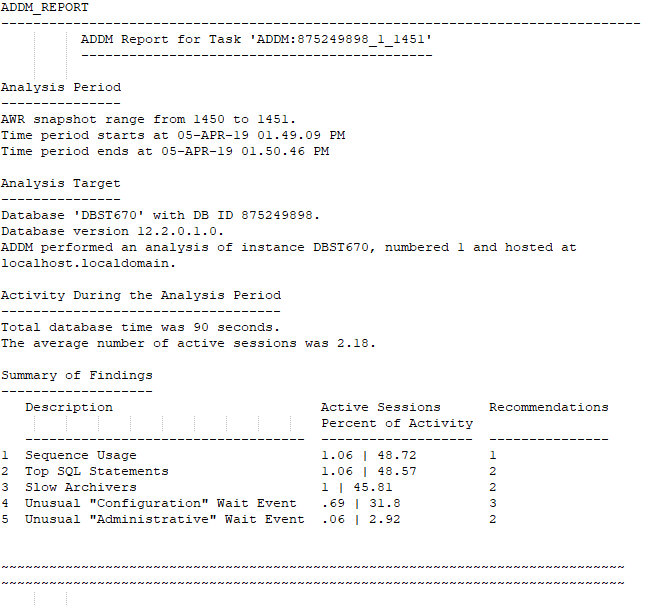


Figure 12. The ADDM report summary.

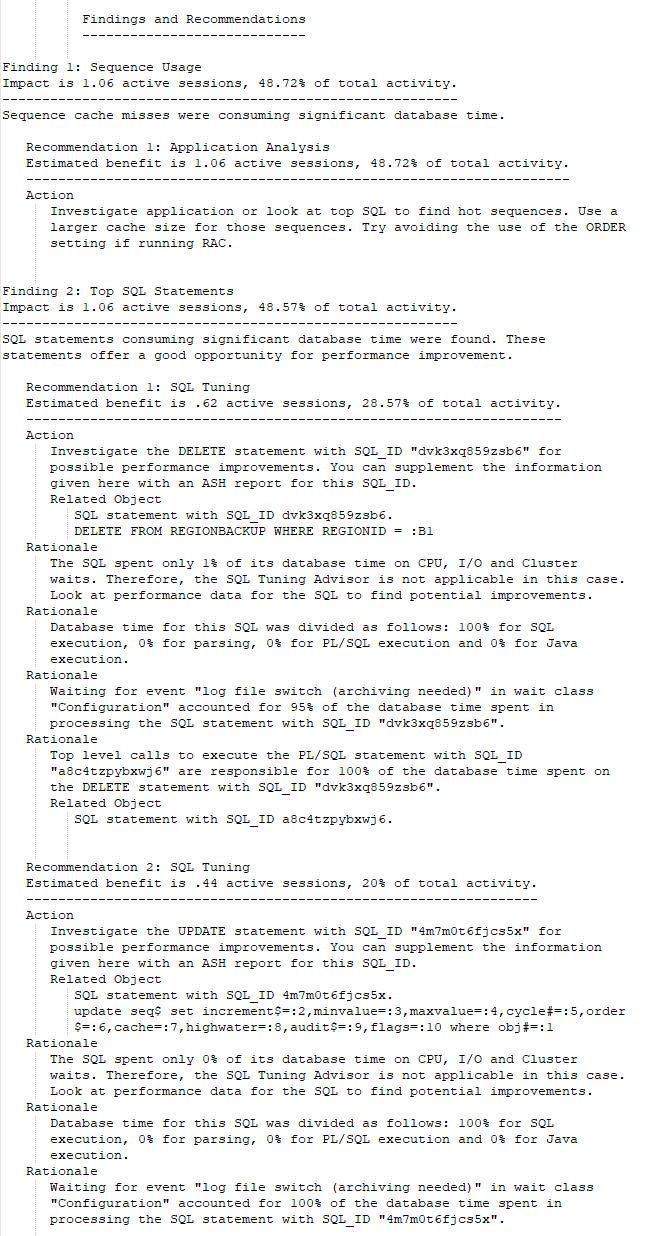


Figure 13. The ADDM report findings 1 and 2.

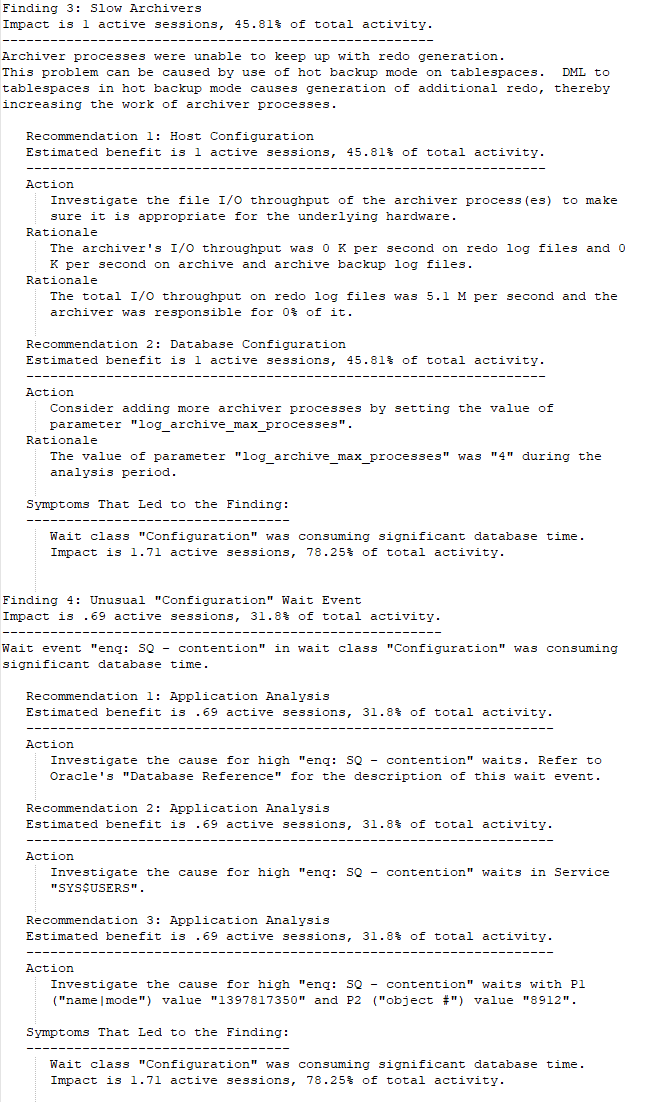


Figure 14. The ADDM report findings 3 and 4.

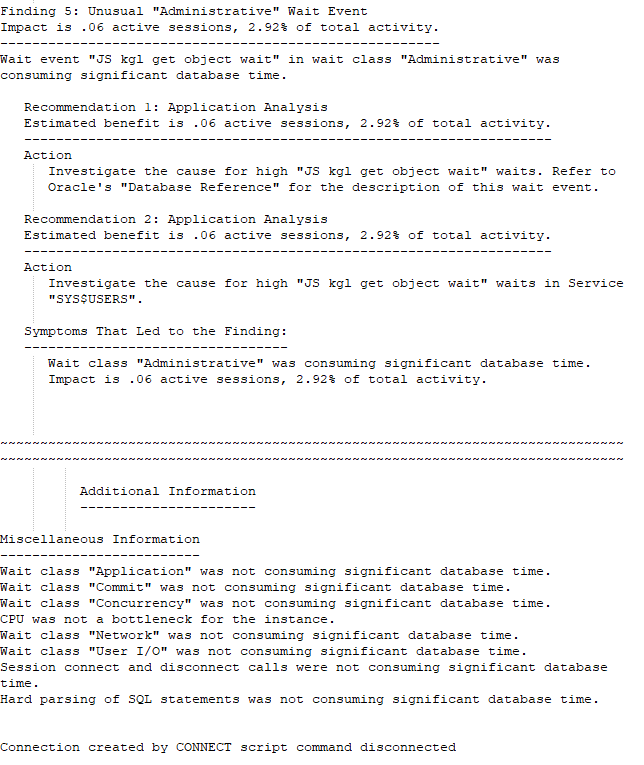


Figure 15. The ADDM report finding 5 and miscellaneous information.

For this project, we will focus on finding 2 recommendation 1 for tuning the DELETE statement. To correct the issue, we comment out the DELETE loop and add a single DELETE statement to delete all rows in the table. We then run a new ADDM report, see figure 16.

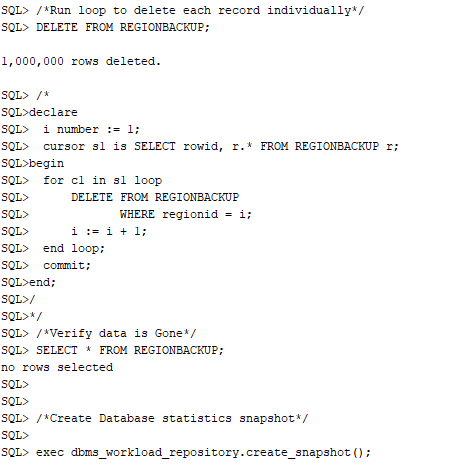
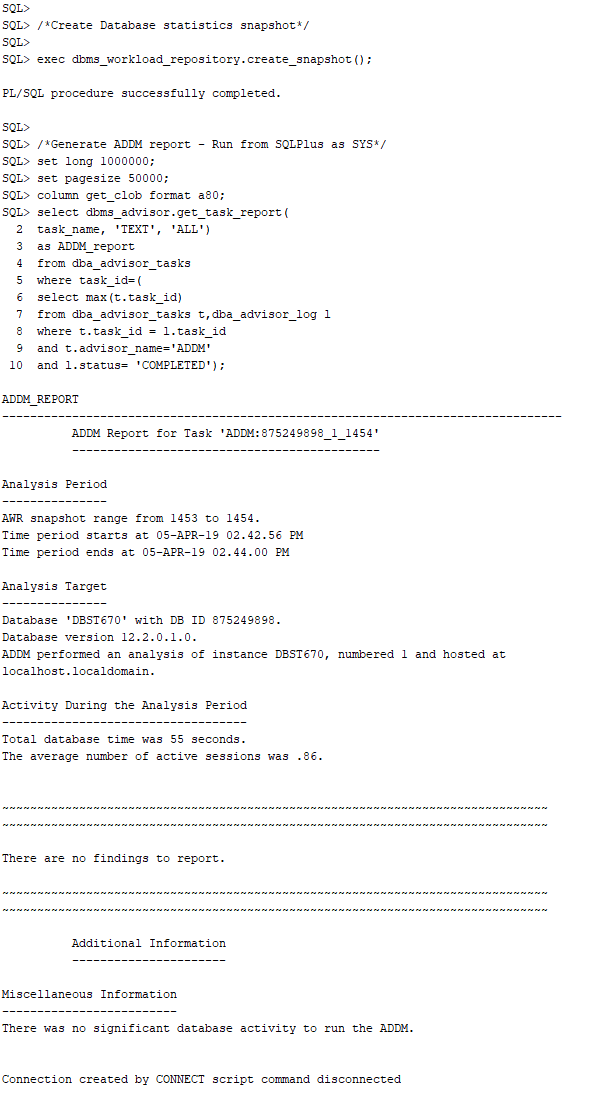


Figure 16. The ADDM report code execution.

As we can note in figure 17, our database time was reduced from 90 seconds to 55 seconds. Additionally, there are no findings or performance recommendations. This test highlights the capabilities of the ADDM utility and how we can use it to find performance improvements in our database.



# Checklist Item 7 – Performance Test Matrix

|  |  |  |
| --- | --- | --- |
| Static Test 1 – Requests to/from client | Read | Write |
| Before: 276359  After: 957072 | Before: 957086  After: 957185 |
| Static Test 2 – CPU used when call started | Before: 16330  After: 19507 | Before: 19524  After: 27014 |
| Static Test 3 – Concurrency wait time | Before: 453  After: 453 | Before: 453  After: 454 |
| Static Test 4 – physical reads | Before: 1022845  After: 1026832 | Before: 1026841  After: 1197247 |
| Static Test 5 – physical writes | Before: 403321  After: 404598 | Before: 406240  After: 520579 |
| Dynamic ADDM Read Test | There are no findings to report. | |
| Dynamic ADDM Write Test | A number of issues were found, to include sequence usage, SQL statement inefficiencies, a slow archiver, and unusual wait events. Focusing on the SQL statement inefficiencies, we found an inefficient DELETE loop causing performance issues. After replacing this with a single DELETE DML statement performance improved significantly, specifically the total database time was reduced from 90 to 55 seconds. | |

# Checklist Items 8-12 - Backup and Recovery

## **Create Recovery Table – Checklist 8**

SQL>

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> /\*---------------------------------------------------\*/

SQL> /\* DROP OBJECTS \*/

SQL> /\*---------------------------------------------------\*/

SQL>

SQL> SHOW USER;

USER is "DATADESIGNLEADUSER"

SQL>

SQL> CONNECT DataDesignLeadUser/abc123;

Connected.

SQL>

SQL> SHOW USER;

USER is "DATADESIGNLEADUSER"

SQL>

SQL> /\*Drop Table\*/

SQL>

SQL> DROP TABLE product\_cat\_backup;

Table PRODUCT\_CAT\_BACKUP dropped.

SQL>

SQL>

SQL> /\*-------------------------------------------------------------------\*/

SQL> /\*A script to create a small table with 3 rows for recovery testing.\*/

SQL> /\*-----------------------------------------------------------------\*/

SQL>

SQL> /\*Create Table\*/

SQL>

SQL> CREATE TABLE product\_cat\_backup (

2 prodcatid NUMBER(10) NOT NULL,

3 prodcattype VARCHAR(25),

4 description VARCHAR(150),

5 CONSTRAINT pk\_prodcat\_backup PRIMARY KEY ( prodcatid )

6 USING INDEX TABLESPACE user\_indx1

7 STORAGE ( INITIAL 50 K NEXT 10 K )

8 )

9 PCTFREE 5 PCTUSED 90 TABLESPACE user\_data1

10 STORAGE ( INITIAL 192 k NEXT 20 k MAXEXTENTS UNLIMITED PCTINCREASE 0 );

Table PRODUCT\_CAT\_BACKUP created.

SQL>

SQL> /\*Insert 3 rows of data from original product category table\*/

SQL>

SQL> INSERT INTO product\_cat\_backup SELECT \* FROM product\_cat WHERE PRODCATID <= 3;

3 rows inserted.

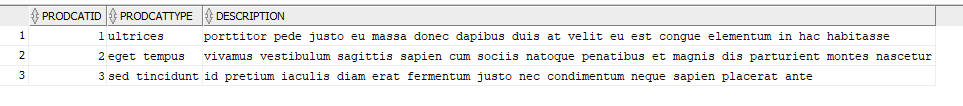
SQL>

SQL> commit;

Commit complete.

SQL> /\*Verify data is there\*/

SQL> SELECT \* FROM product\_cat\_backup;



Connection created by CONNECT script command disconnected

## **Export/Import – Checklist 9 & 10**

First run the export command: “exp owner=DataDesignLeadUser file=DataDesignLeadUser.exp statistics=none*,”* see figure 1.



Figure 1. Example of exporting data from the database.

Return to the script and drop the table, and then confirm it is dropped:

SQL> /\*------------------------------------------------------------------------------------------\*/

SQL> /\*Backup and recovery commands for Recovery testing using export/import and RMAN utilities.\*/

SQL> /\*----------------------------------------------------------------------------------------\*/

SQL>

SQL> /\*Run exp command from Unix command line then return to drop table\*/

SQL> DROP TABLE product\_cat\_backup;

Table PRODUCT\_CAT\_BACKUP dropped.

SQL> SELECT \* FROM product\_cat\_backup;

Error starting at line : 53 in command -

SELECT \* FROM product\_cat\_backup

Error at Command Line : 53 Column : 15

Error report -

SQL Error: ORA-00942: table or view does not exist

00942. 00000 - "table or view does not exist"

\*Cause:

\*Action:

SQL>

SQL> /\*Drop other objects in schema for cleaner import output\*/

SQL> /\*Drop Indexes\*/

SQL>

SQL> DROP INDEX fk\_zip;

Index FK\_ZIP dropped.

SQL>

SQL> DROP INDEX fk\_customerid;

Index FK\_CUSTOMERID dropped.

SQL>

SQL> DROP INDEX fk\_order\_id1;

Index FK\_ORDER\_ID1 dropped.

SQL>

SQL> DROP INDEX fk\_product\_id;

Index FK\_PRODUCT\_ID dropped.

SQL>

SQL> DROP INDEX fk\_order\_id2;

Index FK\_ORDER\_ID2 dropped.

SQL>

SQL> DROP INDEX fk\_emp\_storeid;

Index FK\_EMP\_STOREID dropped.

SQL>

SQL> DROP INDEX fk\_emp\_zip;

Index FK\_EMP\_ZIP dropped.

SQL>

SQL> DROP INDEX fk\_store\_region;

Index FK\_STORE\_REGION dropped.

SQL>

SQL> DROP INDEX fk\_store\_zip;

Index FK\_STORE\_ZIP dropped.

SQL>

SQL> DROP INDEX fk\_prodcatid;

Index FK\_PRODCATID dropped.

SQL>

SQL> DROP INDEX fk\_emp\_ssn;

Index FK\_EMP\_SSN dropped.

SQL>

SQL> /\*Drop Tables\*/

SQL>

SQL> DROP TABLE payment;

Table PAYMENT dropped.

SQL>

SQL> DROP TABLE order\_item;

Table ORDER\_ITEM dropped.

SQL>

SQL> DROP TABLE orders;

Table ORDERS dropped.

SQL>

SQL> DROP TABLE product;

Table PRODUCT dropped.

SQL>

SQL> DROP TABLE customer;

Table CUSTOMER dropped.

SQL>

SQL> DROP TABLE employee;

Table EMPLOYEE dropped.

SQL>

SQL> DROP TABLE stores;

Table STORES dropped.

SQL>

SQL> DROP TABLE zip;

Table ZIP dropped.

SQL>

SQL> DROP TABLE region;

Table REGION dropped.

SQL>

SQL> DROP TABLE product\_cat;

Table PRODUCT\_CAT dropped.

Now, we can run the import command: “imp file=DataDesignLeadUser.exp fromuser=DataDesignLeadUser touser=DataDesignLeadUser,” see figure 2.

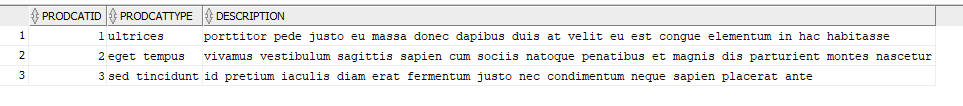


Figure 2. Example of importing data to the database.

From the SQL Script verify the table has returned:

SQL> /\*Run imp command from Unix command line then return to view restored table\*/

SQL> SELECT \* FROM product\_cat\_backup;



## **RMAN – Checklist 9 & 10**

From the command line, we can use RMAN to back up the database, see figure 3.

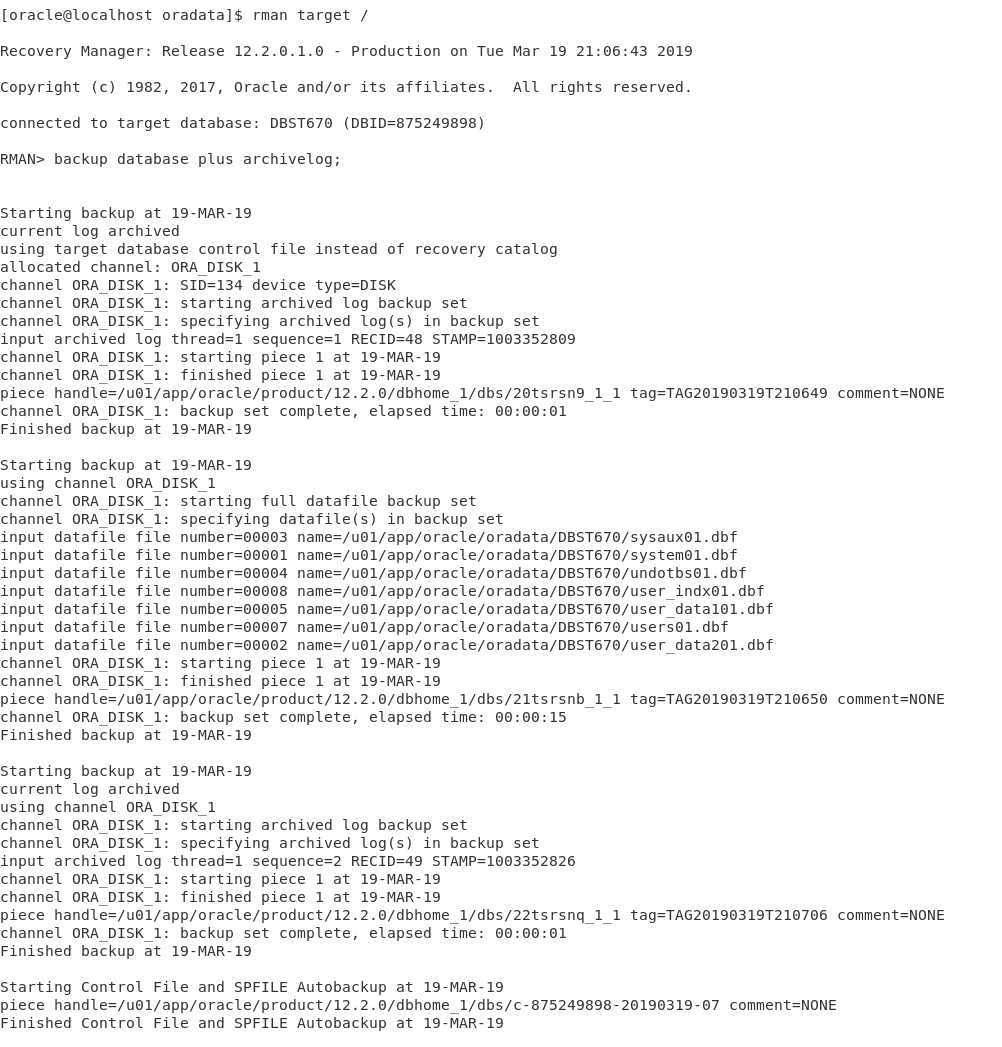


Figure 3. Creating a database back up with RMAN.

We can then list our backups to see what was accomplished, see figure 4.

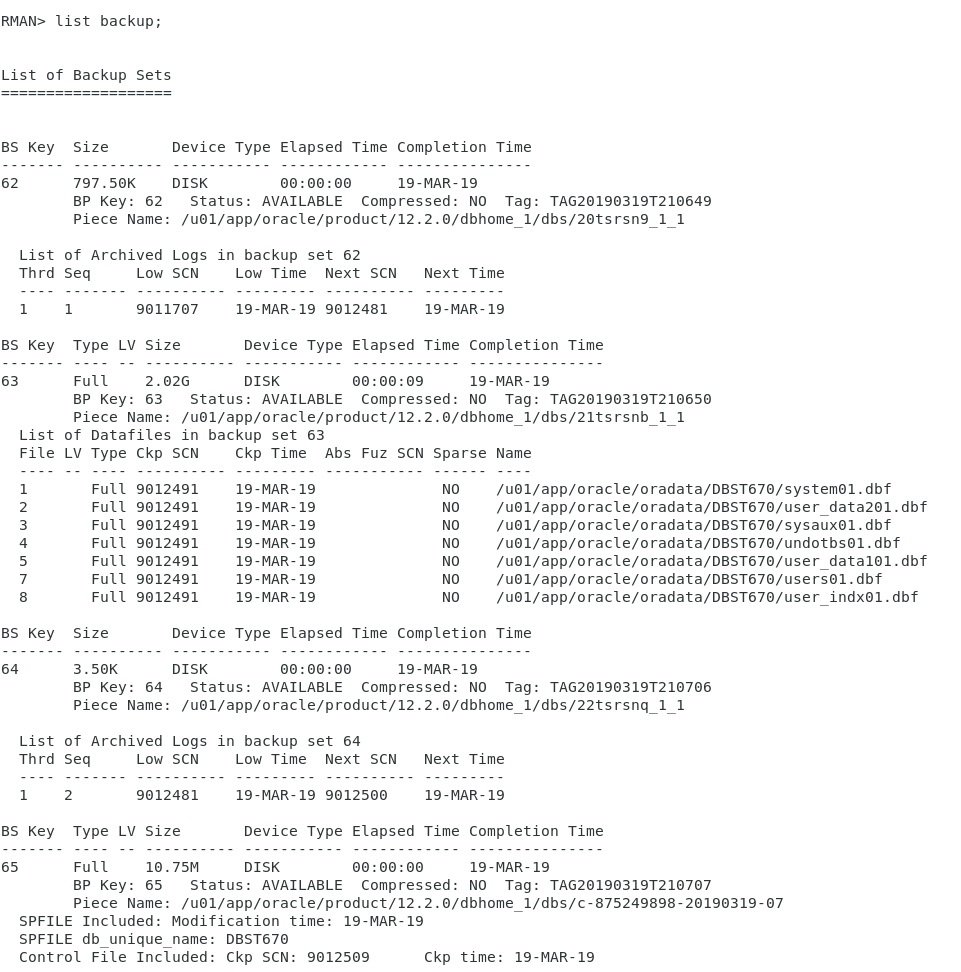


Figure 4. List the existing backups.

We drop the recovery table and check the scn number to ensure it moved past the backup scn number we made before.

SQL> /\*Connect to RMAN from Unix Command Line. Create backup. Return and drop table\*/

SQL> DROP TABLE product\_cat\_backup;

Table PRODUCT\_CAT\_BACKUP dropped.

SQL> /\*Check the current scn, it should be greater than the backup scn we made at this point\*/

SQL> SELECT current\_scn FROM v$database; /\*Run as system\*/



To begin the recovery process, we start the database in a force mount condition, restore the database, recover the database up to the scn number of our original backup, and then open the database with reset logs, see figure 5.

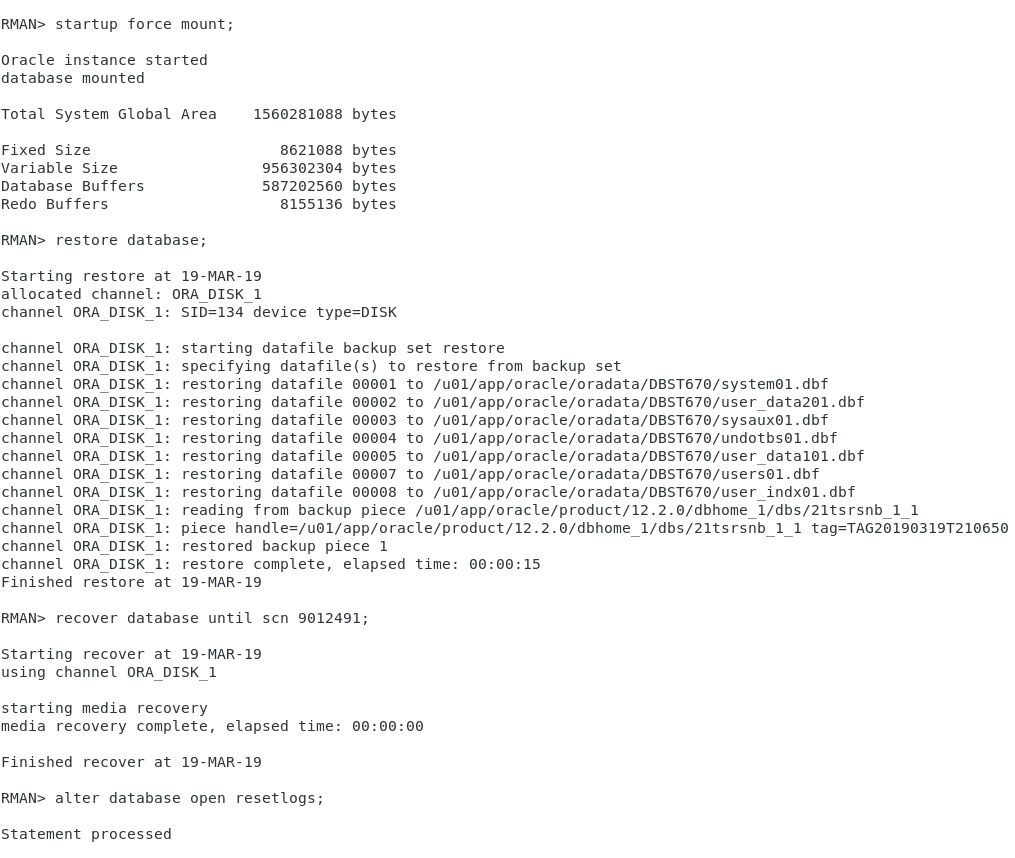
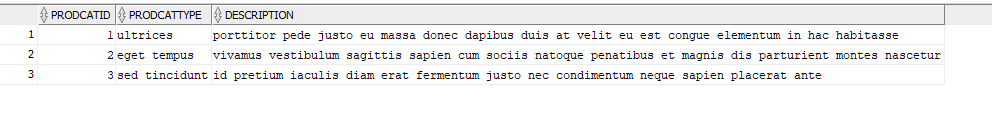


Figure 5. Run a database restore to a specific SCN number in RMAN.

From the SQL script, verify the table and data has returned.

SQL> /\*Restore to point in time and check Status with the select statement.\*/

SQL> SELECT \* FROM product\_cat\_backup;



## **RMAN Versus Import/Export – Checklist 11**

There are distinct differences between how RMAN works and the Import and Export Commands work. The best way to differentiate between the two is to consider RMAN a physical backup and Import/Export a logical backup. When we run an RMAN backup we are creating a complete backup of all the database physical components, i.e. data and control files. This method provides many benefits, such as providing either consistent or inconsistent backup options, backups of the archived redo logs, and automatic back up features. With this method, we can do a complete or incremental backup and restores from a point-in-time based off actual time or a System Change Number (SCN). If there is a complete database failure, this would be the preferred backup solution as it has everything you would need to fully restore your data (15 Backup and Recovery, 2019).

The Import/Export functions are considered a logical backup. When we use the export command, we are taking a copy at that time of specific database objects, such as tables and indexes, and placing them in a binary-formatted dump file. This is a one time issued command and does not provide the automated incremental backup features seen with RMAN. Additionally, while we may have a copy of critical table data at this point, we do not have a complete back up of the database. This form of back up is typically considered supplemental to another. If we relied on exported data files, we would have to rebuild the database manually and then import the data (20 Original Export and Import, 2019).

## **Flashback – Checklist 12**

For this project, we will use restore points to flashback the database to. First, we connect to RMAN and create a restore point, see figure 6.

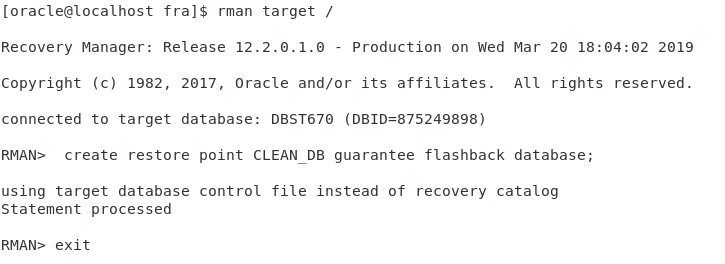


Figure 6. Create a restore point in RMAN.

Then, verify the restore point was created.

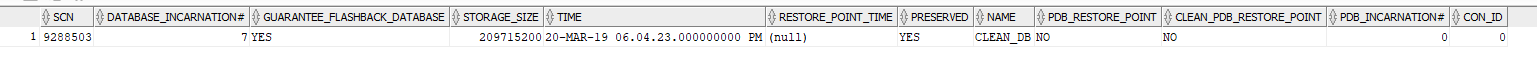
SQL> /\*-----------------------------\*/

SQL> /\*Flashback recovery commands.\*/

SQL> /\*---------------------------\*/

SQL> /\*Create restore point in RMAN, then view it with this select statement\*/

SQL> SELECT \* FROM v$restore\_point;



Drop the table:

SQL> /\*Drop table\*/

SQL> DROP TABLE product\_cat\_backup;

Table PRODUCT\_CAT\_BACKUP dropped.

We can now run the restoration steps, see figure 7.

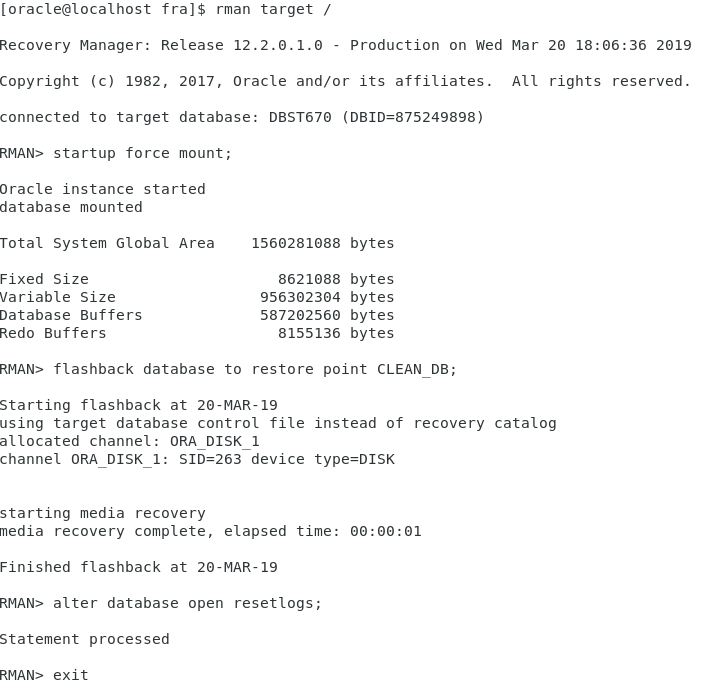
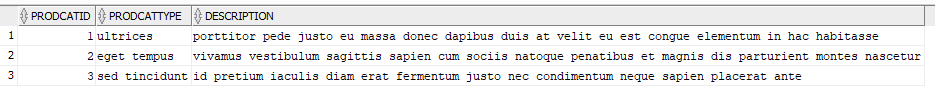


Figure 7. Restoring database to restore point with flashback in RMAN.

Now, we run our select statement again to verify the dropped table has returned:

SQL> /\*Use flashback to restore point then verify table has returned\*/

SQL> SELECT \* FROM product\_cat\_backup;



# Checklist Items 13-19 – Monitor Health, Create Users & Synonyms, & Reverse Engineer

## **Monitor Health of Database – Items 13-15**

SQL>

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> /\*---------------------------------------------------\*/

SQL> /\* Part 2 Checklist Items 13-19 \*/

SQL> /\*---------------------------------------------------\*/

SQL>

SQL> SHOW USER;

USER is "DBAUSER"

SQL>

SQL> CONNECT DBAUser/abc123;

Connected.

SQL>

SQL> SHOW USER;

USER is "DBAUSER"

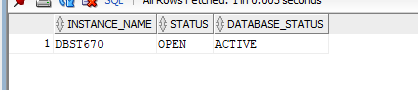
SQL>

SQL> /\*Item 13-15 - Monitor Database status\*/

SQL>

SQL> /\*Check if database status is open and active\*/

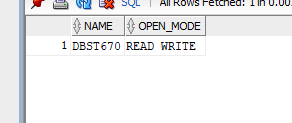
SQL> SELECT instance\_name, status, database\_status FROM V$INSTANCE;



SQL>

SQL> /\*Check if status allows you to read and write to database\*/

SQL> SELECT name, open\_mode FROM V$DATABASE;



SQL>

SQL>

SQL> /\*Get Database reserved, used, and free space size\*/

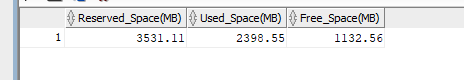
SQL> SELECT "Reserved\_Space(MB)", "Reserved\_Space(MB)" - "Free\_Space(MB)" "Used\_Space(MB)","Free\_Space(MB)"

2 FROM ( SELECT

3 (SELECT round(SUM(bytes/(1024\*1024)),2) FROM dba\_data\_files) "Reserved\_Space(MB)",

4 (SELECT round(SUM(bytes/(1024\*1024)),2) FROM dba\_free\_space) "Free\_Space(MB)"

5 FROM dual);



SQL>

SQL> /\*Show tablespace file location. Also, allocated and free space per tablespace\*/

SQL> SELECT

2 a.tablespace\_name,

3 a.file\_name,

4 round(a.bytes/(1024\*1024), 2) allocated\_MB,

5 round(b.free\_bytes/(1024\*1024),2) free\_MB

6 FROM

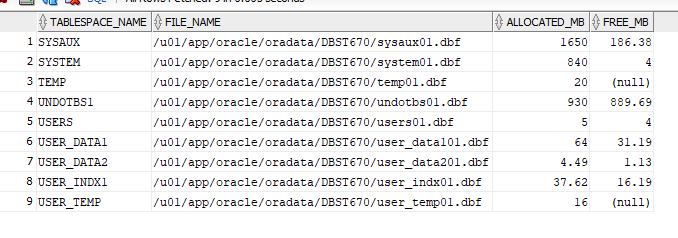
7 (SELECT tablespace\_name, file\_name, bytes

8 FROM dba\_data\_files UNION SELECT tablespace\_name, file\_name, bytes FROM dba\_temp\_files )a,

9 (SELECT tablespace\_name, SUM(bytes) free\_bytes

10 FROM dba\_free\_space b GROUP BY tablespace\_name) b

11 WHERE a.tablespace\_name=b.tablespace\_name(+) ORDER BY a.tablespace\_name;

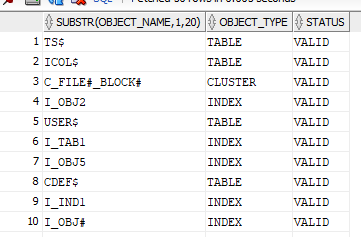


SQL>

SQL>

SQL> /\*Return all objects\*/

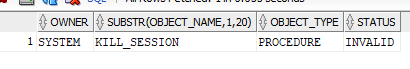
SQL> SELECT substr(object\_name,1,20), object\_type,status FROM all\_objects;



SQL>

SQL> /\*Return all invalid objects\*/

SQL> SELECT owner, substr(object\_name,1,20), object\_type, status FROM all\_objects WHERE status = 'INVALID';



## **Create Users – Item 16**

SQL>

SQL> /\*Item 16 - Create end users and give privileges\*/

SQL>

SQL> /\*Drop Users\*/

SQL>

SQL> DROP USER jkamando CASCADE;

User JKAMANDO dropped.

SQL>

SQL> DROP USER tokusanya CASCADE;

User TOKUSANYA dropped.

SQL>

SQL> DROP USER joluwadamilare CASCADE;

User JOLUWADAMILARE dropped.

SQL>

SQL> DROP USER brussell CASCADE;

User BRUSSELL dropped.

SQL>

SQL> DROP USER rseyoum CASCADE;

User RSEYOUM dropped.

SQL>

SQL> /\*---------------------------------------------------\*/

SQL> /\* CREATE USERS \*/

SQL> /\*---------------------------------------------------\*/

SQL>

SQL> CREATE USER jkamando IDENTIFIED BY abc123

2 DEFAULT TABLESPACE users

3 TEMPORARY TABLESPACE temp;

User JKAMANDO created.

SQL>

SQL> CREATE USER tokusanya IDENTIFIED BY abc123

2 DEFAULT TABLESPACE users

3 TEMPORARY TABLESPACE temp;

User TOKUSANYA created.

SQL>

SQL> CREATE USER joluwadamilare IDENTIFIED BY abc123

2 DEFAULT TABLESPACE users

3 TEMPORARY TABLESPACE temp;

User JOLUWADAMILARE created.

SQL>

SQL> CREATE USER brussell IDENTIFIED BY abc123

2 DEFAULT TABLESPACE users

3 TEMPORARY TABLESPACE temp;

User BRUSSELL created.

SQL>

SQL> CREATE USER rseyoum IDENTIFIED BY abc123

2 DEFAULT TABLESPACE users

3 TEMPORARY TABLESPACE temp;

User RSEYOUM created.

SQL>

SQL>

SQL> /\*Grant necessary permissions\*/

SQL>

SQL> GRANT connect,resource TO jkamando;

Grant succeeded.

SQL>

SQL> GRANT connect,resource TO tokusanya;

Grant succeeded.

SQL>

SQL> GRANT connect,resource TO joluwadamilare;

Grant succeeded.

SQL>

SQL> GRANT connect,resource TO brussell;

Grant succeeded.

SQL>

SQL> GRANT connect,resource TO rseyoum;

Grant succeeded.

SQL>

SQL>

SQL> /\*Alter unprivileged user tablespace\*/

SQL> ALTER USER JKamando DEFAULT TABLESPACE user\_data1 TEMPORARY TABLESPACE user\_temp;

User JKAMANDO altered.

SQL> ALTER USER TOkusanya DEFAULT TABLESPACE user\_data1 TEMPORARY TABLESPACE user\_temp;

User TOKUSANYA altered.

SQL> ALTER USER JOluwadamilare DEFAULT TABLESPACE user\_data1 TEMPORARY TABLESPACE user\_temp;

User JOLUWADAMILARE altered.

SQL> ALTER USER BRussell DEFAULT TABLESPACE user\_data1 TEMPORARY TABLESPACE user\_temp;

User BRUSSELL altered.

SQL> ALTER USER RSeyoum DEFAULT TABLESPACE user\_data1 TEMPORARY TABLESPACE user\_temp;

User RSEYOUM altered.

## **Create Synonyms – Item 17**

SQL>

SQL> /\*Item 17 - Create synonyms\*/

SQL>

SQL> /\*Create synonym for each table from DBAUser\*/

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYCUSTOMERS

2 FOR DATADESIGNLEADUSER.CUSTOMER;

SYNONYM MYCUSTOMERS created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYEMPLOYEE

2 FOR DATADESIGNLEADUSER.EMPLOYEE;

SYNONYM MYEMPLOYEE created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYORDER\_ITEM

2 FOR DATADESIGNLEADUSER.ORDER\_ITEM;

SYNONYM MYORDER\_ITEM created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYORDERS

2 FOR DATADESIGNLEADUSER.ORDERS;

SYNONYM MYORDERS created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYPAYMENT

2 FOR DATADESIGNLEADUSER.PAYMENT;

SYNONYM MYPAYMENT created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYPRODUCT

2 FOR DATADESIGNLEADUSER.PRODUCT;

SYNONYM MYPRODUCT created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYPRODUCT\_CAT

2 FOR DATADESIGNLEADUSER.PRODUCT\_CAT;

SYNONYM MYPRODUCT\_CAT created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYREGION

2 FOR DATADESIGNLEADUSER.REGION;

SYNONYM MYREGION created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYSTORES

2 FOR DATADESIGNLEADUSER.STORES;

SYNONYM MYSTORES created.

SQL> CREATE OR REPLACE PUBLIC SYNONYM MYZIP

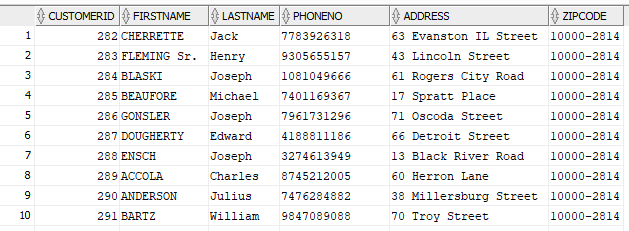
2 FOR DATADESIGNLEADUSER.ZIP;

SYNONYM MYZIP created.

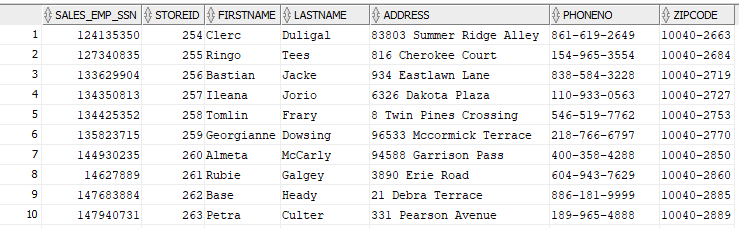
SQL>

SQL> /\*Verify synonym works\*/

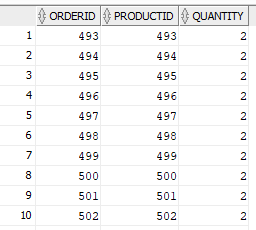
SQL> SELECT \* FROM MYCUSTOMERS;



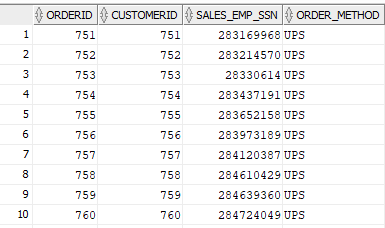
SQL> SELECT \* FROM MYEMPLOYEE;



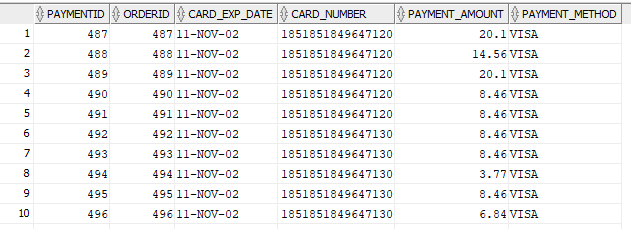
SQL> SELECT \* FROM MYORDER\_ITEM;



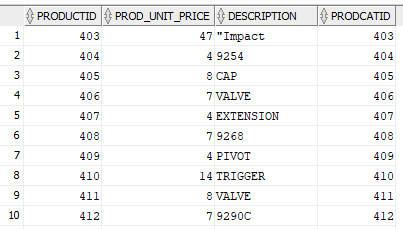
SQL> SELECT \* FROM MYORDERS;



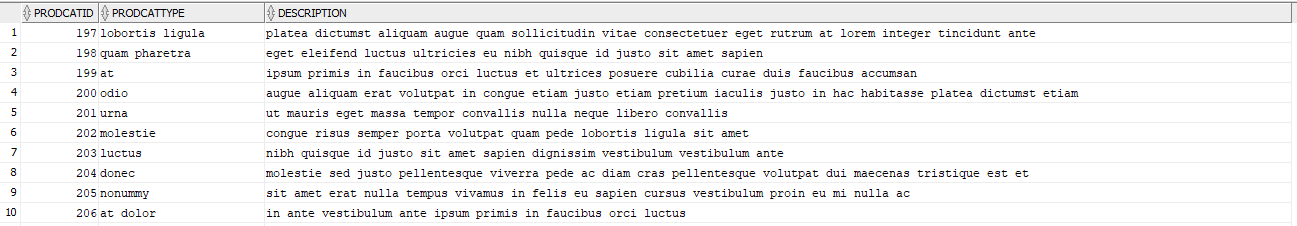
SQL> SELECT \* FROM MYPAYMENT;



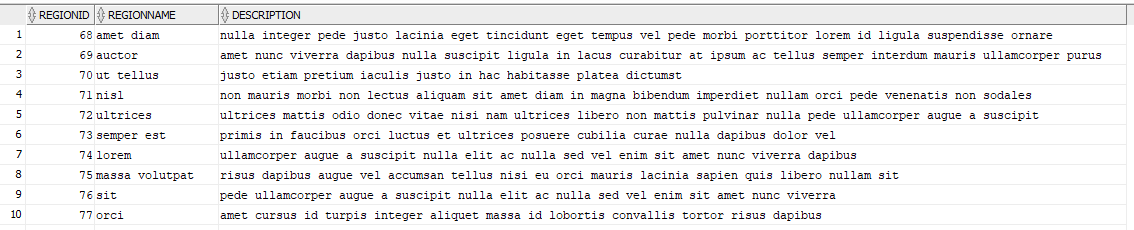
SQL> SELECT \* FROM MYPRODUCT;



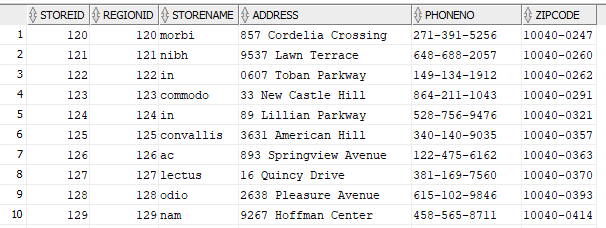
SQL> SELECT \* FROM MYPRODUCT\_CAT;



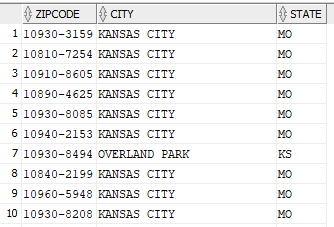
SQL> SELECT \* FROM MYREGION;



SQL> SELECT \* FROM MYSTORES;



SQL> SELECT \* FROM MYZIP;



## **Create reverse engineer on all\_objects views – Item 18**

SQL>

SQL> SET HEADING ON;

SQL> SET PAGES 999;

SQL> SET LONG 90000;

SQL>

SQL> /\*Generate DDLs for tables and indexes in DataDesignLeadUser schema\*/

SQL> SELECT

2 dbms\_metadata.GET\_DDL(object\_type,object\_name,'DATADESIGNLEADUSER')

3 FROM all\_objects

4 WHERE owner = 'DATADESIGNLEADUSER' AND (object\_type = 'TABLE' OR object\_type = 'INDEX');

DBMS\_METADATA.GET\_DDL(OBJECT\_TYPE,OBJECT\_NAME,'DATADESIGNLEADUSER')

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

CREATE TABLE "DATADESIGNLEADUSER"."ZIP"

( "ZIPCODE" CHAR(10) NOT NULL ENABLE,

"CITY" CHAR(25),

"STATE" CHAR(2),

CONSTRAINT "PK\_ZIP" PRIMARY KEY ("ZIPCODE")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 1024000 NEXT 122880 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE

) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

STORAGE(

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

PARTITION BY RANGE ("STATE")

(PARTITION "ZIP\_PART1" VALUES LESS THAN ('N') SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 1122304 NEXT 204800 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1" ,

PARTITION "ZIP\_PART2" VALUES LESS THAN (MAXVALUE) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 1122304 NEXT 204800 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA2" )

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_ZIP" ON "DATADESIGNLEADUSER"."ZIP

" ("ZIPCODE")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 1024000 NEXT 122880 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."CUSTOMER"

( "CUSTOMERID" NUMBER(10,0) NOT NULL ENABLE,

"FIRSTNAME" VARCHAR2(15),

"LASTNAME" VARCHAR2(15),

"PHONENO" CHAR(15),

"ADDRESS" VARCHAR2(35),

"ZIPCODE" CHAR(10),

CONSTRAINT "PK\_CUSTOMER\_ID" PRIMARY KEY ("CUSTOMERID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_ZIP" FOREIGN KEY ("ZIPCODE")

REFERENCES "DATADESIGNLEADUSER"."ZIP" ("ZIPCODE") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 20 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 122880 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_CUSTOMER\_ID" ON "DATADESIGNLEADUS

ER"."CUSTOMER" ("CUSTOMERID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."PRODUCT"

( "PRODUCTID" NUMBER(10,0) NOT NULL ENABLE,

"PROD\_UNIT\_PRICE" NUMBER(10,0),

"DESCRIPTION" VARCHAR2(150),

"PRODCATID" NUMBER(10,0),

CONSTRAINT "PK\_PRODUCTID" PRIMARY KEY ("PRODUCTID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_PRODCATID" FOREIGN KEY ("PRODCATID")

REFERENCES "DATADESIGNLEADUSER"."PRODUCT\_CAT" ("PRODCATID") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 172032 NEXT 24576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA2"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_PRODUCTID" ON "DATADESIGNLEADUSER

"."PRODUCT" ("PRODUCTID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."REGION"

( "REGIONID" NUMBER(10,0) NOT NULL ENABLE,

"REGIONNAME" VARCHAR2(25),

"DESCRIPTION" VARCHAR2(150),

CONSTRAINT "PK\_REGION" PRIMARY KEY ("REGIONID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 196608 NEXT 24576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_REGION" ON "DATADESIGNLEADUSER"."

REGION" ("REGIONID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."STORES"

( "STOREID" NUMBER(10,0) NOT NULL ENABLE,

"REGIONID" NUMBER(10,0),

"STORENAME" VARCHAR2(15),

"ADDRESS" VARCHAR2(35),

"PHONENO" CHAR(15),

"ZIPCODE" CHAR(10),

CONSTRAINT "PK\_STORES" PRIMARY KEY ("STOREID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_STORE\_REGION" FOREIGN KEY ("REGIONID")

REFERENCES "DATADESIGNLEADUSER"."REGION" ("REGIONID") ENABLE,

CONSTRAINT "FK\_STORE\_ZIP" FOREIGN KEY ("ZIPCODE")

REFERENCES "DATADESIGNLEADUSER"."ZIP" ("ZIPCODE") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 98304 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_STORES" ON "DATADESIGNLEADUSER"."

STORES" ("STOREID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."EMPLOYEE"

( "SALES\_EMP\_SSN" NUMBER(9,0) NOT NULL ENABLE,

"STOREID" NUMBER(10,0),

"FIRSTNAME" VARCHAR2(15),

"LASTNAME" VARCHAR2(15),

"ADDRESS" VARCHAR2(35),

"PHONENO" CHAR(15),

"ZIPCODE" CHAR(10),

CONSTRAINT "PK\_SALES\_EMP\_SSN" PRIMARY KEY ("SALES\_EMP\_SSN")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_EMP\_STOREID" FOREIGN KEY ("STOREID")

REFERENCES "DATADESIGNLEADUSER"."STORES" ("STOREID") ENABLE,

CONSTRAINT "FK\_EMP\_ZIP" FOREIGN KEY ("ZIPCODE")

REFERENCES "DATADESIGNLEADUSER"."ZIP" ("ZIPCODE") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 20 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 122880 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_SALES\_EMP\_SSN" ON "DATADESIGNLEAD

USER"."EMPLOYEE" ("SALES\_EMP\_SSN")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."ORDERS"

( "ORDERID" NUMBER(10,0) NOT NULL ENABLE,

"CUSTOMERID" NUMBER(10,0) NOT NULL ENABLE,

"SALES\_EMP\_SSN" NUMBER(9,0) NOT NULL ENABLE,

"ORDER\_METHOD" CHAR(10),

CONSTRAINT "PK\_ORDERID" PRIMARY KEY ("ORDERID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_CUSTOMERID" FOREIGN KEY ("CUSTOMERID")

REFERENCES "DATADESIGNLEADUSER"."CUSTOMER" ("CUSTOMERID") ENABLE,

CONSTRAINT "FK\_EMP\_SSN" FOREIGN KEY ("SALES\_EMP\_SSN")

REFERENCES "DATADESIGNLEADUSER"."EMPLOYEE" ("SALES\_EMP\_SSN") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 60 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 24576 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_ORDERID" ON "DATADESIGNLEADUSER".

"ORDERS" ("ORDERID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."ORDER\_ITEM"

( "ORDERID" NUMBER(10,0) NOT NULL ENABLE,

"PRODUCTID" NUMBER(10,0),

"QUANTITY" NUMBER(10,0),

CONSTRAINT "PK\_ORDER\_ITEM" PRIMARY KEY ("ORDERID", "PRODUCTID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_ORDER\_ID1" FOREIGN KEY ("ORDERID")

REFERENCES "DATADESIGNLEADUSER"."ORDERS" ("ORDERID") ENABLE,

CONSTRAINT "FK\_PRODUCT\_ID" FOREIGN KEY ("PRODUCTID")

REFERENCES "DATADESIGNLEADUSER"."PRODUCT" ("PRODUCTID") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 16384 NEXT 24576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA2"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_ORDER\_ITEM" ON "DATADESIGNLEADUSE

R"."ORDER\_ITEM" ("ORDERID", "PRODUCTID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."PAYMENT"

( "PAYMENTID" NUMBER(10,0) NOT NULL ENABLE,

"ORDERID" NUMBER(10,0),

"CARD\_EXP\_DATE" DATE,

"CARD\_NUMBER" NUMBER(30,0),

"PAYMENT\_AMOUNT" NUMBER(20,2),

"PAYMENT\_METHOD" VARCHAR2(5),

CONSTRAINT "PK\_PAYMENTID" PRIMARY KEY ("PAYMENTID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE,

CONSTRAINT "FK\_ORDER\_ID2" FOREIGN KEY ("ORDERID")

REFERENCES "DATADESIGNLEADUSER"."ORDERS" ("ORDERID") ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 20 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 40960 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA2"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_PAYMENTID" ON "DATADESIGNLEADUSER

"."PAYMENT" ("PAYMENTID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE TABLE "DATADESIGNLEADUSER"."PRODUCT\_CAT"

( "PRODCATID" NUMBER(10,0) NOT NULL ENABLE,

"PRODCATTYPE" VARCHAR2(25),

"DESCRIPTION" VARCHAR2(150),

CONSTRAINT "PK\_PRODCAT" PRIMARY KEY ("PRODCATID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1" ENABLE

) SEGMENT CREATION IMMEDIATE

PCTFREE 5 PCTUSED 90 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 196608 NEXT 24576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_DATA1"

CREATE UNIQUE INDEX "DATADESIGNLEADUSER"."PK\_PRODCAT" ON "DATADESIGNLEADUSER".

"PRODUCT\_CAT" ("PRODCATID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_ZIP" ON "DATADESIGNLEADUSER"."CUSTOMER"

("ZIPCODE")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_CUSTOMERID" ON "DATADESIGNLEADUSER"."ORD

ERS" ("CUSTOMERID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_ORDER\_ID1" ON "DATADESIGNLEADUSER"."ORDE

R\_ITEM" ("ORDERID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_PRODUCT\_ID" ON "DATADESIGNLEADUSER"."ORD

ER\_ITEM" ("PRODUCTID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_ORDER\_ID2" ON "DATADESIGNLEADUSER"."PAYM

ENT" ("ORDERID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_EMP\_STOREID" ON "DATADESIGNLEADUSER"."EM

PLOYEE" ("STOREID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_EMP\_ZIP" ON "DATADESIGNLEADUSER"."EMPLOY

EE" ("ZIPCODE")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_STORE\_REGION" ON "DATADESIGNLEADUSER"."S

TORES" ("REGIONID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_STORE\_ZIP" ON "DATADESIGNLEADUSER"."STOR

ES" ("ZIPCODE")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_PRODCATID" ON "DATADESIGNLEADUSER"."PROD

UCT" ("PRODCATID")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

CREATE INDEX "DATADESIGNLEADUSER"."FK\_EMP\_SSN" ON "DATADESIGNLEADUSER"."ORDERS

" ("SALES\_EMP\_SSN")

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 57344 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USER\_INDX1"

31 rows selected.

## **Create script to detect end-user and what they are running – Item 19**

SQL> /\*Item 19 - Script to detect end user and what they are running\*/

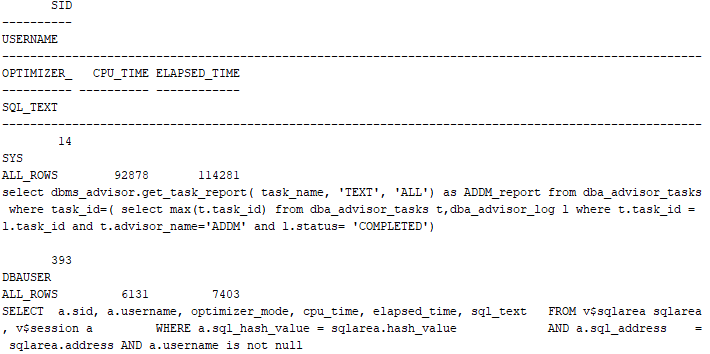
SQL> SET LINESIZE 100;

SQL> SELECT a.sid, a.username, optimizer\_mode, cpu\_time, elapsed\_time, sql\_text

2 FROM v$sqlarea sqlarea, v$session a

3 WHERE a.sql\_hash\_value = sqlarea.hash\_value

4 AND a.sql\_address = sqlarea.address AND a.username is not null;



# References

15 Backup and Recovery. (2019). Retrieved from Oracle Help Center Database Concepts: https://docs.oracle.com/cd/B28359\_01/server.111/b28318/backrec.htm#CNCPT031

20 Original Export and Import. (2019). Retrieved from Oracle Help Center Database Utilities: https://docs.oracle.com/cd/B28359\_01/server.111/b28319/exp\_imp.htm#g1070082