```
-- Create a table based on dba_tables such that we end up with 10000 rows to keep the arithmetic nice and simple

SQL> create table pink_floyd as select owner, table_name, num_rows, blocks from dba_tables;

Table created.

SQL> insert into pink_floyd select * from pink_floyd;

7048 rows created.

SQL> delete pink_floyd where rownum <=4096;

4096 rows deleted.

SQL> commit;

Commit complete.

SQL> select count(*) from pink_floyd;

  COUNT(*)
----------
     10000

-- Create a “normal” index on the table_name column

SQL> create index pink_floyd_table_name_i on pink_floyd(table_name);

Index created.

-- Collects stats on the table

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

PL/SQL procedure successfully completed.

-- Notice how all the columns have accuate stats.

SQL> select column_name, num_distinct, hidden_column, virtual_column from dba_tab_cols where
table_name='PINK_FLOYD';

COLUMN_NAME                    NUM_DISTINCT HID VIR
------------------------------ ------------ --- ---
OWNER                                    74 NO  NO
TABLE_NAME                             5739 NO  NO
NUM_ROWS                                886 NO  NO
BLOCKS                                  152 NO  NO


-- Note average cardinality of the table_name column is ceil(10000/5739) = 2

-- If we run a simple query on the table, searching for a specific table_name:

SQL> select * from pink_floyd where table_name = 'TAB$';


Execution Plan
----------------------------------------------------------
Plan hash value: 4049118941

-------------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                    | Rows  | Bytes |Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                         |     2 |    60 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PINK_FLOYD              |     2 |    60 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | PINK_FLOYD_TABLE_NAME_I |     2 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------


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

   2 - access("TABLE_NAME"='TAB$')


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


-- Note: the number of rows calculated by the CBO matches the actual number of rows returned. This is always a
good thing.


-- However, if we run a similar query, but this time performing a case insensitive search by using the UPPER
function:

SQL> select * from pink_floyd where upper(table_name) = 'TAB$';


Execution Plan
----------------------------------------------------------
Plan hash value: 1152280033

--------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            |   100 |  3000 |    21   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PINK_FLOYD |   100 |  3000 |    21   (0)| 00:00:01 |
--------------------------------------------------------------------------------


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

```
   1 - filter(UPPER("TABLE_NAME")='TAB$')


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

-- Firstly, it negates the use of the index on table_name

-- Also: Oracle is assuming it will now retrieve 100 rows or 1% of the data, not 2 rows as it did previously, as Oracle has no way of knowing exactly how many values could match the outcome of the function.

--If we now create a function-based index to support this query:

```
SQL> create index pink_floyd_upp_tab_name_i on pink_floyd(upper(table_name));

Index created.

SQL> select column_name, num_distinct, hidden_column, virtual_column from dba_tab_cols where
table_name='PINK_FLOYD';

COLUMN_NAME     NUM_DISTINCT HID VIR
--------------- ------------ --- ---
OWNER                     74 NO  NO
TABLE_NAME              5739 NO  NO
NUM_ROWS                 886 NO  NO
BLOCKS                   152 NO  NO
SYS_NC00005$                 YES YES
```

-- Note: Oracle has automatically create a hidden virtual column to support the function-based index, but it has no stats as the table has not been analyzed since the function-based index has been created

-- Runing the same query again …

```
SQL> select * from pink_floyd where upper(table_name) = 'TAB$';


Execution Plan
----------------------------------------------------------
Plan hash value: 1614691703

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                      | Rows  | Bytes| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                           |   100 |  3000|    20   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PINK_FLOYD                |   100 |  3000|    20   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | PINK_FLOYD_UPP_TAB_NAME_I |    40 |      |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------


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

   2 - access(UPPER("TABLE_NAME")='TAB$')


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

-- NOTE: In this instance the CBO has decided to use the index but note Oracle is still assuming it will retrieve 100 rows or 1% of the data, although interestingly, it's assuming a selectivity of 0.4% for the function-based index.

-- Getting the cardinality wrong is often a bad thing and could result in a sub-optimal execution plan …

-- However, if we now collect statistics on this hidden column

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

PL/SQL procedure successfully completed.


SQL> select column_name, num_distinct, hidden_column, virtual_column from dba_tab_cols where
table_name='PINK_FLOYD';

COLUMN_NAME                    NUM_DISTINCT HID VIR
------------------------------ ------------ --- ---
OWNER                                    74 NO  NO
TABLE_NAME                             5739 NO  NO
NUM_ROWS                                886 NO  NO
BLOCKS                                  152 NO  NO
SYS_NC00005$                           5739 YES YES
```

-- We notice we now have accurate statistics on this virtual column …

```
SQL> select * from pink_floyd where upper(table_name) = 'TAB$';


Execution Plan
----------------------------------------------------------
```

```
Plan hash value: 1614691703

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                       | Rows  | Bytes| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                            |     2 |    66|     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PINK_FLOYD                 |     2 |    66|     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | PINK_FLOYD_UPP_TAB_NAME_I |     2 |      |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

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

   2 - access(UPPER("TABLE_NAME")='TAB$')


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


-- And the CBO now accurately determines the correct cardinality of using the index, which again is always a good
thing …
```