**CST8276 – Advanced Database Topics: Lab Assignment 7**

**Student Name: Onur Onel**

**Student ID: 041074824**

**Student email: onel0001@algonquinlive.com**

**Activities:**

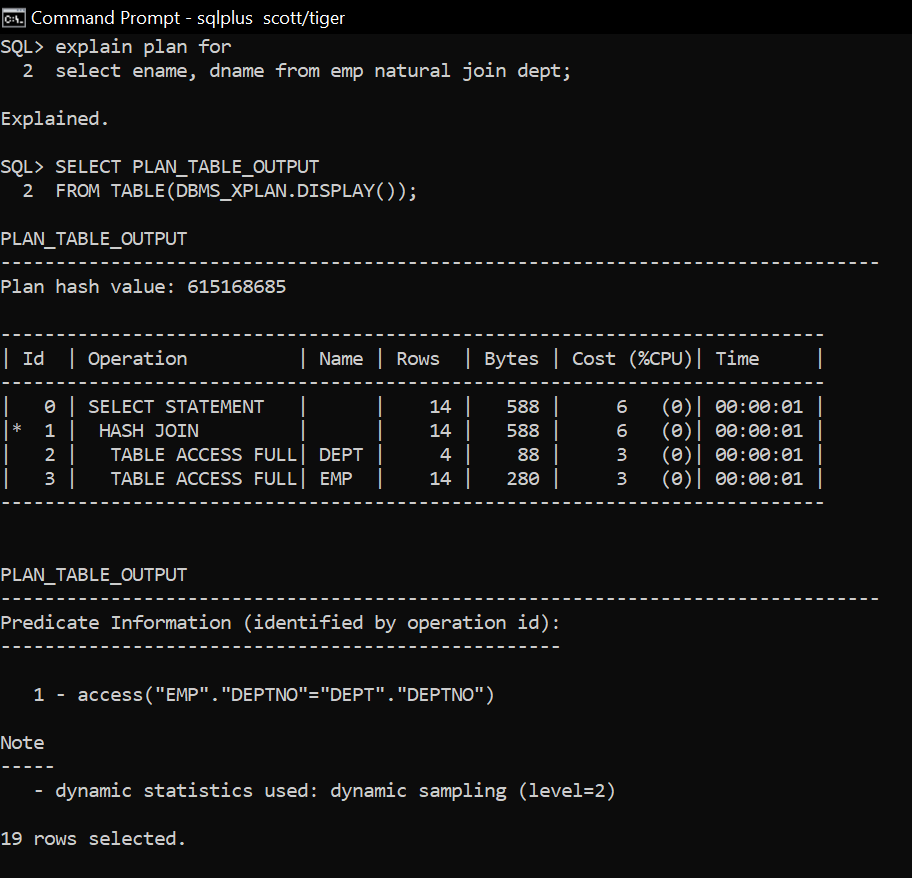
1. Logon as sysdba and unlock the SCOTT account. Also change the password of the scott account into “tiger”.
   1. From the scott user account:
      1. Enter: *explain plan for*

*select ename, dname from emp natural join dept;*

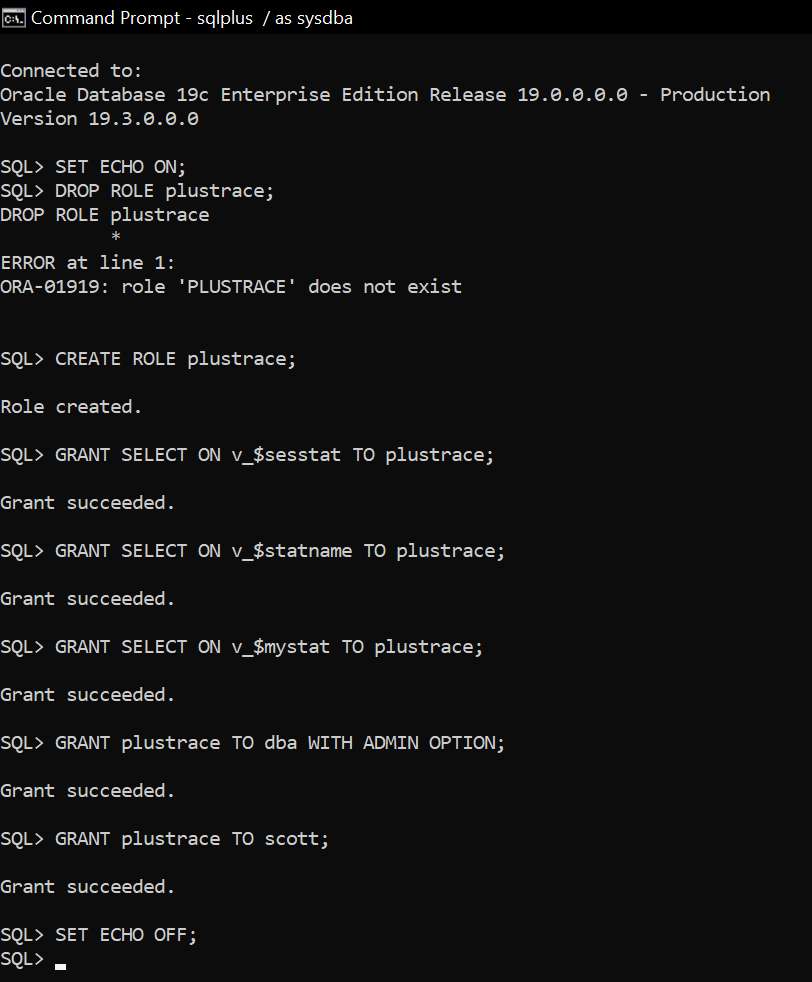
Enter: *SELECT PLAN\_TABLE\_OUTPUT*

*FROM TABLE(DBMS\_XPLAN.DISPLAY());*

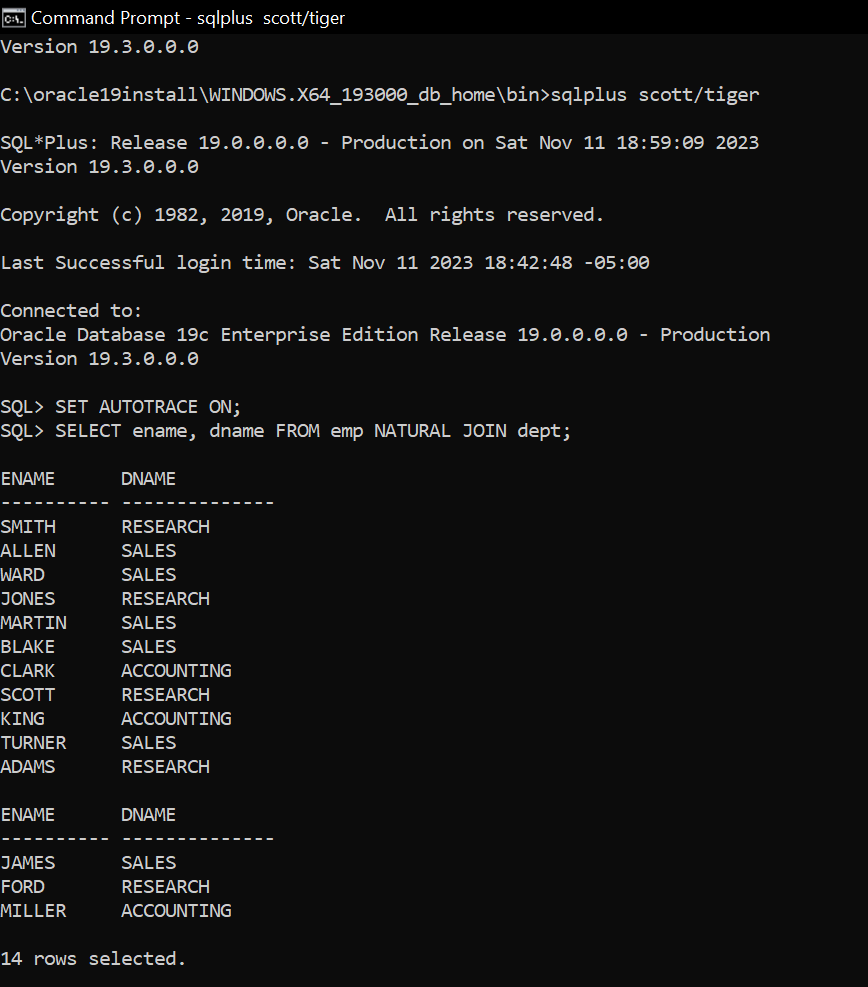
* + 1. In your MS Word submission document, clearly identify the steps that oracle is planning to use in order to execute the query submitted in (1*ai*) , and paste a screen snapshot of the Explain Plan query and the Plan\_Table\_Output.

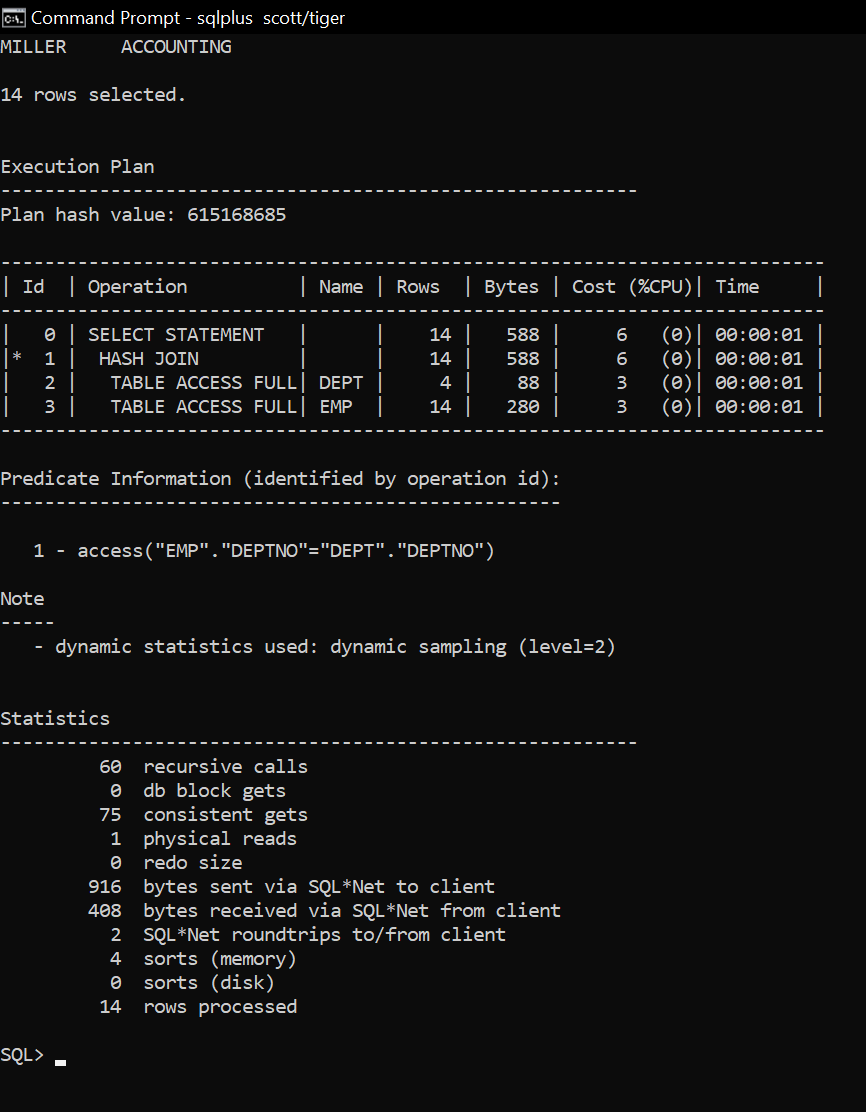


* 1. This step will require you to investigate the “plustrace” role that allows tracing in realtime. You will Set autotrace ON and re-execute the query in (ai).
     1. First, create the plustrace role. The commands required are contained in the plustrc.sql file which is included in the Oracle19c install files. (Mine was located at:D:\Oracle19cinstall\sqlplus\admin)
     2. Then, grant the plustrace role to the scott account. Show your work below.



* 1. Now, execute *set autotrace on;* and re-generate the explain plan for the query in 1ai. Paste a screen snapshot of Plan\_Table\_Output .



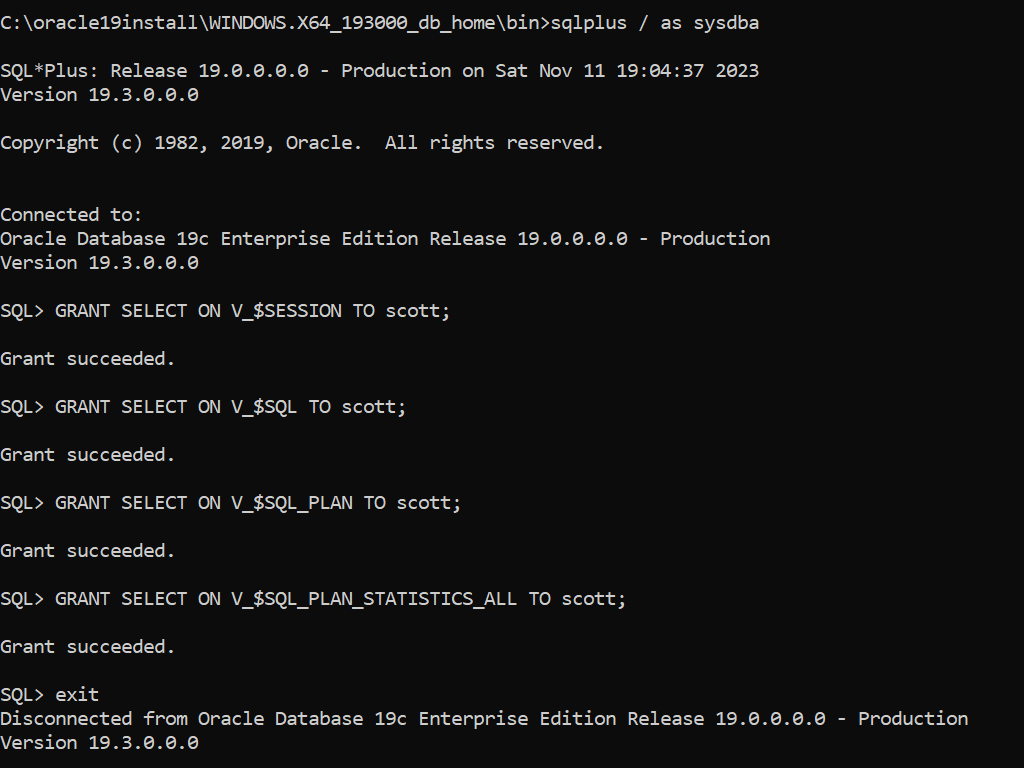


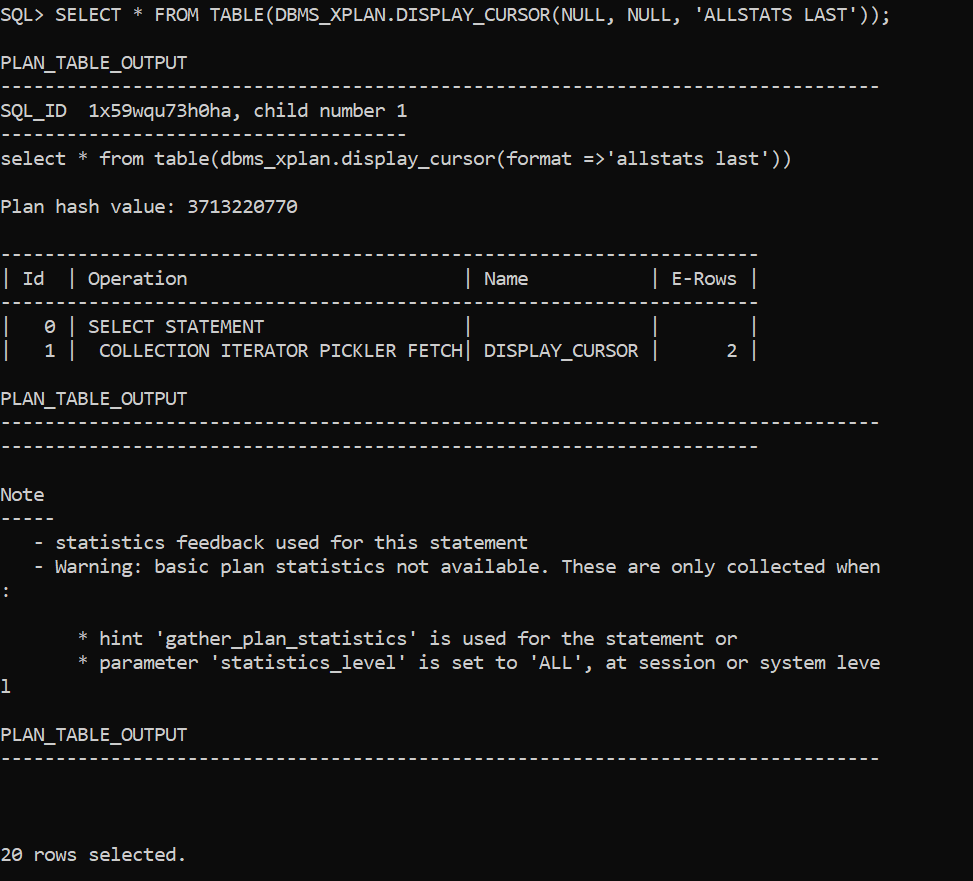
* 1. *Set autotrace OFF*. Modify the select statement from 1ai to give Oracle a hint by inserting */\*+ gather\_plan\_statistics \*/* right after the select word in the select statement.
  2. If you add the hint to “gather\_plan\_statistics”, then Oracle uses the *dbmx\_xplan.display\_cursor* to pullout the format information stored in the library cache of the shared pool. You DO NOT have to run the “EXPLAIN PLAN” command explicitly. After you simply just execute the query (i.e., without the prefix “EXPLAIN PLAN FOR ….”) you can use the statement below to show the plan\_table\_output containing all statistics that are available for the last executed query.

*select \* from table(dbms\_xplan.display\_cursor(format =>'allstats last'));*

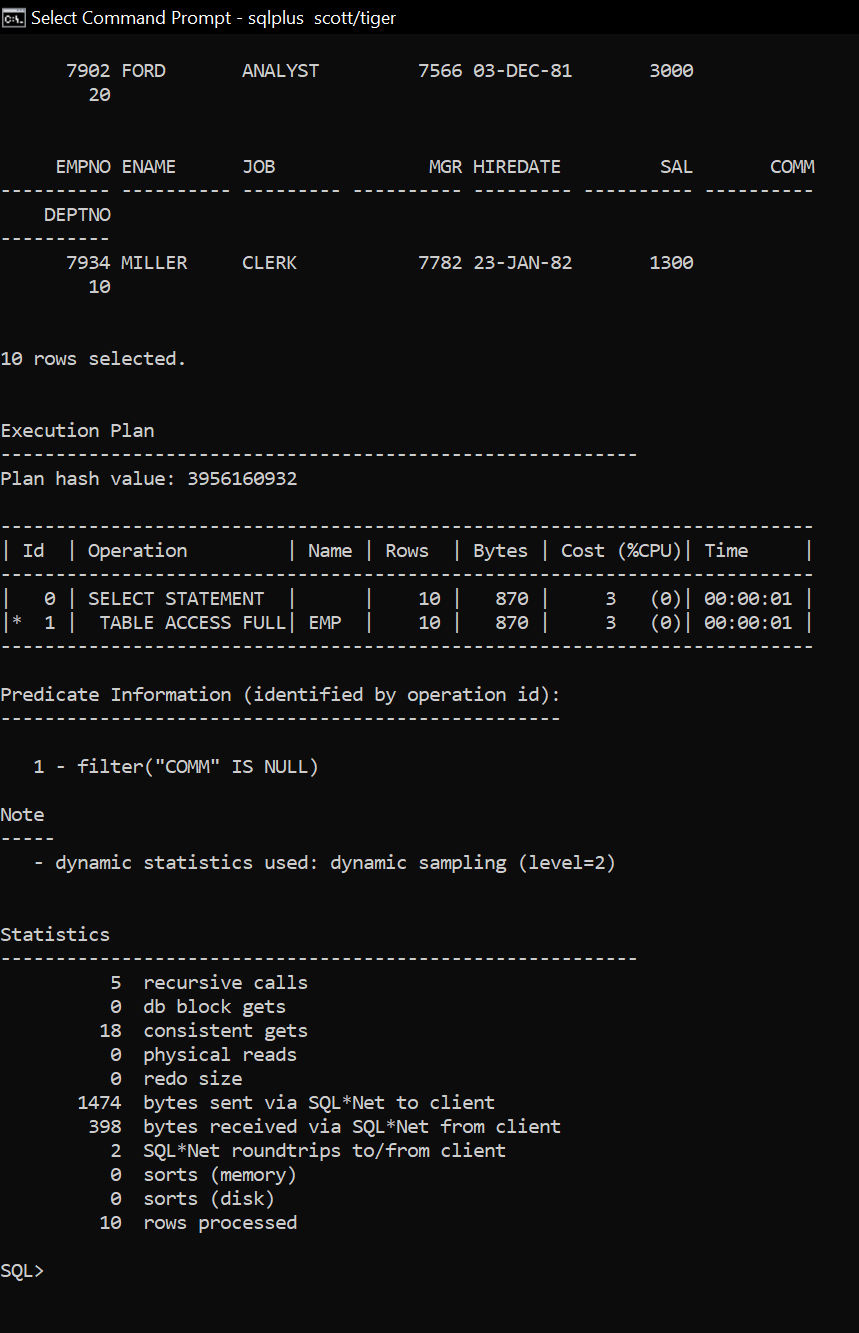
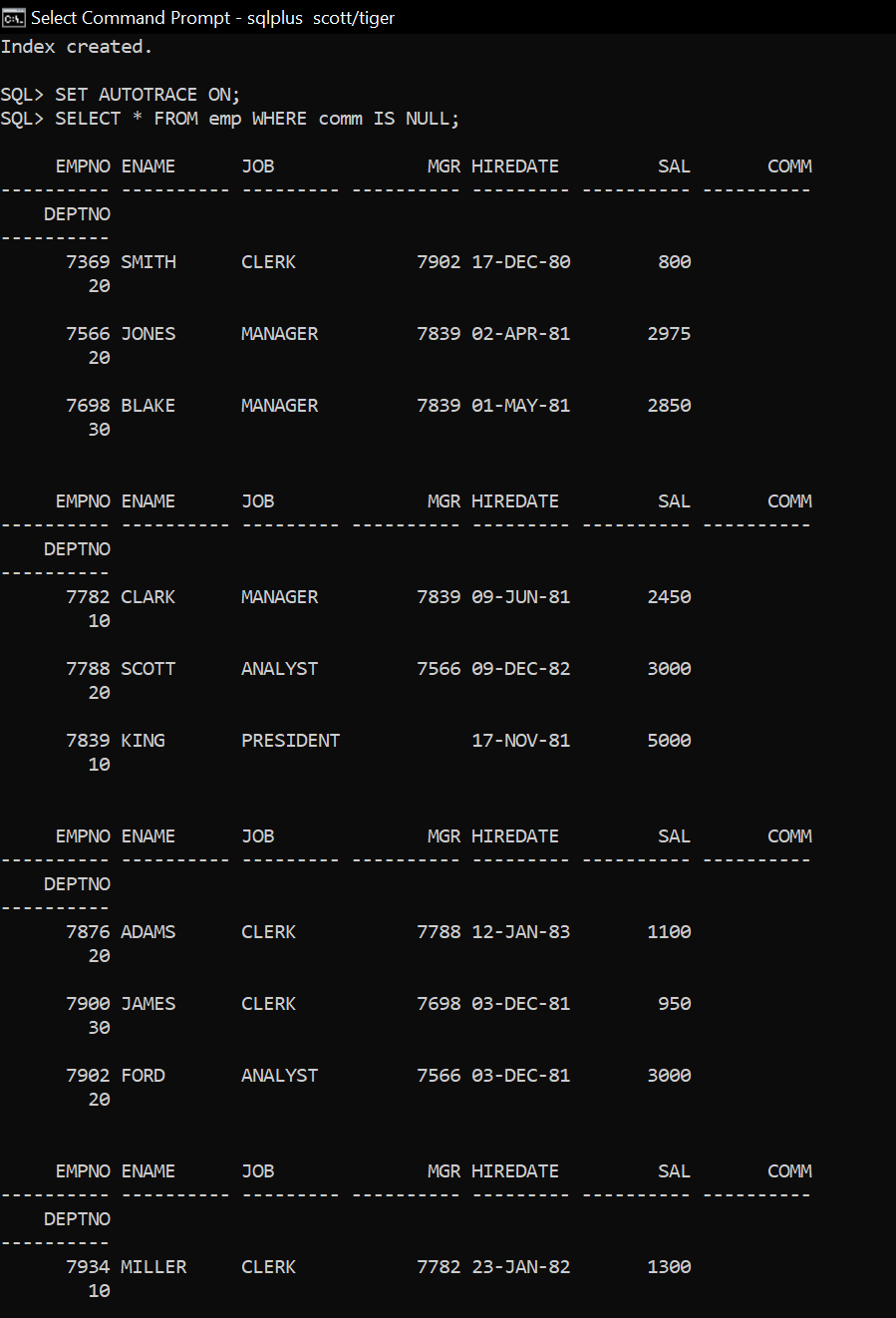
This command gets the statistics for the last query containing the “gather\_plan\_statistics” hint that was executed by the current user. (There is actually an array of the past entries for which plan statistics have been gathered for the user).

Remember you may need to grant the particular user the right privileges in order to display the information that has been stored in the statistical table. Show all your work to execute the query and then display the statistics output. <https://www.oracle.com/technetwork/database/bi-datawarehousing/twp-bp-for-stats-gather-19c-5324205.pdf> )





1. Index null values:
   1. Create an index “*commi*” on field comm in the emp table. Set autotrace ON, and then run a query to find all the employees where comm is null. Include a screenshot of the explain plan and plan table output.

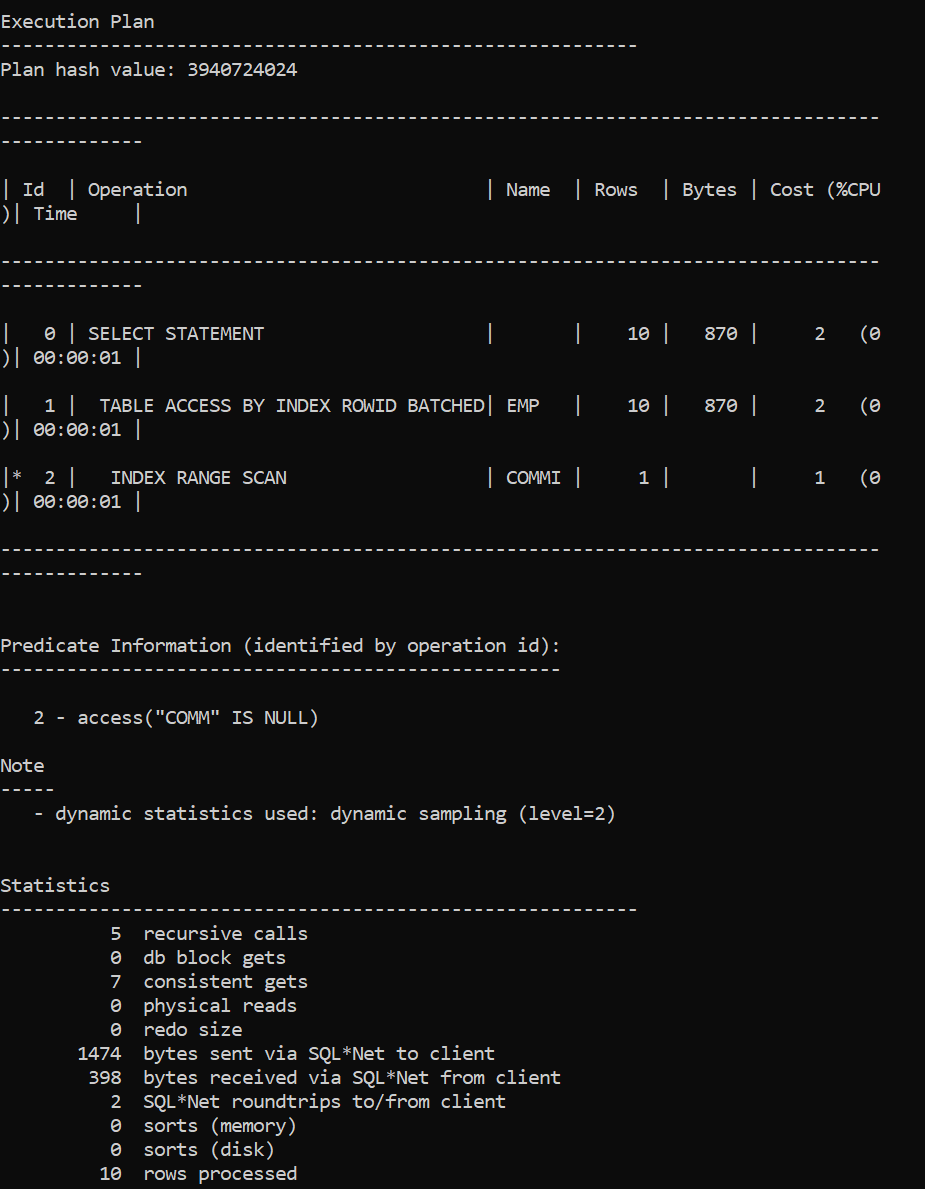


Has oracle used the index *commi* to answer the above query? Why?

Oracle scanned the entire table to fetch the results. The filter condition "COMM" IS NULL is applied directly on the table scan, not utilizing the index.

* 1. Drop the index *commi*, and then create another index on the same field with the same name but with a slightly different structure.

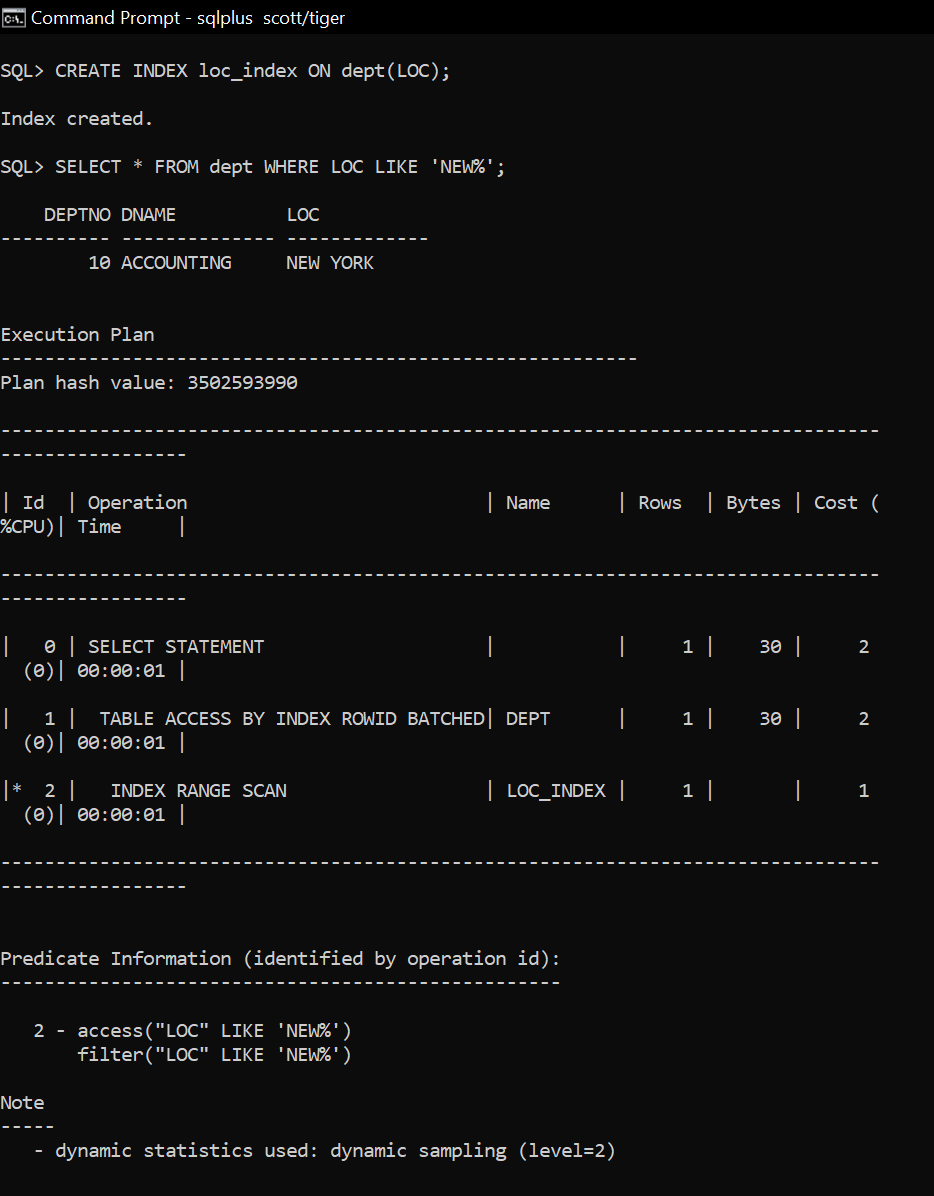
*Create index commi on emp(comm, 1);* Run the same query again as in part 2a. Include a screenshot of the explain plan and plan table output.



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

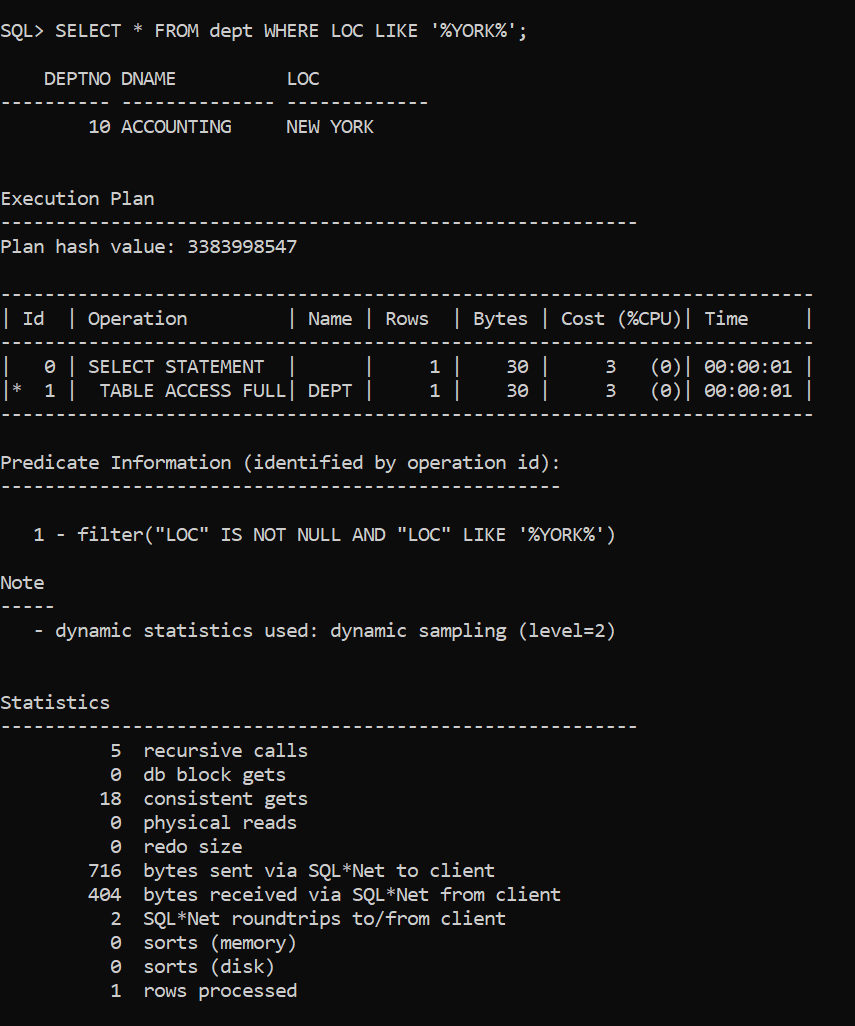
New modification of the index structure to include a constant value allowed Oracle to use the index for efficiently executing the query that filters on comm IS NULL.

1. Index wildcards %:
   1. 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?



Oracle used the index because the query condition LOC LIKE 'NEW%' efficiently matches rows using an index range scan, which is faster than scanning the entire table.

* 1. 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?



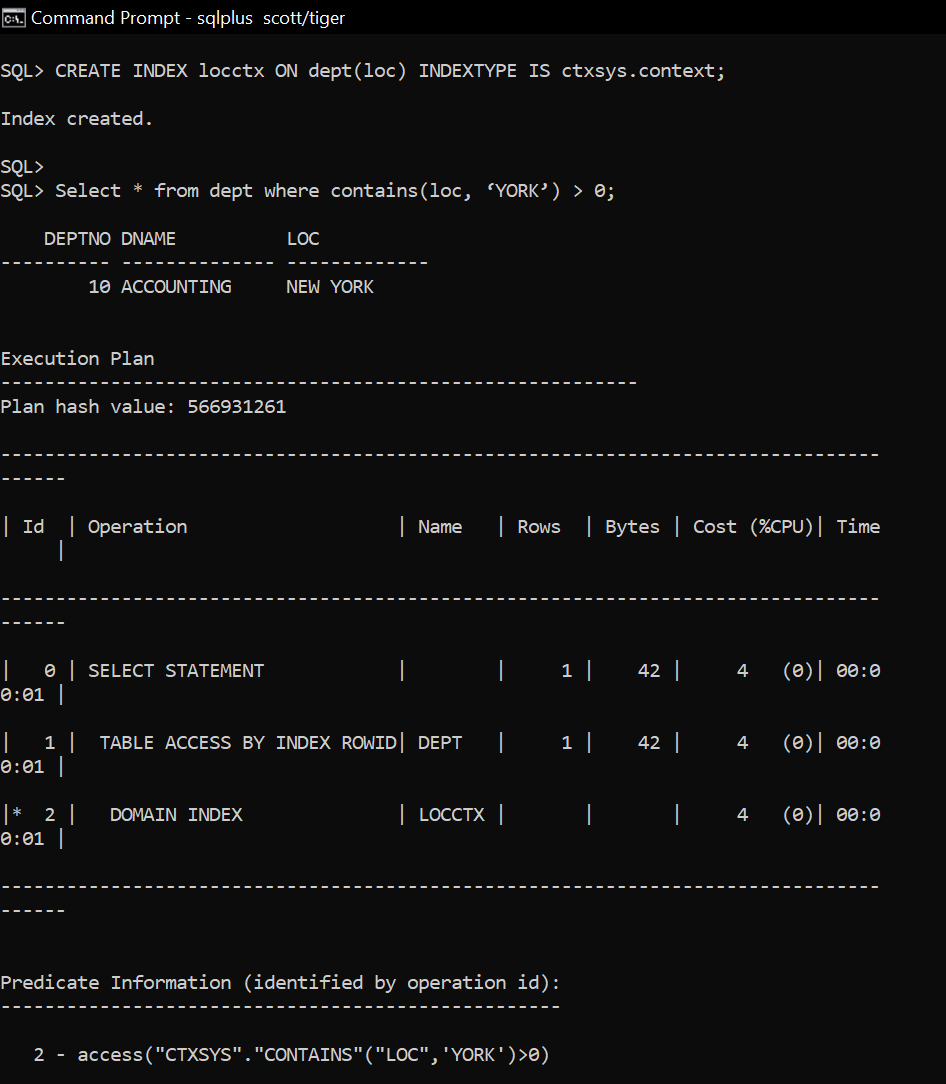
In this case, Oracle did not use the index to answer the query. The execution plan shows a "TABLE ACCESS FULL" operation for the DEPT table, indicating a full table scan was performed instead of using the index.

* 1. Create another index with different structure on the loc field.

*Create index locctx on dept(loc) indextype is ctxsys.context;*

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;*



Oracle has used the LOCCTX index for your query. This is indicated by the DOMAIN INDEX operation in the execution plan, specifically referencing the LOCCTX index. This operation shows that Oracle used the context index to optimize the search for rows where the LOC field contains the word "YORK".

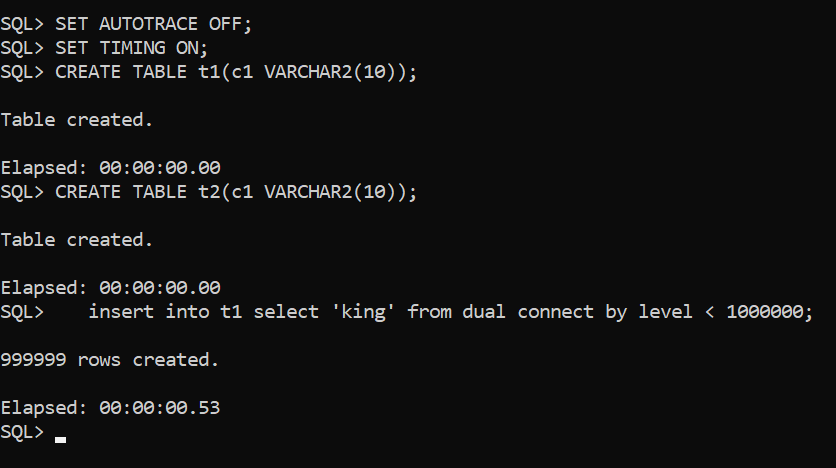
1. When Indexing is good?
   1. *Set autotrace off;* and then *Set timing on*;.   
      Then, create two tables t1 and t2 as follows:

*Create table t1(c1 varchar2(10));*

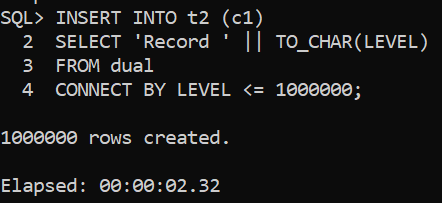
*Create table t2(c1 varchar2(10));*

* 1. Run the below query to insert 1 million records into the table t1

Take note of the time needed to insert into the table t1.

**

* 1. 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 time?



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

In a data warehouse, loading millions of records nightly is faster without indexes. After loading, indexes are created to optimize query performance during daily operations.

You're finished. Please submit.