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

April 21, 2023

Module 6.2 Database Project Assignment (Revised)

Task: Building on the tables and constraints developed in Module 6.1, and using the updated EERD previously developed, complete each of the following problems listed below.

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 eaningful 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**.

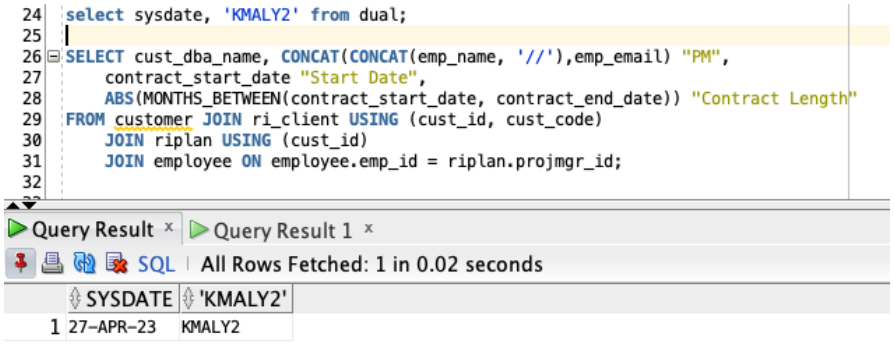
All three SQL scripts are submitted as a text files. Code snippets and screenshots for each table structure and inserts are submitted below in accordance with assignment specifications.

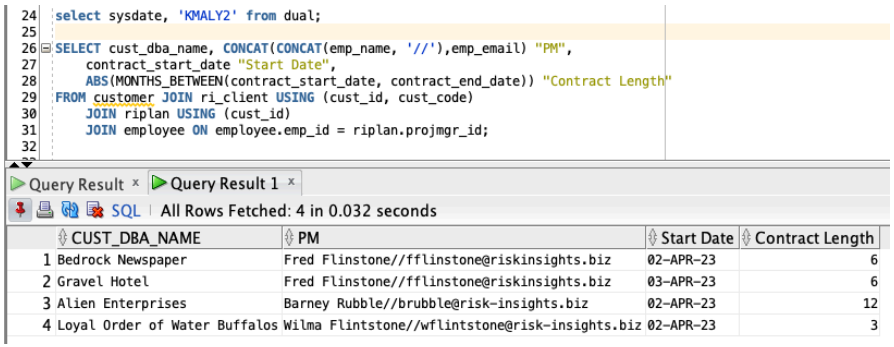
[1] Characters Database, List Of Flintstones Characters

1. Write an SQL query to join 2+ tables where each query contains multiple nested single-row functions. Make sure to use different functions in each query. Explain what each query is intended to do.

Explanation: To understand the Risk Insights revenue projection, the Risk Insights Advisory Board requires a report that details all of the entities that risk insights has under contract; that client program manager and email (in a single field); and the length of each contract.

```
SELECT cust_dba_name,
CONCAT(CONCAT(emp_name, '//'),emp_email) "PM",
    contract_start_date "Start Date",
    ABS(MONTHS_BETWEEN(contract_start_date,
contract_end_date)) "Contract Length"
FROM customer JOIN ri_client USING (cust_id,
cust_code)
    JOIN riplan USING (cust_id)
    JOIN employee ON employee.emp_id =
riplan.projmgr_id;
```

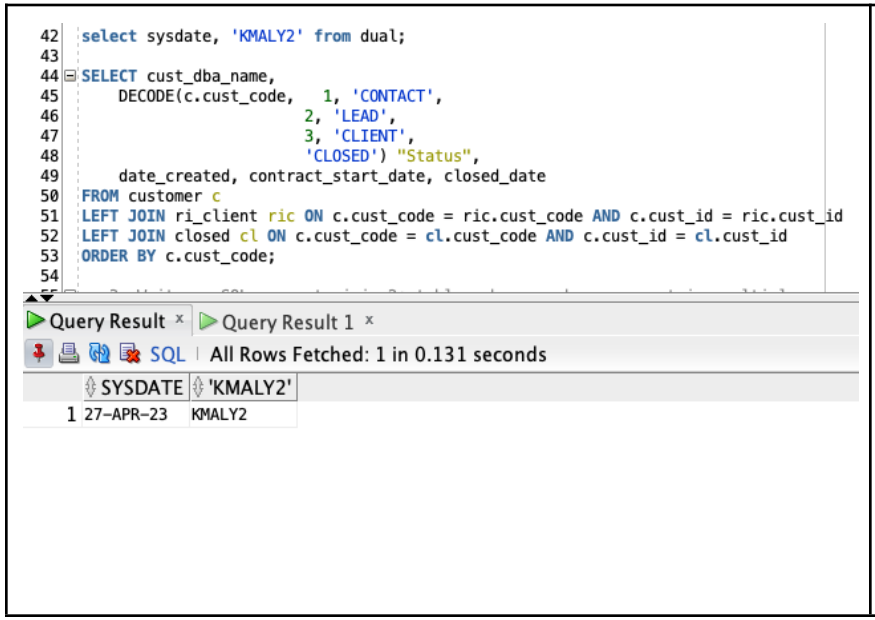




2. Write an SQL query to join 2+ tables where each query contains multiple nested single-row functions. Make sure to use different functions in each query. Explain what each query is intended to do.

Explanation: The Risk Insights Advisory Board would like to know the 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 sysdate, 'KMALY2' from dual;

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 0.131 seconds

	SYSDATE	'KMALY2'
1	27-APR-23	KMALY2

```
select sysdate, 'KMALY2' from dual;

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

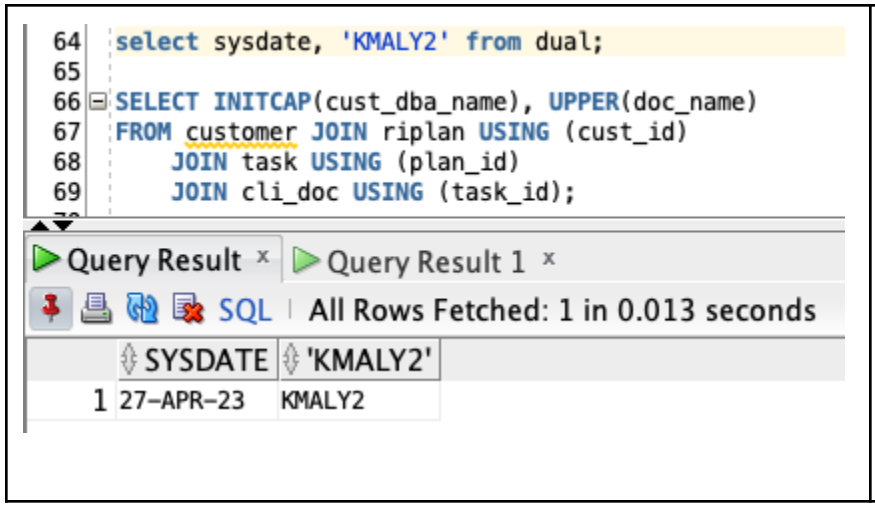
SQL | All Rows Fetched: 12 in 0.057 seconds

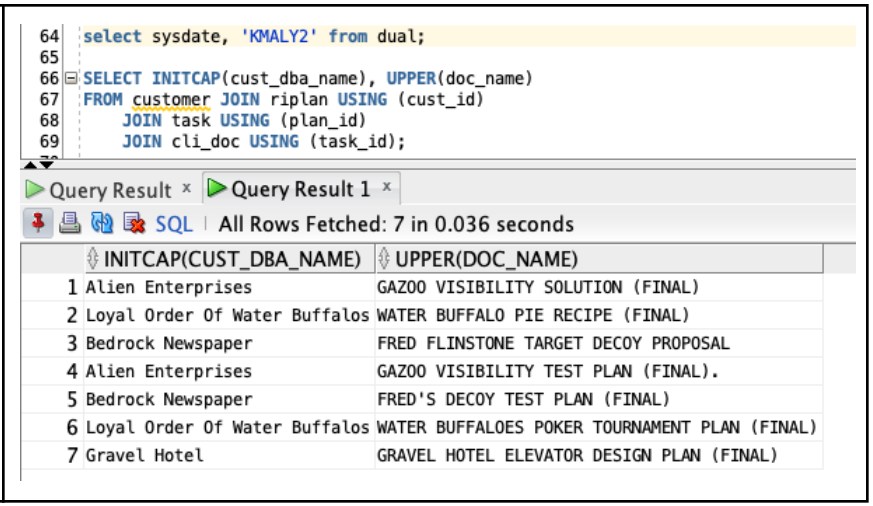
	CUST_DBA_NAME	Status	DATE_CREATED	CONTRACT_START_DATE	CLOSED_DATE
1	Boulder's Rules	CONTACT	03-APR-23	(null)	(null)
2	bedrock babysitters	LEAD	01-APR-23	(null)	(null)
3	PG investigators	LEAD	02-APR-23	(null)	(null)
4	Rockland Club	LEAD	01-APR-23	(null)	(null)
5	bedrock volunteer firefighters	LEAD	01-APR-23	(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

3. Write an SQL query to join 2+ tables where each query contains multiple nested single-row functions. Make sure to use different functions in each query. Explain what each query is intended to do.

Explanation: Prepare a report that serves as the document library, which contains the clients that the documents were written doc and the titles of the documents. The titles of documents must be completely in Upper Case and Companies must be formatted in Title Case.

```
SELECT INITCAP(cust_dba_name),
UPPER(doc_name)
FROM customer JOIN riplan USING (cust_id)
    JOIN task USING (plan_id)
    JOIN cli_doc USING (task_id);
```

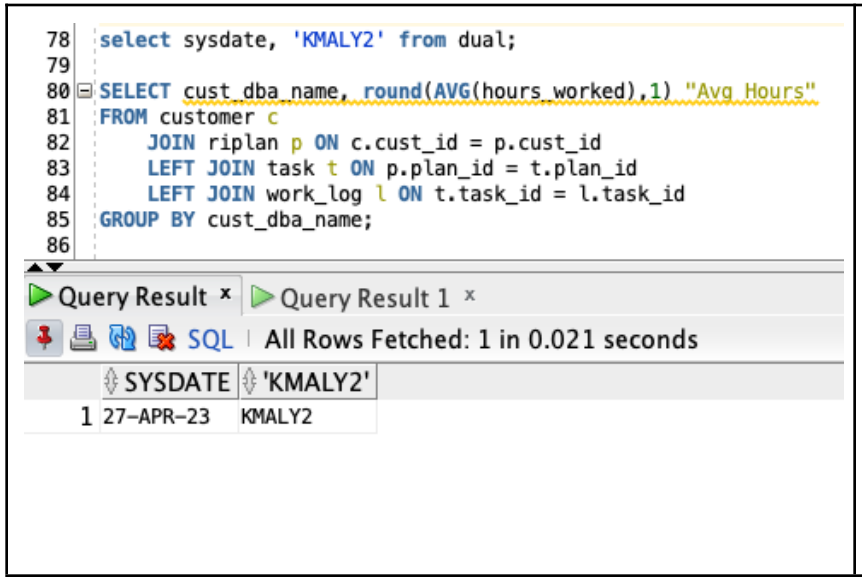




4. Write an SQL query to join 2+ tables where each query contains multiple nested single-row functions. Make sure to use different functions in each query. Explain what each query is intended to do.

Explanation: Compute the average hours worked for each client. Results should be rounded to 6 minute increments.

```
SELECT cust_dba_name, round(AVG(hours_worked),1) "Avg Hours"
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 l ON t.task_id = l.task_id
GROUP BY cust_dba_name;
```

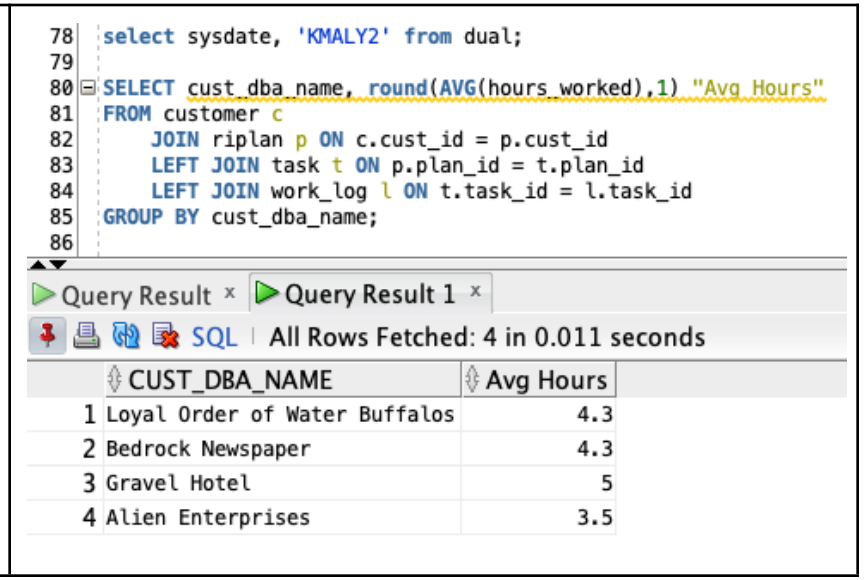


```
78  select sysdate, 'KMALY2' from dual;
79
80  SELECT cust_dba_name, round(AVG(hours_worked),1) "Avg Hours"
81  FROM customer c
82      JOIN riplan p ON c.cust_id = p.cust_id
83      LEFT JOIN task t ON p.plan_id = t.plan_id
84      LEFT JOIN work_log l ON t.task_id = l.task_id
85  GROUP BY cust_dba_name;
86
```

Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.021 seconds

	SYSDATE	'KMALY2'
1	27-APR-23	KMALY2



```
78  select sysdate, 'KMALY2' from dual;
79
80  SELECT cust_dba_name, round(AVG(hours_worked),1) "Avg Hours"
81  FROM customer c
82      JOIN riplan p ON c.cust_id = p.cust_id
83      LEFT JOIN task t ON p.plan_id = t.plan_id
84      LEFT JOIN work_log l ON t.task_id = l.task_id
85  GROUP BY cust_dba_name;
86
```

Query Result | Query Result 1

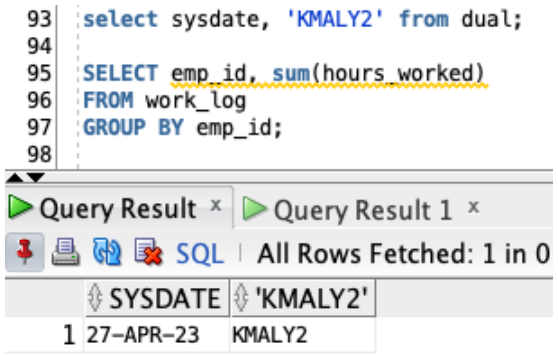
SQL | All Rows Fetched: 4 in 0.011 seconds

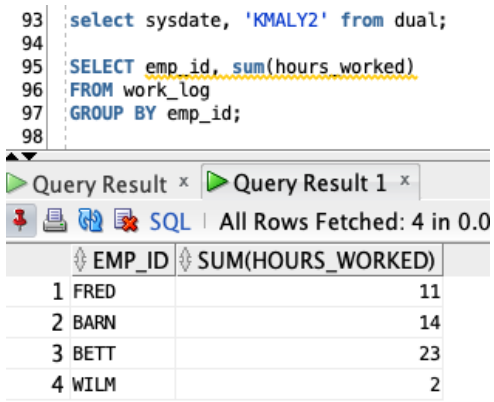
	CUST_DBA_NAME	Avg Hours
1	Loyal Order of Water Buffalos	4.3
2	Bedrock Newspaper	4.3
3	Gravel Hotel	5
4	Alien Enterprises	3.5

5. Write a query which is based on a single table and includes the GROUP BY statement. Explain what the query is supposed to do.

Explanation: Generate a work log summary of the number of hours worked across all task by employee id.

```
SELECT emp_id,
    sum(hours_worked)
FROM work_log
GROUP BY emp_id;
```

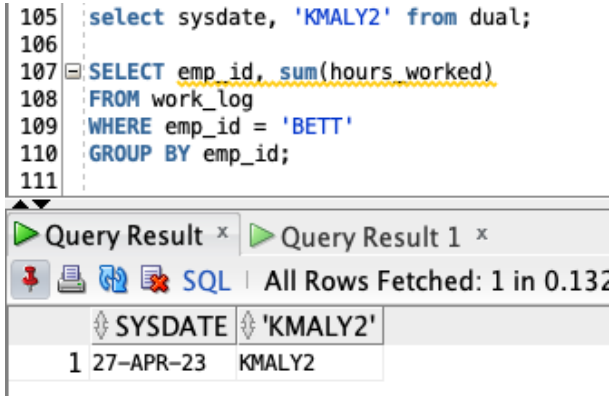


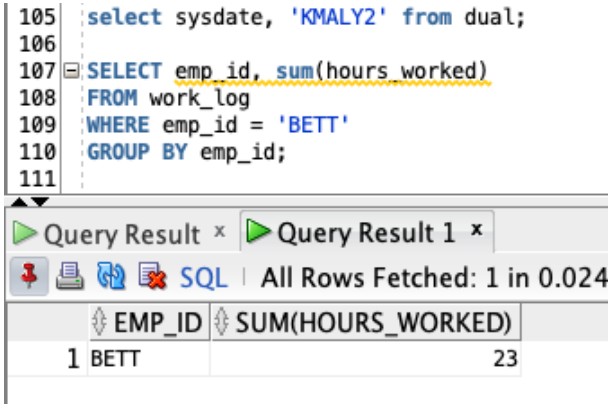


6. Repeat problem 5 but add the WHERE statement. Explain what the query is supposed to do.

Explanation: Generate a work log summary of the number of hours Betty Rubble completed across all tasks.

```
SELECT emp_id,
    sum(hours_worked)
FROM work_log
WHERE emp_id = 'BETT'
GROUP BY emp_id;
```

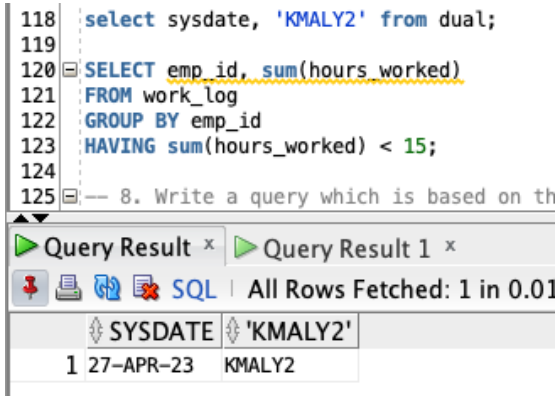


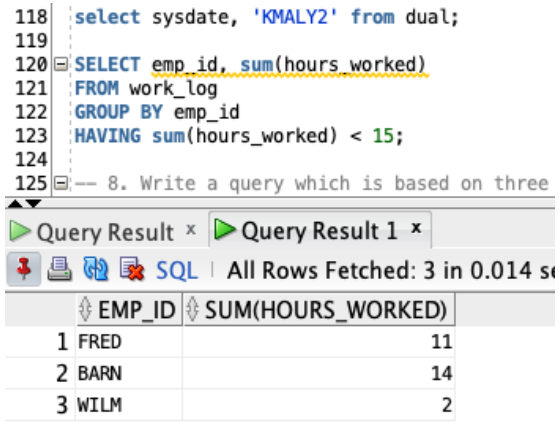


7. Repeat problem 6 but add the HAVING statement. Explain what the query is supposed to do.

Explanation: Identify those employees by emp_id who have worked fewer than 15 hours worked across all tasks.

```
SELECT emp_id, sum(hours_worked)
FROM work_log
GROUP BY emp_id
HAVING sum(hours_worked) < 15;
```

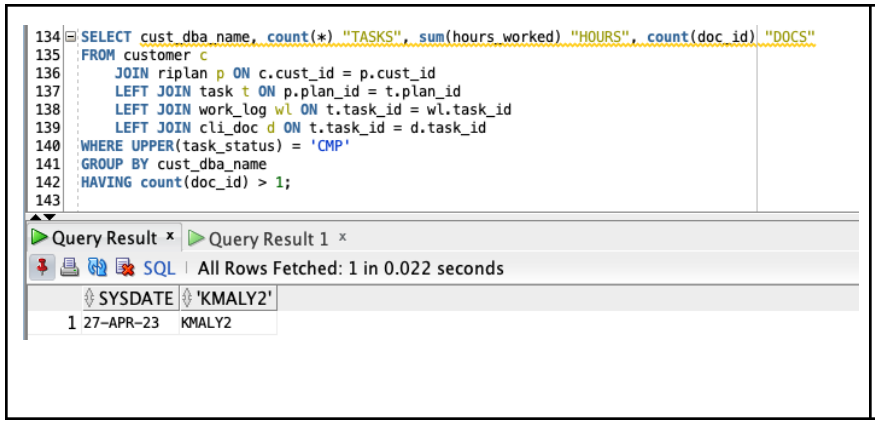


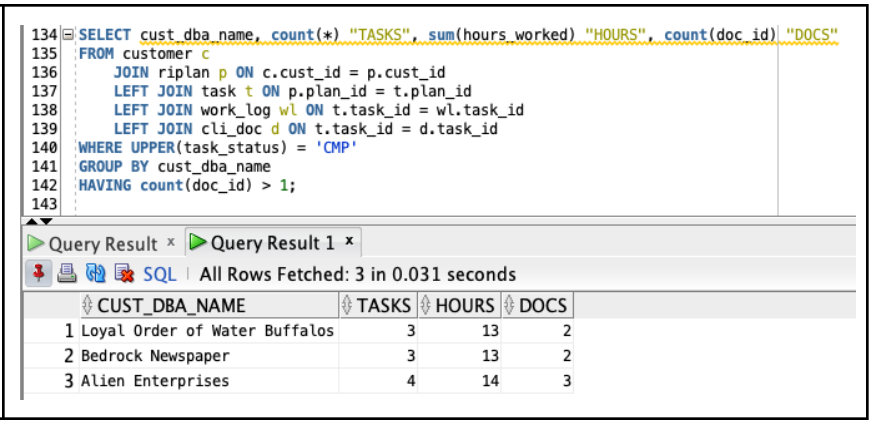


8. Write a query which is based on three tables and includes the GROUP BY, WHERE and HAVING statements. Explain what the query is supposed to do.

Explanation: Prepare a report breaking down the work accomplished by 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.

```
SELECT cust_dba_name, 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
WHERE UPPER(task_status) = 'CMP'
GROUP BY cust_dba_name
HAVING count(doc_id) > 1;
```

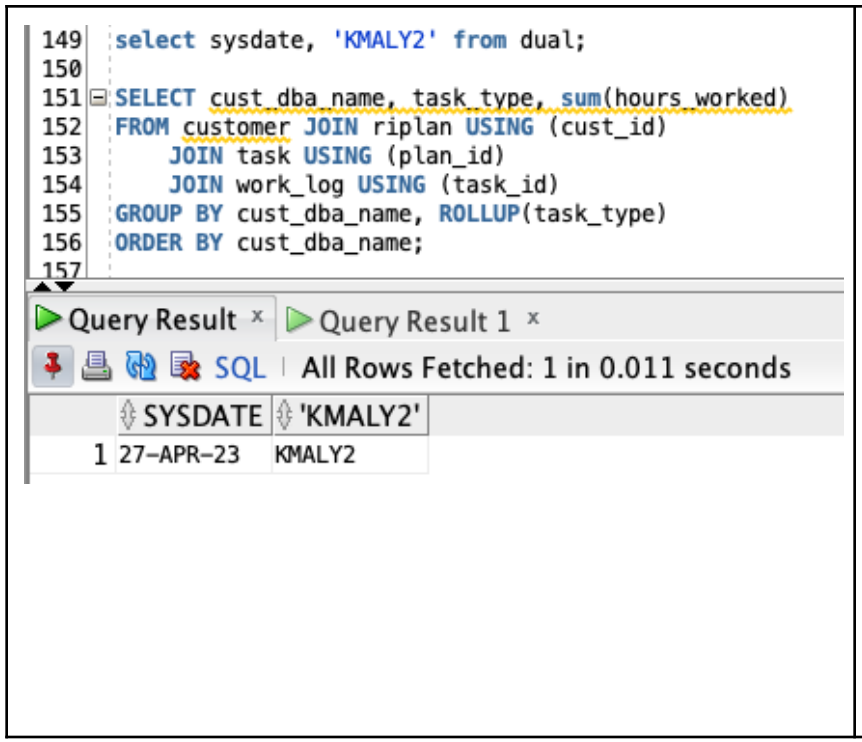




9. Write an SQL query that uses the GROUP BY ROLLUP statement. Explain what the query is supposed to do.

Explanation: Create a table that provides client names, task types and hours worked.

```
SELECT cust_dba_name, task_type, sum(hours_worked)
FROM customer JOIN riplan USING (cust_id)
    JOIN task USING (plan_id)
    JOIN work_log USING (task_id)
GROUP BY cust_dba_name, ROLLUP(task_type)
ORDER BY cust_dba_name;
```

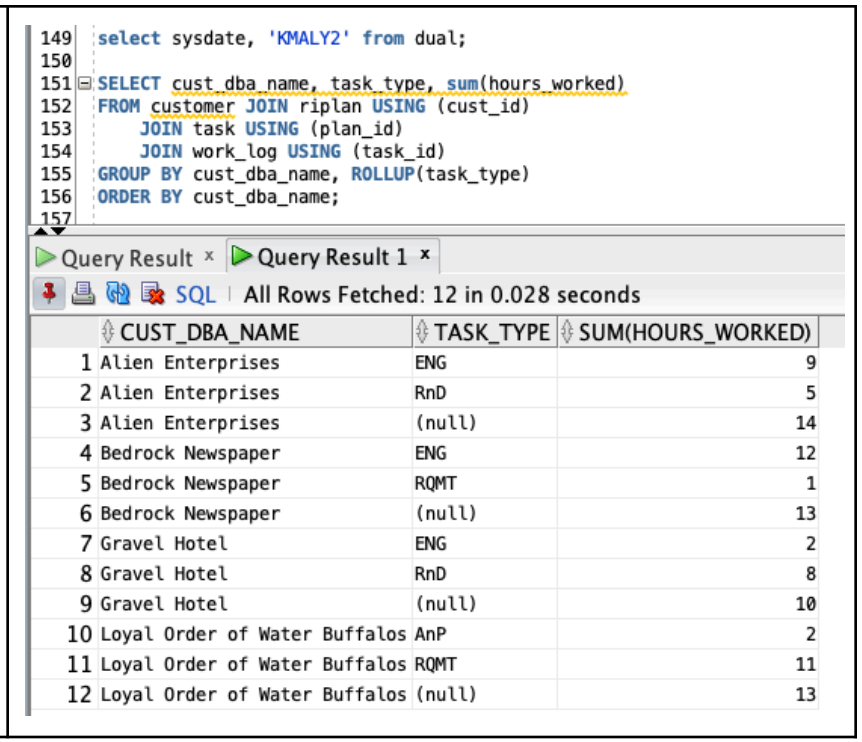


```
149  select sysdate, 'KMALY2' from dual;
150
151  SELECT cust_dba_name, task_type, sum(hours_worked)
152  FROM customer JOIN riplan USING (cust_id)
153      JOIN task USING (plan_id)
154      JOIN work_log USING (task_id)
155  GROUP BY cust_dba_name, ROLLUP(task_type)
156  ORDER BY cust_dba_name;
157
```

Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.011 seconds

	SYSDATE	'KMALY2'
1	27-APR-23	KMALY2



```
149  select sysdate, 'KMALY2' from dual;
150
151  SELECT cust_dba_name, task_type, sum(hours_worked)
152  FROM customer JOIN riplan USING (cust_id)
153      JOIN task USING (plan_id)
154      JOIN work_log USING (task_id)
155  GROUP BY cust_dba_name, ROLLUP(task_type)
156  ORDER BY cust_dba_name;
157
```

Query Result | Query Result 1

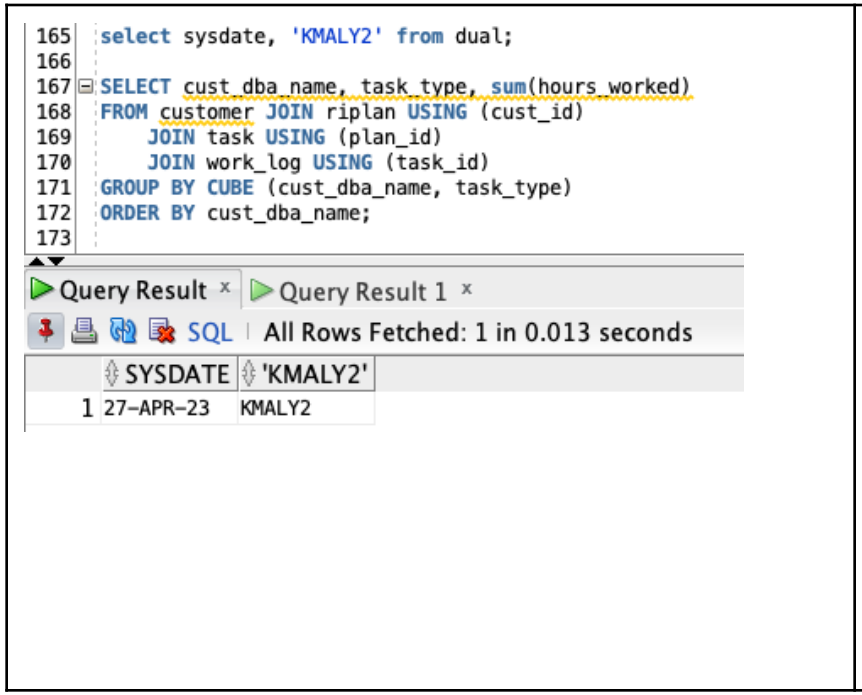
SQL | All Rows Fetched: 12 in 0.028 seconds

	CUST_DBA_NAME	TASK_TYPE	SUM(HOURS_WORKED)
1	Alien Enterprises	ENG	9
2	Alien Enterprises	RnD	5
3	Alien Enterprises	(null)	14
4	Bedrock Newspaper	ENG	12
5	Bedrock Newspaper	RQMT	1
6	Bedrock Newspaper	(null)	13
7	Gravel Hotel	ENG	2
8	Gravel Hotel	RnD	8
9	Gravel Hotel	(null)	10
10	Loyal Order of Water Buffalos	AnP	2
11	Loyal Order of Water Buffalos	RQMT	11
12	Loyal Order of Water Buffalos	(null)	13

10. Write an SQL query that uses the GROUP BY CUBE statement. Explain what the query is supposed to do.

Explanation: Provide a complete summation of hours worked by risk insights employees by client and task type. Include sub and grand totals

```
SELECT cust_dba_name, task_type, sum(hours_worked)
FROM customer JOIN riplan USING (cust_id)
    JOIN task USING (plan_id)
    JOIN work_log USING (task_id)
GROUP BY CUBE (cust_dba_name, task_type)
ORDER BY cust_dba_name;
```

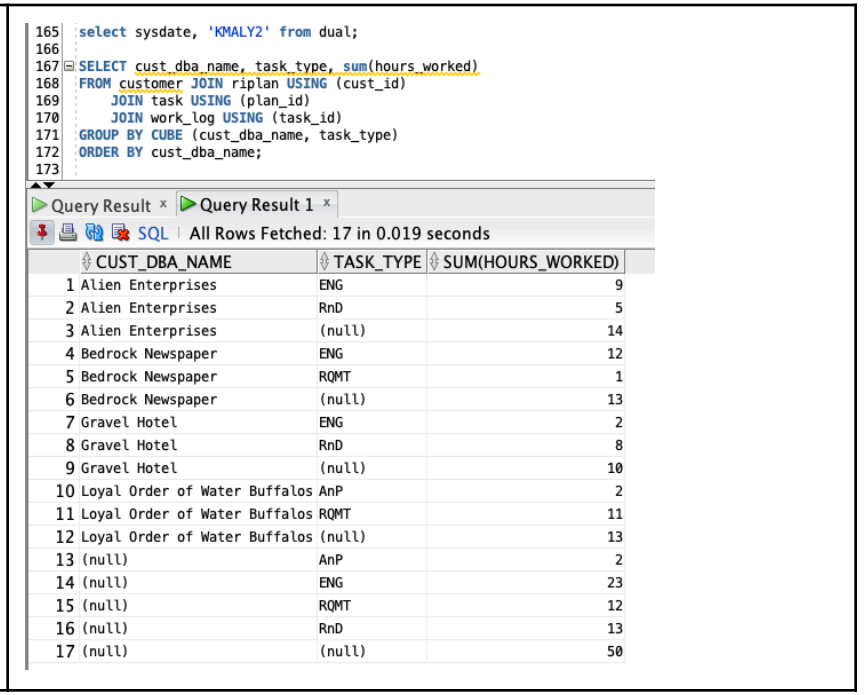


```
165  select sysdate, 'KMALY2' from dual;
166
167  SELECT cust_dba_name, task_type, sum(hours_worked)
168  FROM customer JOIN riplan USING (cust_id)
169      JOIN task USING (plan_id)
170      JOIN work_log USING (task_id)
171  GROUP BY CUBE (cust_dba_name, task_type)
172  ORDER BY cust_dba_name;
173
```

Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.013 seconds

	SYSDATE	'KMALY2'
1	27-APR-23	KMALY2



```
165  select sysdate, 'KMALY2' from dual;
166
167  SELECT cust_dba_name, task_type, sum(hours_worked)
168  FROM customer JOIN riplan USING (cust_id)
169      JOIN task USING (plan_id)
170      JOIN work_log USING (task_id)
171  GROUP BY CUBE (cust_dba_name, task_type)
172  ORDER BY cust_dba_name;
173
```

Query Result | Query Result 1

SQL | All Rows Fetched: 17 in 0.019 seconds

	CUST_DBA_NAME	TASK_TYPE	SUM(HOURS_WORKED)
1	Alien Enterprises	ENG	9
2	Alien Enterprises	RnD	5
3	Alien Enterprises	(null)	14
4	Bedrock Newspaper	ENG	12
5	Bedrock Newspaper	RQMT	1
6	Bedrock Newspaper	(null)	13
7	Gravel Hotel	ENG	2
8	Gravel Hotel	RnD	8
9	Gravel Hotel	(null)	10
10	Loyal Order of Water Buffalos	AnP	2
11	Loyal Order of Water Buffalos	RQMT	11
12	Loyal Order of Water Buffalos	(null)	13
13	(null)	AnP	2
14	(null)	ENG	23
15	(null)	RQMT	12
16	(null)	RnD	13
17	(null)	(null)	50

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