```
------------ Create an indexed table with skewed data in it
SQL> create table skew(c1 number(6), c2 char(20));

Table created.

SQL> insert into skew select 10,10 from dual connect by level <= 10000;

10000 rows created.

SQL> update skew set c1 = 20 where rownum <= 10;

10 rows updated.

SQL> create index skew_idx on skew(c1);

Index created.

SQL> select c1, count(*) from skew group by c1;

        C1   COUNT(*)
---------- ----------
        20         10
        10       9990

------------ Gather statistics
SQL> exec dbms_stats.gather_table_stats(null,'skew')

PL/SQL procedure successfully completed.

------------ Count the number of rows where c1 = 10
SQL> select count(c2) from skew where c1 = 10;

 COUNT(C2)
----------
      9990

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = 10

Plan hash value: 568322376

-------------------------------------------
| Id  | Operation          | Name | Rows  |
-------------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
```

```
|   2 |   TABLE ACCESS FULL| SKEW |  5000 |
-----------------------------------------


------------ Why?
SQL> exec dbms_stats.gather_table_stats(null,'skew', -
> method_opt => 'for all columns size 2', no_invalidate => false)

PL/SQL procedure successfully completed.

------------ Count them again
SQL> select count(c2) from skew where c1 = 10;

 COUNT(C2)
----------
      9990

------------ Did it help?
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = 10

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  9990 |
-----------------------------------------


------------ Now count the number of rows where c1 = 20
SQL> select count(c2) from skew where c1 = 20;

 COUNT(C2)
----------
        10

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
```

```
------------------------
select count(c2) from skew where c1 = 20

Plan hash value: 3493361220

-------------------------------------------------------------
| Id  | Operation                    | Name     | Rows  |
-------------------------------------------------------------
|   0 | SELECT STATEMENT             |          |       |
|   1 |  SORT AGGREGATE              |          |     1 |
|   2 |   TABLE ACCESS BY INDEX ROWID| SKEW     |    10 |
|   3 |    INDEX RANGE SCAN          | SKEW_IDX |    10 |
-------------------------------------------------------------


------------ Now lets use a bind variable
SQL> var x number
SQL> exec :x := 10;

PL/SQL procedure successfully completed.

SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
      9990

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  5000 |
-----------------------------------------


------------ Why?
SQL> alter session set "_optim_peek_user_binds" = true;

Session altered.
```

```
------------ Try again
SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
      9990

------------ Did it help?
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  9990 |
-----------------------------------------


------------ What happens if we bind the value 20?
SQL> exec :x := 20;

PL/SQL procedure successfully completed.

SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
        10

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

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

```
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  9990 |
-----------------------------------------


------------ Would it be nice if we can see the peeked value?
------------ We can use peeked_binds in dbms_xplan!
SQL> select sql_id from v$sql
  2  where sql_text = 'select count(c2) from skew where c1 = :x';

SQL_ID
-------------
7x3dh1yrfuqk1
7x3dh1yrfuqk1

SQL> select *
  2  from table(dbms_xplan.display_cursor('&sql_id',null,'basic rows
peeked_binds'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  5000 |
-----------------------------------------

EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  9990 |
-----------------------------------------

Peeked Binds (identified by position):
```

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

   1 - :X (NUMBER): 10


------------ Flush this thing out!
SQL> select address, hash_value from v$sql
  2  where sql_text = 'select count(c2) from skew where c1 = :x';

ADDRESS  HASH_VALUE
-------- ----------
271FE9A4 2934790721
271FE9A4 2934790721

SQL> exec sys.dbms_shared_pool.purge('&address, &hash_value','c')

PL/SQL procedure successfully completed.

------------ Execute our query again
SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
        10

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 3493361220

-----------------------------------------------------------
| Id  | Operation                    | Name     | Rows  |
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |          |       |
|   1 |  SORT AGGREGATE              |          |     1 |
|   2 |   TABLE ACCESS BY INDEX ROWID| SKEW     |    10 |
|   3 |    INDEX RANGE SCAN          | SKEW_IDX |    10 |
-----------------------------------------------------------


------------ Now see adaptive cursor sharing in action
SQL> select child_number, is_bind_sensitive, is_bind_aware, is_shareable
  2  from v$sql where sql_id = '&sql_id';

CHILD_NUMBER IS_BIND_SENSITIVE IS_BIND_AWARE IS_SHAREABLE
------------ ----------------- ------------- ------------
```

```
           0 Y                 N             Y

------------ Letx bind 10 again
SQL> exec :x := 10;

PL/SQL procedure successfully completed.

SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
      9990

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 3493361220

-----------------------------------------------------------
| Id  | Operation                    | Name     | Rows  |
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |          |       |
|   1 |  SORT AGGREGATE              |          |     1 |
|   2 |   TABLE ACCESS BY INDEX ROWID| SKEW     |    10 |
|   3 |    INDEX RANGE SCAN          | SKEW_IDX |    10 |
-----------------------------------------------------------


------------ See our cursors
SQL> select child_number, is_bind_sensitive, is_bind_aware,is_shareable
  2  from v$sql where sql_id = '&sql_id';

CHILD_NUMBER IS_BIND_SENSITIVE IS_BIND_AWARE IS_SHAREABLE
------------ ----------------- ------------- ------------
           0 Y                 N             Y

------------ Execute it again
SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
      9990

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));
```

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 568322376

-----------------------------------------
| Id  | Operation          | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT   |      |       |
|   1 |  SORT AGGREGATE    |      |     1 |
|   2 |   TABLE ACCESS FULL| SKEW |  9990 |
-----------------------------------------


------------ See our cursors
SQL> select child_number, is_bind_sensitive, is_bind_aware,is_shareable
  2  from v$sql where sql_id = '&sql_id';

CHILD_NUMBER IS_BIND_SENSITIVE IS_BIND_AWARE IS_SHAREABLE
------------ ----------------- ------------- ------------
           0 Y                 N             Y
           1 Y                 Y             Y

------------ Letx bind 20 again
SQL> exec :x := 20;

PL/SQL procedure successfully completed.

SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
        10

------------ See what the optimizer expected
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 3493361220

-----------------------------------------------------
| Id  | Operation                    | Name     | Rows  |
```

```
-----------------------------------------------------
|   0 | SELECT STATEMENT             |          |       |
|   1 |  SORT AGGREGATE              |          |     1 |
|   2 |   TABLE ACCESS BY INDEX ROWID| SKEW     |    10 |
|   3 |    INDEX RANGE SCAN          | SKEW_IDX |    10 |
-----------------------------------------------------


------------ See our cursors
SQL> select child_number, is_bind_sensitive, is_bind_aware,is_shareable
  2  from v$sql where sql_id = '&sql_id';

CHILD_NUMBER IS_BIND_SENSITIVE IS_BIND_AWARE IS_SHAREABLE
------------ ----------------- ------------- ------------
           0 Y                 N             N
           1 Y                 Y             Y
           2 Y                 Y             Y

------------ Each child has an associate selectivity
SQL> desc v$sql_cs_selectivity
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
-------
 ADDRESS                                            RAW(4)
 HASH_VALUE                                         NUMBER
 SQL_ID                                             VARCHAR2(13)
 CHILD_NUMBER                                       NUMBER
 PREDICATE                                          VARCHAR2(40)
 RANGE_ID                                           NUMBER
 LOW                                                VARCHAR2(10)
 HIGH                                               VARCHAR2(10)

------------ See it
SQL> select child_number, predicate,range_id, low, high
  2  from V$SQL_CS_SELECTIVITY where sql_id = '&sql_id' order by
child_number;

CHILD_NUMBER PREDICATE    RANGE_ID LOW        HIGH
------------ ---------- ---------- ---------- ----------
           1 =X                  0 0.899100   1.098900
           2 =X                  0 0.000855   0.001045

------------ What happens if we bind the value 30?
SQL> exec :x := 30;

PL/SQL procedure successfully completed.

SQL> select count(c2) from skew where c1 = :x;

 COUNT(C2)
----------
         0

------------ See what the optimizer expected
```

```
SQL> select * from table(dbms_xplan.display_cursor(null,null,'basic
rows'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
EXPLAINED SQL STATEMENT:
------------------------
select count(c2) from skew where c1 = :x

Plan hash value: 3493361220

-----------------------------------------------------------
| Id  | Operation                     | Name     | Rows  |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |          |       |
|   1 |  SORT AGGREGATE               |          |     1 |
|   2 |   TABLE ACCESS BY INDEX ROWID| SKEW     |     1 |
|   3 |    INDEX RANGE SCAN           | SKEW_IDX |     1 |
-----------------------------------------------------------


------------ See our cursors
SQL> select child_number, is_bind_sensitive, is_bind_aware,is_shareable
  2  from v$sql where sql_id = '&sql_id';

CHILD_NUMBER IS_BIND_SENSITIVE IS_BIND_AWARE IS_SHAREABLE
------------ ----------------- ------------- ------------
           0 Y                 N             N
           1 Y                 Y             Y
           2 Y                 Y             N
           3 Y                 Y             Y

------------ Check their selectivity
SQL> select child_number, predicate,range_id, low, high
  2  from V$SQL_CS_SELECTIVITY where sql_id = '&sql_id' order by
child_number;

CHILD_NUMBER PREDICATE    RANGE_ID LOW        HIGH
------------ ---------- ---------- ---------- ----------
           1 =X                  0 0.899100   1.098900
           2 =X                  0 0.000855   0.001045
           3 =X                  0 0.000450   0.001045

------------ Get some execution info about the children
SQL> desc V$SQL_CS_STATISTICS
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------
-------
 ADDRESS                                            RAW(4)
 HASH_VALUE                                         NUMBER
 SQL_ID                                             VARCHAR2(13)
 CHILD_NUMBER                                       NUMBER
 BIND_SET_HASH_VALUE                                NUMBER
```

```
 PEEKED                                                    VARCHAR2(1)
 EXECUTIONS                                                NUMBER
 ROWS_PROCESSED                                            NUMBER
 BUFFER_GETS                                               NUMBER
 CPU_TIME                                                  NUMBER

------------ See it
SQL> select child_number, executions, rows_processed, buffer_gets,
cpu_time
  2  from v$sql_cs_statistics order by child_number;

CHILD_NUMBER EXECUTIONS ROWS_PROCESSED BUFFER_GETS   CPU_TIME
------------ ---------- -------------- ----------- ----------
           0          1             21           3          0
           1          1           9991          46          0
           2          1             21           3          0
           3          1              1           2          0
```