# CST8276 Lab 7: Query Optimizer

**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 Lab7_*yourName*. Submit the file using BrightSpace, and demonstrate your work to your lab professor before the end of next week's lab session. The lab exercise is worth 2 marks towards your lab grade if correct and submitted on-time or early.**

## Activities:

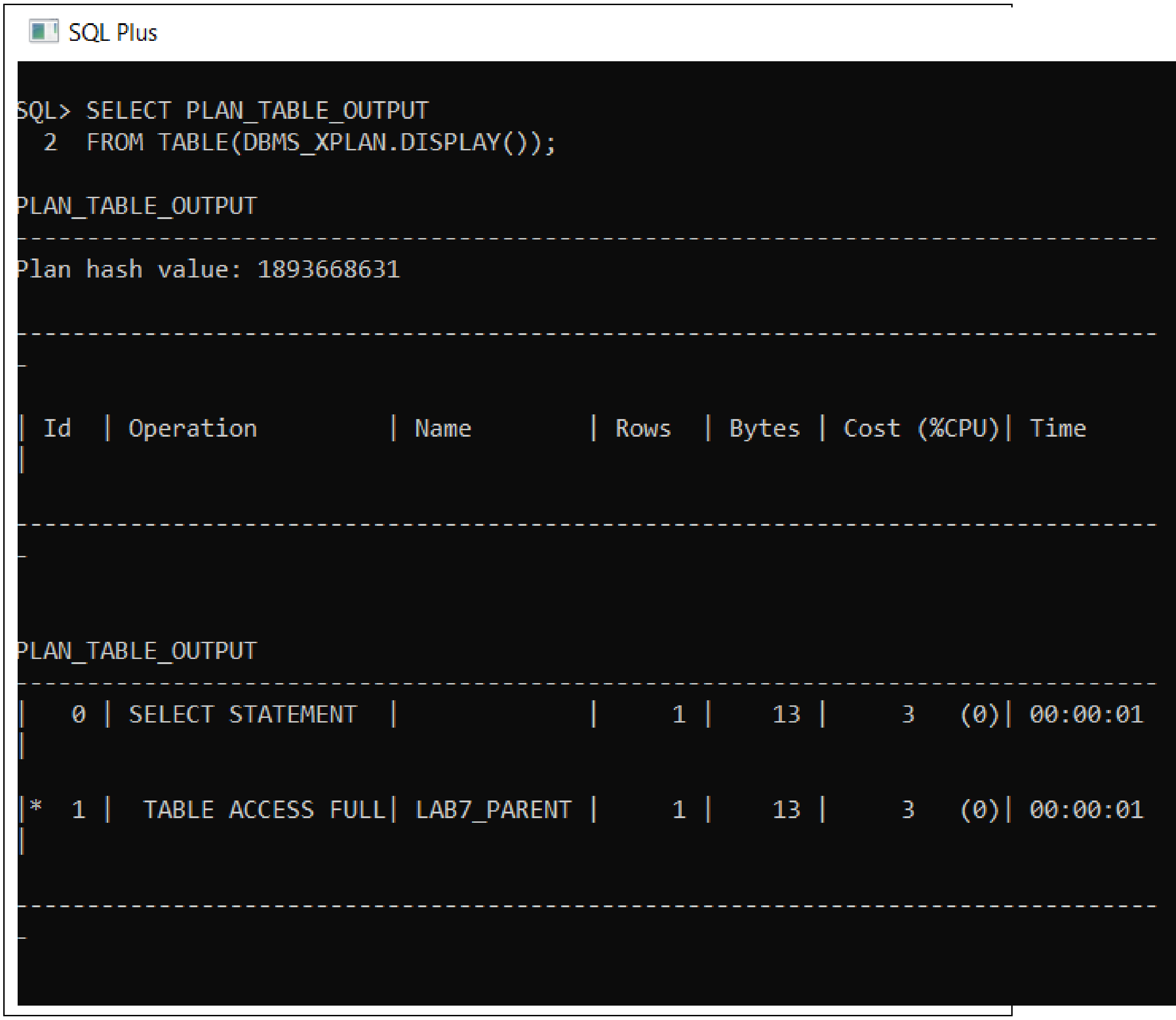
1. **Logon to your general user account and use the supplied script to create a sequence, create two tables, and to populate both tables with several rows of data. (Note: neither table has a primary key or an index). Think of the *lab7_parent* table as a strong entity (e.g., *Employees*) and the *lab7child* table as a weak entity (e.g., *Dependents* of the employee). You may recall a weak entity inherits its parent's primary key and adds this inherited key to a local partial key to create a composite primary key. In our situation the *lab7_child* table will eventually (i.e., several steps later) have a composite primary key of (ee#, fname). Note: If you are unable to create a sequence then from SYS, *grant create sequence to* your general user.**
    a. **From your general user account:**
        *i.* **Enter:**

        ```
        EXPLAIN PLAN FOR
        Select ee#
        From lab7_parent
        Where ee# = 123456;
        ```

        **Enter:**

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

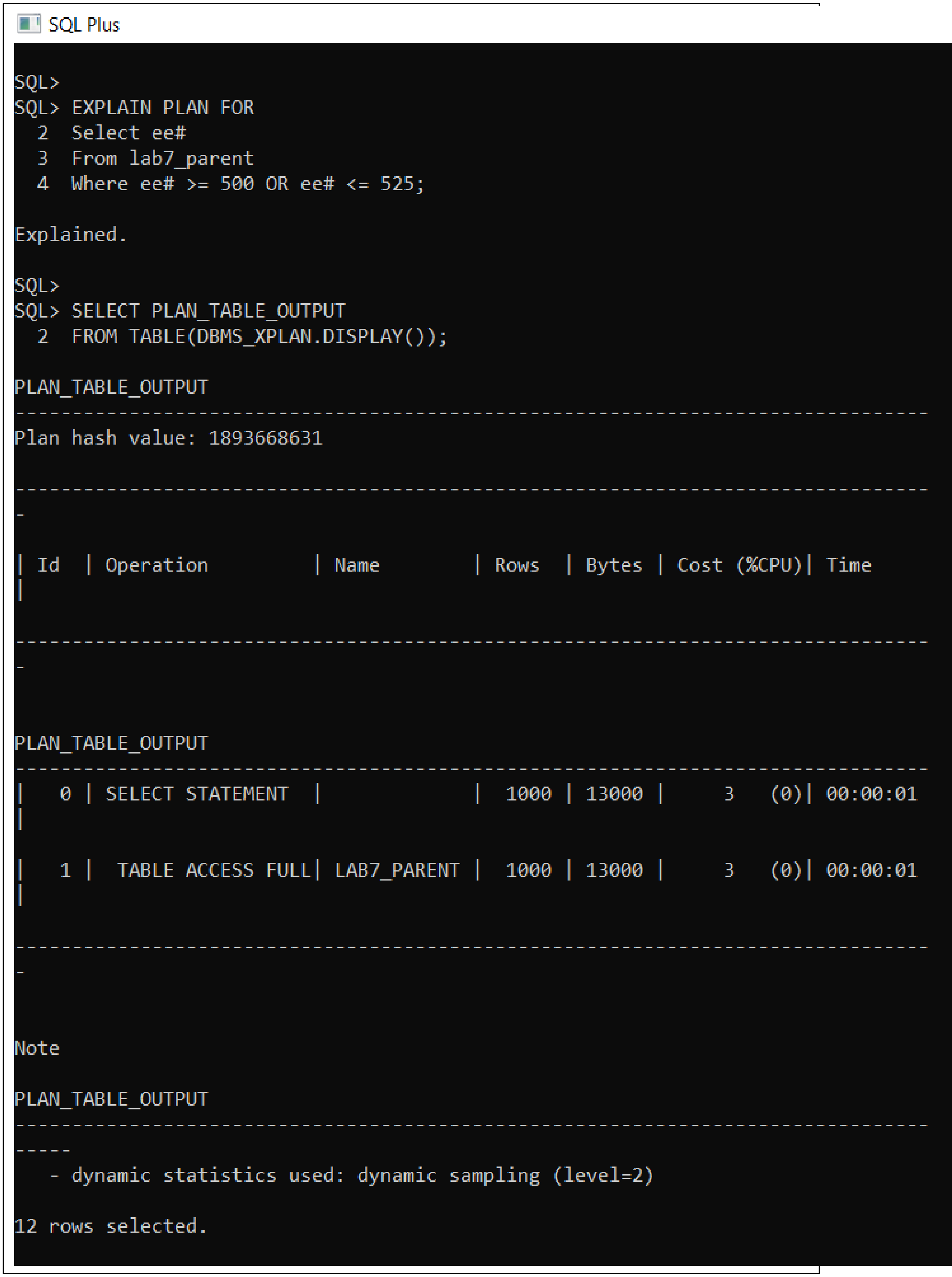
        ii. **Paste a screen snapshot of the Explain Plan query and the Plan_Table_Output into the area below .**



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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1893668631


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

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


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



PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |    13 |     3   (0)| 00:00:01
|

|*  1 |  TABLE ACCESS FULL| LAB7_PARENT |     1 |    13 |     3   (0)| 00:00:01
|


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

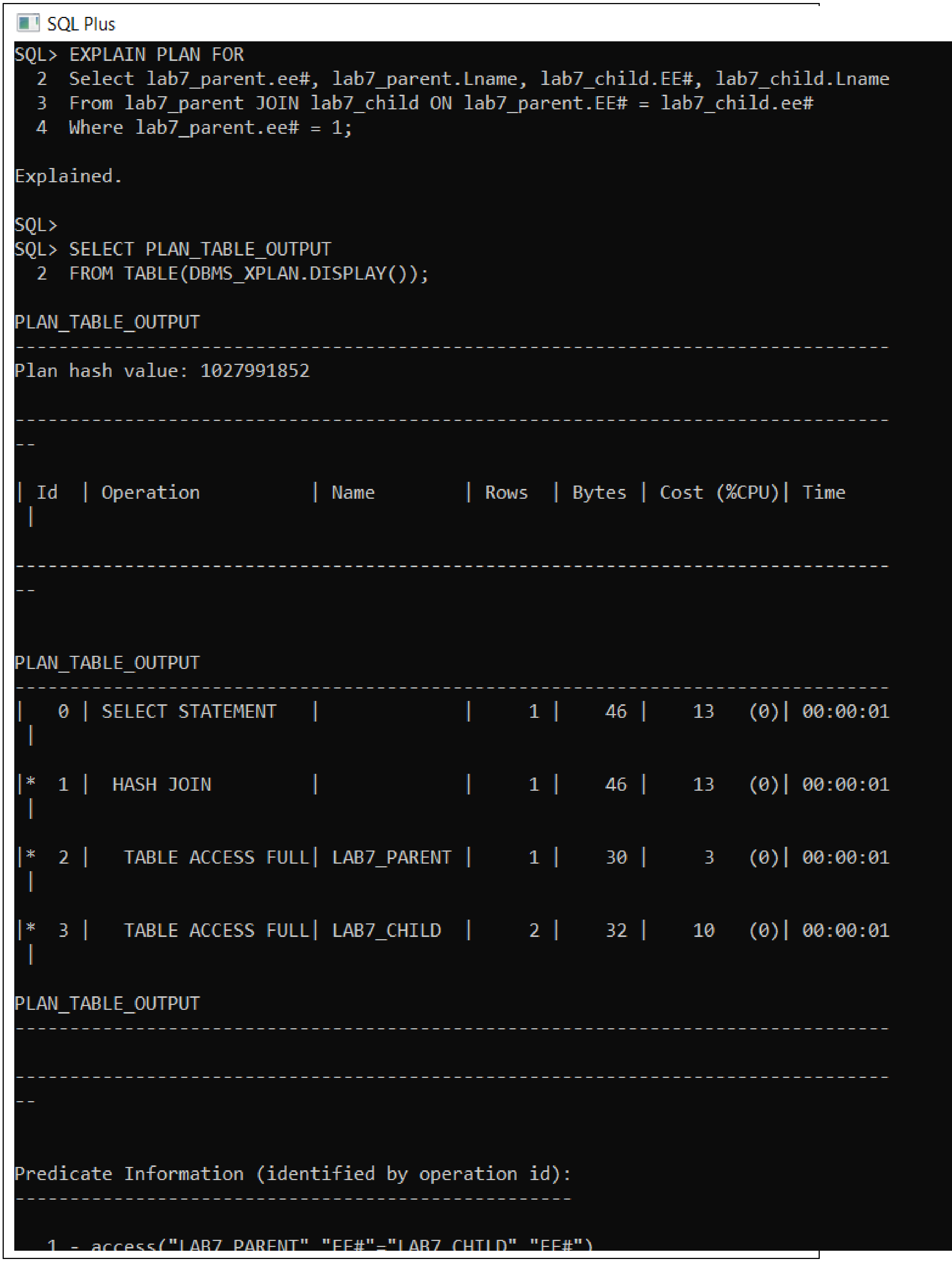
**b.** **Change the WHERE clause in *1ai* so that the query returns a range of approximately 25 ee#s. Copy your Explain Plan and Plan_Table_Output to your submission document.(below)**

# CST8276 Lab 7: Query Optimizer



```
SQL>
SQL> EXPLAIN PLAN FOR
  2  Select ee#
  3  From lab7_parent
  4  Where ee# >= 500 OR ee# <= 525;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1893668631

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |  1000 | 13000 |     3   (0)| 00:00:01
|

|   1 |  TABLE ACCESS FULL| LAB7_PARENT |  1000 | 13000 |     3   (0)| 00:00:01
|

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


Note

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----
   - dynamic statistics used: dynamic sampling (level=2)

12 rows selected.
```

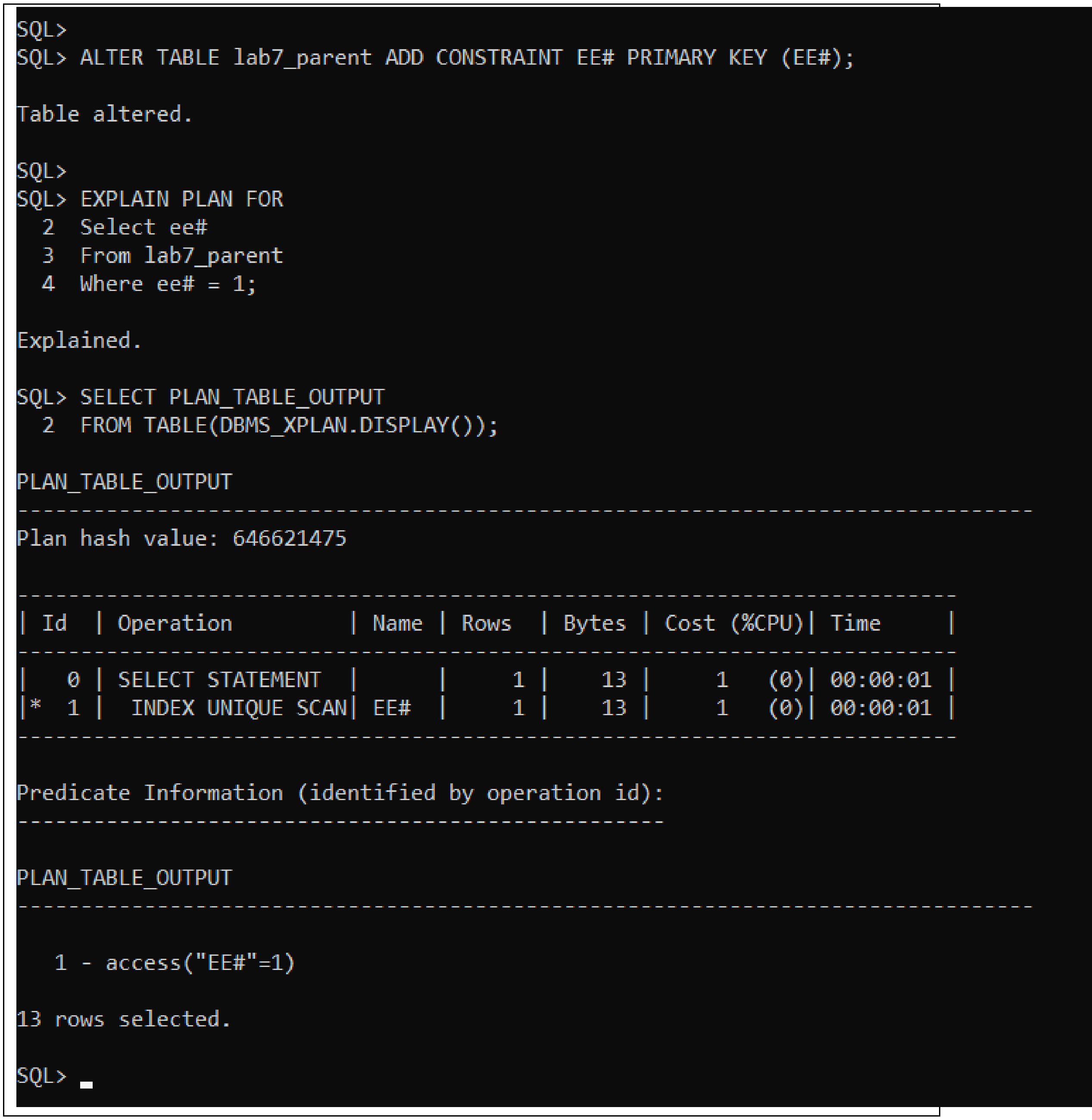
c. **Write an Explain Plan for a query that displays Parent.EE#, Parent.Lname, Child.EE# and Child.Lname. Take a screenshot of the Explain_plan and the Plan_Table_Output**



```
SQL> EXPLAIN PLAN FOR
  2  Select lab7_parent.ee#, lab7_parent.Lname, lab7_child.EE#, lab7_child.Lname
  3  From lab7_parent JOIN lab7_child ON lab7_parent.EE# = lab7_child.ee#
  4  Where lab7_parent.ee# = 1;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1027991852

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |     1 |    46 |    13   (0)| 00:00:01
 |

|*  1 |  HASH JOIN         |             |     1 |    46 |    13   (0)| 00:00:01
 |

|*  2 |   TABLE ACCESS FULL| LAB7_PARENT |     1 |    30 |     3   (0)| 00:00:01
 |

|*  3 |   TABLE ACCESS FULL| LAB7_CHILD  |     2 |    32 |    10   (0)| 00:00:01
 |

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


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


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


   1 - access("LAB7_PARENT"."EE#"="LAB7_CHILD"."EE#")
```

  d. **The main purpose of the preceding activities was to reinforce the idea that a certain type of retrieval – generally considered relatively slow - will be the default approach to retrieving data.**

2. **Add a primary key to the parent table (ee#) and the child table (ee#, fname). Rerun *1ai*. If you obtain the same result as *1ai*, then connect to sys as sysdba and enter: *ALTER SYSTEM FLUSH SHARED_POOL;* and bounce the database (shutdown, startup), reconnect to your general user and rerun the EXPLAIN PLAN query.**
   a. **Your Plan_Table_Output should now be significantly different from the non-indexed version. Take a screenshot of the Explain Plan query and the Plan_Table _Output and add it to your submission document.**



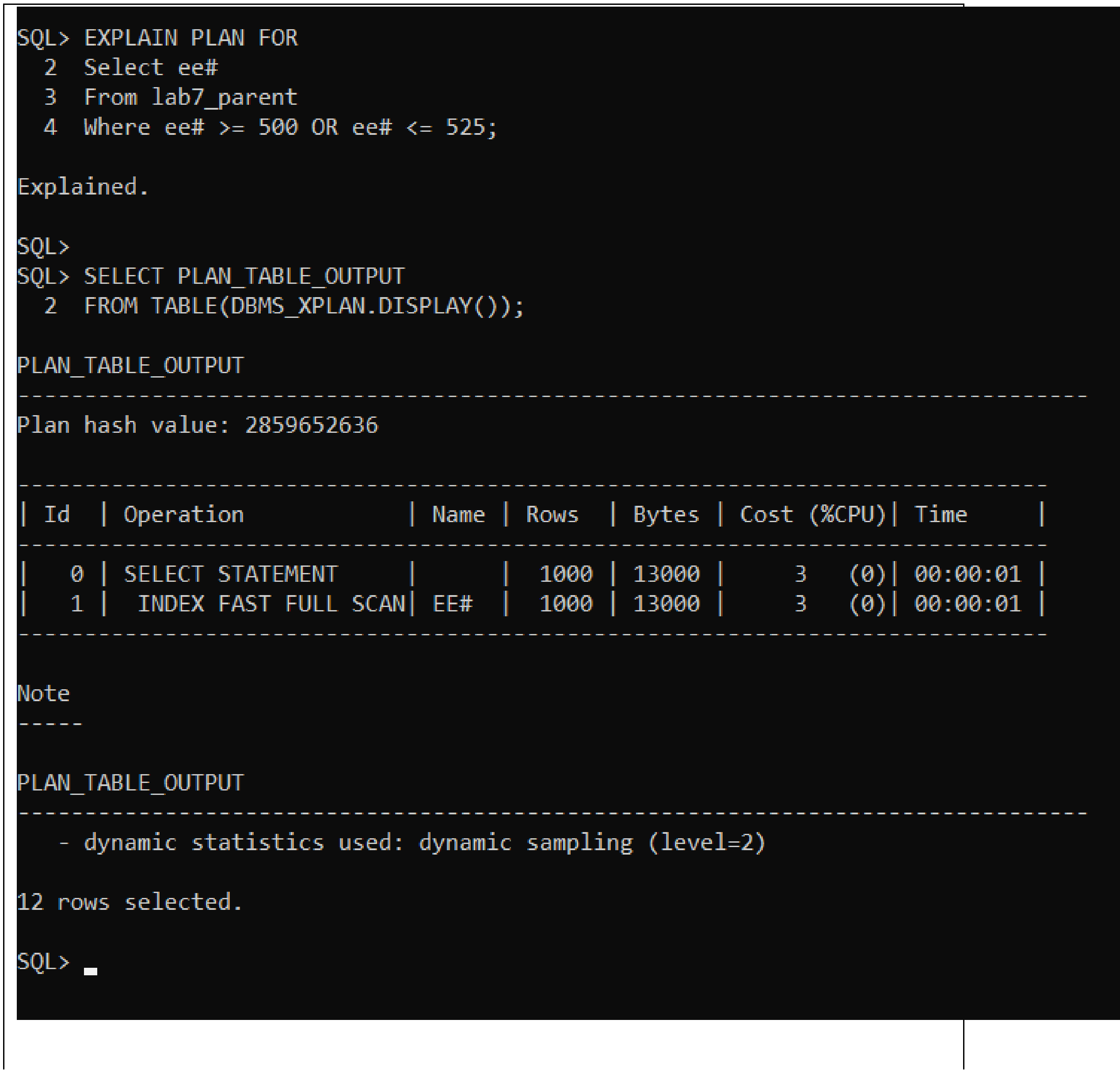
```
SQL>
SQL> ALTER TABLE lab7_parent ADD CONSTRAINT EE# PRIMARY KEY (EE#);

Table altered.

SQL>
SQL> EXPLAIN PLAN FOR
  2  Select ee#
  3  From lab7_parent
  4  Where ee# = 1;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 646621475

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |    13 |     1   (0)| 00:00:01 |
|*  1 |  INDEX UNIQUE SCAN| EE#  |     1 |    13 |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

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

   1 - access("EE#"=1)

13 rows selected.

SQL> _
```

**Explain how your actions in 2 changed the query plan (if your plan doesn't change, explain what you would expect to occur). _**

**Table access is not full anymore, there's an index instead. I suppose the row is accessed directly as it has a primary key now instead of searching all the rows and filtering them**

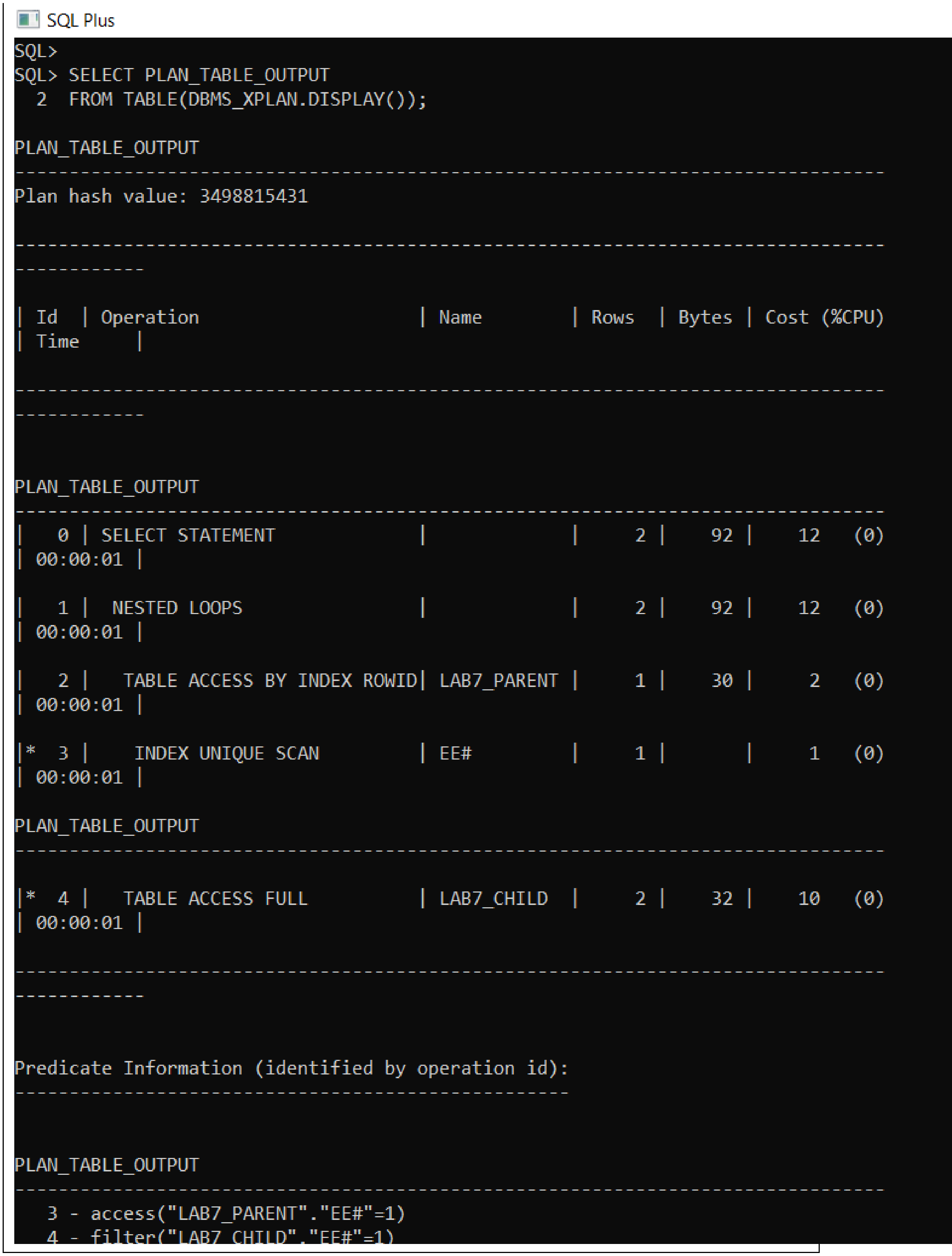
**b. Now that you have added primary keys to both tables, revisit each of the scenarios (i.e., a single ee#, a range of ee#s, a table join**

**on ee#) in Question 1. Include a screenshot of the explain plan and plan table output.**



```
SQL> EXPLAIN PLAN FOR
  2  Select ee#
  3  From lab7_parent
  4  Where ee# >= 500 OR ee# <= 525;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2859652636

-----------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |  1000 | 13000 |     3   (0)| 00:00:01 |
|   1 |  INDEX FAST FULL SCAN| EE#  |  1000 | 13000 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
   - dynamic statistics used: dynamic sampling (level=2)

12 rows selected.

SQL> _
```

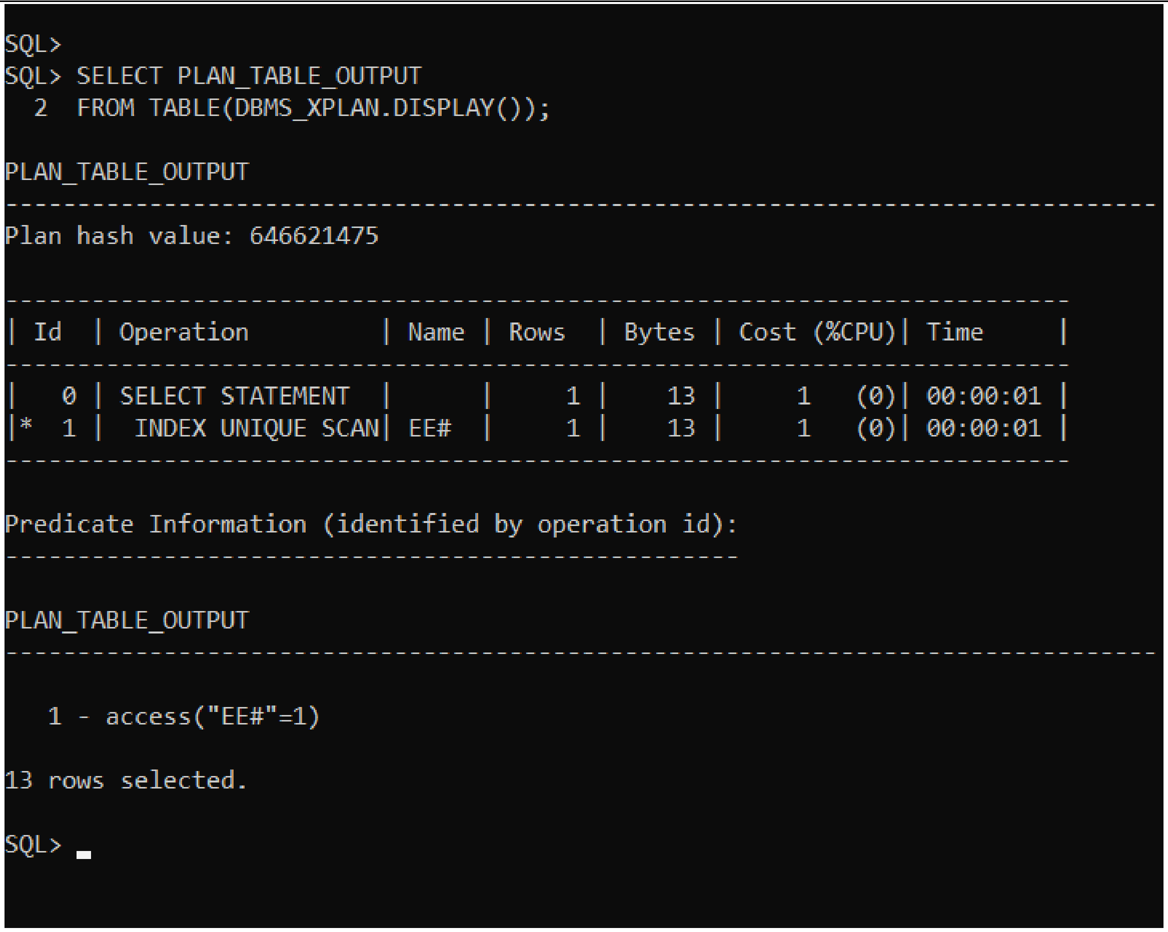
# CST8276 Lab 7: Query Optimizer



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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3498815431

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     2 |    92 |    12   (0)
| 00:00:01 |

|   1 |  NESTED LOOPS                |             |     2 |    92 |    12   (0)
| 00:00:01 |

|   2 |   TABLE ACCESS BY INDEX ROWID| LAB7_PARENT |     1 |    30 |     2   (0)
| 00:00:01 |

|*  3 |    INDEX UNIQUE SCAN         | EE#         |     1 |       |     1   (0)
| 00:00:01 |

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

|*  4 |   TABLE ACCESS FULL          | LAB7_CHILD  |     2 |    32 |    10   (0)
| 00:00:01 |

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


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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
   3 - access("LAB7_PARENT"."EE#"=1)
   4 - filter("LAB7_CHILD"."EE#"=1)
```

3. **Add a foreign key to the child table referencing the parent table. Rerun the table join case. If your results did not change from the pre-foreign key table join results then, from SYS, bounce the database and retry. Include a screenshot of the explain plan and the resulting plan table output. IF there is still no difference, try changing the where clause to include multiple criteria.**

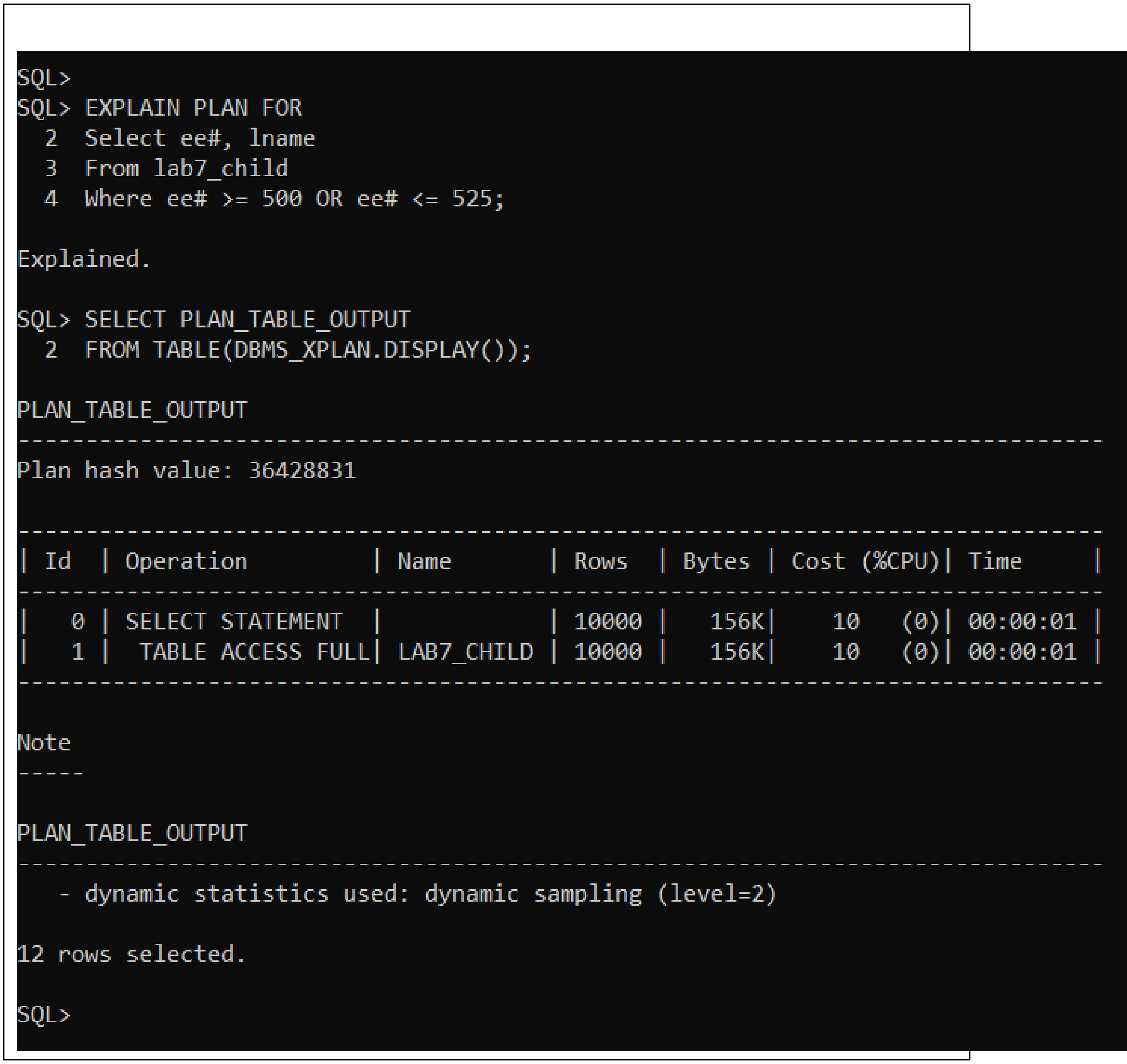
   **Describe what you expected to happen (and why) and show your actual results below.**



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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 646621475

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |    13 |     1   (0)| 00:00:01 |
|*  1 |  INDEX UNIQUE SCAN| EE#  |     1 |    13 |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

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

   1 - access("EE#"=1)

13 rows selected.

SQL>
```

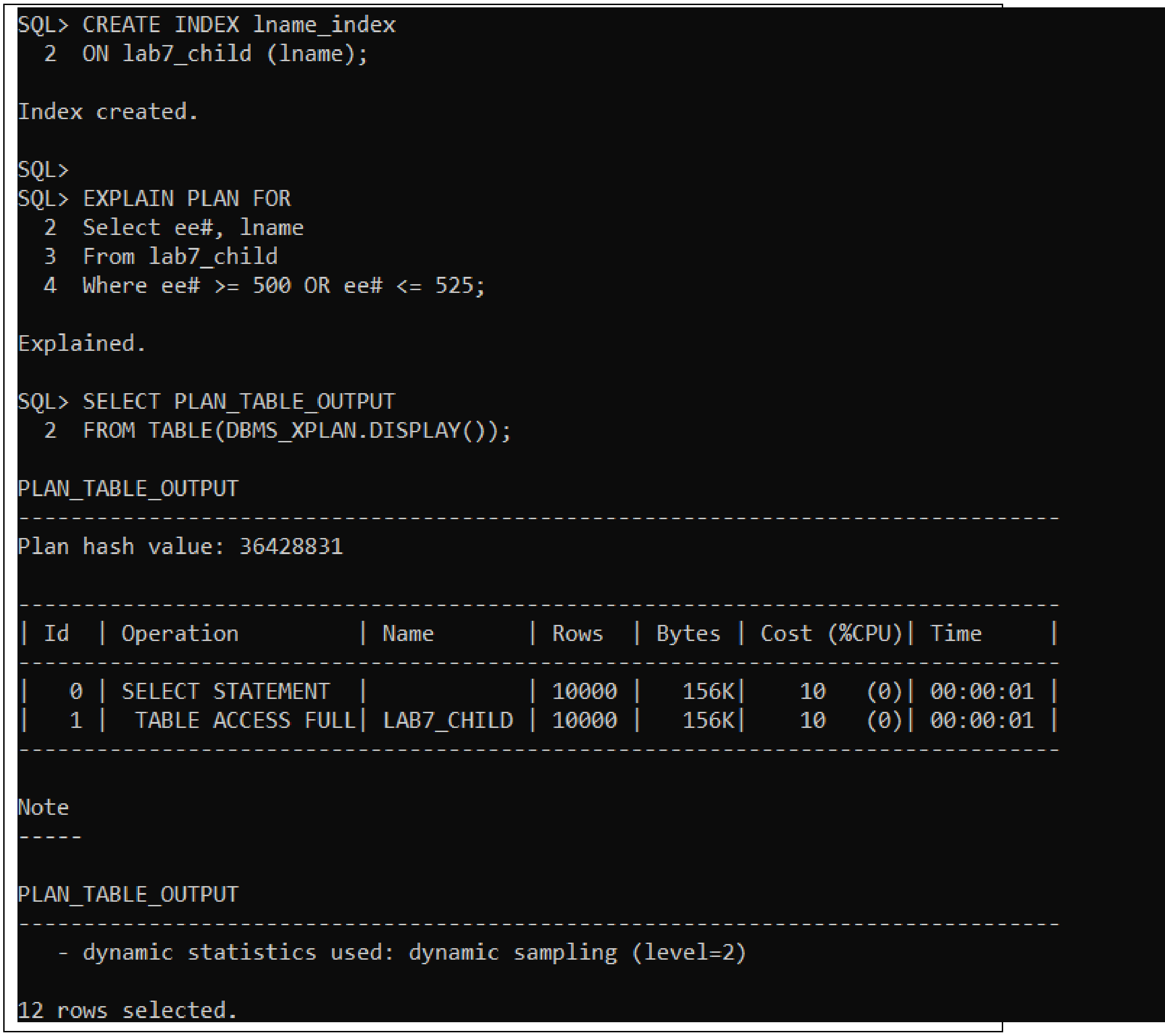
4. **Write an Explain Plan query that contains a child ee# range and a child lname range selection criteria.**

   a. **Copy the explain plan and plan table output.**



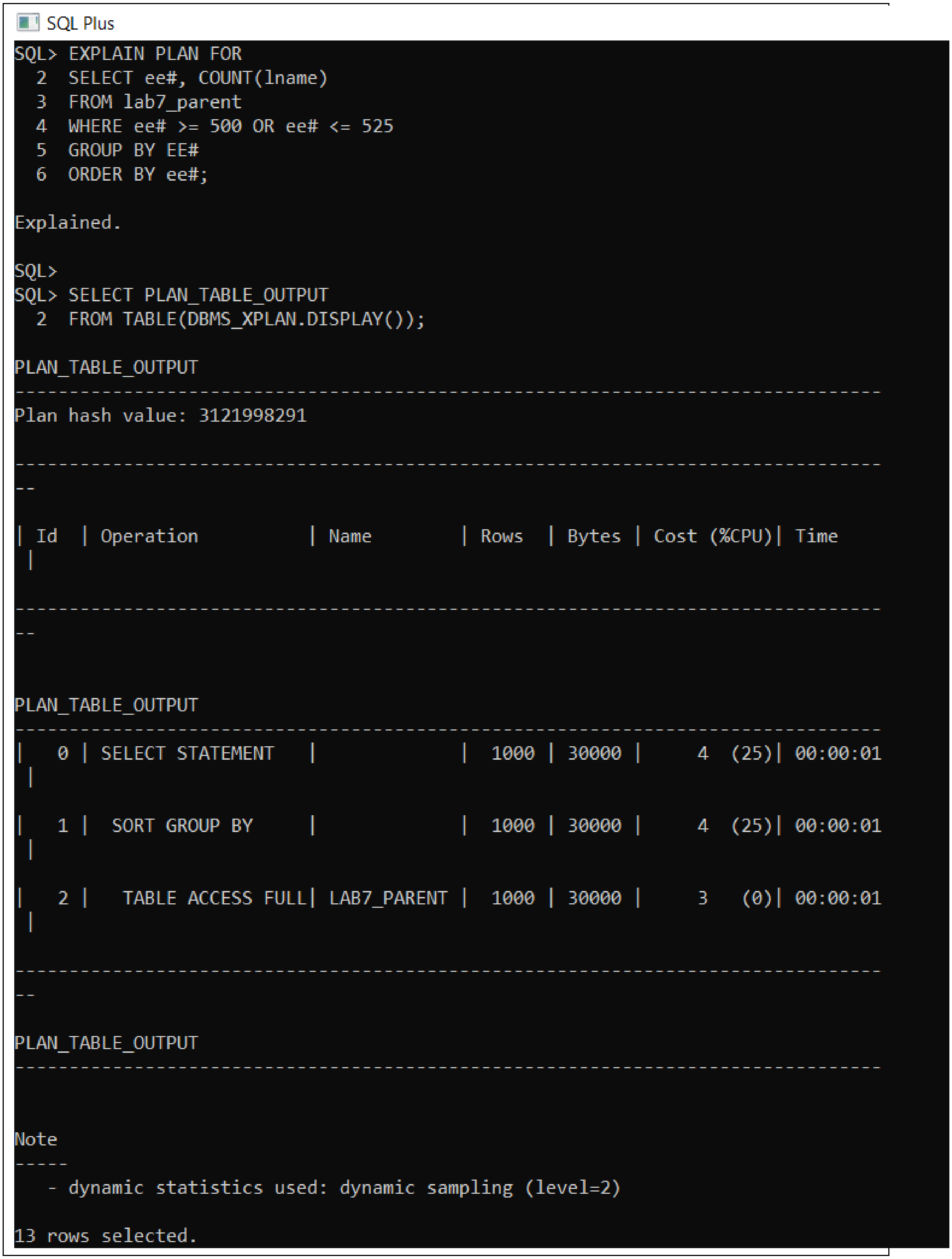
```
SQL>
SQL> EXPLAIN PLAN FOR
  2  Select ee#, lname
  3  From lab7_child
  4  Where ee# >= 500 OR ee# <= 525;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 36428831

--------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            | 10000 |   156K|    10   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| LAB7_CHILD | 10000 |   156K|    10   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
   - dynamic statistics used: dynamic sampling (level=2)

12 rows selected.

SQL>
```

b. **Add an index to child.lname. Rerun the previous child ee# and child lname range query. Copy the explain plan and plan table output. Again, if you don't see any change in results, bounce the instance and rerun.**



```
SQL> CREATE INDEX lname_index
  2  ON lab7_child (lname);

Index created.

SQL>
SQL> EXPLAIN PLAN FOR
  2  Select ee#, lname
  3  From lab7_child
  4  Where ee# >= 500 OR ee# <= 525;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 36428831

--------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            | 10000 |   156K|    10   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| LAB7_CHILD | 10000 |   156K|    10   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
   - dynamic statistics used: dynamic sampling (level=2)

12 rows selected.
```

5. **Write an Explain Plan query that contains parent ee#, count(lname) an ee# range selection criteria and an ordering of the results by ee#. Copy the explain plan and plan table output.**



```
SQL> EXPLAIN PLAN FOR
  2  SELECT ee#, COUNT(lname)
  3  FROM lab7_parent
  4  WHERE ee# >= 500 OR ee# <= 525
  5  GROUP BY EE#
  6  ORDER BY ee#;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3121998291

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |  1000 | 30000 |     4  (25)| 00:00:01
 |

|   1 |  SORT GROUP BY     |             |  1000 | 30000 |     4  (25)| 00:00:01
 |

|   2 |   TABLE ACCESS FULL| LAB7_PARENT |  1000 | 30000 |     3   (0)| 00:00:01
 |

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

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


Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

13 rows selected.
```

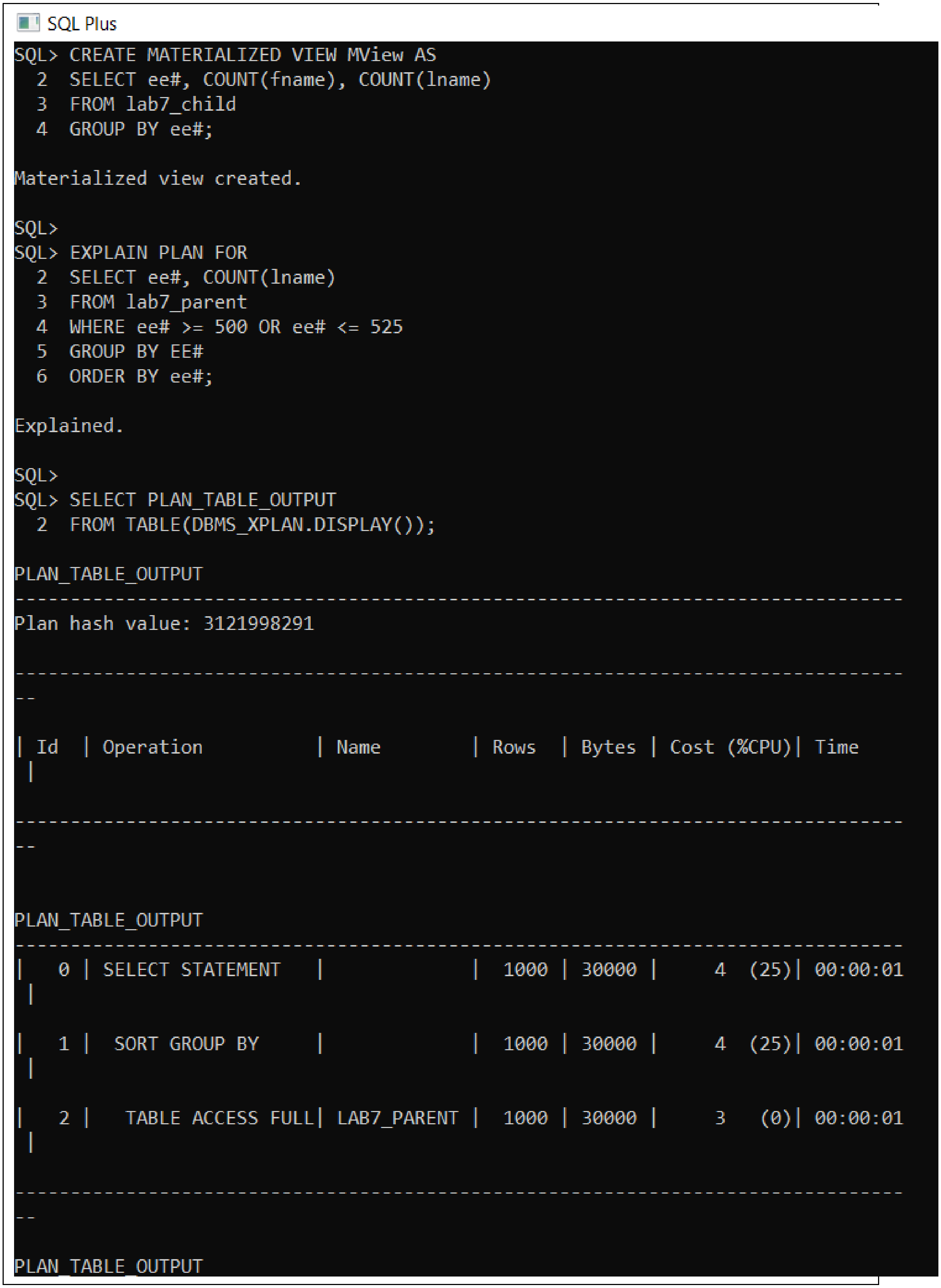
6. **Materialized views are very important performance enhancing objects in large databases and/or data warehouses that either need to join multiple tables or perform significant aggregation from remote instances. The Oracle Data Warehousing Guide, located here, http://docs.oracle.com/database/121/DWHSG/basicmv.htm#DWHSG008 , provides a good description of this feature.**

   **What does a materialized view contain that a regular view does not?**

   Normal views don't have a physical component whereas materialized views are stored on the disc

7. **Connect to SYS as SYSDBA and grant your generalized user the privilege of being able to *create materialized view*. Return to your general user and create a materialize view on parent ee#, count(fname) and count(lname). Re-run the query from Q5 using the newly created materialized view. Copy the explain plan and plan table output.**

# CST8276 Lab 7: Query Optimizer



```
SQL> CREATE MATERIALIZED VIEW MView AS
  2  SELECT ee#, COUNT(fname), COUNT(lname)
  3  FROM lab7_child
  4  GROUP BY ee#;

Materialized view created.

SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT ee#, COUNT(lname)
  3  FROM lab7_parent
  4  WHERE ee# >= 500 OR ee# <= 525
  5  GROUP BY EE#
  6  ORDER BY ee#;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3121998291

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |  1000 | 30000 |     4  (25)| 00:00:01
 |

|   1 |  SORT GROUP BY     |             |  1000 | 30000 |     4  (25)| 00:00:01
 |

|   2 |   TABLE ACCESS FULL| LAB7_PARENT |  1000 | 30000 |     3   (0)| 00:00:01
 |

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

PLAN_TABLE_OUTPUT
```

# CST8276 Lab 7: Query Optimizer

**You're finished. Please submit.**