Oracle Database 12c: SQL Work shop II

Activity Guide

D80194GC11

Edition 1.1

July 2014

D87272

Table of Contents

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1: Overview

Practice Overview

You also learn about your user account that you use in this course. You then start SQL Developer, create a new database connection, and browse your HR tables. You also set some SQL Developer preferences, execute SQL statements, and execute an anonymous PL/SQL block by using SQL Worksheet.

Practice 1-1: Using SQL Developer

Tasks

1. Start SQL Developer by using the desktop icon.
2. Create a database connection using the following information:

   Connection Name: myconnection

   Username: ora21

   Password: ora21

   Hostname: localhost

   Port: 1521

   SID: orcl (or the value provided to you by the instructor)

3. Test the new connection. If the status is Success, connect to the database by using this new connection.
   a. Click the Test button in the New/Select Database Connection window.
   b. If the status is Success, click the Connect button.
4. Browse the structure of the EMPLOYEES table and display its data.
   a. Expand the myconnection connection by clicking the plus sign next to it.
   b. Expand the Tables icon by clicking the plus sign next to it.
   c. Display the structure of the EMPLOYEES table.
   d. View the data of the DEPARTMENTS table.
5. Execute some basic SELECT statements to query the data in the EMPLOYEES table in the SQL Worksheet area. Use both the Execute Statement (or press F9) and the Run Script (or press F5) icons to execute the SELECT statements. Review the results of both methods of executing the SELECT statements on the appropriate tabbed pages.
   a. Write a query to select the last name and salary for any employee whose salary is less than or equal to $3,000.
   b. Write a query to display last name, job ID, and commission for all employees who are not entitled to receive a commission.
6. Set your script pathing preference to /home/oracle/labs/sql2.
   a. Select Tools > Preferences > Database > Worksheet.
   b. Enter the value in the “Select default path to look for scripts” field.
7. Enter the following in the Enter SQL Statement box.

```
SELECT employee_id, first_name, last_name
        FROM    employees;
```

8. Save the SQL statement to a script file by using the File > Save menu item.
   a. Select File > Save.
   b. Name the file intro_test.sql.
   c. Place the file under your /home/oracle/labs/sql2/labs folder.
9. Open and run confidence.sql from your /home/oracle/labs/sql2/labs folder, and observe the output.

Practices for Lesson 2: Introduction to Data Dictionary Views

Chapter 2

Practices for Lesson 2: Overview

Practice overview

This practice covers the following topics:

- Querying the dictionary views for table and column information
- Querying the dictionary views for constraint information
- Adding a comment to a table and querying the dictionary views for comment information

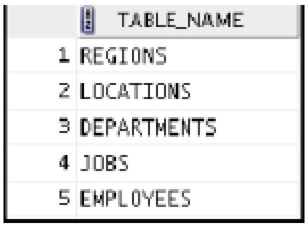
Practice 2-1: Introduction to Data Dictionary Views

Overview

In this practice, you query the dictionary views to find information about objects in your schema.

Tasks

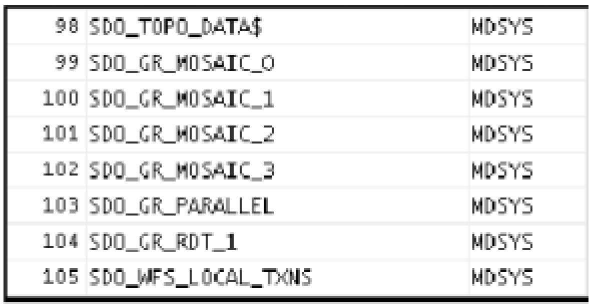
1. Query the USER_TABLES data dictionary view to see information about the tables that you own.



	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES

...

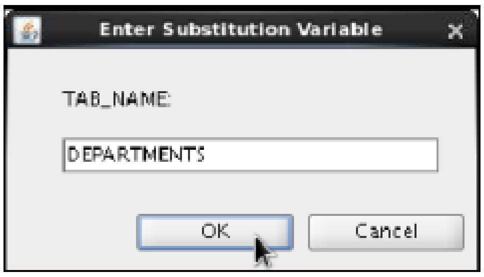
2. Query the ALL_TABLES data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.
   Note: Your list may not exactly match the following list:



98	SDO_TOPO_DATA$	MDSYS
99	SDO_GR_MOSAIC_0	MDSYS
100	SDO_GR_MOSAIC_1	MDSYS
101	SDO_GR_MOSAIC_2	MDSYS
102	SDO_GR_MOSAIC_3	MDSYS
103	SDO_GR_PARALLEL	MDSYS
104	SDO_GR_RDT_1	MDSYS
105	SDO_WFS_LOCAL_TXNS	MDSYS

...

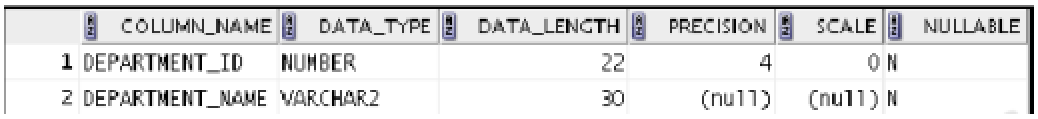
3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the DATA_PRECISION and DATA_SCALE columns. Save this script in a file named lab_02_03.sql.
   For example, if the user enters DEPARTMENTS, the following output results:



Enter Substitution Variable

TAB_NAME:

DEPARTMENTS

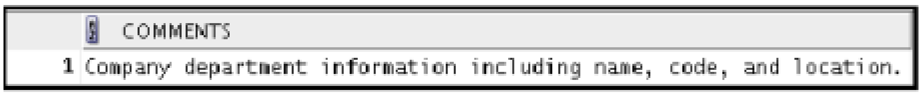


	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	PRECISION	SCALE	NULLABLE
1	DEPARTMENT_ID	NUMBER	22	4	0	N
2	DEPARTMENT_NAME	VARCHAR2	30	(null)	(null)	N

….

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER_CONSTRAINTS and USER_CONS_COLUMNS tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named lab_02_04.sql.

5. Add a comment to the DEPARTMENTS table. Then query the USER_TAB_COMMENTS view to verify that the comment is present.



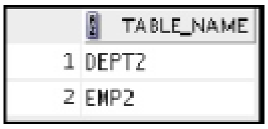
	COMMENTS
1	Company department information including name, code, and location.

6. Run the lab_02_06_tab.sql script as a prerequisite for exercises 6 through 9. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:

   Drops the existing DEPT2 and EMP2 tables

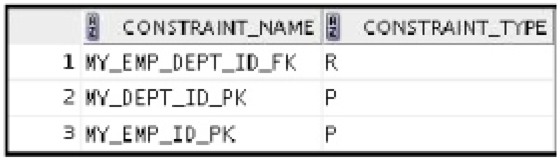
   Creates the DEPT2 and EMP2 tables
   Note: In Practice 2, you should have already dropped the DEPT2 and EMP2 tables so that they cannot be restored.

7. Confirm that both the DEPT2 and EMP2 tables are stored in the data dictionary.



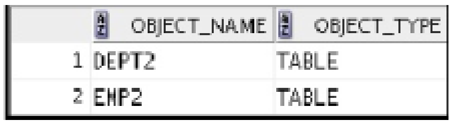
	TABLE_NAME
1	DEPT2
2	EMP2

8. Confirm that the constraints were added, by querying the USER_CONSTRAINTS view. Note the types and names of the constraints.



	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	MY_EMP_DEPT_ID_FK	R
2	MY_DEPT_ID_PK	P
3	MY_EMP_ID_PK	P

9. Display the object names and types from the USER_OBJECTS data dictionary view for the EMP2 and DEPT2 tables.



	OBJECT_NAME	OBJECT_TYPE
1	DEPT2	TABLE
2	EMP2	TABLE

Practices for Lesson 3: Creating Sequences, Synonyms, and Indexes

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

This practice covers the following topics:

- Creating sequences
- Using sequences
- Querying the dictionary views for sequence information
- Creating synonyms
- Querying the dictionary views for synonyms information
- Creating indexes
- Querying the dictionary views for indexes information

Note: Before starting this practice, execute /home/oracle/sql2/code_ex/code_ex_scripts/clean_up_scripts/cleanup_03.sql script.

Practice 3-1: Creating Sequences, Synonyms, and Indexes

Overview

This practice provides you with a variety of exercises in creating and using a sequence, an index, and a synonym.

Note: Execute cleanup_03.sql script from /home/oracle/sql2/code_ex/code_ex_scripts/clean_up_scripts/ before performing the following tasks.

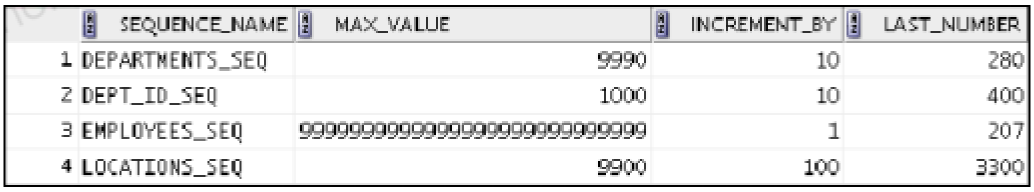
Tasks

1. Create the DEPT table based on the following table instance chart. Confirm that the table is created.

Column Name	ID	NAME
Key Type	Primary key	
Null/Unique		
FK Table		
FK Column		
Data Type	NUMBER	VARCHAR2
Length	7	25

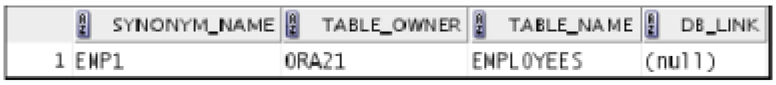
2. You need a sequence that can be used with the PRIMARY KEY column of the DEPT table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence DEPT_ID_SEQ.
3. To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab_03_03.sql. Be sure to use the sequence that you created for the ID

   column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.
4. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script lab_03_04.sql. Run the statement in your script.



	SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
1	DEPARTMENTS_SEQ	9990	10	280
2	DEPT_ID_SEQ	1000	10	400
3	EMPLOYEES_SEQ	9999999999999999999999999999	1	207
4	LOCATIONS_SEQ	9900	100	3300

5. Create a synonym for your EMPLOYEES table. Call it EMP1. Then find the names of all synonyms that are in your schema.



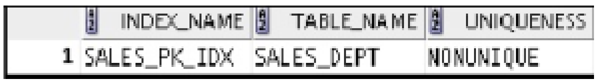
	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	EMP1	ORA21	EMPLOYEES	(null)

6. Drop the EMP1 synonym.

7. Create a nonunique index on the NAME column in the DEPT table.

8. Create the SALES_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column SALES_PK_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30



	INDEX_NAME	TABLE_NAME	UNIQUENESS
1	SALES_PK_IDX	SALES_DEPT	NONUNIQUE

9. Drop the tables and sequences created in this practice.

Practices for Lesson 4: Creating Views

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

This practice covers the following topics:

- Creating a simple view
- Creating a complex view
- Creating a view with a check constraint
- Attempting to modify data in the view
- Querying the dictionary views for view information
- Removing views

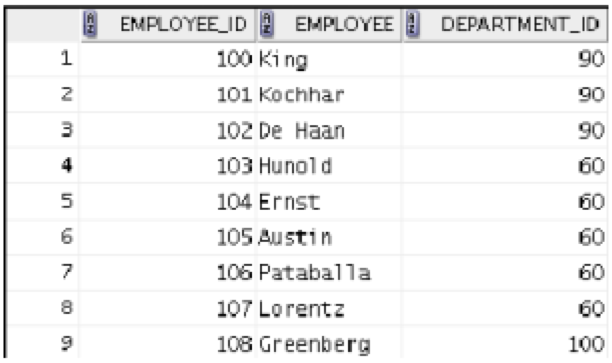
Practice 4-1: Creating Views

Overview:

This lesson's practice provides you with a variety of exercises in creating, using, querying data dictionary views for view information, and removing views.

Tasks:

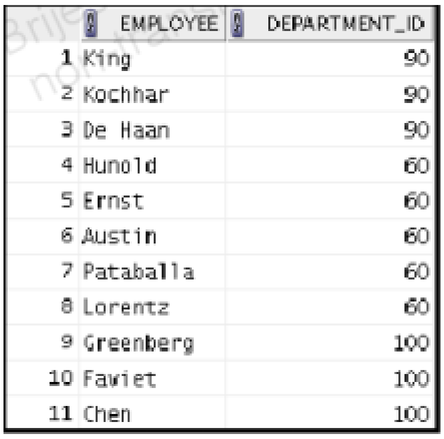
1. The staff in the HR department wants to hide some of the data in the EMPLOYEES table. Create a view called EMPLOYEES_VU based on the employee numbers, employee last names, and department numbers from the EMPLOYEES table. The heading for the employee name should be EMPLOYEE.
2. Confirm that the view works. Display the contents of the EMPLOYEES_VU view.



	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
5	104	Ernst	60
6	105	Austin	60
7	106	Pataballa	60
8	107	Lorentz	60
9	108	Greenberg	100

...

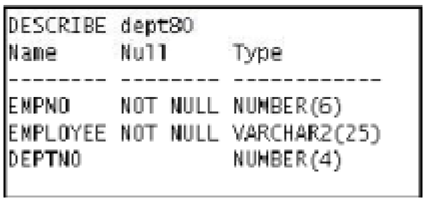
3. Using your EMPLOYEES_VU view, write a query for the HR department to display all employee names and department numbers.



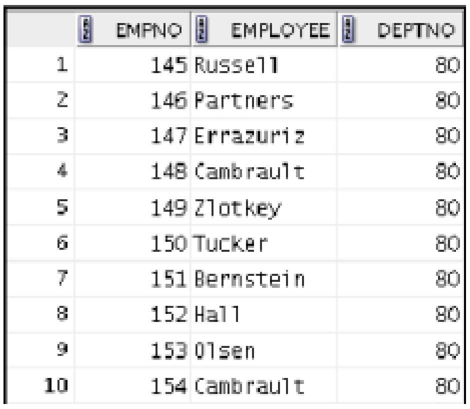
	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60
6	Austin	60
7	Pataballa	60
8	Lorentz	60
9	Greenberg	100
10	Faviet	100
11	Chen	100

...

4. Department 80 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 80. You have been asked to label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.
5. Display the structure and contents of the DEPT80view.



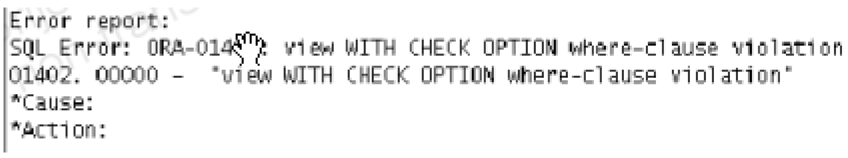
```
DESCRIBE dept80
Name     Null     Type
-------- -------- ------------
EMPNO    NOT NULL NUMBER(6)
EMPLOYEE NOT NULL VARCHAR2(25)
DEPTNO            NUMBER(4)
```



	EMPNO	EMPLOYEE	DEPTNO
1	145	Russell	80
2	146	Partners	80
3	147	Errazuriz	80
4	148	Cambrault	80
5	149	Zlotkey	80
6	150	Tucker	80
7	151	Bernstein	80
8	152	Hall	80
9	153	Olsen	80
10	154	Cambrault	80

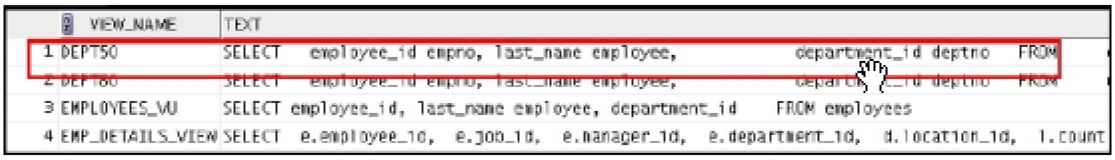
...

6. Test your view. Attempt to reassign Abel to department 80.



```
Error report:
SQL Error: ORA-014  : view WITH CHECK OPTION where-clause violation
01402. 00000 -  "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
```

7. Run lab_04_07.sql to create the dept50 view for this exercise.
You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the USER_VIEWS data dictionary view.
Note: The EMP_DETAILS_VIEW was created as part of your schema.
Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL*Plus, to see more contents of a LONG column, use the SET LONG ^{n} command, where is the value of the number of characters of the LONG column that you want to see.



	VIEW_NAME	TEXT
1	DEPT50	SELECT employee_id empno, last_name employee, department_id deptno FROM
2	DEPT80	SELECT employee_id empno, last_name employee, depar[illegible]_id deptno FROM
3	EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees
4	EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.count

8. Remove the views created in this practice.

Practices for Lesson 5: Managing Schema Objects

Chapter 5

Practices for Lesson 5: Overview

Practice Overview

This practice covers the following topics:

- Adding and dropping constraints
- Deferring constraints
- Creating external tables

Note: Before starting this practice, execute /home/oracle/labs/sql2/code_ex/ /cleanup_scripts/cleanup_05.sql script.

Practice 5-1: Managing Schema Objects

Overview

In this practice, you add, drop, and defer constraints. You create external tables.

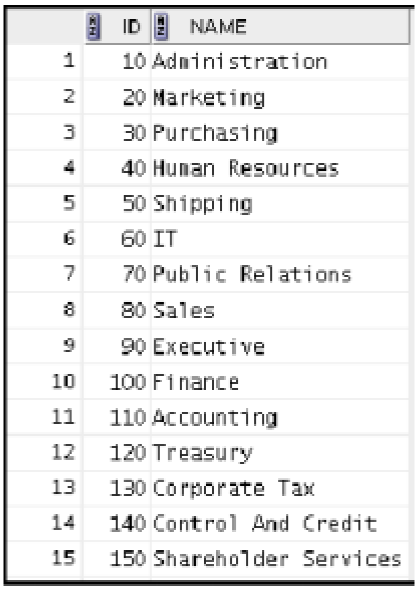
Note: Execute cleanup_05.sql script from /home/oracle/labs/sql2/code_ex/ /cleanup_scripts/ before performing the following tasks.

Tasks

1. Create the DEPT2 table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

2. Populate the DEPT2 table with data from the DEPARTMENTS table. Include only the columns that you need. Confirm that the rows are inserted.



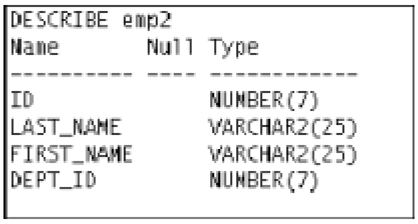
	ID	NAME
1	10	Administration
2	20	Marketing
3	30	Purchasing
4	40	Human Resources
5	50	Shipping
6	60	IT
7	70	Public Relations
8	80	Sales
9	90	Executive
10	100	Finance
11	110	Accounting
12	120	Treasury
13	130	Corporate Tax
14	140	Control And Credit
15	150	Shareholder Services

...

3. Create the EMP2 table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID

Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	V ARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7



```
DESCRIBE emp2
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(25)
FIRST_NAME      VARCHAR2(25)
DEPT_ID         NUMBER(7)
```

4. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.
5. Create a PRIMARY KEY constraint to the DEPT2 table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.
6. Add a foreign key reference on the EMP2 table that that the employee is not ensures assigned to a nonexistent department. Name the my_emp_dept_id_fk. constraint

7. Modify the EMP2 table. Add a COMMISSION column of the NUMBER data type, precision 2, scale 2. Add a constraint to the COMMISSION column that ensures that a commission value is greater than zero.
8. Drop the EMP2 and DEPT2 tables so that they cannot be restored.
9. Create an external table library_items_ext. Use the ORACLE_LOADER access driver.

   Note: The emp_dir directory and library_items.dat file are already created for this exercise. library_items.dat has records in the following format:

   2354,2264, 13.21, 150,

   2355,2289, 46.23, 200,

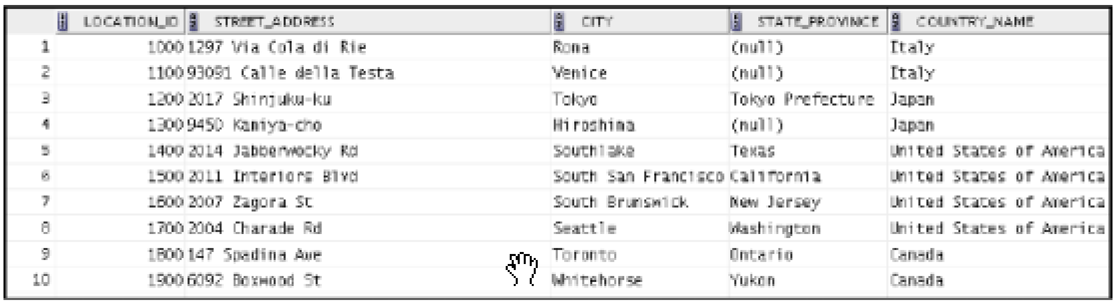
   2355,2264, 50.00, 100,

   a. Open the lab_05_09.sql file. Observe the code snippet to create the library_items_ext external table. Then replace **<**TODO1**>**, **<**TODO2 **>**, **<**TODO3**>**, and <TODO4> as appropriate and save the file as lab_05_09_soln.sql. Run the script to create the external table.
   b. Query the library_items_ext table.

10. The HR department needs a report of the addresses of all departments. Create an external table as dept_add_ext using the ORACLE_DATAPUMP access driver. The report should show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.

    Note: The emp_dir directory is already created for this exercise.

    a. Open the lab_05_10.sql file. Observe the code snippet to create the dept_add_ext external table. Then, replace <TODO1>, <TODO2 >, and <TODO3> with the appropriate code. Replace <oraxx_emp4.exp> and <oraxx_emp5.exp> with the appropriate file names. For example, if you are the ora21 user, your file names are ora21_emp4.exp and ora21_emp5.exp. Save the script as lab_05_10_soln.sql.
    b. Run the lab_05_10_soln.sql script to create the external table.
    c. Query the dept_add_ext table.

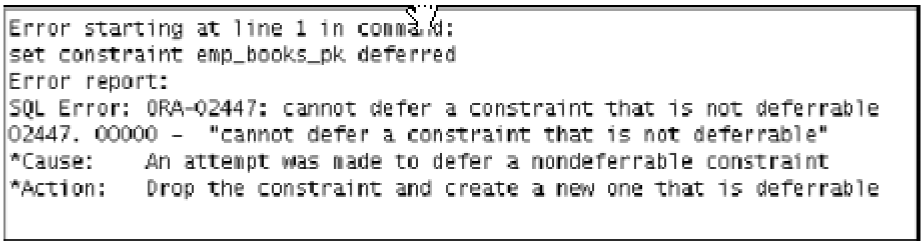


	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1000	1297 Via Cola di Rie	Roma	(null)	Italy
2	1100	93091 Calle della Testa	Venice	(null)	Italy
3	1200	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	1300	9450 Kamiya-cho	Hiroshima	(null)	Japan
5	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
6	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
7	1600	2007 Zagora St	South Brunswick	New Jersey	United States of America
8	1700	2004 Charade Rd	Seattle	Washington	United States of America
9	1800	147 Spadina Ave	Toronto	Ontario	Canada
10	1900	6092 Boxwood St	Whitehorse	Yukon	Canada

Note: When you perform the preceding step, two files oraxx_emp4.exp and oraxx_emp5.exp are created under the default directory emp_dir.

11. Create the emp_books table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

    a. Run the lab_05_11_a.sql file to create the emp_books table. Observe that the emp_books_pk primary key is not created as deferrable.
    b. Run the lab_05_11_b.sql file to populate data into the emp_books table. What do you observe?
    c. Set the emp_books_pk constraint as deferred. What do you observe?



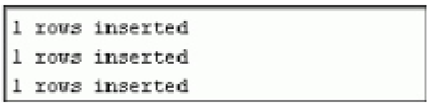
```
Error starting at line 1 in command:
set constraint emp_books_pk deferred
Error report:
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 -  "cannot defer a constraint that is not deferrable"
*Cause:    An attempt was made to defer a nondeferrable constraint
*Action:   Drop the constraint and create a new one that is deferrable
```

    d. Drop the emp_books_pk constraint.

e. Modify the emp_books table definition to add the emp_books_pk constraint as deferrable this time.

f. Set the emp_books_pk constraint as deferred.

g. Run the lab_05_11_g.sql file to populate data into the emp_books table. What do you observe?



```
1 rows inserted
1 rows inserted
1 rows inserted
```

h. Commit the transaction. What do you observe?

Practices for Lesson 6: Retrieving Data by Using Subqueries

Chapter 6

Practices for Lesson 6: Overview

Practice Overview

This practice covers the following topics:

- Creating multiple-column subqueries
- Writing correlated subqueries
- Using the EXISTS operator
- Using scalar subqueries
- Using the WITH clause

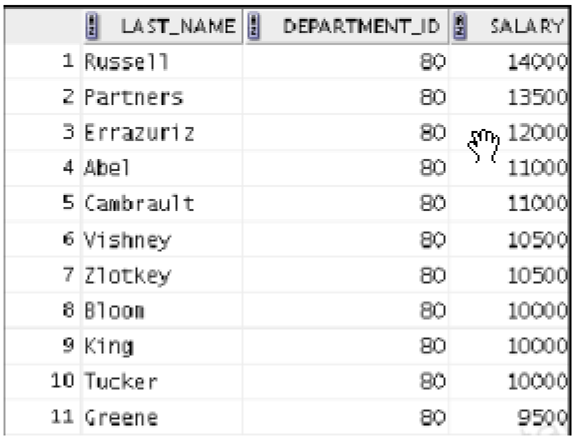
Practice 6-1: Retrieving Data by Using Subqueries

Overview

In this practice, you write multiple-column subqueries, and correlated and scalar subqueries. You also solve problems by writing the WITH clause.

Tasks

1. Write a query to display the last name, department number, and salary of any employee whose department number and salary both match the department number and salary of any employee who earns a commission.

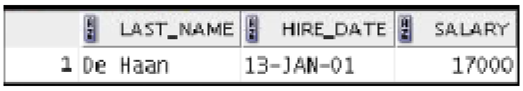


	LAST_NAME	DEPARTMENT_ID	SALARY
1	Russell	80	14000
2	Partners	80	13500
3	Errazuriz	80	12000
4	Abel	80	11000
5	Cambrault	80	11000
6	Vishney	80	10500
7	Zlotkey	80	10500
8	Bloom	80	10000
9	King	80	10000
10	Tucker	80	10000
11	Greene	80	9500

…

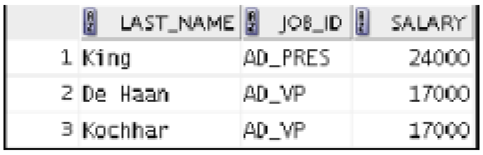
2. Display the last name, department name, and salary of any employee whose salary and job_ID match the salary and job_ID of any employee located in location ID 1700.

3. Create a query to display the last name, hire date, and salary for all employees who have the same salary and manager_ID as Kochhar.
   Note: Do not display Kochhar in the result set.



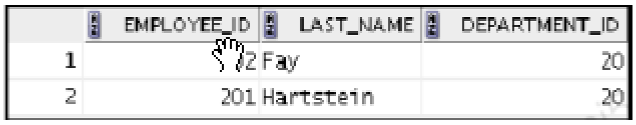
	LAST_NAME	HIRE_DATE	SALARY
1	De Haan	13-JAN-01	17000

4. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (JOB_ID = 'SA_MAN'). Sort the results from the highest to the lowest.



	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	De Haan	AD_VP	17000
3	Kochhar	AD_VP	17000

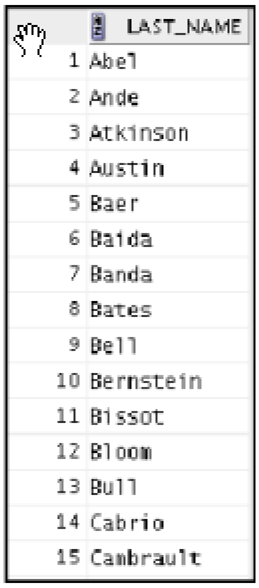
5. Display details such as the employee ID, last name, and department ID of those employees who live in cities the names of which begin with T .



	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	[illegible]2	Fay	20
2	201	Hartstein	20

6. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

7. Find all employees who are not supervisors.
   a. First, do this by using the NOT EXISTS operator.

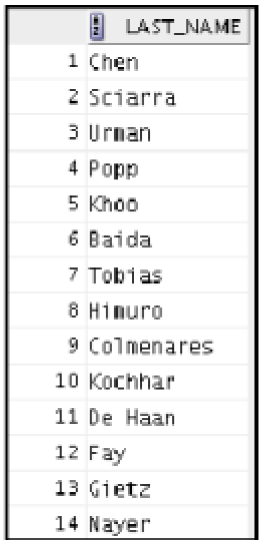


	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda
8	Bates
9	Bell
10	Bernstein
11	Bissot
12	Bloom
13	Bull
14	Cabrio
15	Cambrault

…

   b. Can this be done by using the NOT IN operator? How, or why not? If not, try out using another solution.

8. Write a query to display the last names of the employees who earn less than the average salary in their departments.



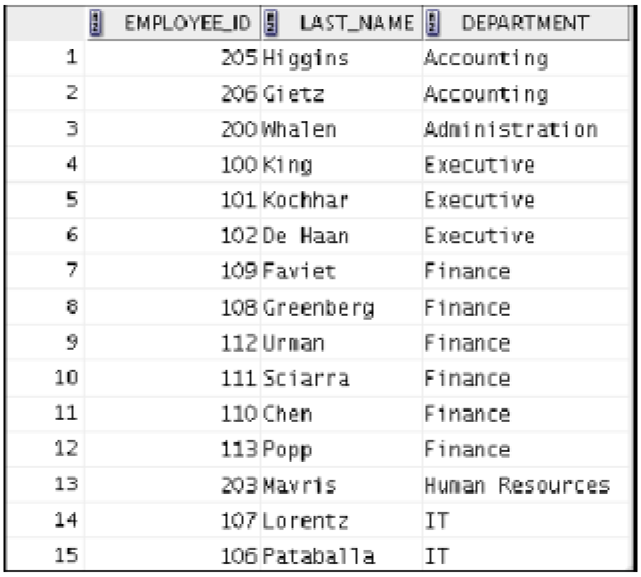
	LAST_NAME
1	Chen
2	Sciarra
3	Urman
4	Popp
5	Khoo
6	Baida
7	Tobias
8	Himuro
9	Colmenares
10	Kochhar
11	De Haan
12	Fay
13	Gietz
14	Mayer

…

9. Write a query to display the last names of the employees who have one or more coworkers in their departments with later hire dates but higher salaries.

10. Write a query to display the employee ID, last names, and department names of all the employees.

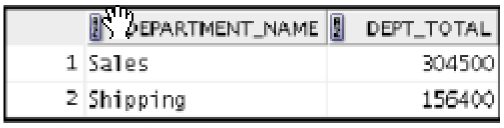
    Note: Use a scalar subquery to retrieve the department name in the SELECT statement.



	EMPLOYEE_ID	LAST_NAME	DEPARTMENT
1	205	Higgins	Accounting
2	206	Gietz	Accounting
3	200	Whalen	Administration
4	100	King	Executive
5	101	Kochhar	Executive
6	102	De Haan	Executive
7	109	Faviet	Finance
8	108	Greenberg	Finance
9	112	Urman	Finance
10	111	Sciarra	Finance
11	110	Chen	Finance
12	113	Popp	Finance
13	203	Mavris	Human Resources
14	107	Lorentz	IT
15	106	Pataballa	IT

...

11. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.



	DEPARTMENT_NAME	DEPT_TOTAL
1	Sales	304500
2	Shipping	156400

Practices for Lesson 7: Manipulating Data by Using Subqueries

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

This practice covers the following topics:

- Using subqueries to manipulate data
- Inserting by using a subquery as a target
- Using the WITH CHECK OPTION keyword on DML statements
- Using correlated subqueries to update and delete rows

Practice 7-1: Manipulating Data by Using Subqueries

Overview

In this practice, you test your knowledge about using subqueries to manipulate data, using the WITH CHECK OPTION keyword on DML statements, and correlated subqueries to update and delete rows.

Tasks

1. Which of the following statements are true?
   a. Subqueries are used to retrieve data by using an inline view.
   b. Subqueries cannot be used to copy data from one table to another.
   c. Subqueries update data in one table based on the values of another table.
   d. Subqueries delete rows from one table based on rows in another table.

2. Fill in the blanks:
   a. You can use a subquery in place of the table name in the ______ clause of the INSERT statement.

   Options:
   1) FROM
   2) INTO
   3) FOR UPDATE
   4) VALUES

3. The WITH CHECK OPTION keyword prohibits you from changing rows that are not in the subquery.
   a. TRUE
   b. FALSE

4. The SELECT list of this subquery must have the same number of columns as the column list of the VALUES clause.
   a. TRUE
   b. FALSE

5. You can use a correlated subquery to delete only those rows that also exist in another table.
   a. TRUE
   b. FALSE

6. To understand the concepts of WITH CHECK OPTION and correlated subqueries, run the demo files for this practice.

Practices for Lesson 8: Controlling User Access

Chapter 8

Practices for Lesson 8: Overview

Practice Overview:

This practice covers the following topics:

- Granting other users privileges to your table
- Modifying another user's table through the privileges granted to you

Practice 8-1: Controlling User Access

Overview

You grant query privilege on your table to another user. You learn how to control access to database objects.

Tasks

1. What privilege should a user be given to log on to the Oracle server? Is this a system privilege or an object privilege?

   ____

2. What privilege should a user be given to create tables?

   ____

3. If you create a table, who can pass along privileges to other users in your table?

   ____

4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?

   ____

5. What command do you use to change your password?

   ____

6. User21 is the owner of the EMP table and grants the DELETE privilege to User22 by using the WITH GRANT OPTION clause. User22 then grants the DELETE privilege on EMP to User23. User21 now finds that User23 has the privilege and revokes it from User22. Which user can now delete from the EMP table?

   ____

7. You want to grant SCOTT the privilege to update data in the DEPARTMENTS table. You also want to enable SCOTT to grant this privilege to other users. What command do you use?

   ____

To complete question 8 and the subsequent ones, you need to connect to the database by using SQL Developer. If you are already not connected, do the following to connect:

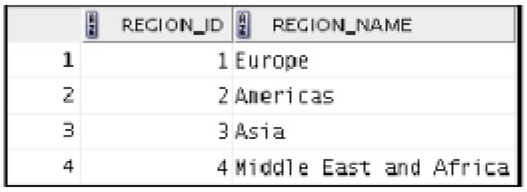
1. **Click the SQL Developer desktop icon.**
2. **In the Connections Navigator, use the** ora21 **account and the corresponding password provided by your instructor to log on to the database.**
3. **Open another SQL Developer session and connect as**ora22**.**

8. Grant another user query privilege on your table. Then, verify whether that user can use the privilege.

   Note: For this exercise, open another SQL Developer session and connect as a different user. For example, if you are currently using ora21, open another SQL Developer session and connect as ora22. Here onwards we would refer the first SQL Developer session as Team 1 and the second SQL Developer session as Team 2.

   a. Grant another user (for example, ora22) privilege to view records in your REGIONS table. Include an option for this user to further grant this privilege to other users.

   b. Have the user query your REGIONS table.

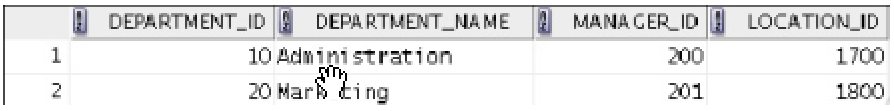


	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

   c. Have the user pass on the query privilege to a third user, ora23.

d. Take back the privilege from the user who performs step b.

9. Grant another user query and data manipulation privileges on your COUNTRIES table. Make sure that the user cannot pass on these privileges to other users.
10. Take back the privileges on the COUNTRIES table granted to another user.
11. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.
12. Query all the rows in your DEPARTMENTS table.



	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800

. .

13. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources as department number 510. Query the other team's table.
14. Create a synonym for the other team's DEPARTMENTS table.
15. Query all the rows in the other team's DEPARTMENTS table by using your synonym.
16. Revoke the SELECT privilege from the other team.
17. Remove the row that you inserted into the DEPARTMENTS table in step 13 and save the changes.
18. Drop the synonyms team 1 and team 2.

Practices for Lesson 9: Manipulating Data

Chapter 9

Practices for Lesson 9: Overview

Practice overview:

This practice covers the following topics:

- Performing multitable INSERTs
- Performing MERGE operations
- Performing flashback operations
- Tracking row versions

Note: Before starting this practice, execute /home/oracle/labs/sql2/code_ex/ script.

Practice 9-1: Manipulating Data

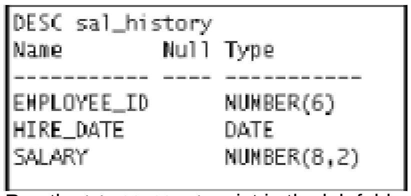
Overview

In this practice, you perform multitable INSERT and MERGE operations, flashback operation, and track row versions.

Note: Execute cleanup_09.sql script from **/**home/oracle/labs/sql2/code_ex/ cleanup_scripts/ before performing the following tasks.

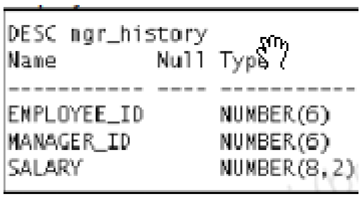
Tasks

1. Run the lab_09_01.sql script in the lab folder to create the SAL_HISTORY table.
2. Display the structure of the SAL_HISTORY table.



```
DESC sal_history
Name        Null Type
----------- ---- -----------
EMPLOYEE_ID      NUMBER(6)
HIRE_DATE        DATE
SALARY           NUMBER(8,2)
```

3. Run the lab_09_03.sql script in the lab folder to create the MGR_HISTORY table.
4. Display the structure of the MGR_HISTORY table.



```
DESC mgr_history
Name        Null Type
----------- ---- -----------
EMPLOYEE_ID      NUMBER(6)
MANAGER_ID       NUMBER(6)
SALARY           NUMBER(8,2)
```

5. Run the lab_09_05.sql script in the lab folder to create the SPECIAL_SAL table.
6. Display the structure of the SPECIAL_ SAL table.
7.
   a. Write a query to do the following:

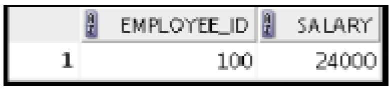
      Retrieve details such as the employee ID, hire date, salary, and manager ID of those employees whose employee ID is less than 125 from the EMPLOYEES table.

      If the salary is more than $20,000, insert details such as the employee ID and salary into the SPECIAL_SAL table.

      If the salary is less than $20,000:

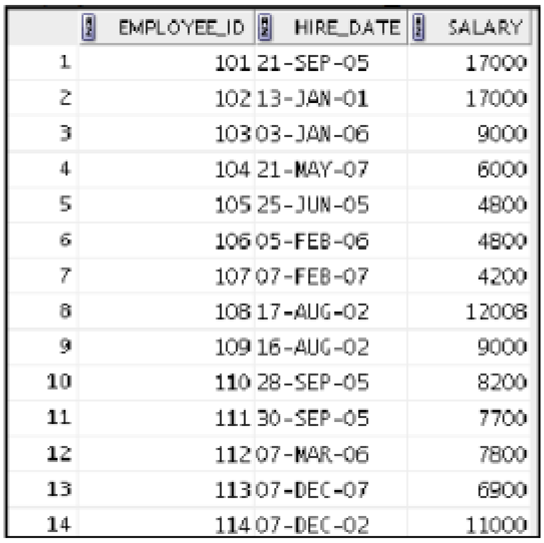
      - Insert details such as the employee ID, hire date, and salary into the SAL_HISTORY table
      - Insert details such as the employee ID, manager ID, and salary into the MGR_HISTORY table

   b. Display the records from the SPECIAL_SAL table.



	EMPLOYEE_ID	SALARY
1	100	24000

c. Display the records from the SAL_HISTORY table.



	EMPLOYEE_ID	HIRE_DATE	SALARY
1	101	21-SEP-05	17000
2	102	13-JAN-01	17000
3	103	03-JAN-06	9000
4	104	21-MAY-07	6000
5	105	25-JUN-05	4800
6	106	05-FEB-06	4800
7	107	07-FEB-07	4200
8	108	17-AUG-02	12008
9	109	16-AUG-02	9000
10	110	28-SEP-05	8200
11	111	30-SEP-05	7700
12	112	07-MAR-06	7800
13	113	07-DEC-07	6900
14	114	07-DEC-02	11000

...

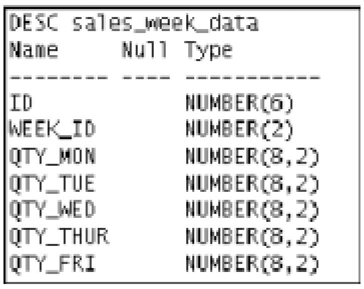
d. Display the records from the MGR_HISTORY table.

8.

a. Run the lab_09_08_a.sql script in the lab folder to create the SALES_WEEK_DATA table.

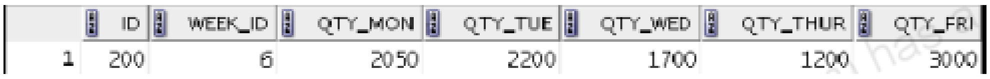
b. Run the lab_09_08_b.sql script in the lab folder to insert records into the SALES_WEEK_DATA table.

c. Display the structure of the SALES_WEEK_DATA table.



```
DESC sales_week_data
Name      Null Type
--------- ---- -----------
ID             NUMBER(6)
WEEK_ID        NUMBER(2)
QTY_MON        NUMBER(8,2)
QTY_TUE        NUMBER(8,2)
QTY_WED        NUMBER(8,2)
QTY_THUR       NUMBER(8,2)
QTY_FRI        NUMBER(8,2)
```

d. Display the records from the SALES_WEEK_DATA table.



	ID	WEEK_ID	QTY_MON	QTY_TUE	QTY_WED	QTY_THUR	QTY_FRI
1	200	6	2050	2200	1700	1200	3000

e. Run the lab_09_08_e.sql script in the lab folder to create the EMP_SALES_INFO table.

f. Display the structure of the EMP_SALES_INFO table.

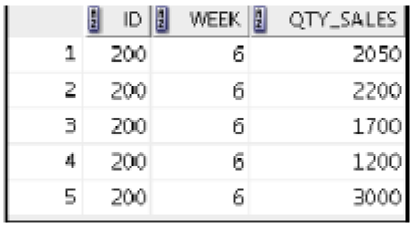
g. Write a query to do the following:

Retrieve details such as ID, week ID, sales quantity on Monday, sales quantity on Tuesday, sales quantity on Wednesday, sales quantity on Thursday, and sales quantity on Friday from the SALES_WEEK_DATA table.

Build a transformation such that each record retrieved from the SALES_WEEK_DATA table is converted into multiple records for the EMP_SALES_INFO table.

Hint: Use a pivoting INSERT statement.

h. Display the records from the EMP_SALES_INFO table.



	ID	WEEK	QTY_SALES
1	200	6	2050
2	200	6	2200
3	200	6	1700
4	200	6	1200
5	200	6	3000

9. You have the data of past employees stored in a flat file called emp.data. You want to store the names and email IDs of all employees, past and present, in a table. To do this, first create an external table called EMP_DATA using the emp.dat source file in the emp _dir directory. Use the lab_09_09.sql script to do this.
10. Run the lab_09_10.sql script to create the EMP_HIST table.
    a. Increase the size of the email column to 45.
    b. Merge the data in the EMP_DATA table created in the last lab into the data in the EMP_HIST table. Assume that the data in the external EMP_DATA table is the most up-to-date. If a row in the EMP_DATA table matches the EMP_HIST table, update the email column of the EMP_HIST table to match the EMP_DATA table row. If a row in the EMP_DATA table does not match, insert it into the EMP_HIST table. Rows are considered matching when the employee's first and last names are identical.
    c. Retrieve the rows from EMP_HIST after the merge.

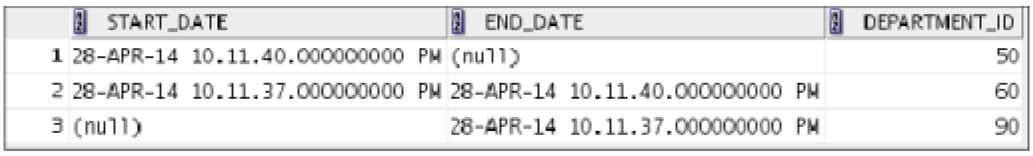
11. Create the EMP2 table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	V ARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

12. Drop the EMP2 table.
13. Query the recycle bin to see whether the table is present.
14. Restore the EMP2 table to a state before the DROP statement.
15. Create the EMP3 table using the lab_09_11.sql script. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar using the Row Versions feature.

```
UPDATE emp3 SET department_id = 60
WHERE last_name = 'Kochhar';
COMMIT;
UPDATE emp3 SET department_id = 50
WHERE last_name = 'Kochhar';
COMMIT;
```

```
SELECT VERSIONS_STARTTIME "START_DATE",
    VERSIONS_ENDTIME "END_DATE",     DEPARTMENT_ID
FROM EMP3
    VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME ='Kochhar';
```



	START_DATE	END_DATE	DEPARTMENT_ID
1	28-APR-14 10.11.40.000000000 PM	(null)	50
2	28-APR-14 10.11.37.000000000 PM	28-APR-14 10.11.40.000000000 PM	60
3	(null)	28-APR-14 10.11.37.000000000 PM	90

16. Drop the EMP2 and EMP3 tables so that they cannot be restored. Check in the recycle bin.

Practices for Lesson 10: Managing Data in Different Time Zones

Chapter 10

Practices for Lesson 10: Overview

Practice Overview:

This practice covers using the datetime functions.

Note: Before starting this practice, execute /home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql script.

Practice 10-1: Managing Data in Different Time Zones

Overview

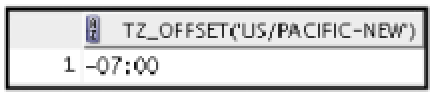
In this practice, you display time zone offsets, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP. You also set time zones and use the EXTRACT function.

Note: Execute cleanup_10.sql script from /home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql before performing the following tasks.

Tasks

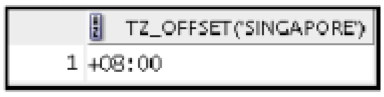
1. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY HH24:MI:SS.
2.
   a. Write queries to display the time zone offsets (TZ_OFFSET) for the following time zones.

   US/Pacific-New



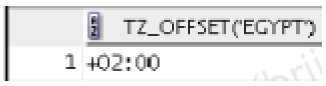
	TZ_OFFSET('US/PACIFIC-NEW')
1	-07:00

   Singapore



	TZ_OFFSET('SINGAPORE')
1	+08:00

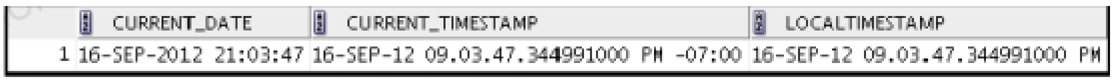
   Egypt



	TZ_OFFSET('EGYPT')
1	+02:00

   b. Alter the session to set the TIME_ZONE parameter value to the time zone offset of US/Pacific-New.

   c. Display CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

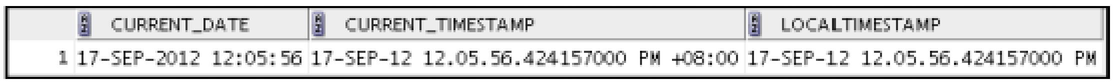


	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	16-SEP-2012 21:03:47	16-SEP-12 09.03.47.344991000 PM -07:00	16-SEP-12 09.03.47.344991000 PM

   d. Alter the session to set the TIME_ZONE parameter value to the time zone offset of Singapore.

   e. Display CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

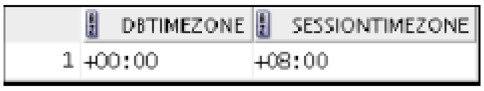
   Note: The output might be different based on the date when the command is executed.



	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	17-SEP-2012 12:05:56	17-SEP-12 12.05.56.424157000 PM +08:00	17-SEP-12 12.05.56.424157000 PM

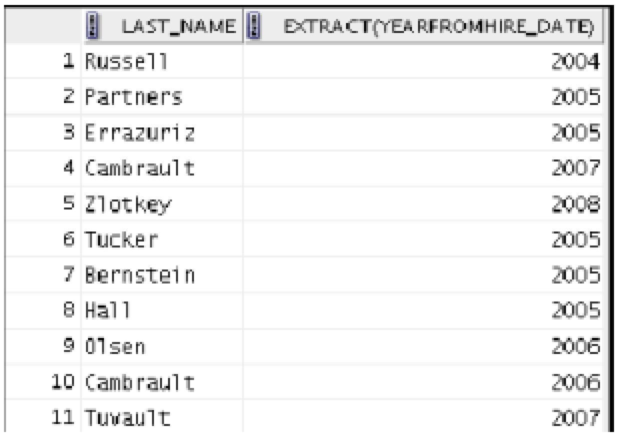
   Note: Observe in the preceding practice that CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP are sensitive to the session time zone.

3. Write a query to display DBTIMEZONE and SESSIONTIMEZONE.



	DBTIMEZONE	SESSIONTIMEZONE
1	+00:00	+08:00

4. Write a query to extract the YEAR from the HIRE_DATE column of the EMPLOYEES table for those employees who work in department 80.



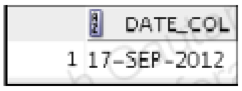
	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	2004
2	Partners	2005
3	Errazuriz	2005
4	Cambrault	2007
5	Zlotkey	2008
6	Tucker	2005
7	Bernstein	2005
8	Hall	2005
9	Olsen	2006
10	Cambrault	2006
11	Tuvault	2007

...

5. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY.
6. Examine and run the lab_10_06.sql script to create the SAMPLE_DATES table and populate it.

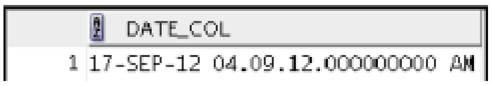
Note: The screenshot dates will change according to the sysdate.

a. Select from the table and view the data.



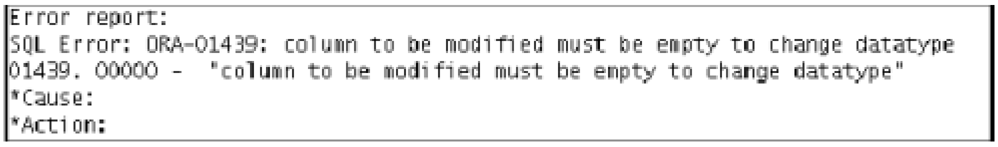
	DATE_COL
1	17-SEP-2012

b. Modify the data type of the DATE_COL column and change it to TIMESTAMP. Select from the table to view the data.



	DATE_COL
1	17-SEP-12 04.09.12.000000000 AM

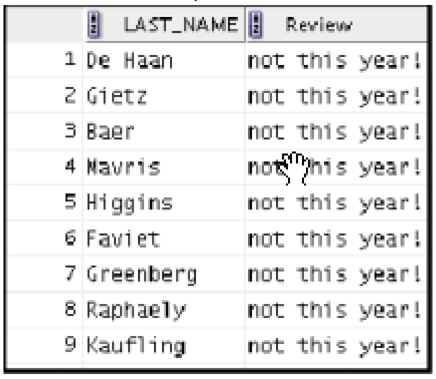
c. Try to modify the data type of the DATE_COL column and change it to TIMESTAMP WITH TIME ZONE. What happens?



```
Error report:
SQL Error: ORA-01439: column to be modified must be empty to change datatype
01439. 00000 -  "column to be modified must be empty to change datatype"
*Cause:
*Action:
```

7. Create a query to retrieve last names from the EMPLOYEES table and calculate the review status. If the year hired was 2008, display Needs Review for the review status; otherwise, display not this year! Name the review status column Review. Sort the results by the HIRE_DATE column.

Hint: Use a CASE expression with the EXTRACT function to calculate the review status.



	LAST_NAME	Review
1	De Haan	not this year!
2	Gietz	not this year!
3	Baer	not this year!
4	Mavris	not this year!
5	Higgins	not this year!
6	Faviet	not this year!
7	Greenberg	not this year!
8	Raphaely	not this year!
9	Kaufling	not this year!

...

8. Create a query to print the last names and the number of years of service for each employee. If the employee has been employed for five or more years, print 5 years of service. If the employee has been employed for 10 or more years, print 10 years of service. If the employee has been employed for 15 or more years, print 15 years of service. If none of these conditions matches, print maybe next year! Sort the results by the HIRE_DATE column. Use the EMPLOYEES table.
   Hint: Use CASE expressions and TO_YMINTERVAL.