```
*** First create a little table, with 3 columns


SQL> create table hist_test (id1 number, id2 number, id3 number);

Table created.


*** The first column is populated by rownum and is perfectly distributed, for now

*** The second column is perfectly distributed with a value between 1 and 10 with 100 even
occurrences of each value

*** The third column is also perfectly distributed but is "special" in that it only has the
one unique value


SQL> insert into hist_test (id1, id2, id3) select rownum, mod(rownum,10)+1, 100 from dual
connect by level <= 1000000;

1000000 rows created.


*** We're now going to "ruin" the perfect distribution of the first column by creating a
rather nasty outlier value

SQL> update hist_test set id1=1000000000 where id1=1000000;

1 row updated.

SQL> COMMIT;

Commit complete.


*** So of the three columns, column one could benefit from a histogram but it would be
pointless to have histograms on the other 2 columns ...

SQL> alter table hist_test add primary key(id1);

Table altered.


*** Let's collect statistics with no histograms, using what were the method_opt default
settings for 9i, 'FOR ALL COLUMNS SIZE 1'

SQL> exec DBMS_STATS.GATHER_TABLE_STATS (null, 'HIST_TEST', method_opt => 'FOR ALL COLUMNS
SIZE 1', estimate_percent=>null);

PL/SQL procedure successfully completed.


*** If we look at dba_tab_histograms, we'll see that indeed there are no histograms
*** Just the standard 2 rows per columns to capture the low and high end points for each
column

SQL> select * from dba_tab_histograms where table_name = 'HIST_TEST' order by column_name,
endpoint_number;

OWNER  TABLE_NAME   COLUMN_NAME  ENDPOINT_NUMBER ENDPOINT_VALUE ENDPOINT_ACT
------ ------------ ------------ --------------- -------------- ------------
BOWIE  HIST_TEST    ID1                        0              1
BOWIE  HIST_TEST    ID1                        1     1000000000
BOWIE  HIST_TEST    ID2                        0              1
BOWIE  HIST_TEST    ID2                        1             10
BOWIE  HIST_TEST    ID3                        0            100
BOWIE  HIST_TEST    ID3                        1            100

6 rows selected.


*** Checking dba_tab_columns also confirms there are no histograms

*** Interestingly, if you look at the density column, they're all perfect.

*** Column 1 would indeed return .000001 (0.0001%) of data for a specific value

*** Column 2 would indeed return .1 (10%) of data for a specific value

*** Column 3 would indeed return 1 (100%) of data for a specific value

SQL> select table_name, column_name, num_distinct, density, num_buckets, histogram from
dba_tab_columns where table_name = 'HIST_TEST';

SQL> select table_name, column_name, num_distinct, density, num_buckets, histogr
am from dba_tab_columns where table_name = 'HIST_TEST';
```

```
TABLE_NAME   COLUMN_NAME  NUM_DISTINCT    DENSITY NUM_BUCKETS HISTOGRAM
------------ ------------ ------------ ---------- ----------- ---------------
HIST_TEST    ID1               1000000    .000001           1 NONE
HIST_TEST    ID2                    10         .1           1 NONE
HIST_TEST    ID3                     1          1           1 NONE


*** Let's now generate some "workload" so Oracle can see that indeed we use all three columns
in SQL statements in our environment ...

SQL> select count(*) from hist_test where id1 = 1;

  COUNT(*)
----------
         1

SQL> select count(*) from hist_test where id2 = 1;

  COUNT(*)
----------
       100

SQL> select count(*) from hist_test where id3 = 1;

  COUNT(*)
----------
         0


*** Let's also run a query using the first column to highlight why a histogram would be
useful here

SQL> select * from hist_test where id1 > 1000000;

       ID1        ID2        ID3
---------- ---------- ----------
1000000000          1        100


Execution Plan
----------------------------------------------------------
Plan hash value: 880336319

-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |   999K|  9755K|   501   (4)| 00:00:07 |
|*  1 |  TABLE ACCESS FULL| HIST_TEST |   999K|  9755K|   501   (4)| 00:00:07 |
-------------------------------------------------------------------------------


*** Notice the expected rows returned is 999K, not 1 and that a Full Table Scan is being
performed !!


*** Let's now gather stats again, but this time we'll let Oracle decide which columns should
have histograms and which columns don't need them
*** by using the FOR ALL COLUMNS SIZE AUTO default option with 10g

SQL> exec DBMS_STATS.GATHER_TABLE_STATS (null, 'HIST_TEST', method_opt => 'FOR ALL COLUMNS
SIZE AUTO', estimate_percent=>null);

PL/SQL procedure successfully completed.


*** Now let's look at what histograms Oracle has generated for us ...

SQL> select * from dba_tab_histograms where table_name = 'HIST_TEST' order by co
lumn_name, endpoint_number;

OWNER  TABLE_NAME   COLUMN_NAME  ENDPOINT_NUMBER ENDPOINT_VALUE ENDPOINT_ACT
------ ------------ ------------ --------------- -------------- ------------
BOWIE  HIST_TEST    ID1                       0              1
BOWIE  HIST_TEST    ID1                       1     1000000000
BOWIE  HIST_TEST    ID2                  100000              1
BOWIE  HIST_TEST    ID2                  200000              2
BOWIE  HIST_TEST    ID2                  300000              3
BOWIE  HIST_TEST    ID2                  400000              4
BOWIE  HIST_TEST    ID2                  500000              5
BOWIE  HIST_TEST    ID2                  600000              6
BOWIE  HIST_TEST    ID2                  700000              7
BOWIE  HIST_TEST    ID2                  800000              8
BOWIE  HIST_TEST    ID2                  900000              9
BOWIE  HIST_TEST    ID2                 1000000             10
BOWIE  HIST_TEST    ID3                 1000000            100

13 rows selected.
```

*** The results are a little disappointing ...

*** Column 1 which perhaps needed a histogram because of the outlier value has actually not had a histogram created

*** Columns 2 which was perfectly distributed now suddenly has a histogram (it now has 10 buckets, one for each value)

*** Column 3 which also really had no need for a histogram has also suddenly been given a histogram, with 1 bucket for it's only value

*** Let's see what impact this now has on the DENSITY values, remembering that the density calculations differ for columns with histograms ...

```
SQL> select table_name, column_name, num_distinct, density, num_buckets, histogram from
dba_tab_columns where table_name = 'HIST_TEST';

TABLE_NAME   COLUMN_NAME  NUM_DISTINCT    DENSITY NUM_BUCKETS HISTOGRAM
------------ ------------ ------------ ---------- ----------- ---------------
HIST_TEST    ID1               1000000    .000001           1 NONE
HIST_TEST    ID2                    10   .0000005          10 FREQUENCY
HIST_TEST    ID3                     1   .0000005           1 FREQUENCY
```

*** Column 1 is fine, its density hasn't changed but remember, this column could really have benefited with a histogram and it doesn't have one

*** Column 2 now has a density value that is totally inaccurate, thanks to the unnecessary histogram

*** Column 3 now has a density value that is also totally inaccurate, thanks to the unnecessary histogram

*** Let's re-run the query that used the first column to see if it now runs better (doubtful if the statistics don't change ...)

```
SQL> select * from hist_test where id1 > 1000000;

       ID1        ID2        ID3
---------- ---------- ----------
1000000000          1        100


Execution Plan
----------------------------------------------------------
Plan hash value: 880336319

--------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |   999K|  9755K|   501   (4)| 00:00:07 |
|*  1 |  TABLE ACCESS FULL| HIST_TEST |   999K|  9755K|   501   (4)| 00:00:07 |
--------------------------------------------------------------------------------
```

*** No change :(

*** Of the 3 columns, Oracle has in one way or the other to differing degrees stuffed up all three columns with the AUTO method_opt option

*** Just to highlight how a histogram would have been useful for the first column, let's create a histogram manually ...

```
SQL> exec DBMS_STATS.GATHER_TABLE_STATS (null, 'HIST_TEST', method_opt => 'FOR COLUMNS ID1
SIZE 254', estimate_percent=>null);

PL/SQL procedure successfully completed.
```

*** Now run the same query again ...

```
SQL> select * from hist_test where id1 > 1000000;

       ID1        ID2        ID3
---------- ---------- ----------
1000000000          1        100


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

Plan hash value: 3591145889

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3937	39370	20 (0)	00:00:01
1	TABLE ACCESS BY INDEX ROWID	HIST_TEST	3937	39370	20 (0)	00:00:01
* 2	INDEX RANGE SCAN	SYS_C006287	3937		11 (0)	00:00:01

*** Although the costings are not perfect, at least they're now "good enough" to generate a more appropriate execution plan

*** Conclusion, be very very careful with the 10g default behaviour for method_opt and considering creating histograms manually on a need to have them basis only ...