Oracle Database Practice Questions

Created by: Lambert Almasi
Contacts:+255714011243
Email:lambert_dublin@yahoo.com

Practice 1

1. Initiate an *i*SQL*Plus session using the user ID and password provided by the instructor.
2. *i*SQL*Plus commands access the database. **False**
3. The following `SELECT` statement executes successfully: **True**

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

4. The following `SELECT` statement executes successfully: **TRue**

```
SELECT *
FROM   job_grades;
```

5. There are four coding errors in this statement. Can you identify them?

```
SELECT          employee_id,
last_name  sal  x  12   ANNUAL
SALARY
FROM       employees;
```

   - **The `EMPLOYEES` table does not contain a column called `sal`. The column is called `SALARY`.**
   - **The multiplication operator is *, not x, as shown in line 2.**
   - **The `ANNUAL SALARY` alias cannot include spaces. The alias should read `ANNUAL_SALARY` or be enclosed in double quotation marks.**
   - **A comma is missing after the column, `LAST_NAME`.**

6. Show the structure of the DEPARTMENTS table. Select all data from the DEPARTMENTS table.

7. Show the structure of the EMPLOYEES table. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first. Provide an alias STARTDATE for the HIRE_DATE column. Save your SQL statement to a file named lab1_7.sql.

8. Run your query in the file lab1_7.sql.

9. Create a query to display unique job codes from the EMPLOYEES table.

If you have time, complete the following exercises:

10. Copy the statement from lab1_7.sql into the *i*SQL*Plus Edit window. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Run your query again.

11. Display the last name concatenated with the job ID, separated by a comma and space, and name the column Employee and Title.

If you want an extra challenge, complete the following exercise:

12. Create a query to display all the data from the EMPLOYEES table. Separate each column by a comma. Name the column THE_OUTPUT.

Practice Questions 2

1. Create a query to display the last name and salary of employees earning more than $12,000. Place your SQL statement in a text file named lab2_1.sql. Run your query.

2. Create a query to display the employee last name and department number for employee number 176.

3. Modify lab2_1.sql to display the last name and salary for all employees whose salary is not in the range of $5,000 and $12,000. Place your SQL statement in a text file named lab2_3.sql.

4. Display the employee last name, job ID, and start date of employees hired between February 20, 1998, and May 1, 1998. Order the query in ascending order by start date.

5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.

6. Modify lab2_3.sql to list the last name and salary of employees who earn between $5,000 and $12,000, and are in department 20 or 50. Label the columns Employee and Monthly Salary, respectively. Resave lab2_3.sql as lab2_6.sql. Run the statement in lab2_6.sql.

7. Display the last name and hire date of every employee who was hired in 1994.

8. Display the last name and job title of all employees who do not have a manager.

9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

If you have time, complete the following exercises.

10. Display the last names of all employees where the third letter of the name is an *a*.

11. Display the last name of all employees who have an *a* and an *e* in their last name.

If you want an extra challenge, complete the following exercises:

12. Display the last name, job, and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to $2,500, $3,500, or $7,000.

13. Modify `lab2_6.sql` to display the last name, salary, and commission for all employees whose commission amount is 20%. Resave `lab2_6.sql` as `lab2_13.sql`. Rerun the statement in `lab2_13.sql`.

Practice Questions 3

1. Write a query to display the current date. Label the column `Date`.
2. For each employee, display the employee number, last_name, salary, and salary increased by 15% and expressed as a whole number. Label the column `New Salary`. Place your SQL statement in a text file named `lab3_2.sql`.

3. Run your query in the file `lab3_2.sql`.
4. Modify your query `lab3_2.sql` to add a column that subtracts the old salary from the new salary. Label the column `Increase`. Save the contents of the file as `lab3_4.sql`. Run the revised query.

5. Write a query that displays the employee's last names with the first letter capitalized and all other letters lowercase and the length of the name for all employees whose name starts with *J*, *A*, or *M*. Give each column an appropriate label. Sort the results by the employees' last names.

6. For each employee, display the employee's last name, and calculate the number of months between today and the date the employee was hired. Label the column `MONTHS_WORKED`. Order your results by the number of months employed. Round the number of months up to the closest whole number.

   Note: Your results will differ.

7. Write a query that produces the following for each employee: `<employee last name> earns <salary> monthly but wants <3 times salary>`. Label the column `Dream Salaries`.

If you have time, complete the following exercises:

8. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with $. Label the column `SALARY`.

9. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column `REVIEW`. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

10. Display the last name, hire date, and day of the week on which the employee started. Label the column `DAY`. Order the results by the day of the week starting with Monday.

If you want an extra challenge, complete the following exercises:

11. Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, put "No Commission." Label the column `COMM`.

12. Create a query that displays the employees' last names and indicates the amounts of their annual salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column `EMPLOYEES_AND_THEIR_SALARIES`.

13. Using the `DECODE` function, write a query that displays the grade of all employees based on the value of the column `JOB_ID`, as per the following data:

14. Rewrite the statement in the preceding question using the `CASE` syntax.

Practice 4 Questions

1. Write a query to display the last name, department number, and department name for all employees.

2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission.

4. Display the employee last name and department name for all employees who have an *a* (lowercase) in their last names. Place your SQL statement in a text file named `lab4_4.sql`.

5. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

6. Display the employee last name and employee number along with their manager's last name and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Place your SQL statement in a text file named `lab4_6.sql`.

7. Modify `lab4_6.sql` to display all employees including King, who has no manager. Place your SQL statement in a text file named `lab4_7.sql`. Run the query in `lab4_7.sql`

If you have time, complete the following exercises.

8. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label.

9. Show the structure of the `JOB_GRADES` table. Create a query that displays the name, job, department name, salary, and grade for all employees.

If you want an extra challenge, complete the following exercises:

10. Create a query to display the name and hire date of any employee hired after employee Davies.

11. Display the names and hire dates for all employees who were hired before their managers, along with their manager's names and hire dates. Label the columns `Employee`, `Emp Hired`, `Manager`, and `Mgr Hired`, respectively.

Practice 5

Determine the validity of the following three statements. Circle either True or False.

1. Group functions work across many rows to produce one result.

   True

2. Group functions include nulls in calculations.

   False. Group functions ignore null values. If you want to include null values, use the NVL function.

3. The `WHERE` clause restricts rows prior to inclusion in a group calculation.

   True

4. Display the highest, lowest, sum, and average salary of all employees. Label the columns `Maximum`, `Minimum`, `Sum`, and `Average`, respectively. Round your results to the nearest whole number. Place your SQL statement in a text file named `lab5_6.sql`.

5. Modify the query in `lab5_4.sql` to display the minimum, maximum, sum, and average salary for each job type. Resave `lab5_6.sql` to `lab5_4.sql`. Run the statement in `lab5_5.sql`.

6. Write a query to display the number of people with the same job.

7. Determine the number of managers without listing them. Label the column `Number of Managers`. *Hint: Use the `MANAGER_ID` column to determine the number of managers.*

8. Write a query that displays the difference between the highest and lowest salaries. Label the column `DIFFERENCE`.

If you have time, complete the following exercises.

9. Display the manager number and the salary of the lowest paid employee for that manager.

Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary

10. Write a query to display each department's name, location, number of employees, and the average salary for all employees in that department. Label the columns `Name`, `Location`, `Number of People`, and `Salary`, respectively. Round the average salary to two decimal places.

If you want an extra challenge, complete the following exercises:

11. Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

12. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

Practice 6 Questions

1. Write a query to display the last name and hire date of any employee in the same department as Zlotkey. Exclude Zlotkey.

2. Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.

3. Write a query that displays the employee numbers and last names of all employees who work in a department with any employee whose last name contains a *u*. Place your SQL statement in a text file named `lab6_3.sql`. Run your query.

4. Display the last name, department number, and job ID of all employees whose department location ID is 1700.

5. Display the last name and salary of every employee who reports to King.

6. Display the department number, last name, and job ID for every employee in the Executive department.

If you have time, complete the following exercises:

7. Modify the query in `lab6_3.sql` to display the employee numbers, last names, and salaries of all employees who earn more than the average salary and who work in a department with any employee with a *u* in their name. Resave `lab6_3.sql` to `lab6_7.sql`. Run the statement in `lab6_7.sql`.

Practice Questions 7

Determine whether the following statements are true or false:

1. The following statement is correct:

```
DEFINE & p_val = 100
```

False

The correct use of DEFINE is `DEFINE p_val=100`. The `&` is used within the SQL code.

2. The DEFINE command is a SQL command.

   False
   The DEFINE command is an iSQL*Plus command.

3. Write a script to display the employee last name, job, and hire date for all employees who started between a given range. Concatenate the name and job together, separated by a space and comma, and label the column Employees. In a separate SQL script file, use the DEFINE command to provide the two ranges. Use the format MM/DD/YYYY. Save the script files as lab7_3a.sql and lab7_3b.sql.

4. Write a script to display the employee last name, job, and department name for a given location. The search condition should allow for case-insensitive searches of the department location. Save the script file as lab7_4.sql.

5. Modify the code in lab7_4.sql to create a report containing the department name, employee last name, hire date, salary, and each employee's annual salary for all employees in a given location. Label the columns DEPARTMENT NAME, EMPLOYEE NAME, START DATE, SALARY, and ANNUAL SALARY, placing the labels on multiple lines. Resave the script as lab7_5.sql and execute the commands in the script.

Practice Questions 8

Insert data into the MY_EMPLOYEE table.

1. Run the statement in the lab8_1.sql script to build the MY_EMPLOYEE table that will be used for the lab.

2. Describe the structure of the MY_EMPLOYEE table to identify the column names.

3. Add the first row of data to the MY_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause.

4. Populate the MY_EMPLOYEE table with the second row of sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

5. Confirm your addition to the table.

6. Write an insert statement in a text file named loademp.sql to load rows into the MY_EMPLOYEE table. Concatenate the first letter of the first name and the first seven characters of the last name to produce the userid.

7. Populate the table with the next two rows of sample data by running the insert statement in the script that you created.

8. Confirm your additions to the table.

9. Make the data additions permanent.

Update and delete data in the MY_EMPLOYEE table.

10. Change the last name of employee 3 to Drexler.
11. Change the salary to 1000 for all employees with a salary less than 900.
12. Verify your changes to the table.
13. Delete Betty Dancs from the `MY_EMPLOYEE` table.
14. Confirm your changes to the table.
15. Commit all pending changes.

Control data transaction to the `MY_EMPLOYEE` table.

16. Populate the table with the last row of sample data by modifying the statements in the script that you created in step 6. Run the statements in the script.
17. Confirm your addition to the table.
18. Mark an intermediate point in the processing of the transaction.
19. Empty the entire table.
20. Confirm that the table is empty.
21. Discard the most recent `DELETE` operation without discarding the earlier `INSERT` operation.
22. Confirm that the new row is still intact.
23. Make the data addition permanent.

Practice 9 Questions

1. Create the `DEPT` table based on the following table instance chart. Place the syntax in a script called `lab9_1.sql`, then execute the statement in the script 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

```
CREATE TABLE dept
      (id
  NUMBER(7),
   name VARCHAR2(25));

DESCRIBE dept
```

2. Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.

3. Create the EMP table based on the following table instance chart. Place the syntax in a script called lab9_3.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

4. Modify the EMP table to allow for longer employee last names. Confirm your modification.

5. Confirm that both the DEPT and EMP tables are stored in the data dictionary. (Hint: USER_TABLES)

6. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, and DEPARTMENT_ID columns. Name the columns in your new table ID, FIRST_NAME, LAST_NAME, SALARY , and DEPT_ID, respectively.

7. Drop the EMP table.

8. Rename the EMPLOYEES2 table to EMP.

9. Add a comment to the DEPT and EMP table definitions describing the tables. Confirm your additions in the data dictionary.

10. Drop the FIRST_NAME column from the EMP table. Confirm your modification by checking the description of the table.

11. In the EMP table, mark the DEPT_ID column in the EMP table as UNUSED. Confirm your modification by checking the description of the table.

12. Drop all the UNUSED columns from the EMP table. Confirm your modification by checking the description of the table.

Practice Questions 10

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

3. Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.

4. Confirm that the constraints were added by querying the USER_CONSTRAINTS view. Note the types and names of the constraints. Save your statement text in a file called lab10_4.sql.

5. Display the object names and types from the USER_OBJECTS data dictionary view for the EMP and DEPT tables. Notice that the new tables and a new index were created.

If you have time, complete the following exercise:

6. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

Practice Questions 11

1. Create a view called EMPLOYEES_VU based on the employee numbers, employee names, and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.
2. Display the contents of the EMPLOYEES_VU view.
3. Select the view name and text from the USER_VIEWS data dictionary view.

   Note: Another view already exists. The EMP_DETAILS_VIEW was created as part of your schema.

   Note: To see more contents of a LONG column, use the iSQL*Plus command SET LONG *n*, where *n* is the value of the number of characters of the LONG column that you want to see.
4. Using your EMPLOYEES_VU view, enter a query to display all employee names and department numbers.
5. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. Label the view columns EMPNO, EMPLOYEE, and DEPTNO. Do not allow an employee to be reassigned to another department through the view.
6. Display the structure and contents of the DEPT50 view.
7. Attempt to reassign Matos to department 80.

If you have time, complete the following exercise:

8. Create a view called SALARY_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the EMPLOYEES, DEPARTMENTS, and JOB_GRADES tables. Label the columns Employee, Department, Salary, and Grade, respectively.

Practice Questions 12

1. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

2. 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 `lab12_2.sql`. Run the statement in your script.

3. Write a script to insert two rows into the `DEPT` table. Name your script `lab12_3.sql`. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

4. Create a nonunique index on the foreign key column (`DEPT_ID`) in the `EMP` table.
5. Display the indexes and uniqueness that exist in the data dictionary for the `EMP` table. Save the statement into a script named `lab12_5.sql`.

Practice Questions 13

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

   The `CREATE SESSION` system privilege

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

   The `CREATE TABLE` privilege

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

   You can, or anyone you have given those privileges to by using the `WITH GRANT OPTION`.

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

   Create a role containing the system privileges and grant the role to the users

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

   The `ALTER USER` statement

6. Grant another user access to your `DEPARTMENTS` table. Have the user grant you query access to his or her `DEPARTMENTS` table.

7. Query all the rows in your `DEPARTMENTS` table.
8. Add a new row to your `DEPARTMENTS` table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table.
9. Create a synonym for the other team's `DEPARTMENTS` table.

10. Query all the rows in the other team's `DEPARTMENTS` table by using your synonym.

11. Query the `USER_TABLES` data dictionary to see information about the tables that you own.

12. Query the `ALL_TABLES` data dictionary view to see information about all the tables that you can access. Exclude tables that you own.

13. Revoke the `SELECT` privilege from the other team.

```
Team 1 revokes the privilege.
```

14. Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

Team 1 executes this INSERT statement.

Practice Questions 14

1. Create the tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.

a. Table name: MEMBER

Column_ Name	MEMBER_ ID	LAST_ NAME	FIRST_NAM E	ADDRESS	CITY	PHONE	JOIN _ DATE
Key Type	PK						
Null/ Unique	NN,U	NN					NN
Default Value							System Date
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	25	25	100	30	15	

b. Table name: TITLE

Column_ Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_ DATE
Key Type	PK					
Null/ Unique	NN,U	NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMEN TARY	
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	60	400	4	20	

c. Table name: TITLE_COPY

Column Name	COPY_ID	TITLE_ID	STATUS
Key Type	PK	PK,FK	
Null/ Unique	NN,U	NN,U	NN
Check			AVAILABLE, DESTROYED, RENTED, RESERVED
FK Ref Table		TITLE	
FK Ref Col		TITLE_ID	
Data Type	NUMBER	NUMBER	VARCHAR2
Length	10	10	15

d. Table name: RENTAL

Column Name	BOOK_ DATE	MEMBER_ ID	COPY_ ID	ACT_RET_ DATE	EXP_RET_ DATE	TITLE_ ID
Key Type	PK	PK,FK1	PK,FK2			PK,FK2
Default Value	System Date				System Date + 2 days	
FK Ref Table		MEMBER	TITLE_ COPY			TITLE_ COPY
FK Ref Col		MEMBER_I D	COPY_ ID			TITLE_ID
Data Type	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
Length		10	10			10

e. Table name: RESERVATION

Column Name	RES_ DATE	MEMBER_ ID	TITLE_ ID
Key Type	PK	PK,FK1	PK,FK2
Null/ Unique	NN,U	NN,U	NN
FK Ref Table		MEMBER	TITLE
FK Ref Column		MEMBER_ID	TITLE_ID
Data Type	DATE	NUMBER	NUMBER
Length		10	10

2. Verify that the tables and constraints were created properly by checking the data dictionary.

3. Create sequences to uniquely identify each row in the MEMBER table and the TITLE table. a. Member number for the MEMBER table: start with 101; do not allow caching of the values. Name the sequence MEMBER_ID_

   b. Title number for the TITLE table: start with 92; no caching. Name the sequence TITLE_ID_SEQ.

   c. Verify the existence of the sequences in the data dictionary.

4. Add data to the tables. Create a script for each set of data to add.
   a. Add movie titles to the TITLE table. Write a script to enter the movie information. Save the statements in a script named lab14_4a.sql. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

   b. Add data to the MEMBER table. Place the insert statements in a script named lab14_4b.sql. Execute commands in the script. Be sure to use the sequence to add the member numbers.

   c. Add the following movie copies in the TITLE_COPY table:
   Note: Have the TITLE_ID numbers available for this exercise

   d. Add the following rentals to the RENTAL table:
   Note: Title number may be different depending on sequence number

5. Create a view named TITLE_AVAIL to show the movie titles and the availability of each copy and its expected return date if rented. Query all rows from the view. Order the results by title.

6. Make changes to data in the tables.
   a. Add a new title. The movie is "Interstellar Wars," which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?" Be sure to add a title copy record for two copies.

   b. Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent "Interstellar Wars." The other is for Mark Quick-to-See, who wants to rent "Soda Gang."

   c. Customer Carmen Velasquez rents the movie "Interstellar Wars," copy 1. Remove her reservation for the movie. Record the information about the rental. Allow the default value for the expected return date to be used. Verify that the rental was recorded by using the view you created.

7. Make a modification to one of the tables.
   a. Add a PRICE column to the TITLE table to record the purchase price of the video. The column should have a total length of eight digits and two decimal places. Verify your modifications.

b. Create a script named `lab14_7b.sql` that contains update statements that update each video with a price according to the following list. Run the commands in the script.

Note: Have the `TITLE_ID` numbers available for this exercise.

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

8. Create a report titled Customer History Report. This report contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab14_8.sql`.

Practice Question 15

1. List the department IDs for departments that do not contain the job ID `ST_CLERK`, using `SET` operators.

2. Display the country ID and the name of the countries that have no departments located in them, using `SET` operators.

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID, using `SET` operators.

4. List the employee IDs and job IDs of those employees who currently have the job title that they held before beginning their tenure with the company.

5. Write a compound query that lists the following:
   - Last names and department ID of all the employees from the `EMPLOYEES` table, regardless of whether or not they belong to any department
   - Department ID and department name of all the departments from the `DEPARTMENTS` table, regardless of whether or not they have employees working in them

Practice Question 16

1. Alter the session to set the `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

   Singapore

   Egypt

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

c. Display the `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
Note: The output might be different based on the date when the command is executed.

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

e. Display the `CURRENT_DATE`, `CURRENT_TIMESTAMP`, `LOCALTIMESTAMP` for this session.
Note: The output might be different, based on the date when the command is executed.

3. Write a query to display the `DBTIMEZONE` and `SESSIONTIMEZONE`.

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

5. Alter the session to set the `NLS_DATE_FORMAT` to `DD-MON-YYYY`.

Practice Questions 18

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.

2. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID1700.

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

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

5. Display the details of the employee ID, last name, and department ID of those employees who live in cities whose name begins with *T*.

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. 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.

b. Can this be done by using the NOT IN operator? How, or why not?

```
SELECT outer.last_name
FROM   employees outer
WHERE  outer.employee_id
NOT IN (SELECT inner.manager_id
        FROM   employees inner);
```

This alternative solution is not a good one. The subquery picks up a NULL value, so the entire query returns no rows. The reason is that all conditions that compare a NULL value result in NULL. Whenever NULL values are likely to be part of the value set, *do not* use NOT IN as a substitute for NOT EXISTS.

Thank You

If You will find difficult to resolve any of Query Contact me

I'll be there to help You.