Vendor: Oracle

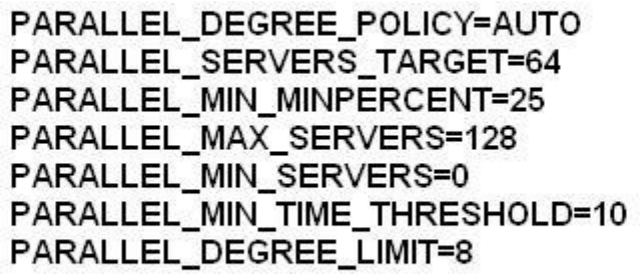
Exam Code: 1Z0-117

Exam Name: Oracle Database 11g Release 2: SQL Tuning Exam

Version: DEMO

QUESTION 1

You instance has these parameter settings:



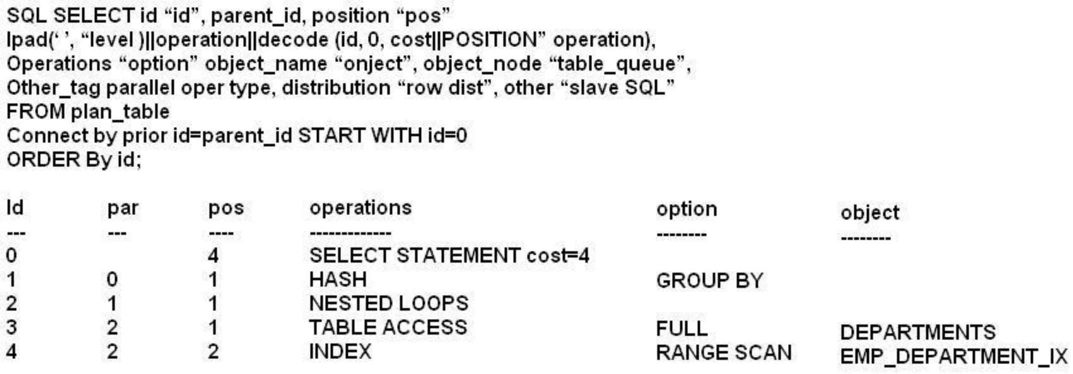
```
PARALLEL_DEGREE_POLICY=AUTO
PARALLEL_SERVERS_TARGET=64
PARALLEL_MIN_MINPERCENT=25
PARALLEL_MAX_SERVERS=128
PARALLEL_MIN_SERVERS=0
PARALLEL_MIN_TIME_THRESHOLD=10
PARALLEL_DEGREE_LIMIT=8
```

Which three statements are true about these settings if no hints are used in a SQL statement?

A. A statement estimated for more than 10 seconds always has its degree of parallelism computed automatically.
B. A statement with a computed degree of parallelism greater than 8 will be queued for a maximum of 10 seconds.
C. A statement that executes for more than 10 seconds always has its degree of parallelism computed automatically.
D. A statement with a computed degree of parallelism greater than 8 will raise an error.
E. A statement with any computed degree of parallelism will be queued if the number of busy parallel execution processes exceeds 64.
F. A statement with a computed degree of parallelism of 20 will be queued if the number of available parallel execution processes is less 5.

Answer: CEF

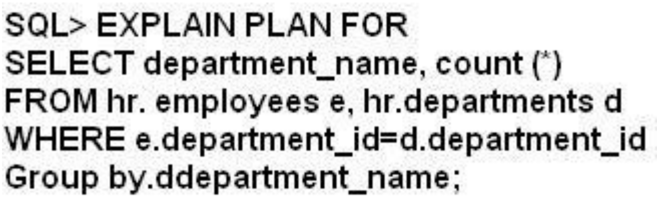
QUESTION 2



```
SQL SELECT id “id”, parent_id, position “pos”
lpad(‘ ’, “level )||operation||decode (id, 0, cost||POSITION” operation),
Operations “option” object_name “onject”, object_node “table_queue”,
Other_tag parallel oper type, distribution “row dist”, other “slave SQL”
FROM plan_table
Connect by prior id=parent_id START WITH id=0
ORDER By id;
```

```
Id      par     pos     operations                  option          object
---     ---     ----    -------------               --------        --------
0               4       SELECT STATEMENT cost=4
1       0       1       HASH                        GROUP BY
2       1       1       NESTED LOOPS
3       2       1       TABLE ACCESS                FULL            DEPARTMENTS
4       2       2       INDEX                       RANGE SCAN      EMP_DEPARTMENT_IX
```

Examine the following SQL statement:



```
SQL> EXPLAIN PLAN FOR
SELECT department_name, count (*)
FROM hr. employees e, hr.departments d
WHERE e.department_id=d.department_id
Group by.ddepartment_name;
```

Examine the exhibit to view the execution plan. Which statement is true about the execution plan?

A. The EXPLAIN PLAN generates the execution plan and stores it in c$SQL_PLAN after executing the query. Subsequent executions will use the same plan.
B. The EXPLAIN PLAN generates the execution plan and stores it in PLAN_TABLE without executing the query. Subsequent executions will always use the same plan.
C. The row with the ID 3 is the first step executed in the execution plan.
D. The row with the ID 0 is the first step executed in the execution plan.
E. The rows with the ID 3 and 4 are executed simultaneously.

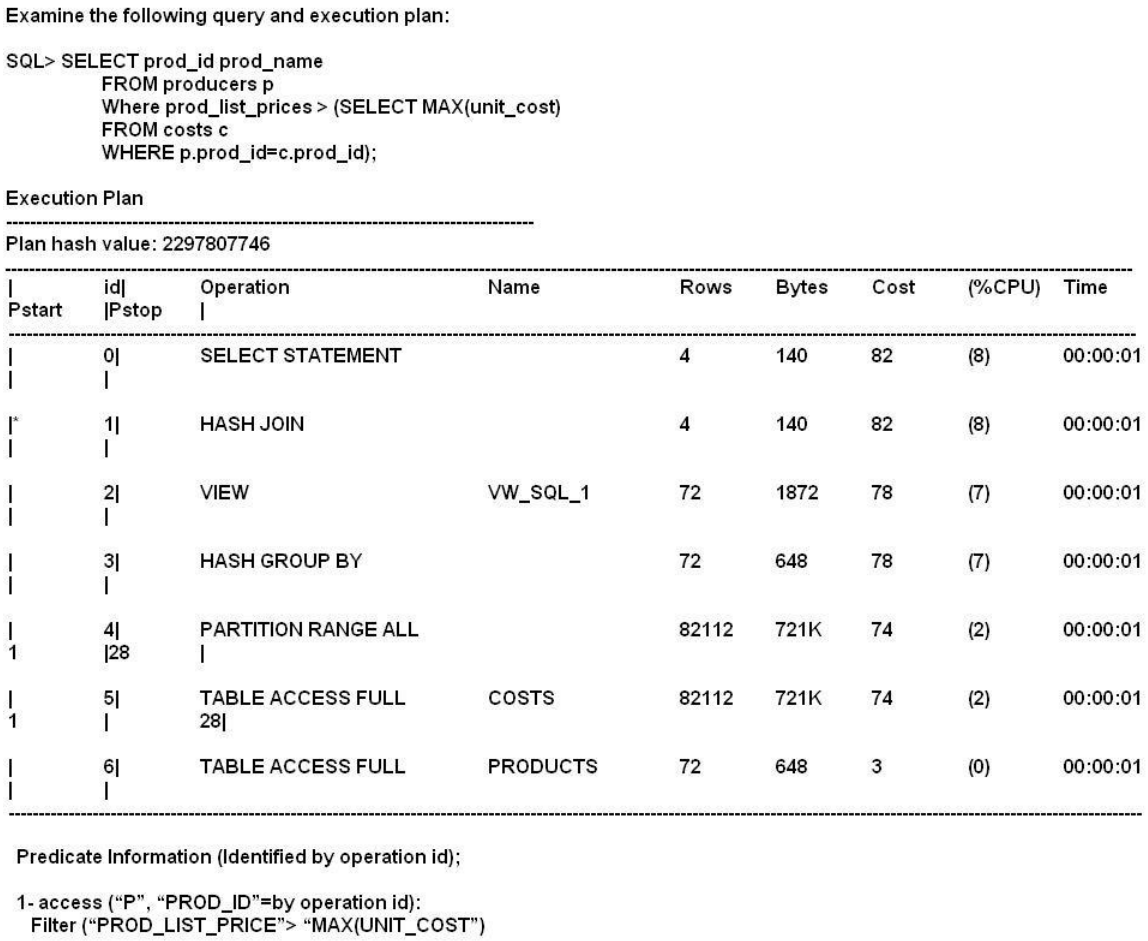
Answer: E
Explanation:
Note the other_tag parallel in the execution plan.
Within the Oracle plan_table, we see that Oracle keeps the parallelism in a column called other_tag. The other_tag column will tell you the type of parallel operation that is being performed within your query.
For parallel queries, it is important to display the contents of the other_tag in the execution.

QUESTION 3
Which two types of SQL statements will benefit from dynamic sampling?

A. SQL statements that are executed parallel
B. SQL statement that use a complex predicate expression when extended statistics are not available.
C. SQL statements that are resource-intensive and have the current statistics
D. SQL statements with highly selective filters on column that has missing index statistics
E. Short-running SQL statements

Answer: BD
Explanation:
B: One scenario where DS is used is when the statement contains a complex predicate expression and extended statistics are not available. Extended statistics were introduced in Oracle Database 11g Release 1 with the goal to help the optimizer get good quality cardinality estimates for complex predicate expressions.
D: DS It is typically used to compensate for missing or insufficient statistics that would otherwise lead to a very bad plan.
Note:
* Dynamic sampling (DS) was introduced in Oracle Database 9i Release 2 to improve the optimizer's ability to generate good execution plans.
* During the compilation of a SQL statement, the optimizer decides whether to use DS or not by considering whether the available statistics are sufficient to generate a good execution plan. If the available statistics are not enough, dynamic sampling will be used. It is typically used to compensate for missing or insufficient statistics that would otherwise lead to a very bad plan. It is typically used to compensate for missing or insufficient statistics that would otherwise lead to a very bad plan. For the case where one or more of the tables in the query does not have statistics, DS is used by the optimizer to gather basic statistics on these tables before optimizing the statement. The statistics gathered in this case are not as high a quality or as complete as the statistics gathered using the DBMS_STATS package. This trade off is made to limit the impact on the compile time of the statement.

QUESTION 4
Examine the following query and execution plan:



Examine the following query and execution plan:

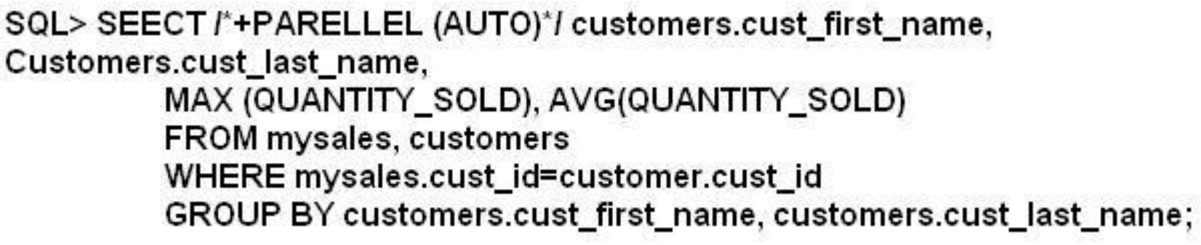
```
SQL> SELECT prod_id prod_name
          FROM producers p
          Where prod_list_prices > (SELECT MAX(unit_cost)
          FROM costs c
          WHERE p.prod_id=c.prod_id);

Execution Plan
-----------------------------------------------------------------------------
Plan hash value: 2297807746
--------------------------------------------------------------------------------------------------------------------------------------------------------
|        id|    Operation             Name         Rows     Bytes    Cost    (%CPU)  Time
Pstart   |Pstop  |
--------------------------------------------------------------------------------------------------------------------------------------------------------
|        0|     SELECT STATEMENT                     4        140      82      (8)     00:00:01
|         |

|*       1|     HASH JOIN                            4        140      82      (8)     00:00:01
|         |

|        2|     VIEW                  VW_SQL_1      72        1872     78      (7)     00:00:01
|         |

|        3|     HASH GROUP BY                       72        648      78      (7)     00:00:01
|         |

|        4|     PARTITION RANGE ALL                 82112     721K     74      (2)     00:00:01
1        |28     |

|        5|     TABLE ACCESS FULL     COSTS         82112     721K     74      (2)     00:00:01
1        |       28|

|        6|     TABLE ACCESS FULL     PRODUCTS      72        648      3       (0)     00:00:01
|         |
--------------------------------------------------------------------------------------------------------------------------------------------------------

 Predicate Information (Identified by operation id);

 1- access ("P", "PROD_ID"=by operation id):
   Filter ("PROD_LIST_PRICE"> "MAX(UNIT_COST")
```

Which query transformation technique is used in this scenario?

A. Join predicate push-down
B. Subquery factoring
C. Subquery unnesting
D. Join conversion

Answer: A
Explanation:
* Normally, a view cannot be joined with an index-based nested loop (i.e., index access) join, since a view, in contrast with a base table, does not have an index defined on it. A view can only be joined with other tables using three methods: hash, nested loop, and sort-merge joins.
* The following shows the types of views on which join predicate pushdown is currently supported.
UNION ALL/UNION view
Outer-joined view
Anti-joined view
Semi-joined view
DISTINCT view
GROUP-BY view

QUESTION 5
You enabled auto degree of parallelism (DOP) for your instance. Examine the query:

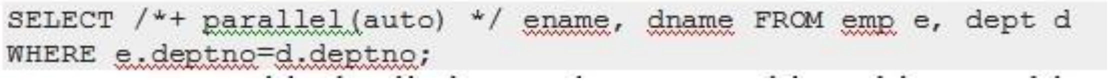


```
SQL> SEECT /*+PARELLEL (AUTO)*/ customers.cust_first_name,
Customers.cust_last_name,
              MAX (QUANTITY_SOLD), AVG(QUANTITY_SOLD)
              FROM mysales, customers
              WHERE mysales.cust_id=customer.cust_id
              GROUP BY customers.cust_first_name, customers.cust_last_name;
```

Which two are true about the execution of this query?

A. Dictionary DOP will be used, if present, on the tables referred in the query.
B. DOP is calculated if the calculated DOP is 1.
C. DOP is calculated automatically.
D. Calculated DOP will always by 2 or more.
E. The statement will execute with auto DOP only when PARALLEL_DEGREE_POLICY is set to AUTO.

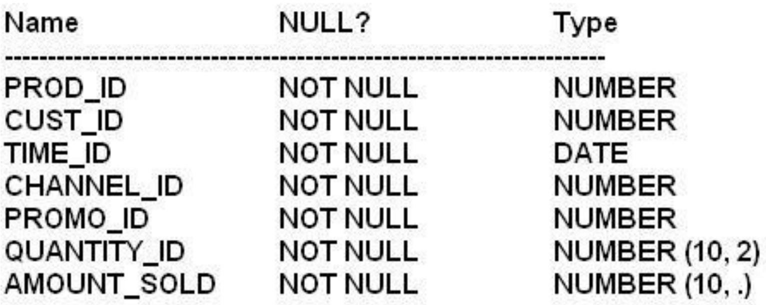
Answer: CD
Explanation:
C: * You can use the PARALLEL hint to force parallelism. It takes an optional parameter: the DOP at which the statement should run. In addition, theNO_PARALLEL hint overrides a PARALLEL parameter in the DDL that created or altered the table.
The following example illustrates computing the DOP the statement should use:



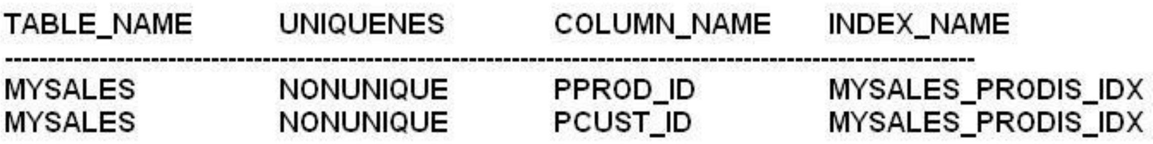
```
SELECT /*+ parallel(auto) */ ename, dname FROM emp e, dept d
WHERE e.deptno=d.deptno;
```

Not A: to override the dictionary dop, we could use hints at object level Not E: statement hints override the PARALLEL_DEGREE_POLICY.
Note:
* Automatic Parallel Degree Policy
When the parameter PARALLEL_DEGREE_POLICY is set to AUTO, Oracle Database automatically decides if a statement should execute in parallel or not and what DOP it should use. Oracle Database also determines if the statement can be executed immediately or if it is queued until more system resources are available. Finally, Oracle Database decides if the statement can take advantage of the aggregated cluster memory or not.
The following is a summary of parallel statement processing when parallel degree policy is set to automatic.
1. A SQL statement is issued.
2. The statement is parsed and the optimizer determines the execution plan.
3. The threshold limit specified by the PARALLEL_MIN_TIME_THRESHOLD initialization parameter is checked.
a. If the execution time is less than the threshold limit, the SQL statement is run serially.
b. If the execution time is greater than the threshold limit, the statement is run in parallel based on the DOP that the optimizer calculates.

QUESTION 6
View Exhibit1 and examine the structure and indexes for the MYSALES table.

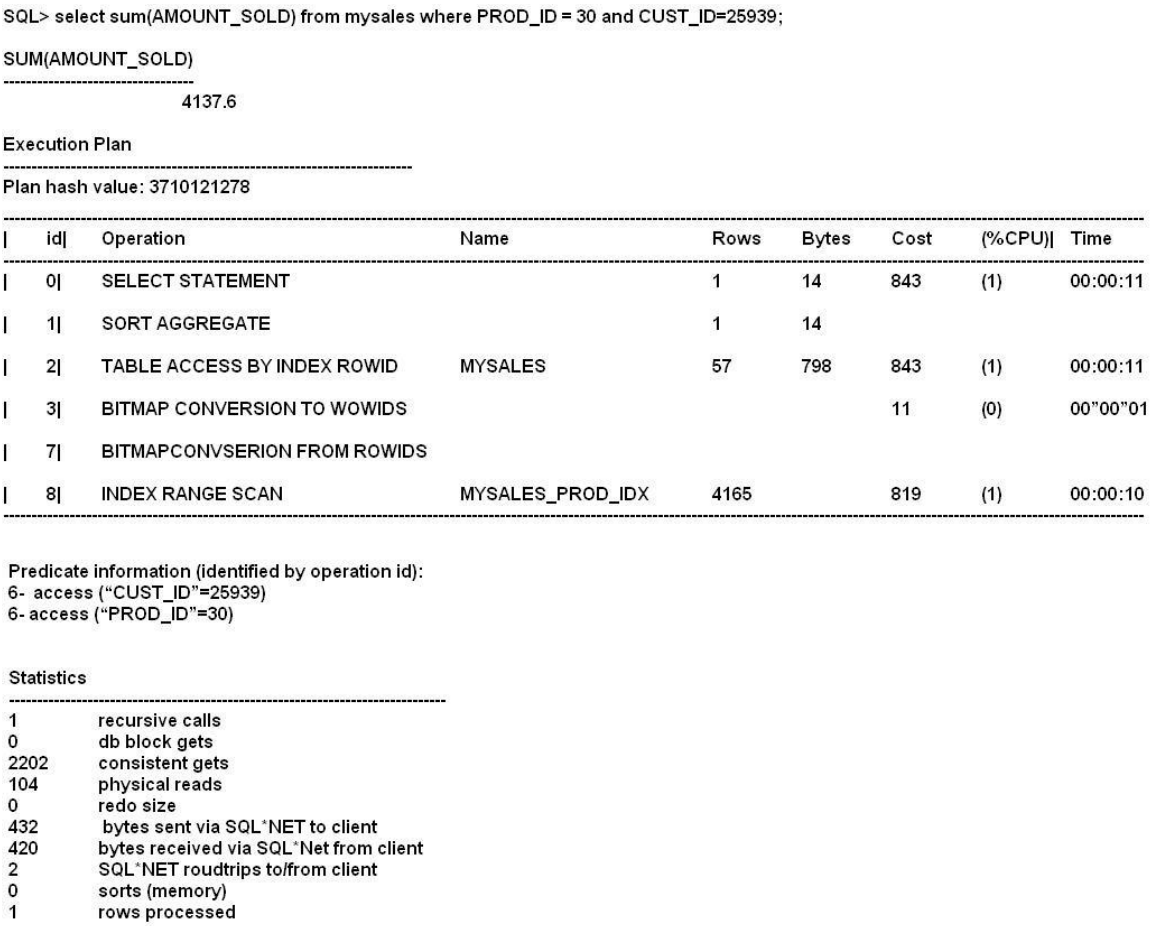


```
Name              NULL?             Type
----------------------------------------------------
PROD_ID           NOT NULL          NUMBER
CUST_ID           NOT NULL          NUMBER
TIME_ID           NOT NULL          DATE
CHANNEL_ID        NOT NULL          NUMBER
PROMO_ID          NOT NULL          NUMBER
QUANTITY_ID       NOT NULL          NUMBER (10, 2)
AMOUNT_SOLD       NOT NULL          NUMBER (10, .)
```



```
TABLE_NAME        UNIQUENES         COLUMN_NAME       INDEX_NAME
---------------------------------------------------------------------------------
MYSALES           NONUNIQUE         PPROD_ID          MYSALES_PRODIS_IDX
MYSALES           NONUNIQUE         PCUST_ID          MYSALES_PRODIS_IDX
```

The application uses the MYSALES table to insert sales record. But this table is also extensively used for generating sales reports. The PROD_ID and CUST_ID columns are frequently used in the WHERE clause of the queries. These columns have few distinct values relative to the total number of rows in the table. The MYSALES table has 4.5 million rows. View exhibit 2 and examine one of the queries and its autotrace output.



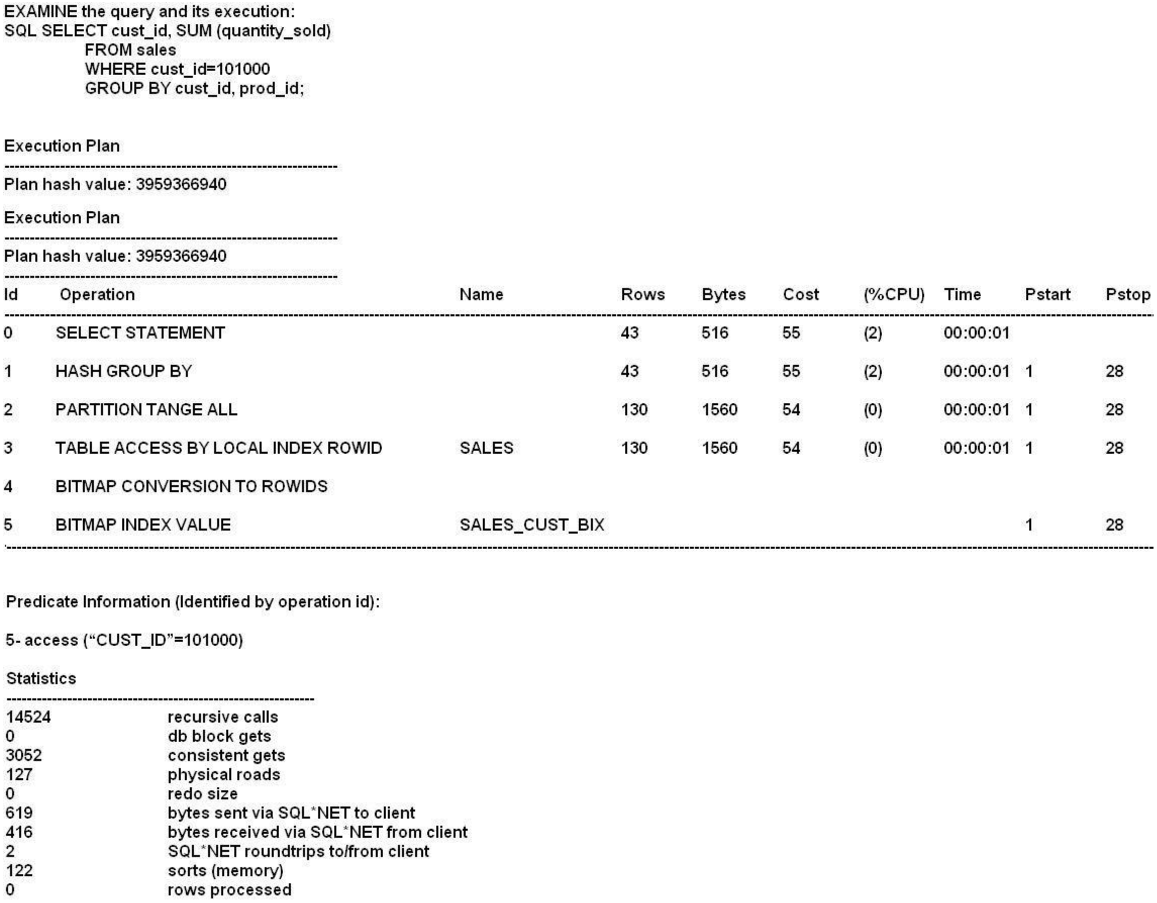
```
SQL> select sum(AMOUNT_SOLD) from mysales where PROD_ID = 30 and CUST_ID=25939;

SUM(AMOUNT_SOLD)
--------------------------------
                          4137.6

Execution Plan
----------------------------------------------------------------------
Plan hash value: 3710121278

------------------------------------------------------------------------------------------------------------------------------------
|   id|   Operation                        Name                  Rows    Bytes   Cost    (%CPU)|   Time
------------------------------------------------------------------------------------------------------------------------------------
|    0|   SELECT STATEMENT                                         1       14      843     (1)       00:00:11

|    1|   SORT AGGREGATE                                           1       14

|    2|   TABLE ACCESS BY INDEX ROWID      MYSALES               57      798     843     (1)       00:00:11

|    3|   BITMAP CONVERSION TO WOWIDS                                              11      (0)       00”00”01

|    7|   BITMAPCONVSERION FROM ROWIDS

|    8|   INDEX RANGE SCAN                 MYSALES_PROD_IDX      4165            819     (1)       00:00:10
------------------------------------------------------------------------------------------------------------------------------------

Predicate information (identified by operation id):
6-  access (“CUST_ID”=25939)
6- access (“PROD_ID”=30)


Statistics
----------------------------------------------------------------------
1              recursive calls
0              db block gets
2202           consistent gets
104            physical reads
0              redo size
432             bytes sent via SQL*NET to client
420            bytes received via SQL*Net from client
2              SQL*NET roudtrips to/from client
0              sorts (memory)
1              rows processed
```

Which two methods can examine one of the queries and its autotrace output?

A. Drop the current standard balanced B* Tree indexes on the CUST_ID and PROD_ID columns and re-create as bitmapped indexes.
B. Use the INDEX_COMBINE hint in the query.
C. Create a composite index involving the CUST_ID and PROD_ID columns.
D. Rebuild the index to rearrange the index blocks to have more rows per block by decreasing the value for PCTFRE attribute.
E. Collect histogram statistics for the CUST_ID and PROD_ID columns.

Answer: BC
Explanation:
B: The INDEX hint explicitly chooses an index scan for the specified table. You can use the INDEX hint for domain, B-tree, bitmap, and bitmap join indexes. However, Oracle recommends using INDEX_COMBINE rather than INDEX for the combination of multiple indexes, because it is a more versatile hint.
C: Combining the CUST_ID and PROD_ID columns into an composite index would improve performance.

QUESTION 7
Examine the Exhibit to view the structure of an indexes for the SALES table.



```
EXAMINE the query and its execution:
SQL SELECT cust_id, SUM (quantity_sold)
          FROM sales
          WHERE cust_id=101000
          GROUP BY cust_id, prod_id;

Execution Plan
--------------------------------------------------------------
Plan hash value: 3959366940
Execution Plan
--------------------------------------------------------------
Plan hash value: 3959366940
--------------------------------------------------------------
Id   Operation                          Name            Rows   Bytes   Cost   (%CPU)  Time       Pstart  Pstop
----------------------------------------------------------------------------------------------------------------
0    SELECT STATEMENT                                   43     516     55     (2)     00:00:01
1    HASH GROUP BY                                      43     516     55     (2)     00:00:01   1       28
2    PARTITION TANGE ALL                                130    1560    54     (0)     00:00:01   1       28
3    TABLE ACCESS BY LOCAL INDEX ROWID  SALES           130    1560    54     (0)     00:00:01   1       28
4    BITMAP CONVERSION TO ROWIDS
5    BITMAP INDEX VALUE                 SALES_CUST_BIX                                           1       28
----------------------------------------------------------------------------------------------------------------

Predicate Information (Identified by operation id):

5- access ("CUST_ID"=101000)

Statistics
----------------------------------------------------------
14524             recursive calls
0                 db block gets
3052              consistent gets
127               physical roads
0                 redo size
619               bytes sent via SQL*NET to client
416               bytes received via SQL*NET from client
2                 SQL*NET roundtrips to/from client
122               sorts (memory)
0                 rows processed
```

The SALES table has 4594215 rows. The CUST_ID column has 2079 distinct values. What would you do to influence the optimizer for better selectivity?

A. Drop bitmap index and create balanced B*Tree index on the CUST_ID column.
B. Create a height-balanced histogram for the CUST_ID column.
C. Gather statistics for the indexes on the SALES table.
D. Use the ALL_ROWS hint in the query.

Answer: D
Explanation:
OPTIMIZER_MODE establishes the default behavior for choosing an optimization approach for the instance.
Values:
FIRST_ROWS_N - The optimizer uses a cost-based approach and optimizes with a goal of best response time to return the first n rows (where n = 1, 10, 100, 1000). FIRST_ROWS - The optimizer uses a mix of costs and heuristics to find a best plan for fast delivery of the first few rows.
ALL_ROWS - The optimizer uses a cost-based approach for all SQL statements in the session and optimizes with a goal of best throughput (minimum resource use to complete the entire statement).

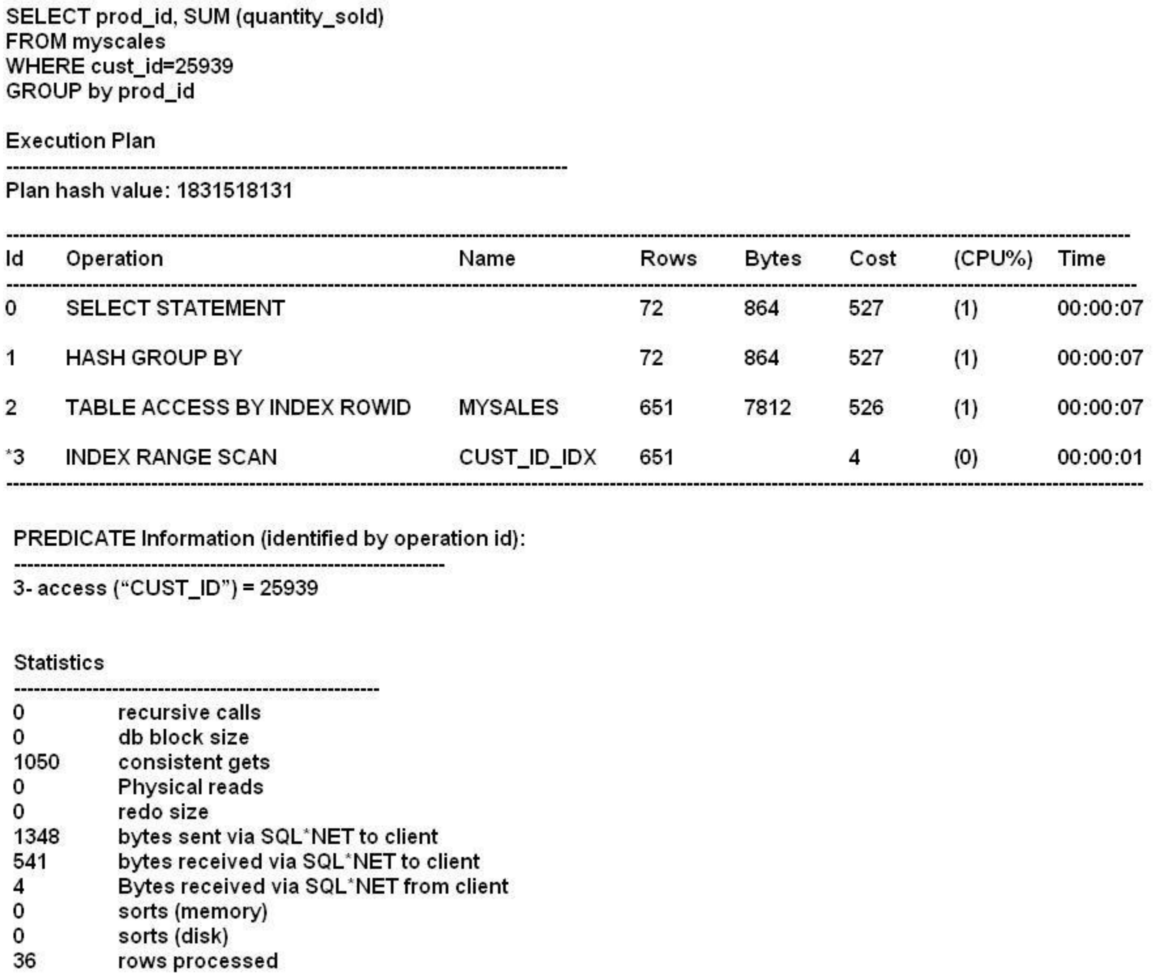
QUESTION 8
Which three are tasks performed in the hard parse stage of a SQL statement executions?

A. Semantics of the SQL statement are checked.
B. The library cache is checked to find whether an existing statement has the same hash value.
C. The syntax of the SQL statement is checked.
D. Information about location, size, and data type is defined, which is required to store fetched values in variables.
E. Locks are acquired on the required objects.

Answer: BDE
Explanation:
Parse operations fall into the following categories, depending on the type of statement submitted and the result of the hash check:
A) Hard parse
If Oracle Database cannot reuse existing code, then it must build a new executable version of the application code. This operation is known as a hard parse, or a library cache miss. The database always perform a hard parse of DDL.
During the hard parse, the database accesses the library cache and data dictionary cache numerous times to check the data dictionary. When the database accesses these areas, it uses a serialization device called a latch on required objects so that their definition does not change (see "Latches"). Latch contention increases statement execution time and decreases concurrency.
B) Soft parse
A soft parse is any parse that is not a hard parse. If the submitted statement is the same as a reusable SQL statement in the shared pool, then Oracle Database reuses the existing code. This reuse of code is also called a library cache hit.
Soft parses can vary in the amount of work they perform. For example, configuring the session cursor cache can sometimes reduce the amount of latching in the soft parses, making them "softer." In general, a soft parse is preferable to a hard parse because the database skips the optimization and row source generation steps, proceeding straight to execution.
Incorrect:
A, C: During the parse call, the database performs the following checks:
Syntax Check

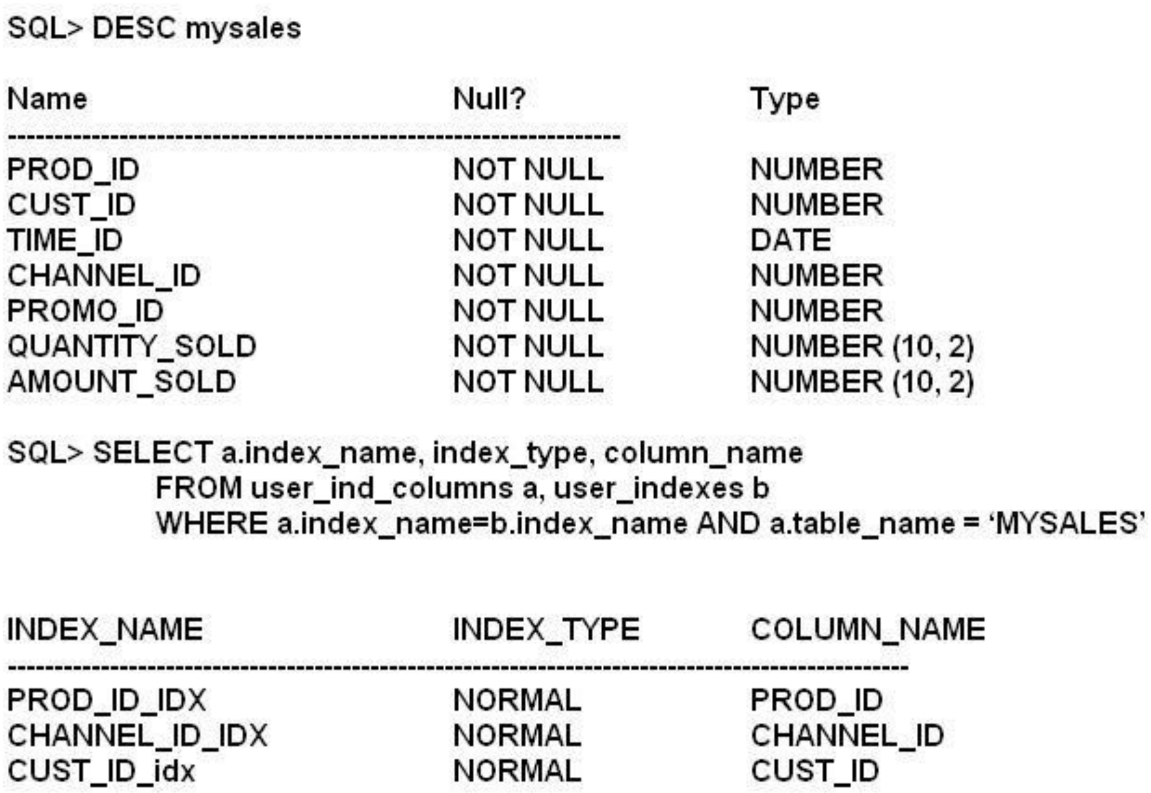
Semantic Check
Shared Pool Check
The hard parse is within Shared Pool check.
Reference: Oracle Database Concepts 11g, SQL Parsing

QUESTION 9
You are administering a database supporting an OLTP application. The application runs a series of extremely similar queries the MYSALES table where the value of CUST_ID changes. Examine Exhibit1 to view the query and its execution plan.



```
SELECT prod_id, SUM (quantity_sold)
FROM myscales
WHERE cust_id=25939
GROUP by prod_id

Execution Plan
--------------------------------------------------------------------------------
Plan hash value: 1831518131

------------------------------------------------------------------------------------------------------------------------------------------------------
Id   Operation                         Name           Rows    Bytes   Cost    (CPU%)   Time
------------------------------------------------------------------------------------------------------------------------------------------------------
0    SELECT STATEMENT                                 72      864     527     (1)      00:00:07

1    HASH GROUP BY                                    72      864     527     (1)      00:00:07

2    TABLE ACCESS BY INDEX ROWID       MYSALES        651     7812    526     (1)      00:00:07

*3   INDEX RANGE SCAN                  CUST_ID_IDX    651             4       (0)      00:00:01
------------------------------------------------------------------------------------------------------------------------------------------------------

 PREDICATE Information (identified by operation id):
 --------------------------------------------------------------
 3- access ("CUST_ID") = 25939


 Statistics
 -----------------------------------------------------
 0          recursive calls
 0          db block size
 1050       consistent gets
 0          Physical reads
 0          redo size
 1348       bytes sent via SQL*NET to client
 541        bytes received via SQL*NET to client
 4          Bytes received via SQL*NET from client
 0          sorts (memory)
 0          sorts (disk)
 36         rows processed
```

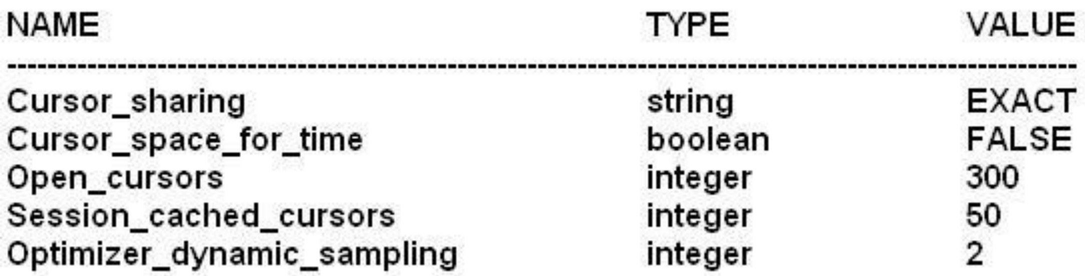
Examine Exhibit 2 to view the structure and indexes for the MYSALES table. The MYSALES table has 4 million records.



```
SQL> DESC mysales

Name                    Null?           Type
------------------------------------------------
PROD_ID                 NOT NULL        NUMBER
CUST_ID                 NOT NULL        NUMBER
TIME_ID                 NOT NULL        DATE
CHANNEL_ID              NOT NULL        NUMBER
PROMO_ID                NOT NULL        NUMBER
QUANTITY_SOLD           NOT NULL        NUMBER (10, 2)
AMOUNT_SOLD             NOT NULL        NUMBER (10, 2)

SQL> SELECT a.index_name, index_type, column_name
        FROM user_ind_columns a, user_indexes b
        WHERE a.index_name=b.index_name AND a.table_name = 'MYSALES'


INDEX_NAME              INDEX_TYPE      COLUMN_NAME
-----------------------------------------------------------------
PROD_ID_IDX             NORMAL          PROD_ID
CHANNEL_ID_IDX          NORMAL          CHANNEL_ID
CUST_ID_idx             NORMAL          CUST_ID
```

Data in the CUST_ID column is highly skewed.

Examine the parameters set for the instance:



```
NAME                                TYPE            VALUE
-----------------------------------------------------------------
Cursor_sharing                      string          EXACT
Cursor_space_for_time               boolean         FALSE
Open_cursors                        integer         300
Session_cached_cursors              integer         50
Optimizer_dynamic_sampling          integer         2
```

Which action would you like to make the query use the best plan for the selectivity?

A. Decrease the value of the OPTIMIZER_DYNAMIC_SAMPLING parameter to 0.
B. Us the /*+ INDEX(CUST_ID_IDX) */ hint in the query.
C. Drop the existing B* -tree index and re-create it as a bitmapped index on the CUST_ID column.
D. Collect histogram statistics for the CUST_ID column and use a bind variable instead of literal values.

Answer: D
Explanation:
Using Histograms
In some cases, the distribution of values within a column of a table will affect the optimizer's decision to use an index vs. perform a full-table scan. This scenario occurs when the value with a where clause has a disproportional amount of values, making a full-table scan cheaper than index

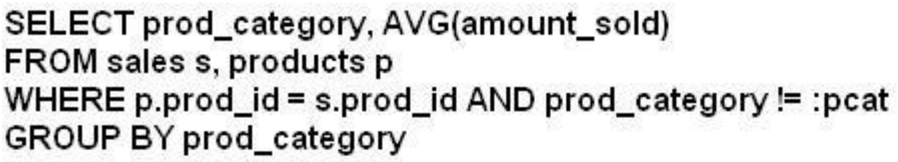
access. A column histogram should only be created when we have data skew exists or is suspected.

QUESTION 10
Which two are the fastest methods for fetching a single row from a table based on an equality predicate?

A. Fast full index scan on an index created for a column with unique key
B. Index unique scan on an created for a column with unique key
C. Row fetch from a single table hash cluster
D. Index range scan on an index created from a column with primary key
E. Row fetch from a table using rowid

Answer: CE
Explanation:
A scan is slower than a row fetch (from hash value or rowid).

QUESTION 11
An application user complains about statement execution taking longer than usual. You find that the query uses a bind variable in the WHERE clause as follows:



```
SELECT prod_category, AVG(amount_sold)
FROM sales s, products p
WHERE p.prod_id = s.prod_id AND prod_category != :pcat
GROUP BY prod_category
```

You want to view the execution plan of the query that takes into account the value in the bind variable PCAT. Which two methods can you use to view the required execution plan?

A. Use the DBMS_XPLAN.DISPLAY function to view the execution plan.
B. Identify the SQL_ID for the statementsand use DBMS_XPLAN.DISPLAY_CURSOR for that SQL_ID to view the execution plan.
C. Identify the SQL_ID for the statement and fetch the execution plan PLAN_TABLE.
D. View the execution plan for the statement from V$SQL_PLAN.
E. Execute the statement with different bind values and set AUTOTRACE enabled for session.

Answer: BD
Explanation:
D: V$SQL_PLAN contains the execution plan information for each child cursor loaded in the library cache.
B: The DBMS_XPLAN package supplies five table functions:
DISPLAY_SQL_PLAN_BASELINE - to display one or more execution plans for the SQL statement identified by SQL handle
DISPLAY - to format and display the contents of a plan table. DISPLAY_AWR - to format and display the contents of the execution plan of a stored SQL statement in the AWR.
DISPLAY_CURSOR - to format and display the contents of the execution plan of any loaded cursor. DISPLAY_SQLSET - to format and display the contents of the execution plan of statements stored in a SQL tuning set.