Instructions

For <u>each</u> problem:

- **Write and execute an SQL query in Oracle Live SQL or SQL*Plus**
- **Execute the following command: select sysdate, 'your name' from dual; where your_name is substituted with your name**
- **Take a screenshot that includes both SQL statements and all results**
- **Copy and paste the screenshot into a Word file containing your solutions**

Assignment:

For the database you designed in the previous assignments, complete the following problems:

- **Write an SQL query that uses a single-row subquery in a WHERE clause. Explain what the query is intended to do.**
- **Write an SQL query that uses a multiple-column subquery in a FROM clause. Explain what the query is intended to do.**
- **Write an SQL query that is based on multiple tables and uses a subquery with the GROUP BY statement and HAVING clause. Explain what the query is intended to do.**
- **Write an SQL query that is based on multiple tables and uses a multiple-row subquery in a WHERE clause. The subquery will include the GROUP BY statement and another multiple-row subquery in a HAVING clause. Explain what the query is intended to do.**
- **Write an SQL query that joins three tables and uses any type of a subquery. Explain what the query is intended to do.**
- **Write an SQL query that is based on multiple tables and uses the DECODE function. Explain what the query is intended to do.**

For Problems 7-10 below, consider the following business scenario:

> **A university wants to keep track of bonuses given to the professors who mentor (supervise) junior faculty members. Data about all professors is available in the Faculty table (see below). In that table, the f_super column represents a faculty id of a mentor (if any). A separate Bonus table is needed to assign and keep track of bonuses. The Bonus table will have a default bonus of 1000. It will be updated once a year to add new mentors and to update bonuses for the existing ones.**

7. Create the Faculty table and populate it with data using the script below: CREATE TABLE faculty (f_id NUMBER(6), f_last VARCHAR2(30) ,f_first VARCHAR2(30), f_mi CHAR(1), loc_id NUMBER(5), f_phone VARCHAR2(10), f_rank VARCHAR2(9), f_super NUMBER(6), CONSTRAINT faculty_f_id_pk PRIMARY KEY(f_id)); INSERT INTO faculty VALUES (1, 'Marx', 'Teresa', 'J', 9, '4075921695', 'Associate', 4); INSERT INTO faculty VALUES (2, 'Zhulin', 'Mark', 'M', 10, '4073875682', 'Full', NULL); INSERT INTO faculty VALUES (3, 'Langley', 'Colin', 'A', 12, '4075928719', 'Assistant', 4); INSERT INTO faculty VALUES (4, 'Brown', 'Jonnel', 'D', 11, '4078101155', 'Full', NULL); Check the result using the select * from faculty; command.

8. Create the Bonus table that consists of two columns: f_id (PK) and bonus. For the f_id column, use the same description as in the Faculty table. For the bonus column, use the NUMBER data type and the DEFAULT constraint to set the values for the bonus column to

1000 (bonus amount). Next, use a subquery to copy ids of mentors given in the Faculty table into the Bonus table. Check the result using the select * from bonus; command.

9. Add two new records to the Faculty table using the command below. These records represent new faculty who came to the university this year. INSERT INTO faculty VALUES (5, 'Sealy', 'James', 'L', 13, '4079817153', 'Associate', 1); INSERT INTO faculty VALUES (6, 'Smith', 'John', 'D', 10, '4238102345', 'Full', NULL); Check the result using the select * from faculty; command.

10. Assume that the same Bonus table is used next year to assign and update bonuses. Use the MERGE statement to modify the Bonus table as follows: - if a mentor already exists in the Bonus table, increase the bonus by 1% - If there is a new mentor in the Faculty table, add him/her to the BONUS table Check the result using the select * from bonus; command.

Note: Include your ERD/EERD from your previous assignment.