```
*** create a table with one column (ID) having many distinct values and one column (CODE) having very few

SQL> CREATE TABLE ziggy_stuff AS SELECT mod(rownum,500000) id, mod(rownum,5) code, 'ZIGGY' name FROM dual
CONNECT BY LEVEL <= 1000000;

Table created.

*** However, add a row that has a very distinct CODE value. Although there are only 6 different CODE
values, there's only one occurance of value 42

SQL> INSERT INTO ziggy_stuff VALUES (42, 42, 'BOWIE');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'ZIGGY_STUFF', cascade=> true,
estimate_percent=> null, method_opt=> 'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.


*** Create a histogram on the CODE value so that the CBO knows there's very few CODEs with a value of 42

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'ZIGGY_STUFF', cascade=> true,
estimate_percent=> null, method_opt=> 'FOR COLUMNS CODE SIZE 10');

PL/SQL procedure successfully completed.


*** First, create an index with the ID column being the leading column

SQL> CREATE INDEX ziggy_stuff_id_code_i ON ziggy_stuff(id, code);

Index created.

SQL> SELECT * FROM ziggy_stuff WHERE id = 42 AND code = 42;

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 975820249

-----------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                        |     1 |    13 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ZIGGY_STUFF            |     1 |    13 |     4   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | ZIGGY_STUFF_ID_CODE_I  |     1 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


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

   2 - access("ID"=42 AND "CODE"=42)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          5  consistent gets
          0  physical reads
          0  redo size
        522  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


*** AS expected, search on both columns and the index is used ...


SQL> SELECT * FROM ziggy_stuff WHERE id = 42;

3 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 975820249

-----------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                        |     2 |    26 |     6   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ZIGGY_STUFF            |     2 |    26 |     6   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | ZIGGY_STUFF_ID_CODE_I  |     2 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


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

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

   2 - access("ID"=42)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          7  consistent gets
          0  physical reads
          0  redo size
        572  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          3  rows processed


*** Search on only the leading column (ID) and again the index can be used effectively


SQL> SELECT * FROM ziggy_stuff WHERE code = 42;

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 4141990364

---------------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |    13 |   307  (15)| 00:00:03 |
|*  1 |  TABLE ACCESS FULL| ZIGGY_STUFF |     1 |    13 |   307  (15)| 00:00:03 |
---------------------------------------------------------------------------------


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

   1 - filter("CODE"=42)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       2605  consistent gets
          0  physical reads
          0  redo size
        522  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


*** However, search on the CODE column only and the index can not be used.
*** As the leading column is very selective, a CODE value of 42 could potentially be referenced within any
of the index leaf blocks

*** Let's now re-create the index but with the columns the other way around (CODE now the leading column)


SQL> DROP INDEX ziggy_stuff_id_code_i;

Index dropped.


SQL> CREATE INDEX ziggy_stuff_code_id_i ON ziggy_stuff(code,id);

Index created.


SQL> SELECT * FROM ziggy_stuff WHERE id = 42 AND code = 42;

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 442388428

-----------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                       |     1 |    13 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ZIGGY_STUFF           |     1 |    13 |     4   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | ZIGGY_STUFF_CODE_ID_I |     1 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


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

```
   2 - access("CODE"=42 AND "ID"=42)


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

*** Again as expected, index is used when both columns are searched

```
SQL> SELECT * FROM ziggy_stuff WHERE code = 42;

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 442388428

-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                        |     1 |    13 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ZIGGY_STUFF            |     1 |    13 |     4   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | ZIGGY_STUFF_CODE_ID_I |     1 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


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

   2 - access("CODE"=42)


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

*** When searching on just the CODE column for the value 42, with the histogram in place, the CBO estimates there's only the one row and so can use the index effectively

```
SQL> SELECT * FROM ziggy_stuff WHERE id = 42;

3 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2304838088

-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                        |     2 |    26 |    11   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ZIGGY_STUFF            |     2 |    26 |    11   (0)| 00:00:01 |
|*  2 |   INDEX SKIP SCAN            | ZIGGY_STUFF_CODE_ID_I |     2 |       |     8   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


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

   2 - access("ID"=42)
       filter("ID"=42)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         19  consistent gets
         10  physical reads
          0  redo size
        572  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          3  rows processed
```

*** When searching on just the ID column, the CBO knows there are only 6 distinct CODE column values

*** The CBO can effectively probe the index in 6 different locations and retrieve all the necessary rows.

*** At 19 consistent gets, it's not as good as the 7 consistent gets with the previous index

*** However, it's not too bad and much better than the approx 2605 consistent gets required for a full table scan

*** Perhaps the second index will suffice, making the overheads associated having a second index unnecessary ...