```
SQL> CREATE TABLE big_dwh_table (id NUMBER, album_id NUMBER, artist_id NUMBER, country_id
NUMBER, format_id NUMBER, release_date DATE, total_sales NUMBER);

Table created.

SQL> CREATE SEQUENCE dwh_seq;

Sequence created.

SQL> create or replace procedure pop_big_dwh_table as
  2  v_id          number;
  3  v_artist_id   number;
  4  begin
  5    for v_album_id in 1..10000 loop
  6        v_artist_id:= ceil(dbms_random.value(0,100));
  7        for v_country_id in 1..100 loop
  8          select dwh_seq.nextval into v_id from dual;
  9          insert into big_dwh_table values (v_id, v_album_id, v_artist_id, vcountry_id,
ceil(dbms_random.value(0,4)),
                                 trunc(sysdate-mod(v_id,ceil(dbms_random.value(0,1000)))),
ceil(dbmsrandom.value(0,500000)));
 10       end loop;
 11    end loop;
 12 commit;
 13 end;
 14 /
Procedure created.

SQL> exec pop_big_dwh_table

PL/SQL procedure successfully completed.



*** Note that the data in the table is stored in ALBUM_ID order



SQL> CREATE TABLE big_dwh_table_2 AS SELECT * FROM big_dwh_table ORDER BY total_sales;

Table created.

*** This table however is ordered by TOTAL_SALES and NOT by ALBUM_ID



SQL> CREATE INDEX big_dwh_album_id_i ON big_dwh_table(album_id);

Index created.



SQL> CREATE INDEX big_dwh_2_album_id_i ON big_dwh_table_2(album_id);

Index created.



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

PL/SQL procedure successfully completed.



SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'BIG_DWH_TABLE_2',
cascade=>true, estimate_percent=>null, method_opt=>'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.
```

```
SQL> SELECT index_name, leaf_blocks, clustering_factor FROM user_indexes WHERE index_name
like 'BIG_DWH%ALBUM_ID_I';


INDEX_NAME                     LEAF_BLOCKS CLUSTERING_FACTOR

------------------------------ ----------- -----------------

BIG_DWH_2_ALBUM_ID_I                  2090            989933

BIG_DWH_ALBUM_ID_I                    2090              4948
```

*** The index on the BIG_DWH_TABLE_2 table has a very bad Clustering Factor but the index on the other table has an excellent Clustering Factor.

*** This first query is order by a column that is not indexed. Therefore it must perform a SORT ORDER BY operation

```
SQL> SELECT * from big_dwh_table WHERE album_id BETWEEN 10 AND 20 ORDER BY id;

1100 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1145799650
-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |  1200 | 36000 |    12   (9)| 00:00:01 |
|   1 |  SORT ORDER BY               |                    |  1200 | 36000 |    12   (9)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| BIG_DWH_TABLE      |  1200 | 36000 |    11   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | BIG_DWH_ALBUM_ID_I |  1200 |       |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

*** However, this query uses an ORDER BY based on the indexed column and so the CBO has decided to use the index and thus eliminate the SORT ORDER BY operation

```
SQL> SELECT * from big_dwh_table WHERE album_id BETWEEN 10 AND 20 ORDER BY album_id;

1100 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 278738356
-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name               | Rows  | Bytes | Cost(%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |  1200 | 36000 |    11   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| BIG_DWH_TABLE      |  1200 | 36000 |    11   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | BIG_DWH_ALBUM_ID_I |  1200 |       |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

*** This query effective retrieves all the rows from the table and yet it still uses the index as it has an excellent Clustering Factor and the cost of reading the whole table via the index thus preventing the sort operation is less than a FTS plus sort operation

```
SQL> SELECT * from big_dwh_table WHERE album_id BETWEEN 1 AND 10000 ORDER BY album_id;

1000000 rows selected.



Execution Plan
----------------------------------------------------------
Plan hash value: 278738356

-------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name               | Rows  | Bytes |  Cost(%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |  1000K|    28M|  7075   (1)| 00:01:25 |
|   1 |  TABLE ACCESS BY INDEX ROWID| BIG_DWH_TABLE      |  1000K|    28M|  7075   (1)| 00:01:25 |
|*  2 |   INDEX RANGE SCAN           | BIG_DWH_ALBUM_ID_I |  1000K|       |  2106   (1)| 00:00:26 |
-------------------------------------------------------------------------------------------------------
```