CST8276 Lab 8: Query Optimizer – Part 2

Purpose: **To provide you with the opportunity to gain additional insight into aspects of query performance.**

Deliverables: **Provide the specified answers and results in a file named Lab8_*yourName*. Submit the file to BrightSpace and demo the results to your lab professor during the lab session. The lab exercise is worth 2 marks towards your lab exercise if correct and submitted on-time or early.**

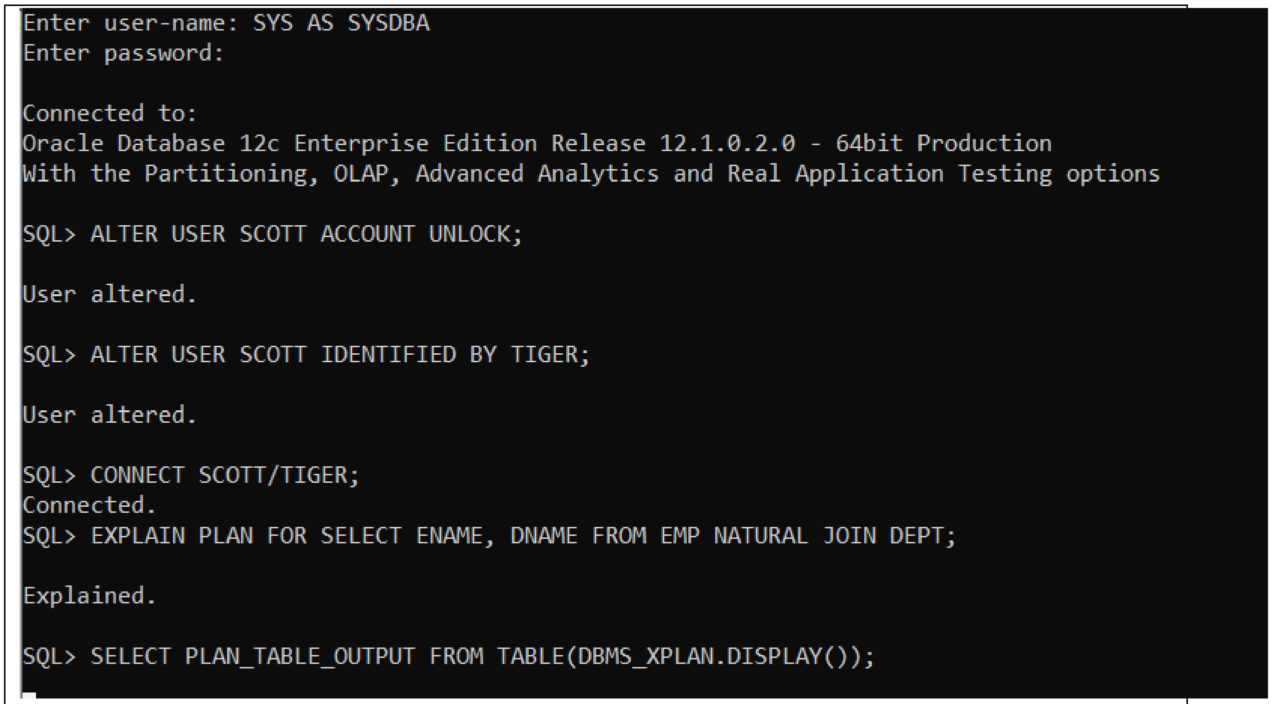
Activities:

1. **Logon as sysdba and unlock the SCOTT account. Also change the password of the scott account into "tiger".**
   a. **From the scott user account:**
      i. **Enter:** ***explain plan for***
      select ename, dname from emp natural join dept;

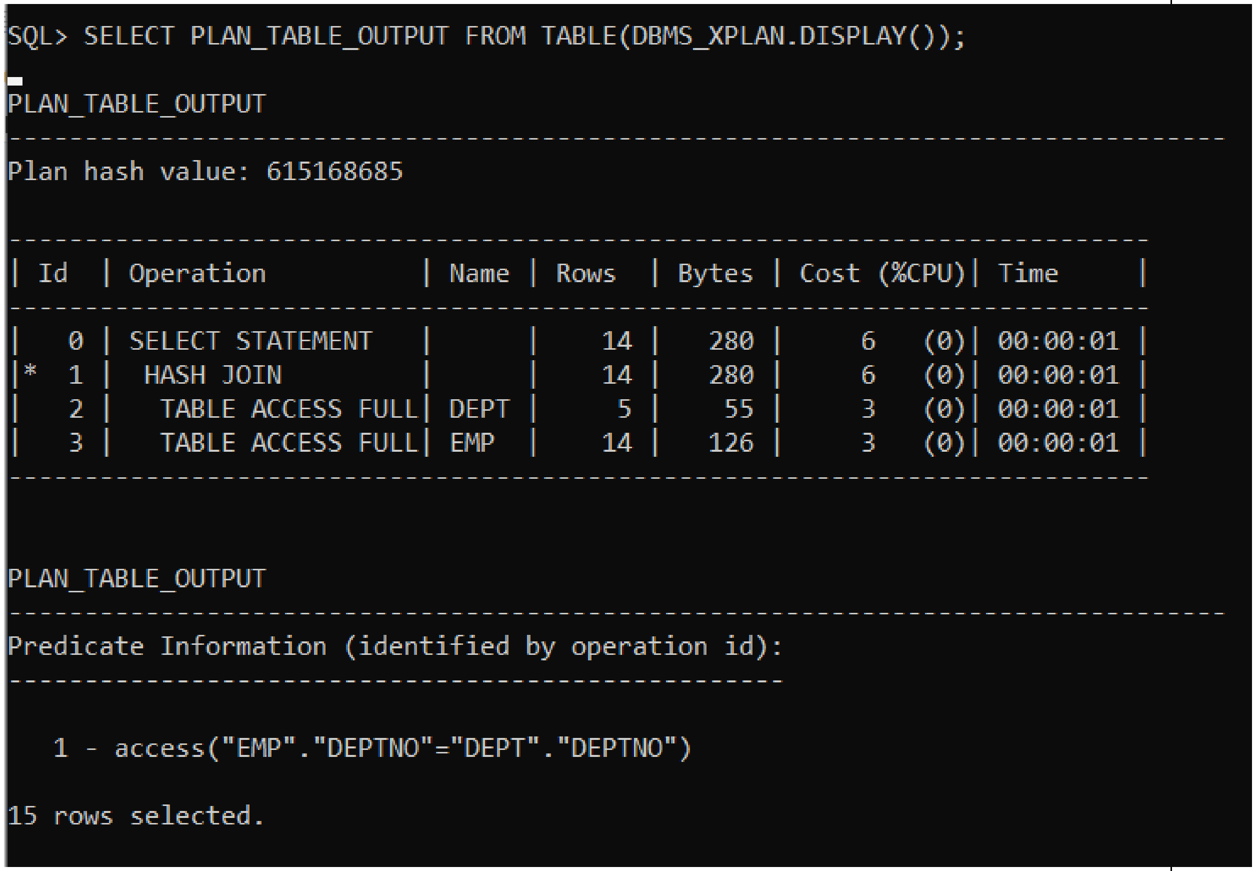
      Enter: ***SELECT PLAN_TABLE_OUTPUT***
      FROM TABLE(DBMS_XPLAN.DISPLAY());

      ii. **In your MS Word submission document, clearly identify the steps that oracle is planning to use in order to execute the query submitted in (*ai*) , and paste a screen snapshot of the Explain Plan query and the Plan_Table_Output.**

CST8276 Lab 8: Query Optimizer – Part 2

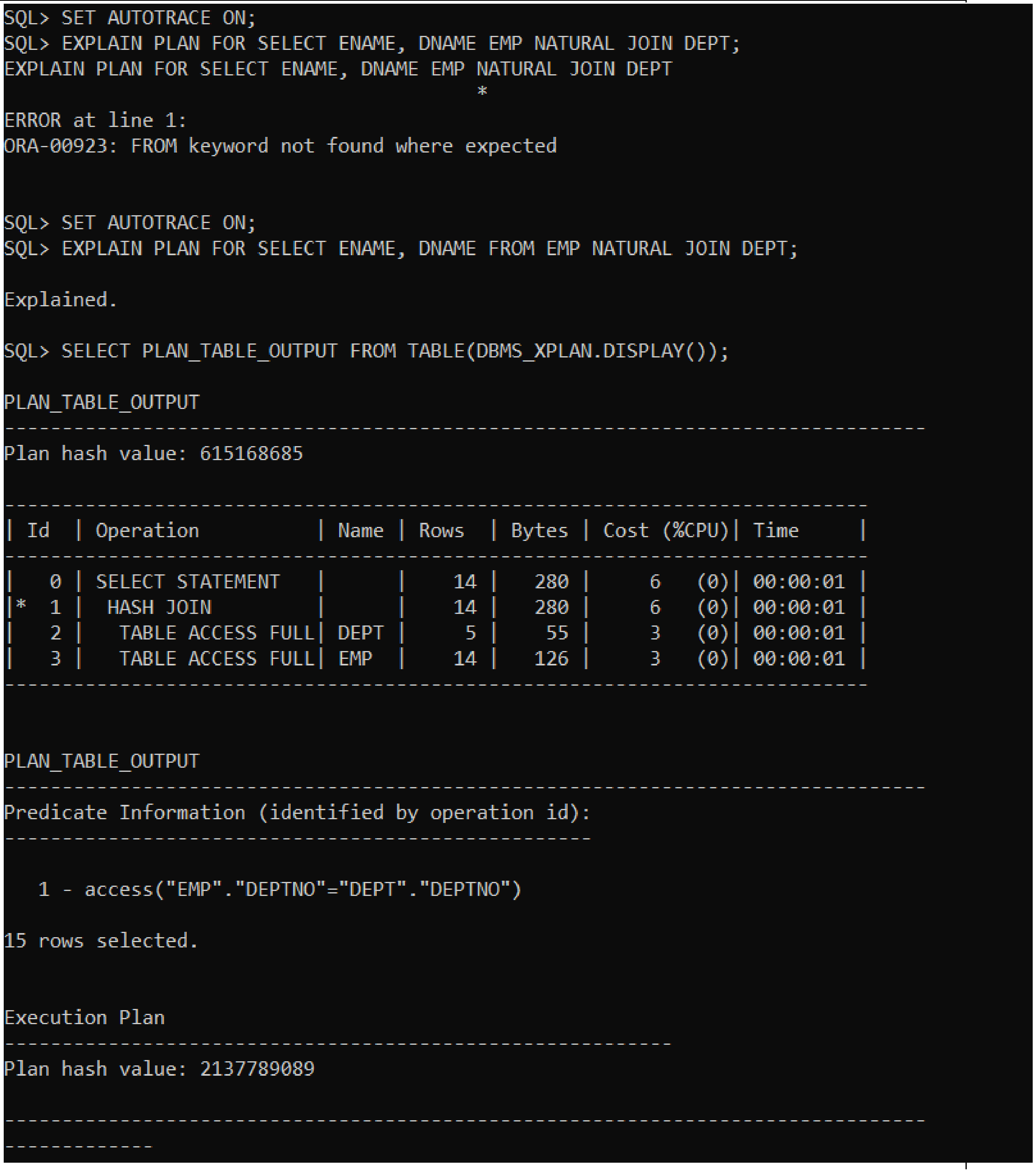


```
Enter user-name: SYS AS SYSDBA
Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> ALTER USER SCOTT ACCOUNT UNLOCK;

User altered.

SQL> ALTER USER SCOTT IDENTIFIED BY TIGER;

User altered.

SQL> CONNECT SCOTT/TIGER;
Connected.
SQL> EXPLAIN PLAN FOR SELECT ENAME, DNAME FROM EMP NATURAL JOIN DEPT;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());
```

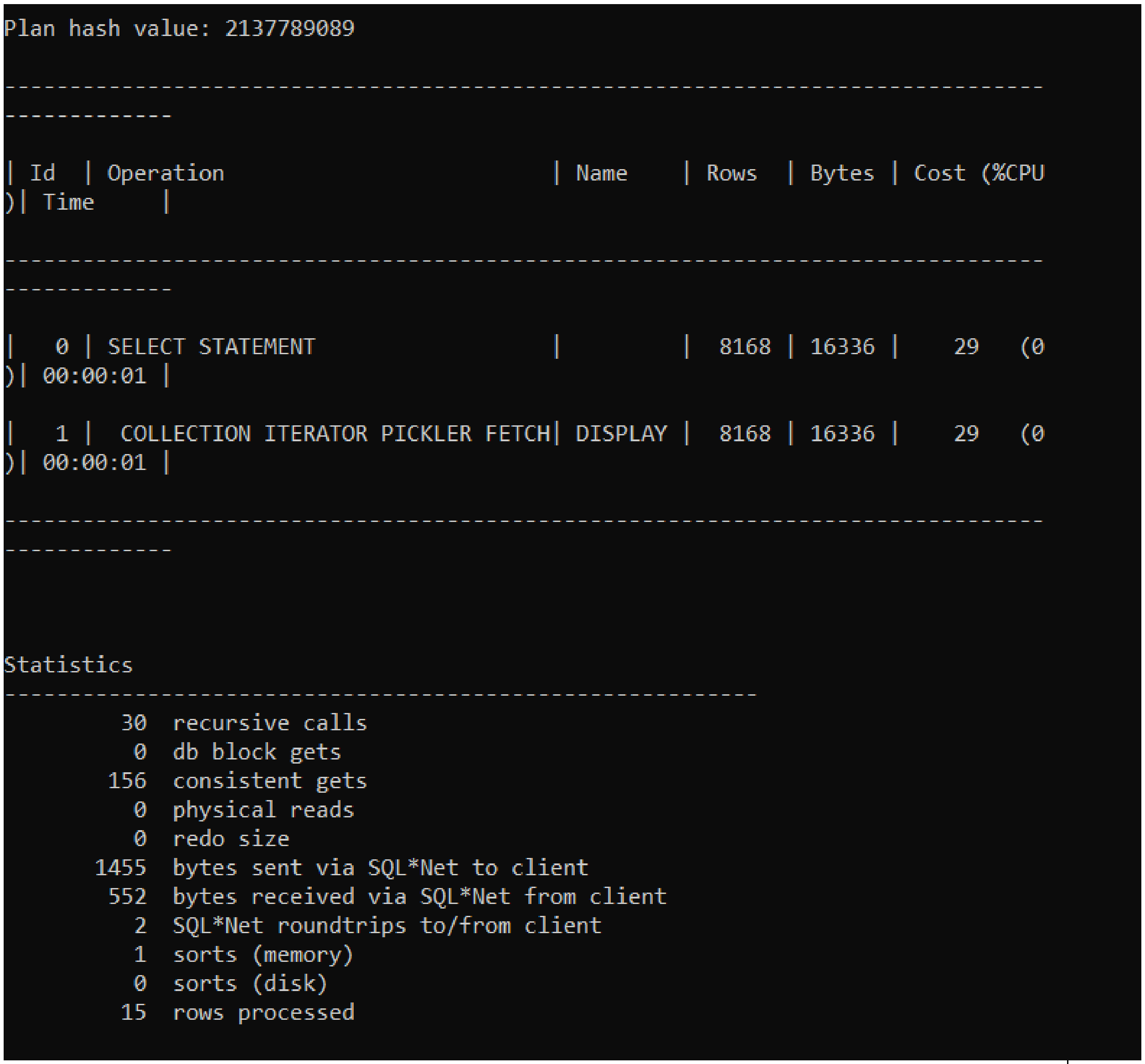


```
SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 615168685

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    14 |   280 |     6   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |      |    14 |   280 |     6   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPT |     5 |    55 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMP  |    14 |   126 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")

15 rows selected.
```

b. Set autotrace ON and re-execute the query in (ai). Paste a screen snapshot of Plan_Table_Output .

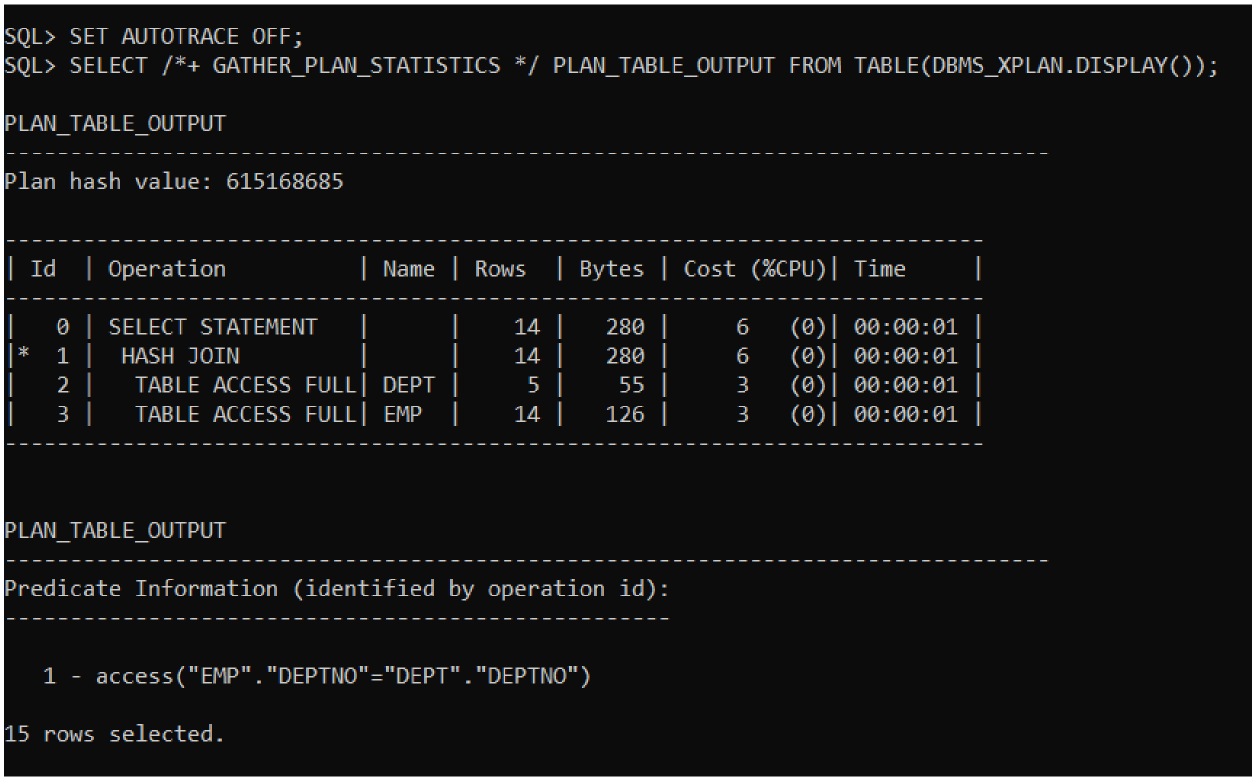


```
SQL> SET AUTOTRACE ON;
SQL> EXPLAIN PLAN FOR SELECT ENAME, DNAME EMP NATURAL JOIN DEPT;
EXPLAIN PLAN FOR SELECT ENAME, DNAME EMP NATURAL JOIN DEPT
                                           *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> SET AUTOTRACE ON;
SQL> EXPLAIN PLAN FOR SELECT ENAME, DNAME FROM EMP NATURAL JOIN DEPT;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 615168685

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    14 |   280 |     6   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |      |    14 |   280 |     6   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPT |     5 |    55 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMP  |    14 |   126 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")

15 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

--------------------------------------------------------------------------------
-------------
```



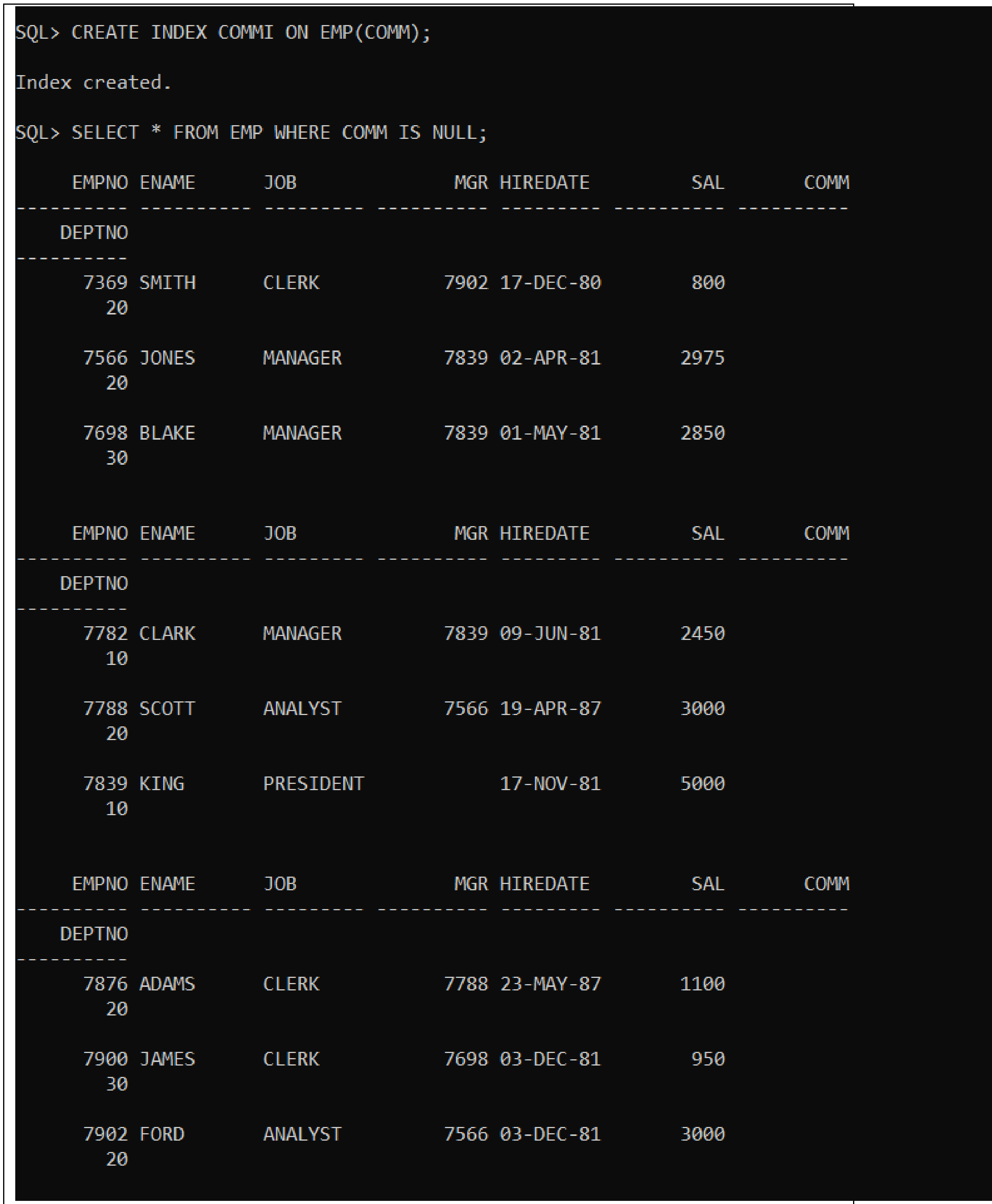
```
Plan hash value: 2137789089

--------------------------------------------------------------------------------
-------------

| Id  | Operation                         | Name    | Rows  | Bytes | Cost (%CPU
)| Time     |

--------------------------------------------------------------------------------
-------------

|   0 | SELECT STATEMENT                  |         |  8168 | 16336 |    29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |  8168 | 16336 |    29   (0
)| 00:00:01 |

--------------------------------------------------------------------------------
-------------



Statistics
----------------------------------------------------------
         30  recursive calls
          0  db block gets
        156  consistent gets
          0  physical reads
          0  redo size
       1455  bytes sent via SQL*Net to client
        552  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
         15  rows processed
```

c. **Set autotrace OFF. Give oracle hints on how to execute the above query. Remember you can give the hints by inserting *`/*+ gather_plan_statistics */`* right after the select word in the select statement.**


```
SQL> SET AUTOTRACE OFF;
SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 615168685

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    14 |   280 |     6   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |      |    14 |   280 |     6   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPT |     5 |    55 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMP  |    14 |   126 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")

15 rows selected.
```

d. **Oracle uses the dbmx_xplan.display_cursor to pullout the format information stored in the library cache of the shared pool. Use the below statement to show the plan_table_output of all statistics that are related to the last executed query.**
select * from table(dbms_xplan.display_cursor(format =>'allstats last'));
Remember you may need to grant the user SCOTT the right privileges in order to display the information that has been stored in the statistical table. Indicate all your work.

2. **Index null values:**
    a. **Create an index "*commi*" on field comm in the emp table. Set autotrace ON, and then run a query to find all the employees with commission equals to null. Include a screenshot of the explain plan and plan table output. Has oracle used the index *commi* to answer the above query?**



```
SQL> CREATE INDEX COMMI ON EMP(COMM);

Index created.

SQL> SELECT * FROM EMP WHERE COMM IS NULL;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7369 SMITH      CLERK           7902 17-DEC-80        800
        20

      7566 JONES      MANAGER         7839 02-APR-81       2975
        20

      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
        30


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
        10

      7788 SCOTT      ANALYST         7566 19-APR-87       3000
        20

      7839 KING       PRESIDENT            17-NOV-81       5000
        10


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7876 ADAMS      CLERK           7788 23-MAY-87       1100
        20

      7900 JAMES      CLERK           7698 03-DEC-81        950
        30

      7902 FORD       ANALYST         7566 03-DEC-81       3000
        20
```

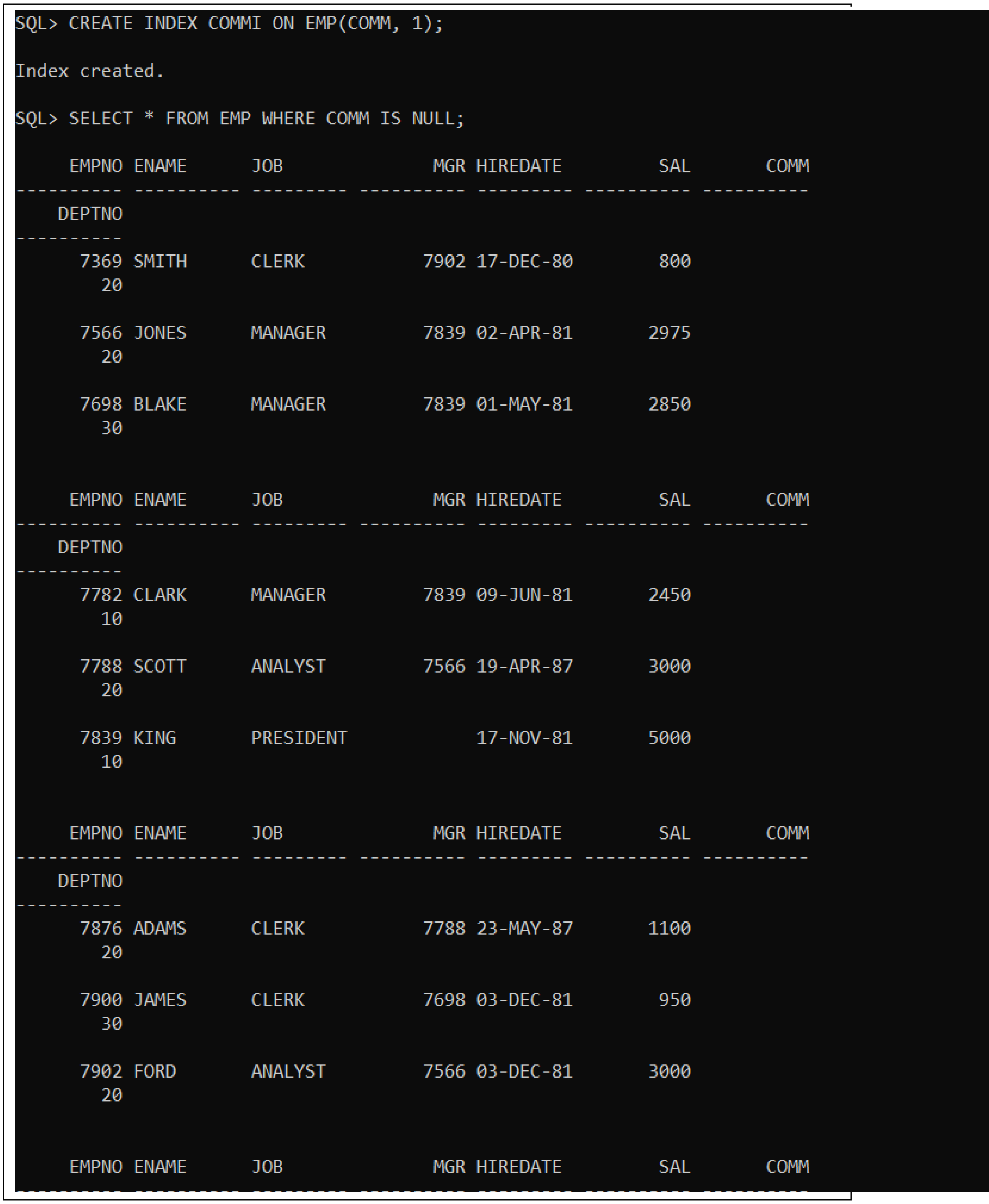
D

Why?

It's because the index value is showing null.

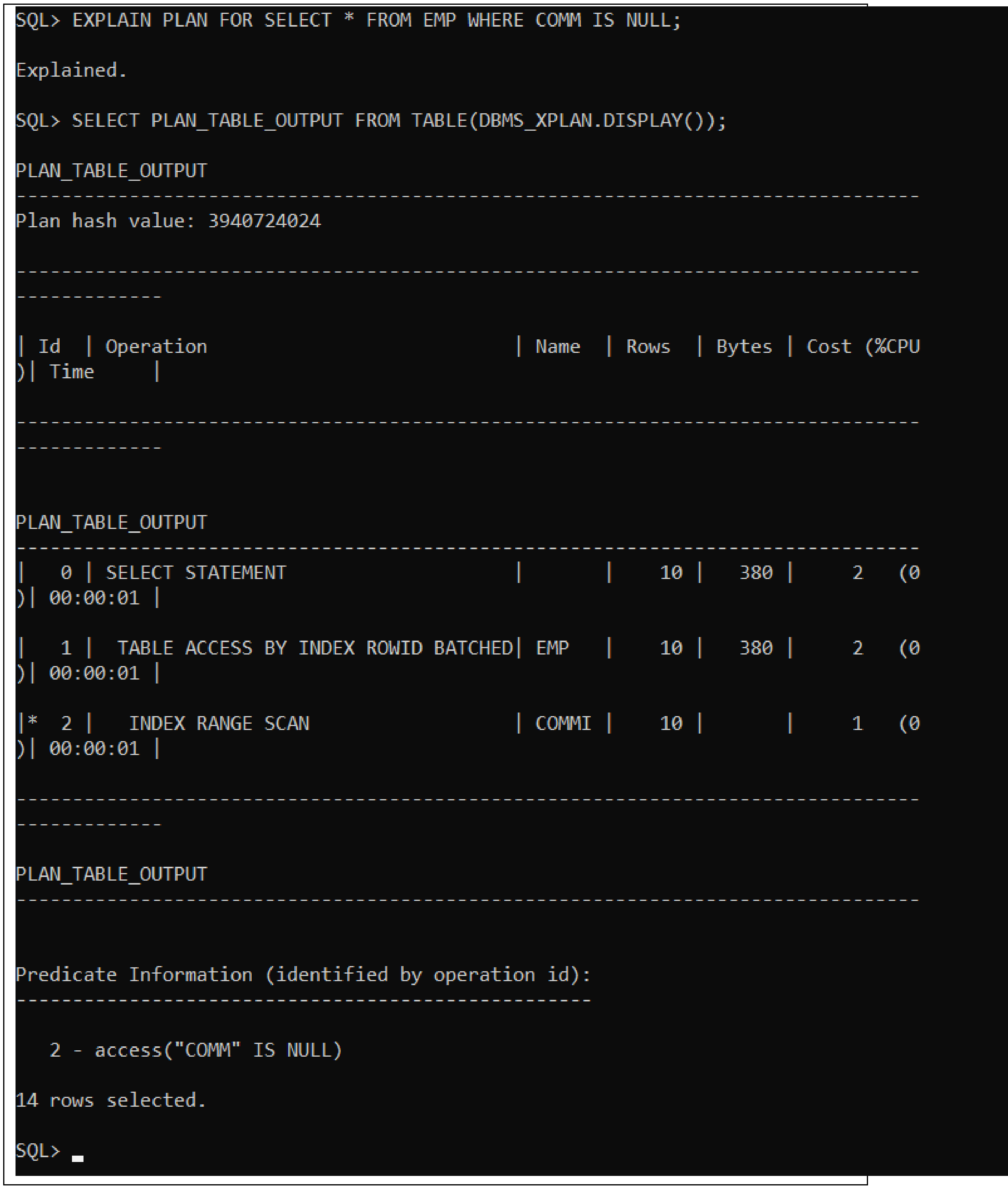
b. **Drop the index *commi,* and then create another index on the same field with the same name but slightly a different structure. *Create index commi on emp(comm, 1)*. Run the same query again as in part 2a. Include a screenshot of the explain plan and plan table output.**

CST8276 Lab 8: Query Optimizer – Part 2



```
SQL> CREATE INDEX COMMI ON EMP(COMM, 1);

Index created.

SQL> SELECT * FROM EMP WHERE COMM IS NULL;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7369 SMITH      CLERK           7902 17-DEC-80        800
        20

      7566 JONES      MANAGER         7839 02-APR-81       2975
        20

      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
        30


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
        10

      7788 SCOTT      ANALYST         7566 19-APR-87       3000
        20

      7839 KING       PRESIDENT            17-NOV-81       5000
        10


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7876 ADAMS      CLERK           7788 23-MAY-87       1100
        20

      7900 JAMES      CLERK           7698 03-DEC-81        950
        30

      7902 FORD       ANALYST         7566 03-DEC-81       3000
        20


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
```

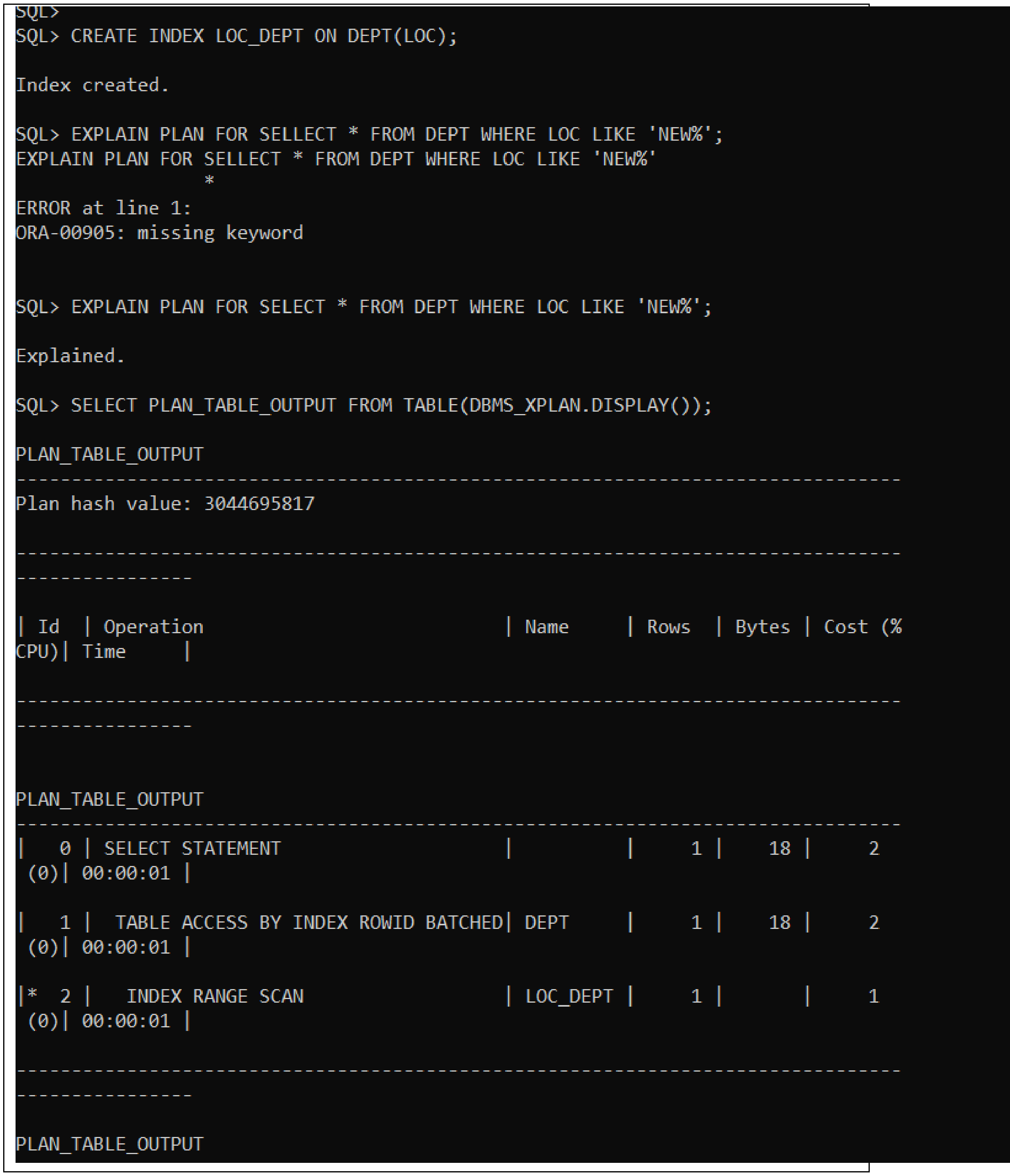
Has oracle used the index *commi* to answer the above query?



```
SQL> EXPLAIN PLAN FOR SELECT * FROM EMP WHERE COMM IS NULL;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3940724024

--------------------------------------------------------------------------------
-------------

| Id  | Operation                            | Name  | Rows  | Bytes | Cost (%CPU
)| Time     |

--------------------------------------------------------------------------------
-------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |       |    10 |   380 |     2   (0
)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| EMP   |    10 |   380 |     2   (0
)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN                   | COMMI |    10 |       |     1   (0
)| 00:00:01 |

--------------------------------------------------------------------------------
-------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("COMM" IS NULL)

14 rows selected.

SQL> _
```

3. **Index wildcards %:**
   a. **Now we will work on the dept table. Create an index on the *LOC* field. Write a query that finds all the departments having their location starts with “NEW”. Include a screenshot of the explain**

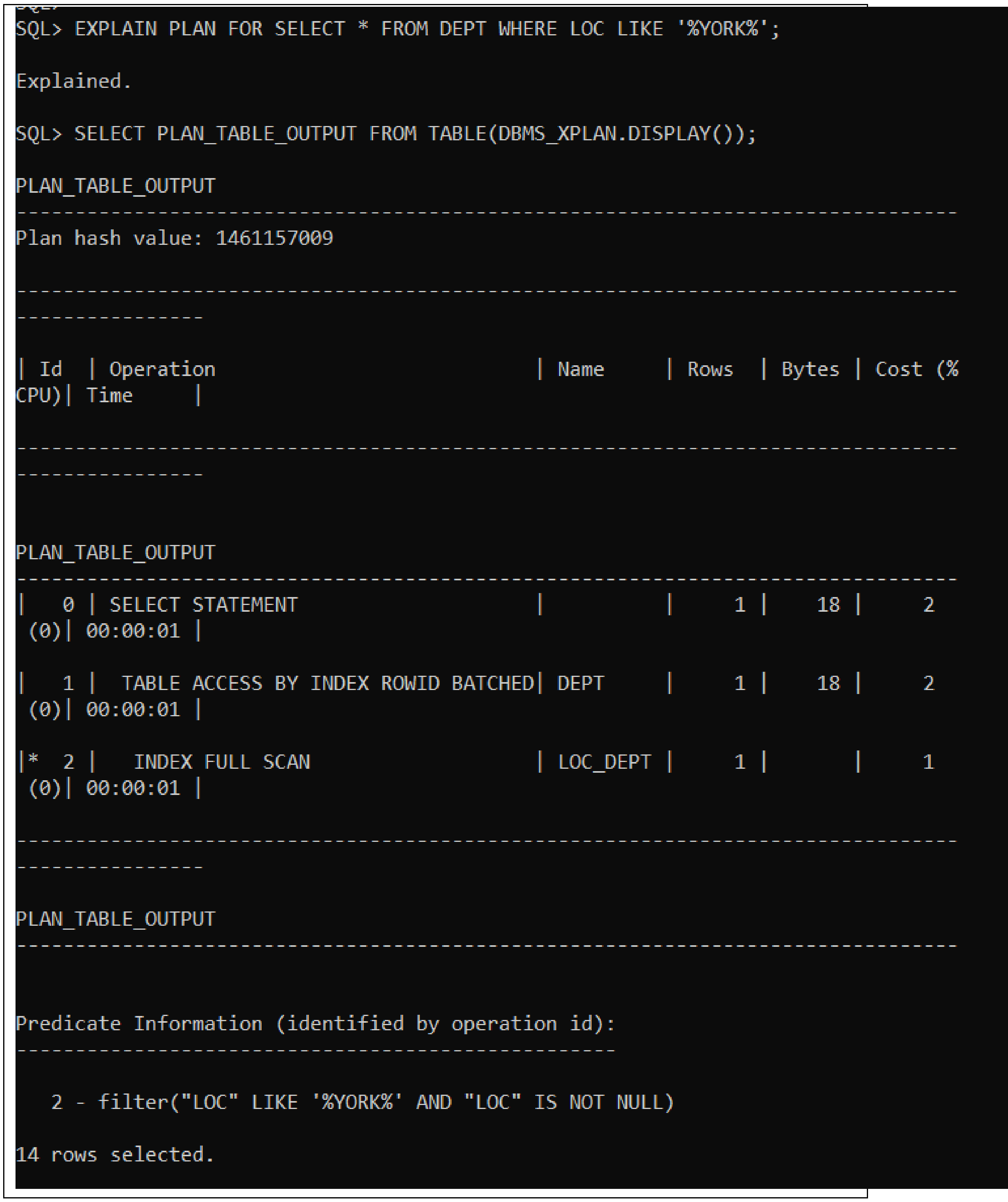
plan and plan table output. Has oracle used the index to answer the query?



```
SQL>
SQL> CREATE INDEX LOC_DEPT ON DEPT(LOC);

Index created.

SQL> EXPLAIN PLAN FOR SELLECT * FROM DEPT WHERE LOC LIKE 'NEW%';
EXPLAIN PLAN FOR SELLECT * FROM DEPT WHERE LOC LIKE 'NEW%'
                 *
ERROR at line 1:
ORA-00905: missing keyword


SQL> EXPLAIN PLAN FOR SELECT * FROM DEPT WHERE LOC LIKE 'NEW%';

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3044695817

--------------------------------------------------------------------------------
----------------

| Id  | Operation                           | Name     | Rows  | Bytes | Cost (%
CPU)| Time     |

--------------------------------------------------------------------------------
----------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |          |     1 |    18 |     2
 (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DEPT     |     1 |    18 |     2
 (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN                  | LOC_DEPT |     1 |       |     1
 (0)| 00:00:01 |

--------------------------------------------------------------------------------
----------------

PLAN_TABLE_OUTPUT
```

b. **Write a query that finds all the departments having their location includes the word "YORK". Include a screenshot of the explain**

plan and plan table output. Has oracle used the index to answer the query?

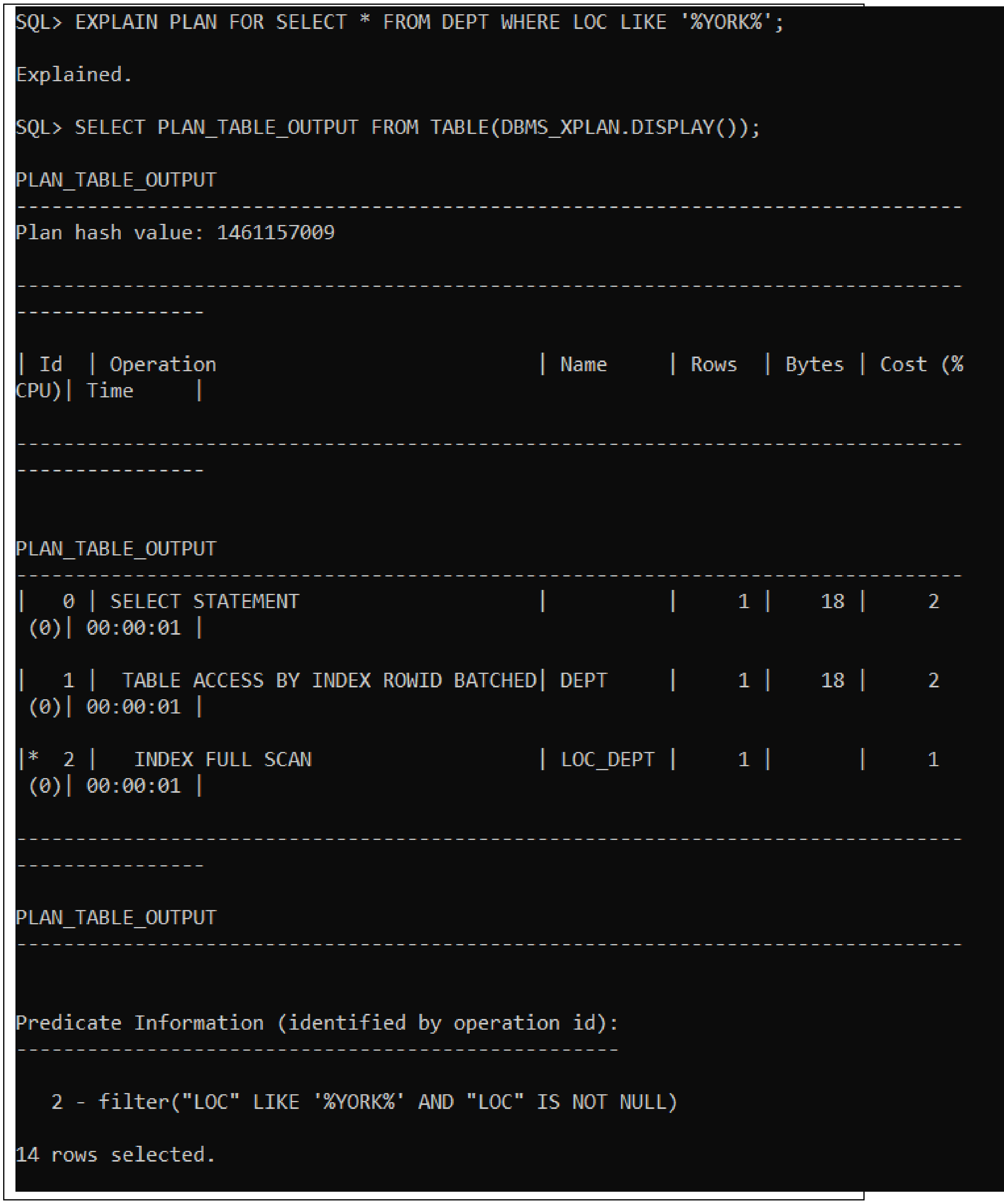


```
SQL> EXPLAIN PLAN FOR SELECT * FROM DEPT WHERE LOC LIKE '%YORK%';

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1461157009

--------------------------------------------------------------------------------
----------------

| Id  | Operation                           | Name     | Rows  | Bytes | Cost (%
CPU)| Time     |

--------------------------------------------------------------------------------
----------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |          |     1 |    18 |     2
 (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DEPT     |     1 |    18 |     2
 (0)| 00:00:01 |

|*  2 |   INDEX FULL SCAN                   | LOC_DEPT |     1 |       |     1
 (0)| 00:00:01 |

--------------------------------------------------------------------------------
----------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("LOC" LIKE '%YORK%' AND "LOC" IS NOT NULL)

14 rows selected.
```

c. **Create another index with different structure on the loc field.**
Create index locctx on dept(loc) indextype is ctxsys.context;

CST8276 Lab 8: Query Optimizer – Part 2

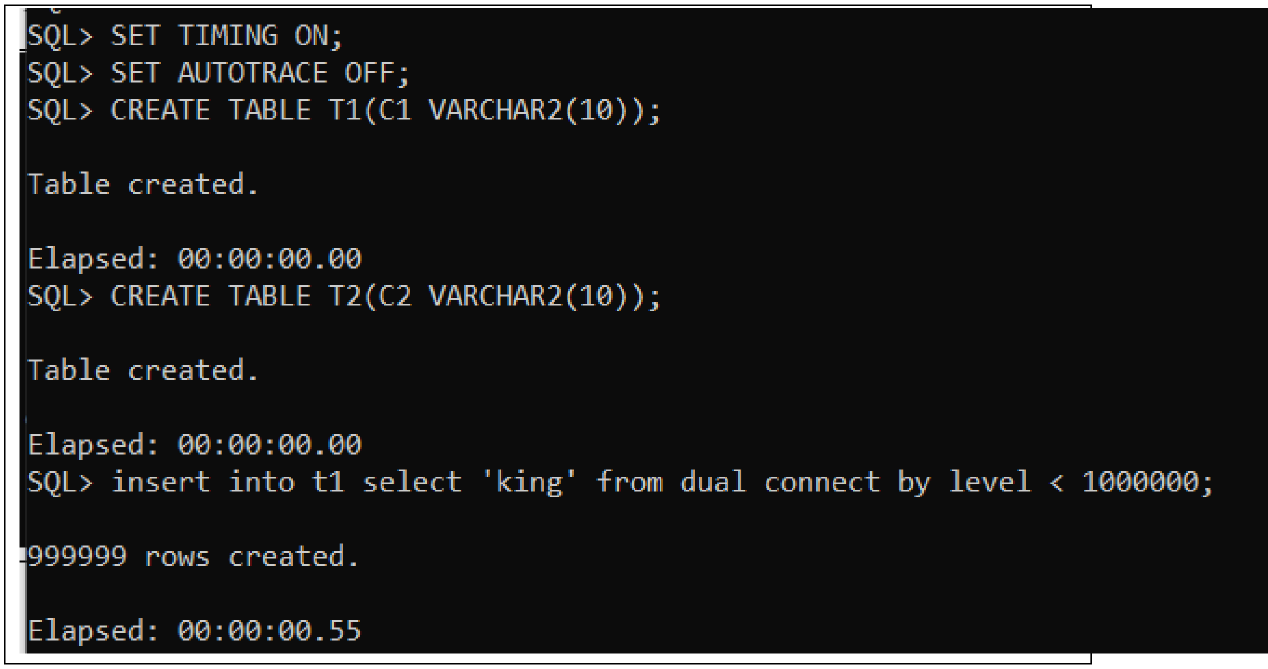
Include a screenshot of the explain plan and plan table output for the following slightly changed from the above query?
Select * from dept where contains(loc, 'YORK') > 0;



```
SQL> EXPLAIN PLAN FOR SELECT * FROM DEPT WHERE LOC LIKE '%YORK%';

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1461157009

--------------------------------------------------------------------------------
----------------

| Id  | Operation                           | Name     | Rows  | Bytes | Cost (%
CPU)| Time     |

--------------------------------------------------------------------------------
----------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |          |     1 |    18 |     2
 (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DEPT     |     1 |    18 |     2
 (0)| 00:00:01 |

|*  2 |   INDEX FULL SCAN                   | LOC_DEPT |     1 |       |     1
 (0)| 00:00:01 |

--------------------------------------------------------------------------------
----------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("LOC" LIKE '%YORK%' AND "LOC" IS NOT NULL)

14 rows selected.
```

Has oracle used the index *locctx*

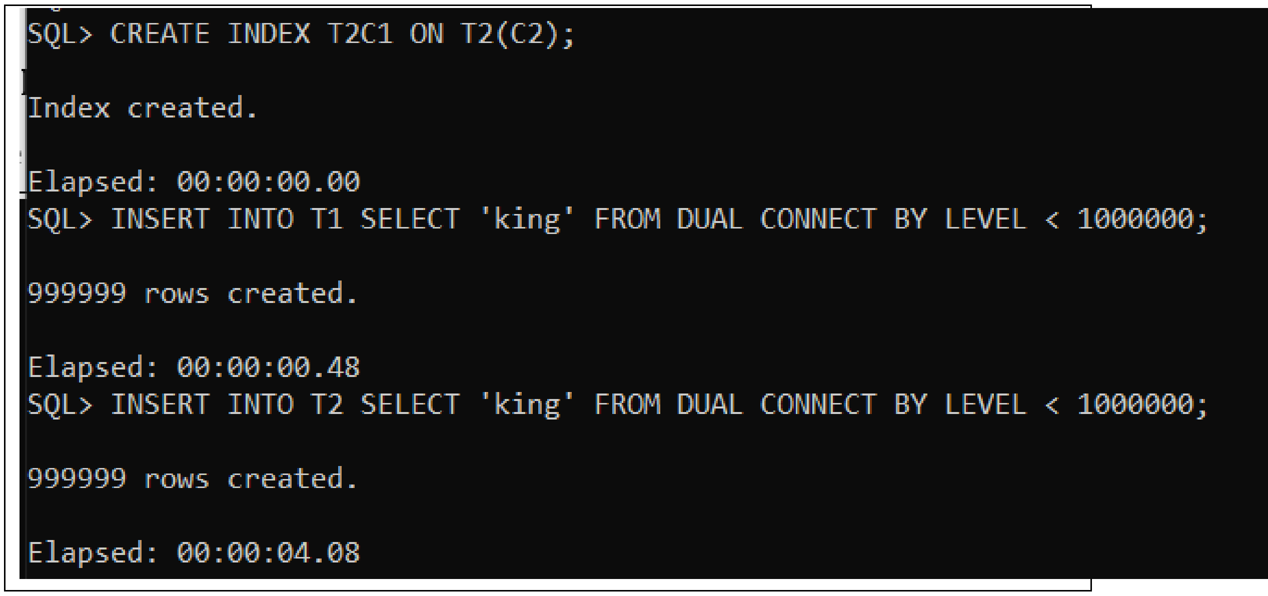
4. **When Indexing is good?**
   a. **Set the autotrace off and Set the timing on. Create two tables t1 and t2.**
   Create table t1(c1 varchar2(10));
   Create table t2(c1 varchar2(10));
   Run the below query to insert 1 million record in the table t1
   insert into t1 select 'king' from dual connect by level < 1000000;
   Take note of the time needed to insert into the table t1.



```
SQL> SET TIMING ON;
SQL> SET AUTOTRACE OFF;
SQL> CREATE TABLE T1(C1 VARCHAR2(10));

Table created.

Elapsed: 00:00:00.00
SQL> CREATE TABLE T2(C2 VARCHAR2(10));

Table created.

Elapsed: 00:00:00.00
SQL> insert into t1 select 'king' from dual connect by level < 1000000;

999999 rows created.

Elapsed: 00:00:00.55
```

   Now create an index on field c1 in the t2 table and then insert one million records to t2. Compare the two running times. Do you think indexing is good all the times?

CST8276 Lab 8: Query Optimizer – Part 2



```
SQL> CREATE INDEX T2C1 ON T2(C2);

Index created.

Elapsed: 00:00:00.00
SQL> INSERT INTO T1 SELECT 'king' FROM DUAL CONNECT BY LEVEL < 1000000;

999999 rows created.

Elapsed: 00:00:00.48
SQL> INSERT INTO T2 SELECT 'king' FROM DUAL CONNECT BY LEVEL < 1000000;

999999 rows created.

Elapsed: 00:00:04.08
```

Give a real-life situation where an index could/should be created after inserting data rather than before inserting data.

If we're using a database where we only want to read the records, indexes can be created to get the records faster, but if there will be constant adds and updates, they shouldn't be used since it's slower to make those operations when there are indexes.

You're finished. Please submit.