Oracle Database 11*g*: OCM Exam Preparation Workshop

Activity Guide • Volume I

D69748GC20
Edition 2.0
June 2013
D82217

ORACLE

Author

Setsuko Fujitani

Technical Contributors and Reviewers

Sharath Bhujani
Joel Goodman
Setsuko Fujitani
Lakshmi Narapareddi

Editors

Vijayalakshmi Narasimhan
Rashmi Rajagopal
Malavika Jinka

Publishers

Sujatha Nagendra
Joseph Fernandez

Contents

Appendix A: Practices

Appendix B: Solutions

Appendix C: Manual Solutions

Appendix A: Practices

Practice 1-1: Configuring the Initial Oracle Network Environment

Your Tasks

1. First, stop the default listener and then create a `LISTENER` listener. Use the following information:

Object	Setting
Listener name	`LISTENER`
Host	`<fully qualified host name of your odd PC>`
Protocol	TCP/IP
Port	`1521`

2. Configure local naming methods for the new `orcl` database and `PROD1` database (already exists). The `orcl` database will be created in the practice titled "Practice 1-2: Creating an Oracle Database". Use the following information:

Object	Setting
Service Name	`orcl.oracle.com`
Protocol	TCP/IP
Port	`1521`
Host	`IP address or fully qualified host name of your odd PC`
Net Service Name	`orcl`

Object	Setting
Service Name	`PROD1.us.oracle.com`
Protocol	TCP/IP
Port	`1521`
Host	`IP address or fully qualified host name of your odd PC`
Net Service Name	`PROD1`

Practice 1-2: Creating an Oracle Database

Background: You are about to begin creating your first Oracle database. You anticipate that several similar databases will be needed in the near future. Therefore, you decide to create your `orcl` database, as well as a database template and the database creation scripts. Locate the scripts in the `/home/oracle/labs` directory (which is the directory that you use most often throughout this course).

Note: Completing the database creation is critical for all following practice sessions.

Your Tasks

1. Create a database with the following settings:

Object	Setting
Global Database Name	`orcl.oracle.com`
SID	`orcl`
Password for All Accounts	`oracle`
Character Set	`AL32UTF8`
National Character Set	`AL16UTF16`
Database File Directory	`/u01/app/oracle/oradata/orcl`
Block Size	8 KB
Fast Recovery Area	`/u01/app/oracle/fast_recovery_area`
Fast Recovery Area Size	10 GB
Undo Management	AUTO
Memory Size	Automatic Memory Management 800 MB
Controlfile name	`control0*.ctl`
Default tablespace	`USERS`
UNDO tablespace	`UNDOTBS1`
Temporary tablespace	`TEMP`

2. Configure Enterprise Manager Database Control using Enterprise Manager Configuration Assistant (EMCA). The repository should already exist in the database in the `sysaux` tablespace.

Practice 1-3: Managing the Oracle Instance

Background: You have just installed the Oracle software and created a database. You want to ensure that you can start and stop the database and see the application data.

Your Tasks

1. View the initialization parameters of the `orcl` database. Set the `JOB_QUEUE_PROCESSES` parameter to `30`.
2. Shut down the database instance by using Enterprise Manager.
3. When the Status of the instance is Down, use SQL*Plus to verify that you are **not** able to connect as the `SYSTEM` user.
4. Restart the database instance.
5. In the alert log, view the phases that the database went through during startup.

Practice 1-4: Managing Undo Data

Background: In the `orcl` database, a new version of your application will include several reports based on very long-running queries. Configure your system to support these reports.

Your Tasks

1. Use the Undo Advisor to calculate the amount of undo space required to support a report that takes 60 minutes to run, on the basis of an analysis period of the last seven days.
2. Resize the undo tablespace to support the retention period required by the new reports (or 400 MB, whichever is smaller). Do this by increasing the size of the existing data file.

Practice 1-5: Managing Database Storage Structures

Background: You need to create a new tablespace for the `INVENTORY` application in the `orcl` database. All scripts are located in the `/home/oracle/labs` directory.

Your Tasks

1. Create a new, locally managed tablespace called `INVENTORY`. Use the following specifications:

Object	Setting
Tablespace name	`INVENTORY`
Extent Management	Locally Managed
Type	Permanent
Status	Read Write
Use Bigfile tablespace	*Deselected*
Data File Name	`inventory01.dbf`
File Size	`5 MB`
Autoextend	*Deselected*
Extent Allocation	Automatic
Segment Space Management	Automatic
Enable Logging	*Selected*

   Review the SQL that will be run to create this tablespace.

2. Run the `lab_01_05_02.sql` script to create and populate a table (called `X`) in the `INVENTORY` tablespace. What error do you eventually see?

   _____

3. Define space for 50 MB in the tablespace instead of 5 MB, while keeping the same single data file in the tablespace. What is the `ALTER` statement that is executed to make this change?

   _____

4. Run the `lab_01_05_04.sql` script that drops the table and re-executes the original script that previously returned the space error.

   Note that the same number of row inserts are attempted, and there is no error because of the increased size of the tablespace.

5. Create a new, bigfile tablespace called `BIGTBS`. Use the following specifications:

Object	Setting
Tablespace name	`BIGTBS`
Extent Management	Locally Managed
Status	Read Write

Data File Name	`bigtbs.dbf`
File Size	`5 MB`

6. Drop the `INVENTORY` tablespace and the `BIGTBS` tablespace.

Practice 1-6: Configuring the Oracle Network Environment

Background: You need to connect to the instructor's `orcl` database instance. Work with your instructor to enable connections by using different methods. Ensure that users can use connect-time failover to take advantage of a backup listener.

Your Tasks

1. Make a copy of your `tnsnames.ora` file. It is in the `$ORACLE_HOME/network/admin` directory.

2. Modify your local Names Resolution file so that you can connect to your instructor's `orcl` database instance. Name this connection **`testorcl`**.

Object	Setting
Connection name	`testorcl`
DB ID Method	`SID`
SID value	`orcl`
Protocol	TCP/IP
Port	`1521`
Host	IP or fully qualified host name of your instructor's PC

3. Test your changes to the network configuration by using SQL*Plus. Use `system` as the username, `oracle` as the password, and `testorcl` as the connect string. To see the information related to the instructor, select the `instance_name` and `host_name` columns from the `v$instance` table. You should see the instructor's host name.

4. Create a `LISTENER2` listener to support connect-time failover. Use port 1561 for this listener. Use the **Static Database Registration** tab to connect the listener to your database. Use the following information:

Object	Setting
Listener name	`LISTENER2`
Host	`<fully qualified host name of your odd PC>`
Service name	`orcl.oracle.com`
Protocol	TCP/IP
Port	`1561`
SID	`orcl`
Oracle Home Directory	`/u01/app/oracle/product/11.2.0/dbhome_1`

5. Start the `LISTENER2` listener.

Practice 1-7: Oracle Shared Server

You notice that your system is performing poorly during peak load times. After investigating, you find that user sessions are consuming so much memory that your system is swapping excessively. Configure your system to reduce the amount of memory that is consumed by user sessions.

Tasks

- Investigate the impact of dedicated server connections on your system.
- Configure your system to use shared servers.
- Investigate the impact of shared server connections on your system.

1. Investigate the impact of dedicated server connections on your system.
   a) Open four terminal sessions on your server.
   b) Check to see how many Oracle processes are running. (Your answer will vary.)
   c) Now start SQL*Plus sessions in your remaining three terminal sessions. Verify that none of your three SQL*Plus sessions are using shared servers.
   d) Your default service name is configured to use dedicated server processes. Notice that two new processes are created with each SQL*Plus session that is started. Remember that each of your SQL*Plus sessions starts *two* resource-consuming processes: one is the SQL*Plus session itself (which will normally consume resources on the client or middle tier), and the other is the dedicated server process (which consumes resources on the server).

2. Configure your system to use shared servers with a minimum of two shared servers and dispatchers attribute for TCP/IP should be set to a minimum of two dispatchers.

3. Investigate the impact of shared server connections on your system.
   a) Exit your three SQL*Plus sessions.
   b) Check to see how many Oracle processes are running. (Your answer will vary.)
   c) Reconnect to your instance in the three terminal windows.
   d) The connection shown in the preceding step uses EZConnect to connect to the shared server. How else could you have connected?
   e) Verify that you are connected using a shared server.
   f) Count the number of Oracle processes. Notice that your three SQL*Plus sessions started only three new processes. (These are the three SQL*Plus processes.) All three of these would normally have been located on the client machine or application

server, which means that these three new sessions would not have added *any* new processes on the server.

4. Configure two additional local naming methods for the `orcl` database. The `shared_orcl` alias always uses a shared server connection and the `orcl` alias always uses a dedicated server connection.

Practice 1-8: Using the Password Security Feature

You decide to increase the security of your passwords by enforcing case sensitivity for the password for privileged users.

If there is no HR schema on the orcl database, run the HR.sh script that is located at /home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

1. Confirm the Password Case Sensitivity settings for the instance for database users. View the value of the SEC_CASE_SENSITIVE_LOGON parameter. Connect to the orcl instance as HR with hr as the password. Attempt to connect with the username and password combinations: HR/hr, hr/hr, and hr/HR.
2. Make the password file non-case-sensitive.
3. Confirm the Password Case Sensitivity settings for the instance for privileged users. Attempt to connect to the orcl instance as the SYSBDA user using a net service name and:
   - A lowercase password
   - An uppercase password
   - A mixed-case password
4. Make the password file case-sensitive to match the database password usage.

Practice 2-1: Creating a Recovery Catalog and Registering a Database

1. Create a tablespace for the recovery catalog and a recovery catalog owner in your `PROD1` database. The tablespace name is `RCTS`. The username is `RCUSER` with the password `oracle`. This user is granted a privilege of recovery catalog owner.
2. Connect to the recovery catalog database (`PROD1`) with the appropriate recovery catalog owner name (`RCUSER`) using RMAN. Create the recovery catalog in the `RCTS` tablespace.
3. Using RMAN, connect to your target database `orcl` and the recovery catalog database `PROD1`.
4. Using RMAN, execute the `resync catalog` command to resynchronize the control file and the recovery catalog. What happens? Why?
5. Register the target database in the recovery catalog.
6. You can use Enterprise Manager to create the recovery catalog and register the database. (If you manually create the recovery catalog and manually register the database, skip this step.)
7. Create an RMAN script named `whole_backup` to make a whole database backup. **Do not execute the `whole_backup` script at this time.**
8. Use the `PRINT` command to query the recovery catalog and verify the creation of your `whole_backup` script.

Practice 2-2: Configuring Your Database

1. Configure your database (`orcl`) in `ARCHIVELOG` mode.
2. Verify that the Fast Recovery Area has been configured for your database and increase the Fast Recovery Area size to 12 GB.
3. Set Preferred Credentials in Enterprise Manager.
4. Use Recovery Manager (RMAN) to connect to your target database. Make a note of the database identifier (DBID) of your database.

   Database Identifier: _____
5. Use the RMAN `SHOW ALL` command to view the configuration settings in your database, and then exit from your RMAN session.

Practice 2-3: Using RMAN to Create and Manage Backups

1. For the `orcl` database, configure autobackup of the control file and the server parameter file.
2. Configure backup optimization and enable block change tracking. Specify **`/u01/app/oracle/oradata/orcl/chg_track.f`** for the name of the block change tracking file.
3. Create a whole database backup as the base backup of the incremental backup using the Oracle-suggested backup strategy.
4. View information about your backups.
5. Create an archival backup of the `orcl` database for Long-Term storage using the tag (long01) into the `/home/oracle/my-files/backup` directory.

Practice 2-4: Using RMAN to Recover a Data File

In this practice, you use RMAN to recover a lost data file for `orcl`.

1. Use SQL*Plus to query the `HR.REGIONS` table. Make a note of the number of rows in the `HR.REGIONS` table.
2. At the operating system prompt, execute the `lab_02_04_02.sh` script that is located in `/home/oracle/labs` to simulate a failure in your database. This script deletes the `EXAMPLE` tablespace data file.
3. Use SQL*Plus to query the `HR.JOBS` table.
4. Perform database recovery of the `EXAMPLE` tablespace data file.
5. Return to your SQL*Plus session and again attempt to query the `HR.JOBS` table.

Practice 2-5: Recovering Control Files

In this practice, you recover your control file by using autobackup.

1. Use SQL*Plus to view information about the control files in the `orcl` database. Query `V$CONTROLFILE`.
2. Simulate a failure in your environment by executing the `lab_02_05_02.sh` script that is located in `/home/oracle/labs` to delete all your control files.
3. You need some more information about your control files. Query `V$CONTROLFILE_RECORD_SECTION` to learn more about the contents of your control file.
4. You have lost all your control files and will need to recover them from the control file autobackup. Use Recovery Manager to recover the control files.

Practice 2-6: Enabling Flashback Database

1. Enable Flashback Database for `orcl`.
2. Use the `ALTER DATABASE` command to enable supplemental logging.

Practice 2-7: Using Flashback Database

1. In `orcl`, verify that Flashback Database is enabled.
2. Create a Guaranteed Restore Point named `BEFORE_TRUNCATE`.
3. Determine the number of rows in the `HR.JOB_HISTORY` table. Record the number of rows: _____
4. Truncate the `HR.JOB_HISTORY` table.
5. Determine the number of rows in the `HR.JOB_HISTORY` table.
6. Use Flashback Database to restore the `HR.JOB_HISTORY` table rows.
7. Return to your SQL*Plus session. Query the `HR.JOB_HISTORY` table again to be sure that the data has been restored. For cleanup, drop the restore point `BEFORE_TRUNCATE`.

Practice 3-1: Using External Tables

If there is no SH schema on the orcl database, run the SH.sh script that is located at /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. This practice will demonstrate the use of external tables to load data into a data warehouse. The table will be called sales_delta_XT and it will use data that is contained in the salesDec01.dat file, which is located in the /home/oracle/labs directory. Before you can create an external table, you will need to create a directory object in the database that will point to the directory on the file system where the data files will reside. Optionally, you can separate the location for the logfile, badfile, and discarded files from the location of the data files. Create the directory objects and grant the access privilege for the directories to the SH user.

2. When creating an external table, you are defining two parts of information:
   - The metadata information for the table representation inside the database
   - The HOW access parameter definition to extract the data from the external file

   After the creation of this meta information, the external data can be accessed from within the database, without the necessity of an initial load.

   Execute the following statements as the SH user to create the external table:

```
CREATE TABLE sales_delta_XT (
   PROD_ID NUMBER,
   CUST_ID NUMBER,
   TIME_ID DATE,
   CHANNEL_ID CHAR(2),
   PROMO_ID NUMBER,
   QUANTITY_SOLD NUMBER(3),
   AMOUNT_SOLD NUMBER(10,2)
   )
   ORGANIZATION external (
   TYPE oracle_loader
     DEFAULT DIRECTORY data_dir
     ACCESS PARAMETERS  (
       RECORDS DELIMITED BY NEWLINE
       BADFILE log_dir:'sh_sales.bad'
       LOGFILE log_dir:'sh_sales.log_xt'
       FIELDS TERMINATED BY "|" (
          prod_id, cust_id,
          time_id CHAR(11) DATE_FORMAT DATE MASK "DD-MON-YYYY",
          channel_id, promo_id, quantity_sold, amount_sold)
       )
       location('salesDec01.dat')
   );
```

3. The data in the external file can now be accessed without any further action. To demonstrate this capability, execute the following SQL commands:

```
SELECT COUNT(*) FROM sales_delta_xt;
SELECT MAX(time_id) FROM sales_delta_xt;
```

4. Load the data serially from the `sales_delta_xt` external table into the `SALES` fact table. Roll back the operation when you have finished. Execute the following SQL statements:

```
INSERT /*+ APPEND */ INTO sales (
  PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
  PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD
  )
  SELECT PROD_ID, CUST_ID, TIME_ID,
    case CHANNEL_ID
      when 'S' then 3
      when 'T' then 9
      when 'C' then 5
      when 'I' then 4
      else 2
    end,
    PROMO_ID, sum(QUANTITY_SOLD), sum(AMOUNT_SOLD)
    FROM SALES_DELTA_XT
    GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;
```

5. Using the Data Pump driver, create an external table called `sales_ch` from the `SALES` table that contains all the records that have a `channel_id` of 4. Use the same `data_dir` directory that you created in step 1. Name the Data Pump output file that will be created as `'sales_ch.exp'`. Execute the following SQL statements to create the external table:

```
CREATE TABLE sales_ch
  ORGANIZATION external (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY data_dir
    LOCATION ('sales_ch.exp')
  )
  AS SELECT * from sales
       WHERE channel_id = '4';
```

   Make sure that you can access the `sales_ch` table. Look at the `sales_ch.exp` file that is being used by the new table (and created in the process).

6. Drop the `data_dir` and `log_dir` directories.

Practice 3-2: Moving Data

If there is no HR schema on the orcl database, run the HR.sh script that is located at /home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

Background: In the recent past, you received a number of questions about the HR schema. To analyze them, without interfering with the daily activities, you decide to use the Data Pump Wizard to export the HR schema to file. When you perform the export, you are not sure into which database you will be importing this schema.

In the end, you find out that the only database for which the management approves an import is the orcl database. So you perform the import with the Data Pump Wizard, remapping the HR schema to a newly created HR_TEST schema in the HR_TEST tablespace.

1. Create a tablespace HR_TEST and a user HR_TEST by using SQL*Plus.

```
HR_TEST tablespace:
  DATAFILE : /u01/app/oracle/oradata/orcl/hr_test01.dbf
  SIZE : 10MB

HR_TEST user:
  DEFAULT TABLESPACE : HR_TEST
  QUOTA : UNLIMITED on HR_TEST
  PRIVIREGE : CREATE SESSION
```

2. Export the HR schema from the orcl database.

3. As the SYSTEM user, import the exported HR schema back into the orcl database, remapping it to the previously created HR_TEST schema and HR_TEST tablespace.

Practice 3-3: Materialized Views

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. On the orcl database, connect as SH with the password sh and estimate the number of rows a materialized view corresponding to the following query would contain:

```
SELECT c.cust_id, SUM(amount_sold) AS dollar_sales,
COUNT(amount_sold) AS cnt_dollars, COUNT(*) AS cnt
FROM   sales s, customers c
WHERE  s.cust_id= c.cust_id
GROUP  BY c.cust_id;
```

2. Execute the following SQL to create a table called CUST_ID_SALES_AGGR. Compare the estimated size that you obtained in step 1 with the actual number of rows in the existing CUST_ID_SALES_AGGR table. This table contains the data corresponding to the query in step 1. What is your conclusion?

```
CREATE TABLE cust_id_sales_aggr
AS
SELECT c.cust_id,
       SUM(amount_sold) AS dollar_sales,
       COUNT(amount_sold) AS cnt_dollars,
       COUNT(*) AS cnt
FROM sales s, customers c
WHERE s.cust_id= c.cust_id
GROUP BY c.cust_id
/
```

3. Determine the number of objects in the SH schema.

4. Assume that the CUST_ID_SALES_AGGR table indeed corresponds with the result of the query in step 1. Use this knowledge to create a materialized view on the prebuilt CUST_ID_SALES_AGGR table. Make sure that this materialized view can be used for future query rewrites and is fast refreshable on demand.

5. Count the number of objects in the SH schema again, to check for new objects. From the data dictionary, identify the objects having CUST_ID_SALES_AGGR as name. What type of objects are these? What is your conclusion?

6. Create a new materialized view named `MV1`. This materialized view should be defined exactly in the same way as `CUST_ID_SALES_AGGR` (created in the previous step), except that `MV1` should not be defined on the prebuilt table `CUST_ID_SALES_AGGR`, and its `SELECT` list should not contain the following two expressions: `COUNT(amount_sold)` and `COUNT(*)`. Count the number of objects in the `SH` schema again. What new objects have been created? What is your conclusion?

7. Compare the staleness status of the materialized views that you have created. What is your conclusion?

8. Execute the `lab_03_03_08.sql` script. This script adds one row to the `CUSTOMERS` table. Check the staleness status and the compile status of both the materialized views.

9. How would you refresh the staleness status of both the materialized views to reflect their current status? When refreshed, check the status. What happens and why?

Practice 3-4: Using SQL*Loader

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create an SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

The salesDec01.dat file that is located in /home/oracle/labs has been extracted from an OLTP database and needs to be loaded into the data warehouse. You need a transformation, because the data is not in the right format. The QUANTITY_SOLD and AMOUNT_SOLD columns must be summed, grouping on all other columns, before loading the data into the target table.

Because SQL*Loader cannot sum values while loading, the data must first be loaded into a staging table, and then transformed while being inserted into the target table.

1. Connect to SQL*Plus as the SH user and create a staging table SALES_DEC01 to load the data into. The table must have the same structure as the SALES table.
   Hint: Use the clause WHERE 1=0.
   Modify the channel_id column to accept CHAR data because the initial data that you will load is not in exactly the same format as the SALES fact table.

```
CREATE TABLE sales_dec01 AS
SELECT *
FROM   sales
WHERE  1=0;

ALTER TABLE sales_dec01 MODIFY (channel_id CHAR(2) null);
```

2. Load the data from the salesDec01.dat file into the SALES_DEC01 staging table, using the sales_dec01.ctl control file that is located in /home/oracle/labs. Verify the information in the control file before loading.

3. Load the data from the SALES_DEC01 staging table into the SALES target table. The data needs to be transformed; the QUANTITY_SOLD and AMOUNT_SOLD columns must be summed, grouping on all other columns. In addition, the CHAR values for channel_id must be converted to NUMBER as is the same column in the SALES table. As the SH user, execute the following SQL statements to perform this task:

```
INSERT /*+ APPEND */ INTO sales
  ( PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
    PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD )
  SELECT
    PROD_ID, CUST_ID, TIME_ID,
    case CHANNEL_ID
    when 'S' then 3
    when 'T' then 9
    when 'C' then 5
    when 'I' then 4
    else 2
    end,
    PROMO_ID,
    sum(QUANTITY_SOLD),
    sum(AMOUNT_SOLD)
  FROM sales_dec01
  GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;
```

4. When you have successfully loaded the data into the SALES table, drop the staging table sales_dec01.

Practice 3-5: Loading Data from Transportable Tablespaces

If there is no SH schema on the orcl database, run the SH.sh script that is located at /home/oracle/labs. The script will create an SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Assume that you want to move the January 2000 sales data from your data warehouse to the SALES table in your data mart. The data must be placed into a separate tablespace in order to be transported. So, create a tablespace called tt_temp_sales to hold the January sales data. After creating the tablespace, create a table called temp_jan_sales using the CREATE TABLE ... AS SELECT statement. Execute the following CREATE TABLESPACE statements as the SYSTEM user and execute the following CREATE TABLE statements as the SH user on the orcl database:

   ```
   CREATE TABLESPACE tt_temp_sales DATAFILE '/tmp/tt_temp_sales.dbf'
   SIZE 30M REUSE autoextend on;

   CREATE TABLE temp_jan_sales NOLOGGING TABLESPACE tt_temp_sales AS
   SELECT * FROM sales
   WHERE time_id BETWEEN '31-DEC-1999' AND '01-FEB-2000';
   ```

2. To prevent any changes to the tt_temp_sales tablespace, set it to read-only by executing the following SQL statement:

   ```
   ALTER TABLESPACE tt_temp_sales READ ONLY;
   ```

3. Create a /home/oracle/orcl directory and a directory object called orcl_dir that points to /home/oracle/orcl where the log and dump files for the Data Pump export will be written. Next, use the expdp utility as shown as follows to export the tt_transfer tablespace metadata.

   ```
   CREATE DIRECTORY orcl_dir as '/home/oracle/orcl';

   expdp system/oracle DIRECTORY=orcl_dir
   DUMPFILE=meta_tt_temp_sales.dmp
   TRANSPORT_TABLESPACES=tt_temp_sales
   ```

4. You will transport the tablespace to the PROD1 database. Create the /home/oracle/PROD1 directory and copy the data file and the dump file to the directory. Verify that the files have been successfully copied.

5. Set tt_temp_sales tablespace to read/write by executing the following SQL statement on the orcl database:

   ```
   ALTER TABLESPACE tt_temp_sales READ WRITE;
   ```

6. On the PROD1 database, create a directory called PROD1_dir that points to /home/oracle/PROD1 that will be used by the Data Pump import, and create a user named SH by using the following SQL statement.

```
DROP USER sh CASCADE;
CREATE USER sh IDENTIFIED BY sh;
GRANT create session TO sh;
```

When this is done, use the impdp utility as shown as follows to make the tt_transfer tablespace accessible to the PROD1 database:

```
impdp system/oracle DIRECTORY=PROD1_dir
DUMPFILE=meta_tt_temp_sales.dmp logfile=imp_tt.log
transport_datafiles=/home/oracle/PROD1/tt_temp_sales.dbf
```

7. When the import is finished, verify that the temp_jan_sales table is accessible.

8. Drop the tt_temp_sales tablespace and the orcl_dir directory from the orcl database.

9. Drop the tt_temp_sales tablespace and the PROD1_dir directory from the PROD1 database.

Practice 3-6: Automatic Parallelism Feature

1. In the `orcl` database, verify the current value of the initialization parameters that are used for parallel execution.

2. Enable automatic parallelism for this session.

3. Tune the parameter to be executed in serial if the computed elapsed time is below 20 seconds in this session.

4. Verify the current value of the initialization parameters dealing with parallel execution.

Practice 3-7: Enabling Parallel DML

1. In the `orcl` database, connect as the `SH` user and create the partitioned table `LITTLE_SALES` by executing the following SQL statements:

```
CREATE TABLE little_sales
PARTITION BY HASH (time_id)
(PARTITION LS1,PARTITION LS2)
PARALLEL
AS
SELECT * FROM sales WHERE 1=2;
```

   Note that the `LITTLE_SALES` table has only two partitions and the dictionary degree of parallelism (DOP) is set to `DEFAULT`. Execute the following SQL statements to insert some rows into the `LITTLE_SALES` table:

```
INSERT INTO little_sales
SELECT *
FROM sales
WHERE ROWNUM<5000;
commit;
```

   Is the `INSERT` statement executed in parallel? You can confirm it by examining the `V$PQ_SESSTAT` view.

2. Enable parallel DML in your session. Execute the `INSERT` statement again and confirm the output of selecting the `v$pq_sesstat` view.

3. Drop the `LITTLE_SALES` table.

Practice 3-8: Managing Processes for Parallel SQL Execution

1. In the `orcl` database, connect as `SYSDBA` and query the current values of the initialization parameters dealing with parallel execution that are of interest to you. Execute the following SQL statements:

```
SELECT name, value FROM v$parameter
WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
'PARALLEL_MIN_PERCENT');
```

2. Connect as the `SH` user and execute the following statements to create the `TEMP_CHANNELS` table:

```
CREATE TABLE temp_channels
PARALLEL 5 AS
SELECT *
FROM sh.channels
WHERE channel_id IN ('2','3','4');
```

3. Connect as `SYSDBA` and change the `PARALLEL_MAX_SERVERS` parameter to 5 and the `PARALLEL_ADAPTIVE_MULTI_USER` to `FALSE`. Verify that the change is in effect.

4. In a single statement, create a `TEMP_SALES` table as the `SH` user with minimal logging and with a DOP set to 5. This table should be a subset of the `SALES` table; it should contain only rows with `CHANNEL_ID` values `'2'` or `'3'`. Execute the following SQL statements to create the table. Examine `V$PQ_SESSTAT` to determine the DOP for the operation.

```
CREATE TABLE temp_sales NOLOGGING PARALLEL 5 AS
SELECT *
FROM sh.sales
WHERE channel_id IN ('2','3');
```

5. Execute the following SQL statements. This script joins the two tables, `TEMP_SALES` and `TEMP_CHANNELS`, using a DOP of 5. Examine `V$PQ_SESSTAT` to determine the DOP for the operation.

```
SELECT  count(*)
FROM temp_sales s, temp_channels c
WHERE (s.channel_id) = (c.channel_id);
```

6. Connect as `SYSDBA` and restore the value of `PARALLEL_MAX_SERVERS` to its original value. Reconnect as user `SH` and rerun the query in step 5, and inspect `V$PQ_SESSTAT` when the query has finished. Compare the results with those in step 5. What are your conclusions?

7. After you have finished, drop the `TEMP_CHANNELS` and `TEMP_SALES` tables.

Practice 3-9: Star Schema Tuning

In this practice, you optimize a query to use star transformation and access the benefits of using this optimizer technique.

1. From a terminal session, connect as the `oracle` user and execute the `setup_star_schema_lab.sh` script that is located in your `/home/oracle/labs` directory.

2. From the same terminal window, start a SQL*Plus session connected as the `SH` user and do not disconnect from it until this practice finishes. Before executing the following SQL statement, ensure that you flush both the shared pool and the buffer cache to avoid caching issues as much as possible. After this, analyze the execution of the following query (you can use the `first_run.sql` script):

```
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
               SUM(s.amount_sold) sales_amount
FROM sales s, times t, customers c, channels ch
WHERE s.time_id = t.time_id                    AND
      s.cust_id = c.cust_id                    AND
      s.channel_id = ch.channel_id             AND
      c.cust_state_province = 'CA'             AND
      ch.channel_desc in ('Internet','Catalog') AND
      t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-04')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;
```

   What are your conclusions?

3. Without modifying the `SH` schema, how can you improve the execution plan for the query mentioned in step 2? Verify your solution and explain why it is probably a better solution. You can use `second_run.sql`.

4. How would you enhance the previous optimization without changing the `SH` schema? You can use `third_run.sql`.

5. How do you eliminate one access on the `CUSTOMERS` table from the previous execution plan for the same `SELECT` statement seen in step 3?

6. Try to apply your finding. What happens and why?

7. Fix the issue that you found and apply your solution from step 5 again.

8. Verify that you solved the problem from step 5. You can use `fourth_run.sql`.

9. Clean up your environment by removing the index that you created and by returning the constraint to its original state by using the `cleanup_star_schema_lab.sql` script.

Practice 4-1: Implementing Fine-Grained Auditing

In the orcl database, there is a business requirement that a record must be logged whenever employee salary information is accessed. INSERT, UPDATE, and DELETE are recorded in a journal table by using triggers. Create a proof of concept solution for SELECT accesses. Create a user PFAY and prove that SELECT accesses will be recorded.

1. Create a security officer account that has privileges to create user accounts, grant privileges, and administer fine-grained auditing and fine-grained access control. This account is named SEC with the password sec. In this and subsequent practices, security functionality is implemented in a single user. Create this user, giving it the following properties:

```
Name is SEC
Password is sec
Can create a session and grant the privilege to other users to
   create a session
Can select from any table in the database
Can create or drop any context in the database
Can create, alter, and drop users
Can create roles and can alter and drop any roles
Can create tables, procedures, and triggers (including the
   ADMINISTER DATABASE TRIGGER privilege, which allows the user to
   create database triggers)
Can execute DBMS_SESSION. This privilege is granted from the SYS
   user to PUBLIC by default.
```

2. As the SEC user, create the PFAY user and grant SELECT access to the HR.EMPLOYEES table to PFAY.
   Create the PFAY user with the password e!car0 (*e*-exclamation point-*c*-*a*-*r*-zero).
   Grant PFAY the required access.
3. Make sure that EXAMPLE is the default tablespace for the SEC user and that SEC has the ability to create objects in the EXAMPLE tablespace.
4. Enable the SEC user to execute the DBMS_FGA package.
5. Create a Fine-Grained Auditing (FGA) policy with the following properties:
   Object: HR.EMPLOYEES
   Name: AUDIT_EMPS_SALARY
   Audits: Any access to the SALARY column
   Policy: Disabled
6. As the PFAY user, select SALARY from the HR.EMPLOYEES table.
7. As the SYSTEM user, display the audit record from the previous SELECT statement.

   Note: The time stamp that is shown is the time when step 6 was executed.

Practice 4-2: Using an Encrypted Tablespace

1. Create an Oracle encryption wallet.

   As the database software owner, create a directory for the wallet.

   `/u01/app/oracle/admin/orcl/wallet`

2. Modify the `sqlnet.ora` file (located at `/u01/app/oracle/product/11.2.0/dbhome_1/network/admin`) to use the encryption wallet. Using your preferred editor, add the following lines:

```
ENCRYPTION_WALLET_LOCATION=
     (SOURCE=
        (METHOD = FILE)
        (METHOD_DATA =
           (DIRECTORY = /u01/app/oracle/admin/orcl/wallet)))
```

3. In the `orcl` database, create a database master encryption key.
4. Create an encrypted tablespace named `ENCTS` that uses the default encryption algorithm with a size of 10 MB. Use the name of the data file `$ORACLE_HOME/dbs/encts.dat`.
5. View the data dictionary.
6. Drop the `ENCTS` tablespace.

Practice 4-3: Using Flashback Versions Query

If there is no HR schema on the orcl database, run the HR.sh script that is located in/home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

1. Execute the following SQL to query the HR.LOCATIONS table for location ID 1400.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

2. Execute the following SQL to update the POSTAL_CODE column in the HR.LOCATIONS table, simulating user error.

```
UPDATE hr.locations
SET postal_code = postal_code + 100
WHERE location_id = 1400;
commit;
```

3. Execute the following SQL to query the POSTAL_CODE column in HR.LOCATIONS and view the change.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

4. Execute the following SQL to update the POSTAL_CODE column in the HR.LOCATIONS table, simulating user error.

```
UPDATE hr.locations
SET postal_code = postal_code + 100
WHERE location_id = 1400;
commit;
```

5. Perform Flashback Versions Query to correct user errors.
6. Return to your SQL*Plus session. Query the HR.LOCATIONS table to confirm the Flashback operation.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

Practice 4-4: Creating an Application Context

In this practice, you create an application context, set the context using a secure package, and test the context.

1. Connect as `SYSTEM` using `ORACLE@orcl` as password. Using the `SYS_CONTEXT` procedure, display the following session-related attributes:

   `CURRENT_USER`

   `SESSION_USER`

   `PROXY_USER`

   `IP_ADDRESS`

   `NETWORK_PROTOCOL`

   `AUTHENTICATION_TYPE`

   `AUTHENTICATION_DATA`

   `CLIENT_IDENTIFIER`

   You can use one of the following techniques to call `SYS_CONTEXT`:

   ```
   SELECT sys_context('userenv','…')FROM dual;
   EXEC dbms_output.put_line(syscontext('userenv','…'));
   ```

   (Make sure to issue `SET SERVEROUTPUT ON` before executing `DBMS_OUTPUT`.)

2. Implement a local application context with the following properties:

   Name: `EMP_USER`

   Package Name: `current_emp`

   Owned by: `SEC`

   It contains the following attributes, which are listed with the column from the `HR.EMPLOYEES` table that is used to obtain the attribute value:

Attribute	Column from `HR.EMPLOYEES`
`ID`	`EMPLOYEE_ID`
`NAME`	`FIRST_NAME || ' ' || LAST_NAME`
`EMAIL`	`EMAIL`

3. The row in the EMPLOYEES table that is used to populate the attributes is selected by comparing the EMAIL column to the SESSION_USER attribute from SYS_CONTEXT.

   The procedure that sets the application context has the following properties:

   Owned by: SEC user

   Part of: CURRENT_EMP package

   Name: SET_EMP_INFO

   It is called from a logon trigger named EMP_LOGON that is also owned by SEC. This trigger applies to all users.

   Execute $HOME/labs/lab_04_04_03_a.sql to create the package and package body.
   Execute $HOME/labs/lab_04_04_03_b.sql to create the logon trigger.

4. Test the context that you created by performing the following steps:
   a) Create a user named SKING with the CREATE SESSION privilege.
   b) Log in as SKING.
   c) Use SYS_CONTEXT to verify that the EMP_USER context attributes are set. If you use DBMS_OUTPUT, remember to issue the SET SERVEROUTPUT ON command.
5. Review lab_04_04_05.sql, connect as SKING, and execute this script, which lists all the application context attributes that are set in the current session. Because Label Security is installed, the LBAC$LABELS LBAC$LASTSEQ attributes are part of the context but not populated because Label Security is not yet configured.
6. Log in as SEC and select information about the application context that you created from the data dictionary.
7. What happens when you call DBMS_SESSION.SET_CONTEXT to set an attribute in the EMP_USER context? Assume that SKING wants to change the context setting.

Practice 4-5: Implementing a Fine-Grained Access Control Policy

In this practice, you create, enable, and test a Fine-Grained Access Control (FGAC) policy.

1. How does FGAC determine which rows belong in the Virtual Private Database (VPD) for the current user?
2. How does FGAC know which tables are defined in the VPD?
3. The `SEC` user also needs the privilege to create policies. As `SYSTEM`, grant `SEC` the ability to execute the package that creates policies.
4. What privilege exempts the user from access policies? Why does the `SEC` user need this privilege? Grant it to `SEC`.
5. The `lab_04_05_05.sql` script creates the package used by the security policy to return a predicate.
   a) Review and execute the script.
   b) What predicate does the policy use to limit the rows returned from the `EMPLOYEE` table?

      ```
      employee_id = SYS_CONTEXT('emp_user', 'id')
      ```

   c) How does this predicate limit the rows?
6. Test the policy function.
7. Implement a policy with the following characteristics:

   The policy limits the rows that are selected from the `HR.EMPLOYEES` table.

   The policy is named `HR_EMP_POL`.

   The function that is used to return a predicate is `SEC.HR_POLICY_PKG.LIMIT_EMP_EMP`.
8. Set up the `SKING` user so that the user can access the `HR.EMPLOYEES` table.
9. As `SKING`, execute the `lab_04_04_05.sql` script, which displays the current context attributes.
10. Which rows are returned when `SKING` queries the `HR.EMPLOYEE` table without a `WHERE` clause? Try it.
11. As the `SEC` user, delete the FGAC policy just created.

Practice 4-6: Using Flashback Data Archive

In this practice, you use Flashback Data Archive.

1. Ensure that you are pointing to the `orcl` database. Using SQL*Plus, connect to the database as the `SYS` user and execute the `lab_04_06_01.sql` script from the `/home/oracle/labs` directory. The script creates a small `FLA_TBS1` tablespace, creates the `ARCHIVE_ADMIN` user with the `ARCHIVE_ADMIN` password, and unlocks the `HR` user with the `hr` password. The password is case-sensitive by default.

2. Give the `ARCHIVE_ADMIN` user administrative privileges for creating, maintaining, and dropping Flashback Data Archives.

3. In SQL*Plus, connect as the `ARCHIVE_ADMIN` user with the `ARCHIVE_ADMIN` password.

4. Create a Flashback Data Archive named `fla1` using `fla_tbs1` tablespace and set the retention to 1 year.

5. Give the privilege to use the `fla1` archive to the `HR` user.

6. Now, switch to the role of a flashback archive user. Connect as the `HR` user with the `hr` password. Enable this flashback archive for the `EMPLOYEES` table.

7. To view and increase the salary of Mr. Fox three times by 1000, execute the `lab_04_06_07.sql` script. This produces activity in the Flashback Data Archive.

8. Query the internal name of the archive table.

9. As the `HR` user, choose a time after the creation of the Flashback Data Archive and before you executed the erroneous DML. To view Mr. Fox's employee record as of that time, execute the following query (replace `'10' MINUTES` with your chosen historic date, format examples: `'50' SECOND`, `'10' DAY`, `'5' MONTH`):

   Note: You receive an ORA-1466 error, if you specify a time before the Flashback Data Archive was started. Reduce the time to a smaller interval and try again. If you still see the salary of 12600, increase your time interval.

```
SELECT employee_id, last_name, salary
FROM hr.employees AS OF TIMESTAMP
(SYSTIMESTAMP - INTERVAL '10' MINUTE)
WHERE last_name = 'Fox';
```

10. As the HR user, you realize that the recent updates were mistakes. Revert to the original values for your chosen historic date (for example, 10 minutes ago).

```
UPDATE hr.employees
SET salary = (SELECT salary FROM hr.employees
AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '10' MINUTE)
WHERE last_name = 'Fox')
WHERE last_name = 'Fox';
```

11. As the ARCHIVE_ADMIN user, drop the fla1 Flashback Data Archive.
Note: Dropping a Flashback Data Archive includes dropping the internal tamper-proofed history table. You cannot drop this table directly due to auditing and security requirements. Dropping a Flashback Data Archive does not drop the tablespaces in which they are stored, because the tablespaces might contain other data.

12. Connected as the SYS user, clean up your environment by deleting the fla_tbs1 tablespace and the ARCHIVE_ADMIN user.

Practice 4-7: Using SecureFiles

1. In the `orcl` database, create a tablespace, `secf_tbs`, by using the following SQL statements. Enable `sh` to use the `secf_tbs` tablespace.

```
create tablespace secf_tbs datafile
'/u01/app/oracle/oradata/orcl/secf_tbs.dbf' size 10M;
```

2. Connect `orcl` database by `sh`, create the `T1` table in the `secf_tbs` tablespace with the following columns:

   - `EMPNO NUMBER`
   - `C1 CLOB`

   Enable compression (High) and deduplication for the `C1` column.

3. Clean up your environment by dropping table t1 and deleting the secf_tbs tablespace.

Practice 4-8: Using Reference Partitioning

Reference partitioning enables tables with a parent/child relationship to be logically equipartitioned by inheriting the partition key from the parent table without duplicating the key columns. In this practice, you will create a range-partitioned table called ORDERS, and then create a table called ORDER_ITEMS, which is reference-partitioned on a foreign key that is contained in the ORDERS table.

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Start a SQL*Plus session connected to the orcl database as the SYS user. From your SQL*Plus session, execute the lab_04_08_01.sql script.
2. Still in your SQL*Plus session, connect as the SH user. Execute the lab_04_08_02.sql script to create the range-partitioned ORDERS table.
3. Create a reference-partitioned ORDER_ITEMS table as follows:

```
Table Name:ORDER_ITEMS
Columns:order_id number(12) not null
  , product_id number not null
  , quantity number not null
  , sales_amount number not null
References:order_id references order_id of ORDERS
```

4. View information about the tables.

Practice 4-9: Using Interval Partitioning

Interval partitioning fully automates the creation of range partitions. Managing the creation of new partitions can be a cumbersome and highly repetitive task. This is especially true for predictable additions of partitions covering small ranges, such as adding new daily partitions. Interval partitioning automates this operation by creating partitions on demand.

1. Connect as the `sh` user, and create an interval-partitioned `NEWSALES` table as follows:

```
Table Name:NEWSALES
Columns:prod_id number(6) not null
  , cust_id number not null
  , time_id date not null
  , channel_id char(1) not null
  , promo_id number(6) not null
  , quantity_sold number(3) not null
  , amount_sold number(10,2) not null
Partition Method: range,interval
Partition Column: time_id
Interval :1'DAY'
Initial partition:
p_before_1_jan_2005 values less than (to_date('01-01-
2005','dd-mm-yyyy'))
```

2. Find the information about the `NEWSALES` table in the dictionary view.

3. Execute the `lab_04_09_03.sql` script to insert new data into the `NEWSALES` table that forces the creation of a new partition (segment).

4. Check the information about the new partition.

Practice 5-1: EXPLAIN PLAN and SQL*Plus AUTOTRACE

If there is no SH schema on the `orcl` database, run the `SH.sh` script that is located at `/home/oracle/labs`. The script will create the SH schema. You can ignore the errors on the Data Pump Import utility.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Open a terminal window. Set the current directory to **`labs`**. Start SQL*Plus. Log in to the `SH` schema as `SH` with the password `SH`. Then check the existence and column structure of the `CUSTOMERS` table. View the structure of the `PLAN_TABLE` table.

   Also list the existing indexes on the `CUSTOMERS` table by using the `lab_05_01_01.sql` script.

2. Explain the following SQL statement by using `EXPLAIN PLAN` and SQL*Plus `AUTOTRACE`.

```
SELECT cust_first_name,cust_last_name
FROM   customers
WHERE  cust_id =100
```

Practice 5-2: SQL Trace and TKPROF

1. Open a terminal window. Set the current directory to `labs`. Start SQL*Plus. Log in to the `orcl` database as the `SH` user with the password `sh`. Make sure that `AUTOTRACE` is disabled.

2. Provide an identifier for the trace file to help you locate it. Drop all indexes (except the index of the primary key) on the `CUSTOMERS` table. Enable SQL Trace. Analyze the following SQL statement by using SQL Trace and `TKPROF`.

```
SELECT max(cust_credit_limit)
FROM customers
WHERE cust_city ='Paris';
```

3. Now create an index on the `CUST_CITY` column, and then run the same query again.

4. Disable tracing.

5. Determine the location of the trace files by using the `SHOW PARAMETER DIAGNOSTIC` command and making a note of the `DIAGNOSTIC_DEST` destination.

6. Exit your session.

7. Change directory to the `DIAGNOSTIC_DEST` destination.

8. Locate your file by the file identifier that you gave in step 2. Look for a file called **`orcl_ora_xxxxx_OCMWS.trc`**.

9. View the difference the in execution plans and statistics of the SQL statement with and without an index. You can use **gedit** to do this. Change back to your home directory, and then to the `labs` directory.

10. Now take a look at `DBMS_MONITOR`. Start two sessions, one connected as `SYS as SYSDBA` and the other connected as `SH`.

11. From the `SYSDBA` session, determine the session ID (**`sid`**) and serial number (**`serial#`**) from `v$session` for the `SH` user, and then describe the `DBMS_MONITOR` package. Then, from the `SYSDBA` session, enable tracing using the `sid` and `serial#` values for the other session, including the waits and bind information.

12. From the SH session, execute the following SQL statement, and then exit your session.

```
SELECT c.cust_last_name, t.calendar_year, sum(s.amount_sold)
  FROM sales    s JOIN
      customers c USING (cust_id) JOIN
      times t USING (time_id)
  GROUP BY c.cust_last_name, t.calendar_year
  ORDER BY c.cust_last_name, t.calendar_year;
```

13. From the remaining SYSDBA session, determine your DIAGNOSTIC_DEST location, locate the trace file, and view the contents. Determine the location of the trace files by using the SHOW PARAMETER DIAGNOSTIC command and making a note of the DIAGNOSTIC_DEST destination.

14. Exit your session.

15. Change the directory to the DIAGNOSTIC_DEST destination that you retrieved by the previous query.

16. Locate your file with the following command:

```
$ ls -ltr
```

17. View the bottom-most file that is the most recent.

18. Convert the file to readable format.

19. You can use gedit to view the file. Change back to your home directory and then to the labs directory.

Practice 5-3: Access Paths

1. Open a terminal window. Change the working directory to `/home/oracle/labs`. Connect to the `orcl` database as `SH` with the password of `sh` by SQL*Plus. Drop all existing indexes (except the index of the primary key) on the `CUSTOMERS` table, and then create three indexes on the following columns: `CUST_GENDER`, `CUST_POSTAL_CODE`, and `CUST_CREDIT_LIMIT`.

2. Enable `AUTOTRACE` and run the following SQL statement. The `WHERE` clause contains three predicates. Execute this statement and take note of the indexes used, the cost of the execution plan, and the amount of I/O performed.

```
SELECT c.*
  FROM customers c
  WHERE cust_gender = 'M'
    AND cust_postal_code = '40804'
    AND cust_credit_limit = 10000;
```

3. Drop the indexes (except the index of the primary key) again and replace them with a single concatenated index of three columns `CUST_GENDER`, `CUST_POSTAL_CODE`, and `CUST_CREDIT_LIMIT`. Then run the SQL statement of the step 2 again.

4. Drop all indexes (except the index of the primary key) on the `CUSTOMERS` table. Create three bitmapped indexes on `CUST_GENDER`, `CUST_POSTAL_CODE`, and `CUST_CREDIT_LIMIT`, and then run the SQL statement of the step 2 again. This statement has a complicated `WHERE` clause. Bitmapped indexes are good for this type of statement. You see several bitmap operations in the execution plan.

5. Finally, investigate the benefits of function-based indexes. Drop all indexes on the `CUSTOMERS` table. First, create a normal index on the `CUST_LAST_NAME` column and run the following SQL statement:

```
SELECT cust_id, country_id
  FROM customers
  WHERE lower( cust_last_name) like 'gentle';
```

   Create a function-based index that utilizes the `LOWER` function on the `CUST_LAST_NAME` column.

Practice 5-4: Gathering Optimizer Statistics

1. Connect to the `orcl` database as the `sh` user and create a `MY_CUST` table that is identical to the `CUSTOMERS` table.
   ```
   CREATE table my_cust AS SELECT * FROM customers;
   ```

2. Query `USER_TABLES` to verify the existence of statistics for the `MY_CUST` table.

3. Run the following query on the table, and then view the execution plan using `AUTOTRACE TRACEONLY EXPLAIN`.
   ```
   SELECT * FROM my_cust WHERE cust_id < 50;
   ```

4. Disable `AUTOTRACE`. Create an index `MY_CUST_ID_IDX` on the `CUST_ID` column.

5. Run the query again and view the execution plan by using `AUTOTRACE`.

6. How does the optimizer know to use the index?
   Hint: Query `USER_INDEXES` and `USER_TABLES` by using `tabstats.sql` and `indstats.sql` to get the answer.

7. Drop the index that you created. Then verify the index statistics again.
   What do you see?

8. Identify the last analyzed date and sample size for all the tables in your schema.

9. Identify the types of histograms for all the columns in your schema.
   Hint: Query `USER_TAB_COL_STATISTICS`.

10. Gather statistics for all the tables on the `SH` schema.

11. Now consider histograms. First flush the shared pool. Then run **`lab_05_04_11.sql`** that creates the `NEW_CUST` table and populates it with skewed data. It also creates an index and a histogram on the skew data column `CUST_ID`. After this script is run, the `CUST_ID` column has 1,000 rows with a value of 1, one row with a value of 2, and one row with a value of 3.

    Now run the following statement and use `AUTOTRACE` to get the execution plan.
    ```
    SELECT count(ord_total) FROM new_cust where cust_id = 1;
    ```
    You see that this is a full table scan. Now try running the following SQL. Is the index used?
    ```
    SELECT count(ord_total) FROM new_cust where cust_id = 2;
    ```
    The optimizer uses the histogram to determine whether to use an index.

Practice 5-5: Using Deferred Optimizer Statistics

In this practice, you manipulate both deferred statistics publishing and statistics extensions. The basic idea of this practice is to test various statistic gatherings on a particular table before publishing the best ones in your production environment.

1. Confirm whether the `ORACLE_SID` was set to `orcl`. Execute the `stats_setup.sh` script. This script creates a new user called `STATS` and creates and populates a new table called `STATS.TABJFV`.

2. Start a SQL*Plus session connected as user `STATS`. ***Do not disconnect from that session***. Make sure that you delete all existing statistics on `STATS.TABJFV` and verify that none exist either as public statistics or as pending statistics. Use the `delete_stats.sql` script.

3. Determine the publishing mode for `STATS.TABJFV` statistics, and set it to `PENDING` mode.

4. Collect statistics on `STATS.TABJFV` and investigate the result. What do you observe? Use the `collect_pending.sql` script.

5. From a terminal window, connected as the `STATS` user in a SQL*Plus session (do not exit from this session after this step), disable dynamic sampling for your session and determine the number of rows that the optimizer can currently estimate for the following query:

   ```
   select count(*) from tabjfv where c1 = 1 and c2 = 1;
   ```
   What is your conclusion?

   Hint: Use an explain plan to view the execution plan and SQL statement as follows:

   ```
   select plan_table_output
   from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
   ```

6. Now, switch your session to use pending statistics that were previously collected.

7. Determine again the optimizer's estimation of the number of rows returned by your query. What do you observe?

8. Create a statistics extension to group `C1` and `C2` to indicate that both columns are correlated in `STATS.TABJFV`. When done, gather statistics again on your table with maximum precision for your extension.

9. Determine again the optimizer's estimation of the number of rows returned by your query. What do you observe?

10. Execute `stats_cleanup.sh` to clean up your environment.

Practice 5-6: Using Result Cache

In this practice, you explore the various possibilities to cache query results in the SGA. Perform the following steps to understand the use of Query Result Cache.

1. Change to the `$HOME/labs` directory and execute the `result_cache_setup.sh` script.

2. In your terminal window, log in to SQL*Plus as the `QRC` user. From now on, do not disconnect from this session. Determine the current content of the query cache by using the following statement:

```
select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;
```

3. Set timing on and execute the following query. You can use the `rc_query1.sql` script. Note the time that it takes to execute.

```
select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,
 cachejfv c5
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b'
 and c5.c='b';
```

4. Determine the execution plan of the previous query by using `explain_rc_query1.sql`. What do you observe?

5. Determine the current content of the query cache. What do you observe?

6. Flush the buffer cache of your instance and rerun the query that was executed in step 3. What do you observe?

7. Insert a new row into the `CACHEJFV` table by using the following statement:

```
insert into cachejfv values('c');
commit;
```

   What do you observe?

8. Execute your first query again and check the result cache. What do you observe?

9. Generate a detailed result cache memory report.

10. Execute your first query again. What do you observe?

11. Clear the result cache. Query `V$RESULT_CACHE_OBJECTS` to verify the clear operation.

Practice 5-7: Managing Resources

In this practice, you create an APPUSER consumer group and assign it to the default DEFAULT_PLAN resource plan. Then you map a couple of Oracle users and your major OS user to resource groups. Activate the resource plan and test your assignments.

Log in as the SYS user (with oracle as the password, connect as SYSDBA) and perform the necessary tasks either through Enterprise Manager Database Control or through SQL*Plus. All scripts for this practice are in the /home/oracle/labs directory.

If there is no HR, SCOTT, OE, or PM user, execute lab_05_07.sql to create sample users by using SQL*Plus.

```
$ cd /home/oracle/labs
$ sqlplus / as sysdba @lab_05_07.sql
```

1. Using Enterprise Manager Database Control, create a resource group called APPUSER. At this point, **do not** add users to the group.

2. Add the APPUSER and LOW_GROUP consumer groups to the DEFAULT_PLAN resource plan. Change the level 3 CPU resource allocation percentages: 60% for the APPUSER consumer group and 40% for the LOW_GROUP consumer group.

3. Configure Consumer Group Mappings, so that the HR Oracle user belongs to the APPUSER consumer group and the SCOTT Oracle user to the LOW_GROUP consumer group. For the SCOTT user, confirm that his ORACLE_USER attribute has a higher priority than the CLIENT_OS_USER attribute.

4. Configure Consumer Group Mappings so that the oracle OS user belongs to the SYS_GROUP consumer group.

5. Assign the PM Oracle user to the following consumer groups: APPUSER, LOW_GROUP, and SYS_GROUP.

6. Activate the DEFAULT_PLAN resource group.

7. Test the consumer group mappings. Start two SQL*Plus sessions: the first with the system/oracle@orcl connect string and the second with the scott/scott@orcl connect string. Test other mappings as well.

8. Revert to your original configuration by deactivating the DEFAULT_PLAN resource group, undoing all consumer group mappings (use rsc_cleanup.sh), and finally by deleting the APPUSER resource group.

Practice 5-8: Using SQL Access Advisor

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema. You can ignore the errors on the Data Pump Import utility.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. From a terminal session, connected as the oracle user, execute the sqlaccessadv_setup.sh script. This script generates the necessary data that you use throughout this practice. In particular, it generates the SQL Tuning Set that is used to represent the workload that you want to analyze.

2. Using Enterprise Manager, create a SQL Access Advisor tuning task based on the captured workload that is in the SH.SQLSET_MY_ACCESS_WORKLOAD SQL tuning set by using the SQLACCESS_WAREHOUSE template.

3. After this is done, investigate the proposed recommendations.

Practice 5-9: Creating Baselines

Using the `orcl` database, create baselines. (This exercise requires that the default snapshot collection has been running since the database was started.)

1. Create a baseline named Monday over past snapshots of Monday and compute statistics over the static baseline.

2. Create a repeating baseline in the future. Set the following:

   Baseline Name: `afterwork`
   Start Time: 6:00 PM
   Duration: 12 hours
   Frequency: Daily
   Start Time: Set it to the Sunday preceding the class.
   End Time: Set it to the Saturday following the class.
   Retention Time: 28 days

3. View the data dictionary views.

Practice 5-10: Using SQL Plan Management

Background: SQL Plan Management (SPM) is a new Oracle Database 11*g* feature that provides controlled execution plan evolution.

With SPM, the optimizer automatically manages execution plans and ensures that only known or verified plans are used.

When a new plan is found for a SQL statement, it will not be used until it has been verified to have comparable or better performance than the current plan.

1. Before you can start this practice, you need to set up a new user. Execute the `spm_setup.sh` script to set up the environment for this practice. This script creates the `SPM` user that you use throughout the practice.

2. The first component of SPM is Plan Capture. There are two main ways to capture plans: automatically (on the fly), or bulk load. You look at automatic capture first. Connect to the `orcl` database as the `spm` user with the password `spm` and enable automatic plan capture so that the SPM repository is automatically populated for any repeatable SQL statement.

3. Execute the following query in your SQL*Plus session (no space in `/*LOAD…`):
   `select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order by prod_id;`
   Use the `spm_query1.sql` script to execute the query.

4. Because this is the first time that you have seen this SQL statement, it is not yet repeatable, so there is no plan baseline for it. To confirm this, you can check that the plan baseline was not loaded. Check whether there are any plan baselines that exist for your statement.

5. Re-execute the query in step 3 by using `spm_query1.sql`.

6. The SQL statement is now known to be repeatable and a plan baseline is automatically captured. Check that the plan baseline was loaded for the previous statement. What do you observe?

7. Now, change or alter the optimizer mode to use `FIRST_ROWS` optimization and re-execute your statement. Describe what happened.

8. Now reset the optimizer mode to default values and disable auto capture of plan baselines.

9. Purge the plan baselines and confirm that the SQL plan baseline is empty. Use `purge_auto_baseline.sql`.

10. Now, you will see how to directly load plan baselines from the cursor cache. Before you begin, you need some SQL statements. Still connected to your SQL*Plus session, check the execution plan for the following SQL statement, and then execute it (use the `explain_spm_query3.sql` and `spm_query3.sql` scripts):

```
select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order
by prod_id;
```

11. Now change the optimizer mode to use `FIRST_ROWS` optimization and re-execute the step 10. What do you observe?

12. Reset the optimizer mode to `ALL_ROWS`.

13. Now that the cursor cache is populated, you need to get the SQL ID for your SQL statement by using the following SQL.Use the SQL ID to filter the content of the cursor cache and load the baselines with these two plans.

    Use the `load_cc_baseline.sql` script.

14. Confirm that the baselines were loaded.

15. Purge plan baselines by using `purge_cc_baseline.sql`.

16. Drop the `spm` user to clean up.

Practice 5-11: Using the SQL Performance Analyzer

In this practice, you simulate exporting a SQL Tuning Set (STS) from a 10*g* database and import it back into an 11*g* test environment. There, you access the performance of the SQL statements that you imported before upgrading the 10*g* database.

If you do not use an spfile, at first, create an spfile from pfile and restart `orcl` database.

1. From a terminal window, which is referred to as the first session, execute the `setup_SPAbig10g.sh` script to set up your simulated 10*g* environment. In this simulated environment, the `OPTIMIZER_FEATURES_ENABLE` parameter is set to `10.2.0.2`.

```
$ cd /home/oracle/labs
$ ./setup_SPAbig10g.sh
```

2. (*Perform steps 2 and 3 at the same time*) Generate a SQL Tuning Set (STS) called `STS_JFV` that captures SQL statements from the cursor cache for approximately 12 minutes every five seconds. Make sure that you try to capture only statements from the `SQL_JFV` module in the `APPS` schema. Also, this STS should belong to the `SYS` user. Use the `capsts10g.sh` script to perform this step.

```
$./capsts10g.sh
```

3. (*Perform steps 2 and 3 at the same time*) From a second terminal window, connected as the `oracle` user, execute your workload by using the `wrkl10g_jfv.sh` script. This script runs a workload of 45 statements that will be captured in `STS_JFV` automatically.

```
$./wrkl10g_jfv.sh
```

4. After approximately 12 minutes, both sessions should have finished. Connect to the `orcl` database as the `sys` user, check the content of `STS_JFV`, and stage it in a table called `APPS.STS_JFV_TAB`.

5. By using Data Pump Export, export the `APPS` schema to the default Data Pump directory (`DATA_PUMP_DIR`).

6. Now restart your 11*g* environment to restore the database to an 11*g* environment. Use the `setup_SPAbig11g.sh` script to perform this step.

```
$./setup_SPAbig11g.sh
```

7. Drop the `apps` user from `orcl` database. By using Data Pump Import, import the `APPS` schema into your system.

8. Unpack the previously imported staging table in the `SYS` schema.

9. As the `SYS` user, use Enterprise Manager Database Control to test the behavior of `STS_JFV` in the simulated 10*g* environment and compare it to the 11*g* environment. You will do this by changing the `OPTIMIZER_FEATURES_ENABLE` parameter. What are your conclusions?

Practice 6-1: Deploying the Management Agent

To begin monitoring your targets, you need to deploy the Management Agent. In the lesson, you learned that there are several methods available for deploying the agent. The method this practice is going to use is to install the agent by using Deployments tab.

1. Log in to the Grid Control console by using `sysman` as the user and `Oracle123` as the password. You will receive a security alert for the first-time login, click OK and then you need to add an exception on the "Secure Connection Failed" page. After logging in, click the **Deployments** tab. On the Deployments page, in the Agent Installation section, click the **Install Agent** link. The **Select the type of Agent Deployment that you want to perform** page appears, showing the various installation options. Click **Fresh Install** to perform a new installation of the Management Agent into the database server.

   `https://<your management server host>:7799/em`

`Hosts`	`Your database server host(odd machine)`
`OS user`	`oracle/oracle`
`Destination`	`/u01/app/oracle/product/11.1.0`
`The agent registration password`	`Oracle123`

Practice 6-2: Configuring Your Database

Now that the Management Agent has been deployed to your targets, you need to configure monitoring credentials for the `EMREP` database.

1. Log in to the Grid Control console as the `sysman` user using the `Oracle123` password. Click the **Targets** tab and find the database target. Note that on the Databases subtab, the status of the `EMREP` database shows as unavailable. This is because Grid Control does not have monitoring credentials configured yet for this database. Configuring the database involves setting the monitor password for the `dbsnmp` user on your database to the appropriate value, in this case, `Oracle123`.

Practice 6-3: Creating Administrators

You create a Super Administrator that will be using your Grid Control console, so you can avoid using `sysman` if you want.

1. Create an administrator called **Sys#_Admin** and make this administrator a Super Administrator. Assign a password of `Oracle123` to this administrator.

Practice 6-4: Administering and Maintaining the Oracle Database

This practice takes you through some basic administration tasks for the `orcl` database by using Grid Control.

1. What is the value of the `pga_aggregate_target` initialization parameter?
2. Add another data file to the `USERS` tablespace.
3. Schedule a full database backup to happen at 2:00 PM two weeks from today.

Practice 6-5: Changing Metric Thresholds

In this practice, you change the warning and critical metric thresholds.

1. Change the threshold for the Tablespace Space Used (%) metric. You change the threshold for the tablespace `USERS` to 80% and 90% for warning and critical thresholds, respectively.

Practice 6-6: Setting Up Email Addresses

You need to define the email addresses to receive notifications. You can specify multiple addresses if you want to be notified in different ways. In this practice, you define email addresses to receive notifications in different formats.

1. Log in as Sys#_Admin and using the Preferences link, define email addresses to receive notifications. Define two email addresses. The first email address should have a long message format and the second email should have a short message format.

Practice 6-7: Defining Notification Schedule

Notification schedules allow Grid Control to determine who a notification should be sent to when an alert is triggered.

1. Define a schedule assuming that you work from 9:00 AM to 5:00 PM every weekday, except Thursday. On Thursday, you work from 3:00 PM to 11:00 PM. You want to receive an email in the long format on all your working days and receive emails in the long and short format on Thursday.

Practice 6-8: Subscribing to Notification Rules

Notification rules are sets of conditions that determine when a notification occurs. Grid Control provides a few out-of-the-box notification rules for some of the most common problem situations. Administrators can subscribe to them by selecting the **Subscribe (Send E-mail)** check box for the respective notification rule.

Subscribe to the out-of-the-box notification rule Agents Unreachable so that you can be alerted whenever any agent is not reachable. Use the Public Rules section on the Preferences page.

Practice 6-9: Monitoring the Scheduler

Background: Because your job tasks are regularly increasing, you decide to automate routine tasks. First, you monitor existing scheduler elements, and then you create scheduler components and test them.

In this practice, you use Enterprise Manager Grid Control to define and monitor the Scheduler and automate tasks. Regularly, click **Show SQL** to review all statements that are new to you.

Log in to Enterprise Manager Grid Control as the `sysman` user. Log in to the `orcl` database as the `SYS` user (with `oracle` as password, connect as `SYSDBA`) or as the `HR` user (with `hr` as password, connect as `Normal`), as indicated. Perform the necessary tasks either through Enterprise Manager Grid Control or through SQL*Plus. All scripts for this practice are in the `/home/oracle/labs` directory.

1. Log in to Enterprise Manager Grid Control as the `sysman` user. Log in to the `orcl` database as the `SYS` user and grant the following roles to the `HR` user:

   - `CONNECT` role
   - `RESOURCE` role
   - `DBA` role

   Because you are going to use the `HR` user to administer jobs through Grid Control, you need to make sure that `HR` is registered as a possible administrator.

2. On the Server tabbed page, click the **Jobs** link in the Oracle Scheduler region. Are there any jobs?

   Question: Are there any jobs?

   _____

3. Review the Programs page in Enterprise Manager. (**Hint:** Use the browser's **Back** button.)

   Question: Are there any existing programs?

   _____

4. Review the Scheduler Schedules page in Enterprise Manager.

   Question: Are there any existing schedules?

   _____

5. Review the Scheduler Windows page in Enterprise Manager. Are there any existing windows? Which resource plan is associated with each window?

   Question 1: Are there any existing windows? What are their names?

   _____

   _____

Question 2: Which resource plan is associated with the `WEEKNIGHT_WINDOW` window?

_____

Question 3: Which resource plan is associated with the `WEEKEND_WINDOW` window?

_____

6. Review the Scheduler Job Classes page in Enterprise Manager. Are there any existing job classes? If so, which resource consumer group is associated with each job class?

   Question 1: Are there any existing job classes?

   _____

   Question 2: Which resource consumer group is associated with the `DEFAULT_JOB_CLASS` job class?

   _____

   Question 3: Which resource consumer group is associated with the `AUTO_TASKS_JOB_CLASS` job class?

   _____

   _____

Practice 6-10: Creating Scheduler Components

In this practice, you use Enterprise Manager Grid Control to create Scheduler objects and automate tasks.

Prerequisite: Ensure that you complete step 1 in Practice 6-9, which gives the `HR` user administrative privileges.

1. While you are logged in to the database as the `HR` user, on the Grid Control `orcl` database page, create a simple job that runs a SQL script:
   - General:
     Name: `CREATE_LOG_TABLE_JOB`
     Owner: `HR`
     Description: Create the `SESSION_HISTORY` table
     Logging level: `RUNS`
     Command type: PL/SQL
     PL/SQL block: `BEGIN execute immediate('create table session_history(snap_time TIMESTAMP WITH LOCAL TIME ZONE, num_sessions NUMBER)'); END;`
   - Schedule:
     Repeating: Do not Repeat
     Start: Immediately
   - Options:
     No special options
2. Create a program called `LOG_SESS_COUNT_PRGM` that logs the current number of database sessions into a table. Use the following code, which is provided in the `lab_06_10_02.sql` script:

```
DECLARE
sess_count   NUMBER;
BEGIN
SELECT COUNT(*) INTO sess_count FROM V$SESSION;
INSERT INTO session_history VALUES (systimestamp, sess_count);
COMMIT;
END;
```

3. Create a schedule named `SESS_UPDATE_SCHED` that is owned by `HR` that executes every three seconds. Use SQL*Plus and the `DBMS_SCHEDULER.CREATE_SCHEDULE` procedure to create the schedule.

```
BEGIN
  DBMS_SCHEDULER.CREATE_SCHEDULE (
   schedule_name => 'SESS_UPDATE_SCHED',
   start_date => SYSTIMESTAMP,
   repeat_interval => 'FREQ=SECONDLY;INTERVAL=3',
   comments => 'Every three seconds');
END;
/
```

Return to Enterprise Manager Grid Control, and verify that the SESS_UPDATE_SCHED schedule was created.

Hint: You may have to refresh the page for the Schedule to appear.

4. Using Enterprise Manager Grid Control, create a job named LOG_SESSIONS_JOB that uses the LOG_SESS_COUNT_PRGM program and the SESS_UPDATE_SCHED schedule. Make sure that the job uses FULL logging.
5. In your SQL*Plus session, check the HR.SESSION_HISTORY table for rows.

   Question: If there are rows in the table, are the time stamps three seconds apart?

   _____

6. Use Enterprise Manager Grid Control to alter the SESS_UPDATE_SCHED schedule from every three seconds to every three minutes. Then use SQL*Plus to verify that the rows are now being added every three minutes: query the HR.SESSION_HISTORY table, ordered by the SNAP_TIME column.
7. **This is your mandatory cleanup task.** Use Enterprise Manager to drop the LOG_SESSIONS_JOB and CREATE_LOG_TABLE_JOB jobs, the LOG_SESS_COUNT_PRGM program, and the SESS_UPDATE_SCHED schedule. Use SQL*Plus to drop the SESSION_HISTORY table, and exit your session.

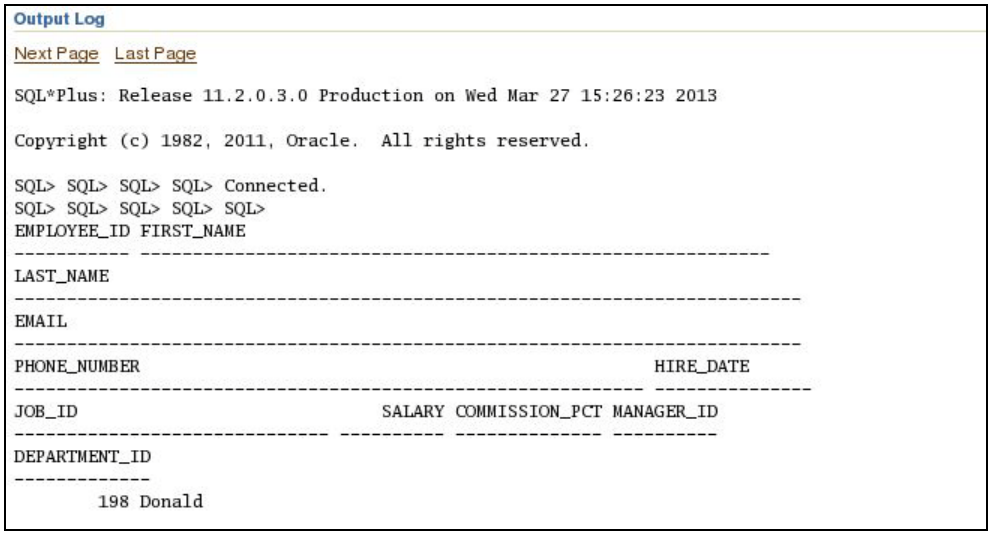
   Note: Make sure that you do not delete the wrong schedule.

Practice 6-11: Using the Job System

This practice is to help you get familiar with the Job System. You create various jobs, single and multitask, and interact with these jobs. To understand the Job functionality, you perform the following tasks:

- Create a job that runs a SQL script immediately (save this job to the library).

1. Create a simple job called "Team# SQL Job" that runs a SQL script against the database on your database host. This job should perform a `SELECT` statement on the `employees` table in the `HR` schema. Use **`select * from hr.employees;`** for the SQL script. Save the job to the Job Library, so that you can make changes to it later. Go to the Job Library and run the job. The result of this job should be similar to:



```
Output Log
Next Page  Last Page

SQL*Plus: Release 11.2.0.3.0 Production on Wed Mar 27 15:26:23 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

SQL> SQL> SQL> SQL> Connected.
SQL> SQL> SQL> SQL> SQL>
EMPLOYEE_ID FIRST_NAME
----------- --------------------------------------------------------------
LAST_NAME
--------------------------------------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE_NUMBER                                                    HIRE_DATE
-------------------------------------------------------------- ---------------
JOB_ID                          SALARY COMMISSION_PCT MANAGER_ID
------------------------------ ---------- -------------- ----------
DEPARTMENT_ID
-------------
        198 Donald
```

Practice 7-1: Adding a Physical Standby Database to your Configuration

1. Add a new physical standby database to your configuration.

   Primary Database
   Database Name: `orcl`
   Instance Name: `orcl`
   Database Unique Name: `orcl`
   Grid Control Target Name: `orcl.oracle.com`

   Physical Standby Database
   Database Name: `orcl`
   Instance Name: `stdby`
   Database Unique Name: `stdby`
   Type of standby: physical
   Grid Control Target Name: `stdby.oracle.com`

   Settings:
   Host: odd PC
   File Location: `/u01/app/oracle/oradata/stdby`

2. Configure Data Guard Broker and configure an additional listener.

Practice 7-2: Using Real-Time Query

1. Enable Real-Time Query and insert a new row into `HR.REGIONS` table of primary database. Then confirm that the inserted new row can be queried against the physical standby database while Redo Apply is active.

Practice 7-3: Performing Switchover

1. Perform the switchover to the standby database.

2. Switch back to `orcl` (the original primary database) and confirm that `orcl` is the primary database.

Practice 7-4: Creating and Managing a Snapshot Standby Database

1. Configure the Flashback Database on your primary and physical standby databases.
2. Convert your physical standby database to a snapshot standby database.
3. Update your database.

   Primary:
   ```
   insert into hr.regions values(10,'Australia');
   commit;
   ```

   Standby:
   ```
   insert into hr.regions values(20,'Australia');
   commit;
   ```

   Query the `hr.regions` table in your snapshot standby database.
4. Convert your snapshot standby database back to a physical standby database.
5. Verify the shipment of redo data.

Practice 7-5: Configuring RMAN Parameters

1. In your RMAN session (connected to your primary database), configure the backup retention policy to allow recovery for seven days.

2. Specify that the archived redo log files can be deleted after they are applied to the standby database.

Practice 7-6: Setting the Data Protection Mode

1. Set the protection mode to maximum availability.

Practice 7-7: Enabling Fast-Start Failover

1. Set fast-start failover properties and enable fast-start failover.

2. Start the observer.

3. Test fast-start failover by aborting `orcl`.

4. Perform a switchover to your `orcl` database and stop the observer.

Practice A-1: Installing Oracle Grid Infrastructure

In this practice, you install Oracle Grid Infrastructure.

1. Use the Oracle Universal Installer (runInstaller) and install Oracle Grid Infrastructure using the following information.

Hosts	host01, host02
OS user	oracle/oracle
GNS	Do not use
ORACLE_BASE	/u01/app/grid
Software Location	/u01/app/11.2.0/grid
Inventory	/u01/app/oraInventory
cluster name	cluster01
SCAN name	cluster01-scan.example.com
Voting Disk/OCR Location	ASM:DATA diskgroup
Diskgroup	DATA(Normal Redundancy) (ASMDISK01,ASMDISK02,ASMDISK03,ASMDISK04) FRA(External Redundancy) (ASMDISK05,ASMDISK06,ASMDISK07,ASMDISK08)
Network	eth0:Public eth1&eth2:Private

Practice A-2: Installing Oracle Database Software

Now that Grid Infrastructure has been installed, you install Oracle Database 11*g* Release 2 software.

1. Use the Oracle Universal Installer (runInstaller) and install Oracle Database 11*g* Release 2 software using the following information:

ORACLE_BASE	/u01/app/oracle
Software Location	/u01/app/oracle/product/11.2.0/dbhome_1
Inventory	/u01/app/oraInventory

Practice A-3: Creating a RAC database

In this practice, you create a two-node RAC database.

1. Launch the Database Configuration Assistant (DBCA) and create a two-node database using the following information:

Template	General purpose
Type	Policy-managed
Server pool name	sp1
Database Name	orcl
SYS/SYSTEM password	oracle
Database files location	DATA diskgroup
Fast Recovery Area	FRA diskgroup
Fast Recovery Area Size	4GB
Memory Target	800MB
Sample schema	Yes
Enterprise Manager	Not needed