Keith Maly
AIT-524 (DL-1)
Dr. Rytikova

April 21, 2023

Module 6.2 Database Project Assignment (Revised)

Task: For questions 1-6, building on the tables, constraints, and date developed through Module 6.2, and using the updated EERD provided at the end of this assignment, complete each of the following problems listed below. For questions 7-10, use the materials provided in the assignment

Background: Upon reviewing the requirements for this assignment after completing assignment 6.1, it was determined that additional records would be required in the task, work log, and client documents tables; with the intent that each query developed for Module 6.2 query would return meaningful results. No functional changes were made to the database design in this iteration.

To prepare the database for this module, all previous tables were dropped. The script for this schema was loaded into Oracle 12c and is titled **2023_04_07_Risk_Insights_Build.sql**.

With the schema in place, a second script was developed to populate the tables using filename **2023_04_21_Risk_Insights_Data_Load_62.sql**. Data populated within the database continues to be drawn from entities from the cartoon, 'The Flintstones.'[1]

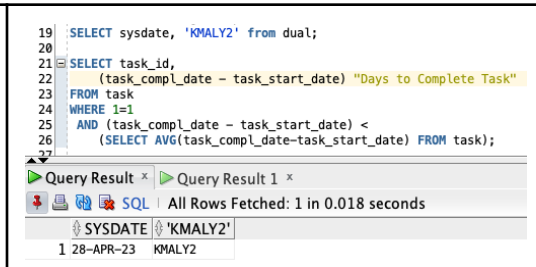
The SQL code to complete assignment for Module 6.1 is captured in total within the file titled **2023_04_23_M62_Functions.sql**.

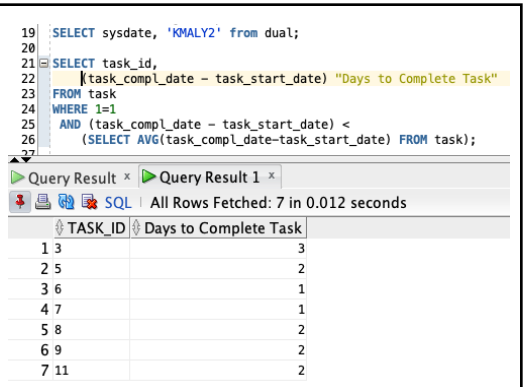
All three SQL scripts are submitted as text files. Code snippets and screenshots for each table structure and inserts are submitted below in accordance with assignment specifications.

[1] Characters Database, List Of Flintstones Characters

1. Write an SQL query that uses a single-row subquery in a WHERE clause. Explain what the query is intended to do. Explanation: Identify the tasks, by id number, that were completed faster than the average completion time.

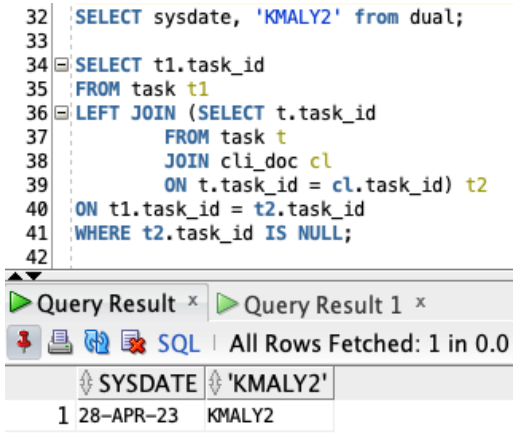
```
SELECT task_id,
    (task_compl_date - task_start_date)
    "Days to Complete Task"
FROM task
WHERE 1=1
 AND (task_compl_date - task_start_date) <
      (SELECT
        AVG(task_compl_date-task_start_date)
        FROM task);
```

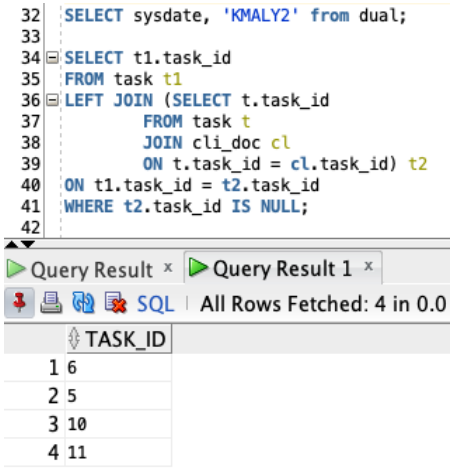




2. Write an SQL query that uses a multiple-column subquery in a FROM clause. Explain what the query is intended to do. Explanation: List the client tasks, by their id number, that do not have associated documents in the document library

```
SELECT t1.task_id
FROM task t1
LEFT JOIN (SELECT t.task_id
        FROM task t
        JOIN cli_doc cl
        ON t.task_id = cl.task_id) t2
ON t1.task_id = t2.task_id
WHERE t2.task_id IS NULL;
```

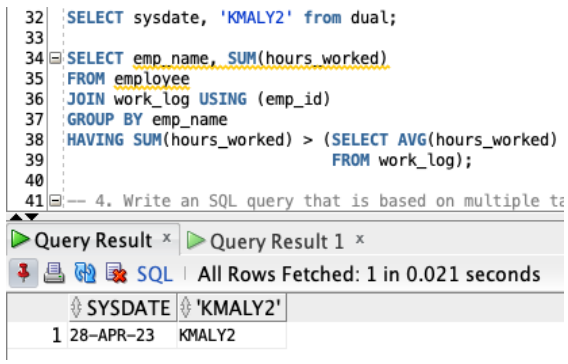


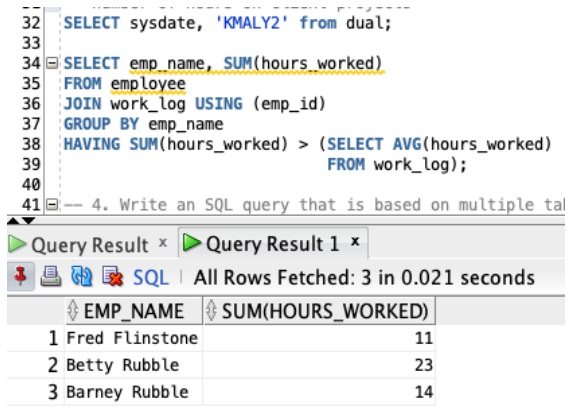


3. Write an SQL query that is based on multiple tables and uses a subquery with the GROUP BY statement and HAVING clause. Explain what the query is intended to do.

Explanation: Identify the workers, by name, who logged more than the average number of hours on client projects

```
SELECT emp_name,
       SUM(hours_worked)
FROM employee
JOIN work_log USING (emp_id)
GROUP BY emp_name
HAVING SUM(hours_worked) >
       (SELECT AVG(hours_worked)
       FROM work_log);
```



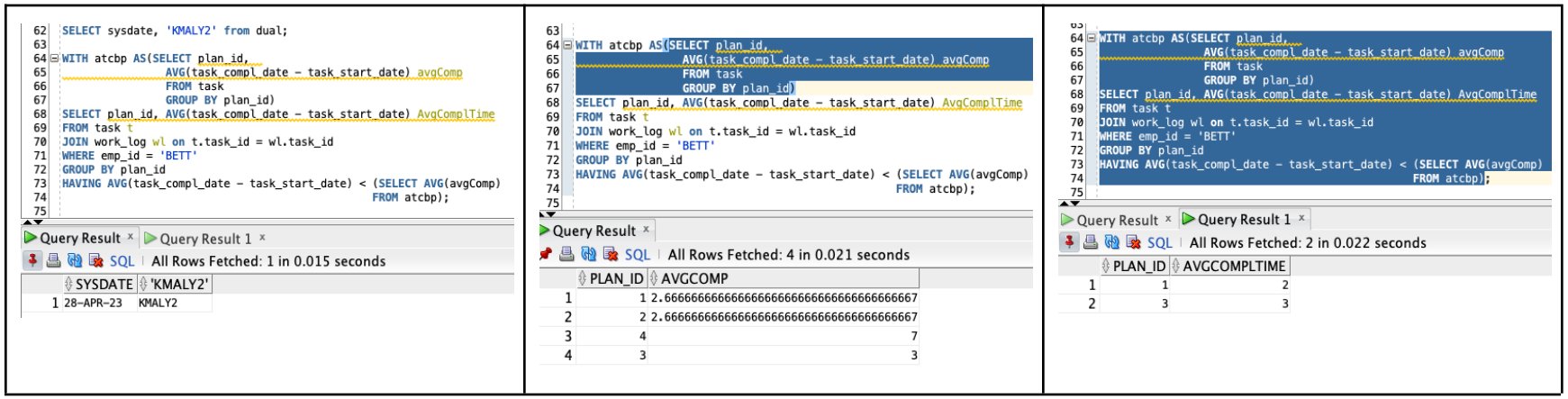


4. Write an SQL query that is based on multiple tables and uses a multiple-row subquery in a WHERE clause. The subquery will include the GROUP BY statement and another multiple-row subquery in a HAVING clause. Explain what the query is intended to do.

Explanation: Fred believes that Betty is his highest performing employee. To show the impact she is having across programs, Fred wants a report that identifies the project plans that Betty has worked on where her average time to complete tasks is better than the overall average time to complete tasks across all clients

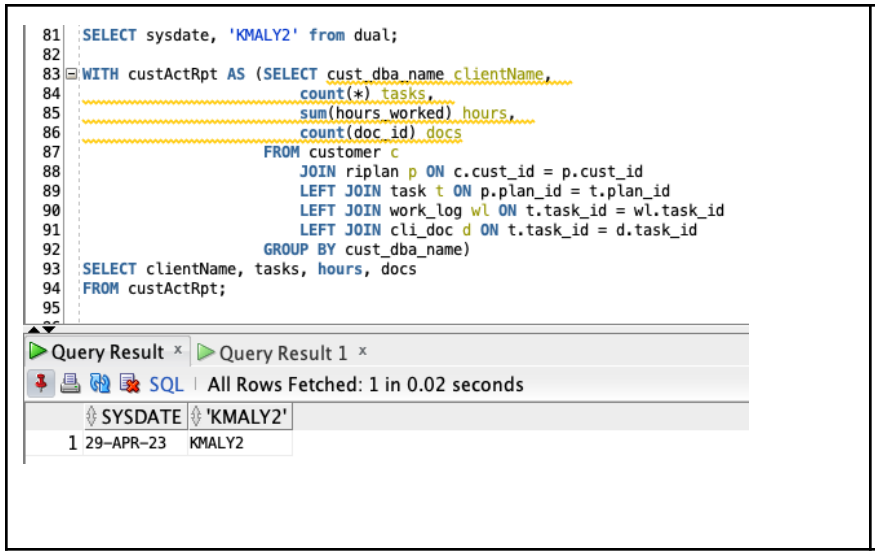
Query:

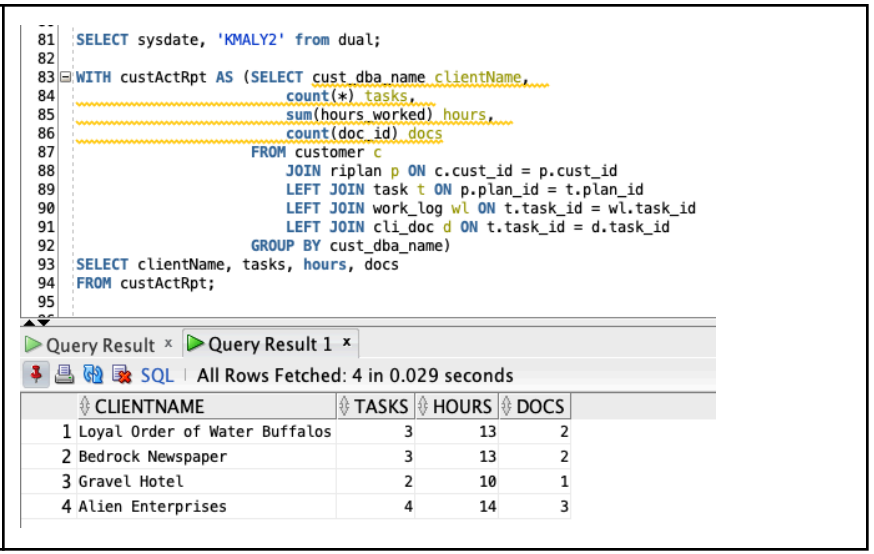
```
WITH atcbp AS(SELECT plan_id,
        AVG(task_compl_date - task_start_date) avgComp
        FROM task
        GROUP BY plan_id)
SELECT plan_id, AVG(task_compl_date - task_start_date) AvgComplTime
FROM task t
JOIN work_log wl on t.task_id = wl.task_id
WHERE emp_id = 'BETT'
GROUP BY plan_id
HAVING AVG(task_compl_date - task_start_date) < (SELECT AVG(avgComp) FROM atcbp);
```



5. Write an SQL query that joins three tables and uses any type of a subquery. Explain what the query is intended to do. Explanation: Prepare a report breaking down the work accomplished or in progress for the client, which includes the company name, number of plans, number of tasks completed, and documents prepared. Do not include clients that have no prepared documents.

```
WITH custActRpt AS (SELECT cust_dba_name clientName,
                      count(*) tasks,
                      sum(hours_worked) hours,
                      count(doc_id) docs
                   FROM customer c
                      JOIN riplan p ON c.cust_id = p.cust_id
                      LEFT JOIN task t ON p.plan_id = t.plan_id
                      LEFT JOIN work_log wl ON t.task_id = wl.task_id
                      LEFT JOIN cli_doc d ON t.task_id = d.task_id
                   GROUP BY cust_dba_name)
SELECT clientName, tasks, hours, docs
FROM custActRpt;
```

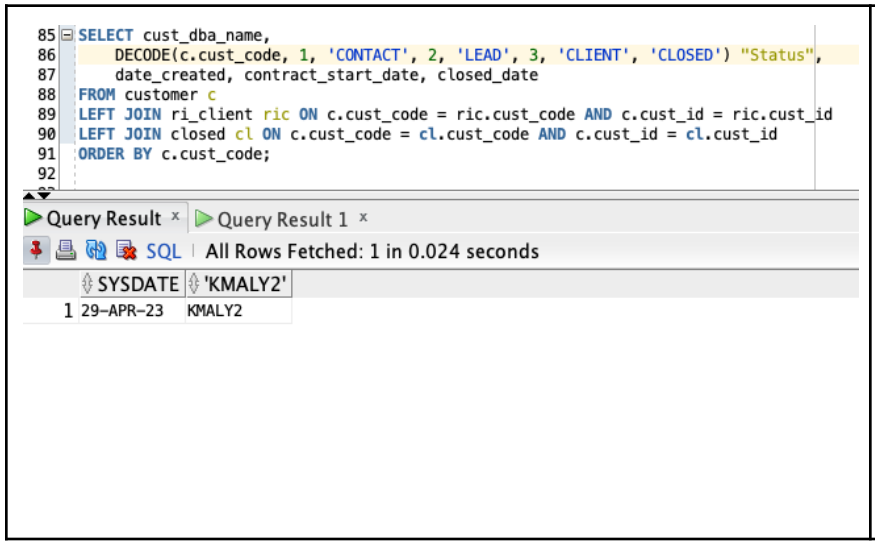




6. Write an SQL query that is based on multiple tables and uses the DECODE function. Explain what the query is intended to do.

Explanation: Create a report status of all entities that Risk Insights has engaged since incorporating and the key dates associated with each.

```
SELECT cust_dba_name, DECODE(c.cust_code, 1, 'CONTACT', 2, 'LEAD', 3, 'CLIENT', 'CLOSED') "Status",
   date_created, contract_start_date, closed_date
FROM customer c
LEFT JOIN ri_client ric ON c.cust_code = ric.cust_code AND c.cust_id = ric.cust_id
LEFT JOIN closed cl ON c.cust_code = cl.cust_code AND c.cust_id = cl.cust_id
ORDER BY c.cust_code;
```

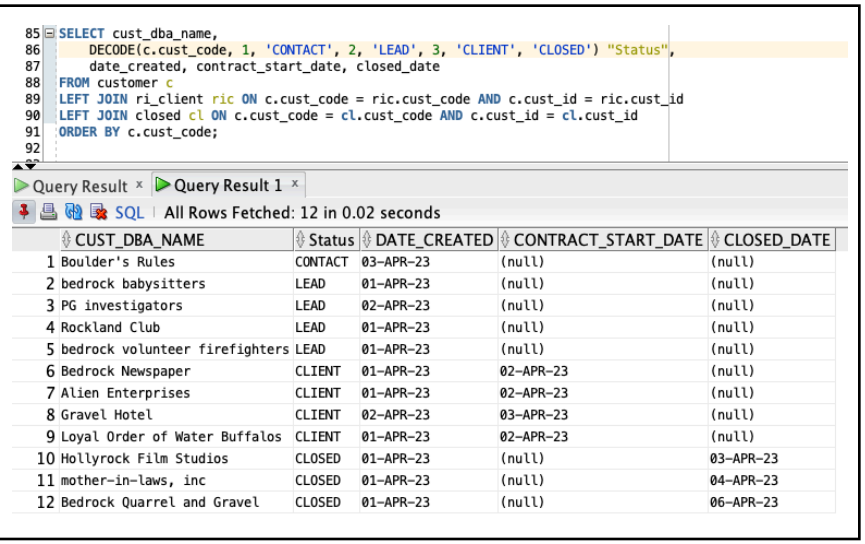


```
SELECT cust_dba_name,
    DECODE(c.cust_code, 1, 'CONTACT', 2, 'LEAD', 3, 'CLIENT', 'CLOSED') "Status",
    date_created, contract_start_date, closed_date
FROM customer c
LEFT JOIN ri_client ric ON c.cust_code = ric.cust_code AND c.cust_id = ric.cust_id
LEFT JOIN closed cl ON c.cust_code = cl.cust_code AND c.cust_id = cl.cust_id
ORDER BY c.cust_code;
```

Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.024 seconds

	SYSDATE	'KMALY2'
1	29-APR-23	KMALY2



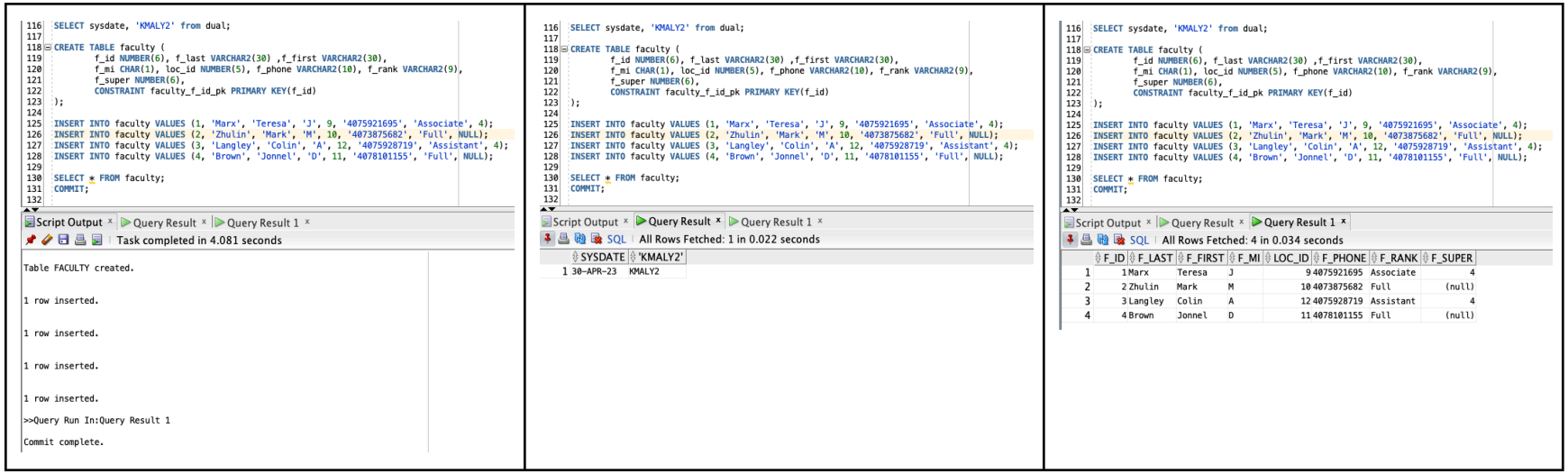
Query Result | Query Result 1

SQL | All Rows Fetched: 12 in 0.02 seconds

	CUST_DBA_NAME	Status	DATE_CREATED	CONTRACT_START_DATE	CLOSED_DATE
1	Boulder's Rules	CONTACT	03-APR-23	(null)	(null)
2	bedrock babysitters	LEAD	01-APR-23	(null)	(null)
3	PG investigators	LEAD	02-APR-23	(null)	(null)
4	Rockland Club	LEAD	01-APR-23	(null)	(null)
5	bedrock volunteer firefighters	LEAD	01-APR-23	(null)	(null)
6	Bedrock Newspaper	CLIENT	01-APR-23	02-APR-23	(null)
7	Alien Enterprises	CLIENT	01-APR-23	02-APR-23	(null)
8	Gravel Hotel	CLIENT	02-APR-23	03-APR-23	(null)
9	Loyal Order of Water Buffalos	CLIENT	01-APR-23	02-APR-23	(null)
10	Hollyrock Film Studios	CLOSED	01-APR-23	(null)	03-APR-23
11	mother-in-laws, inc	CLOSED	01-APR-23	(null)	04-APR-23
12	Bedrock Quarrel and Gravel	CLOSED	01-APR-23	(null)	06-APR-23

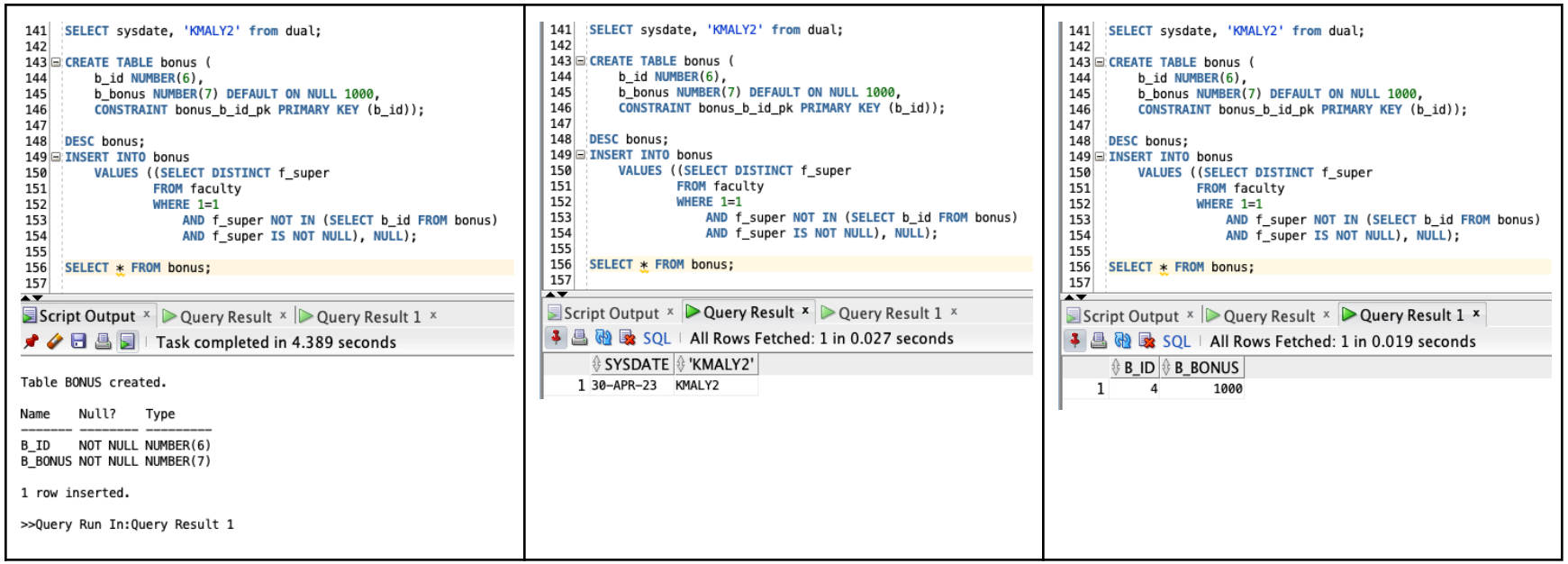
7. Create the Faculty table and populate it with data using the script below: Check the result using the query: select * from faculty;

```
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);

SELECT * FROM faculty;
```



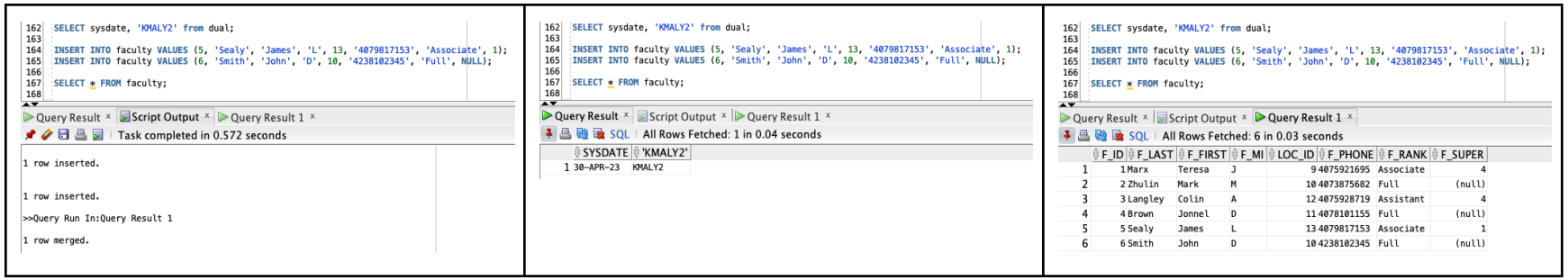
8. Create the Bonus table that consists of two columns: f_id (PK) and bonus. For the f_id column, use the same description as in the Faculty table. For the bonus column, use the NUMBER data type and the DEFAULT constraint to set the values for the bonus column to 1000 (bonus amount). Next, use a subquery to copy ids of mentors given in the Faculty table into the Bonus table. Check the result using the select * from bonus; command.

```
CREATE TABLE bonus (b_id NUMBER(6), b_bonus NUMBER(7) DEFAULT ON NULL 1000,
   CONSTRAINT bonus_b_id_pk PRIMARY KEY (b_id), CONSTRAINT bonus_b_id_fk FOREIGN KEY (b_id)
                                                    REFERENCES faculty(f_id));
DESC bonus;
INSERT INTO bonus
   VALUES ((SELECT DISTINCT f_super FROM faculty WHERE f_super IS NOT NULL), NULL);

SELECT * FROM bonus;
```



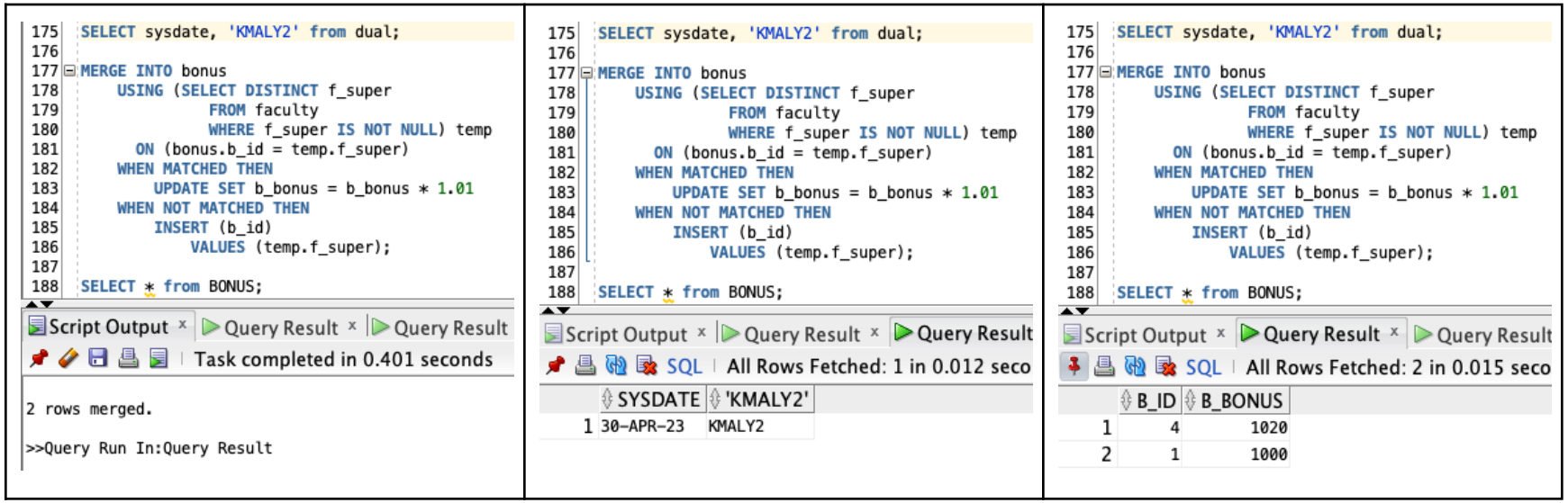
9. Add two new records to the Faculty table using the command below. These records represent new faculty who came to the university this year. Using the command SELECT * from faculty,check the result.

INSERT INTO faculty VALUES (5, 'Sealy', 'James', 'L', 13, '4079817153', 'Associate', 1);
INSERT INTO faculty VALUES (6, 'Smith', 'John', 'D', 10, '4238102345', 'Full', NULL);
SELECT * FROM faculty;



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.

```
MERGE INTO bonus
   USING (SELECT DISTINCT f_super FROM faculty WHERE f_super IS NOT NULL) temp
    ON (bonus.b_id = temp.f_super)
   WHEN MATCHED THEN
      UPDATE SET b_bonus = b_bonus * 1.01
   WHEN NOT MATCHED THEN
      INSERT (b_id) VALUES (temp.f_super);
```



AIT-524 / DL1: Revised EERD (Module 6.2)
Keith Maly (kmaly2@gmu.edu)
April 21, 2023

CHECK Constraints: ending dates within tables must be after starting dates; customer codes may only be 00, 01, 02, or 03.
UNIQUE The only anticipated 'unique' data element outside of primary keys is the clients EIN
Per Oracle Docs, all attributes in sub/supertype design must be NOT NULL