ITEMSAMPLE

ItemID	ItemCost
1001	2000
1002	5000
1003	10000
1004	6000
1005	3000
1006	7000
1007	1000
1008	2500
1009	15000
1010	17500

Display the top-3 most expensive items.

Display the top-3 most expensive items.

```
SELECT ItemID, ItemCost
FROM ItemSample I
ORDER BY ItemCost DESC;
```

```
SQL> SELECT ItemID, ItemCost
  2  FROM ItemSample I
  3  ORDER BY ItemCost DESC;

    ITEMID   ITEMCOST
---------- ----------
      1010      17500
      1009      15000
      1003      10000
      1006       7000
      1004       6000
      1002       5000
      1005       3000
      1008       2500
      1001       2000
      1007       1000

10 rows selected.
```

The query above only shows the order of all item from the most to the least expensive.

Display the top-3 most expensive items.

The two queries below use ROWNUM and RANK, both of which are operators provided by ORACLE DBMS. They can be used to answer the query, but we do not allow the use of these in our assignment.

```
SELECT ItemID, ItemCost
FROM (SELECT ItemID, ItemCost
      FROM ItemSample I
      ORDER BY ItemCost DESC)
WHERE ROWNUM <= 3;
```

```
SQL> SELECT ItemID, ItemCost
  2  FROM (SELECT ItemID, ItemCost
  3        FROM ItemSample I
  4        ORDER BY ItemCost DESC)
  5  WHERE ROWNUM <= 3;

    ITEMID   ITEMCOST
---------- ----------
      1010      17500
      1009      15000
      1003      10000
```

```
SELECT ItemID, ItemCost
FROM (SELECT ItemID, ItemCost,
      RANK () OVER (ORDER BY ItemCost DESC)
      AS RANK
     FROM ItemSample I
     ORDER BY RANK)
WHERE RANK <= 3;
```

```
SQL> SELECT ItemID, ItemCost
  2  FROM (SELECT ItemID, ItemCost, RANK () OVER (ORDER BY ItemCost DESC) AS RANK
  3        FROM ItemSample I
  4        ORDER BY RANK)
  5  WHERE RANK <= 3;

    ITEMID   ITEMCOST
---------- ----------
      1010      17500
      1009      15000
      1003      10000
```

Display the top-3 most expensive items.

```
SELECT  I1.ItemID, I1.ItemCost
FROM    ItemSample I1
WHERE   3 >=
        (SELECT COUNT(*)
         FROM ItemSample I2
         WHERE I1.ItemCost <= I2.ItemCost)
ORDER BY I1.ItemCost DESC;
```

The suggested solution is to use nested query called correlated sub-query as shown.

In this type of query, for each outer query tuple, we perform the inner query.

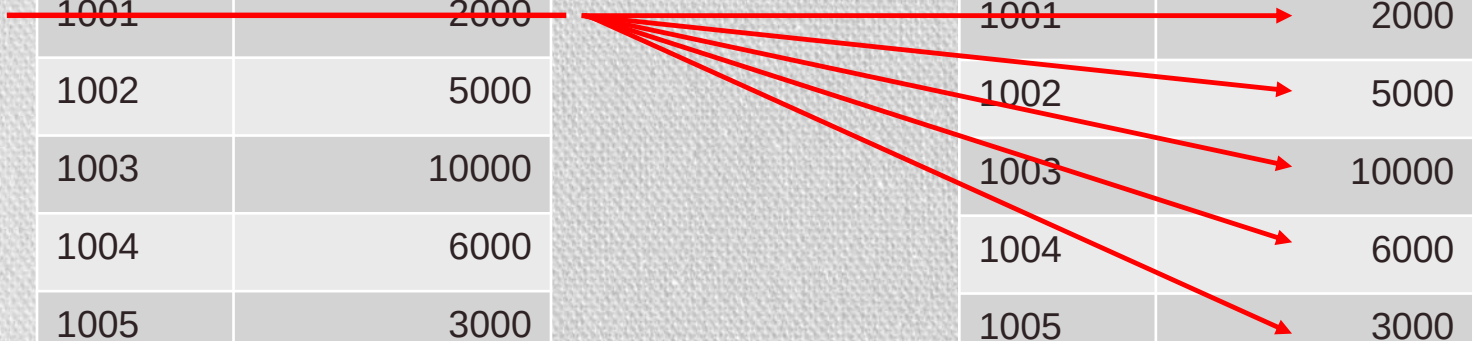
```
SQL> SELECT I1.ItemID, I1.ItemCost
  2  FROM ItemSample I1
  3  WHERE  3 >=
  4   (SELECT COUNT(*) FROM ItemSample I2
  5    WHERE I1.ItemCost <= I2.ItemCost)
  6  ORDER BY I1.ItemCost DESC;

    ITEMID   ITEMCOST
---------- ----------
      1010      17500
      1009      15000
      1003      10000
```

ITEMSAMPLE I1

ItemID	ItemCost
1001	2000
1002	5000
1003	10000
1004	6000
1005	3000

ITEMSAMPLE I2

ItemID	ItemCost
1001	2000
1002	5000
1003	10000
1004	6000
1005	3000



NOTE:
For this illustration, let's say we have 5 instances/records only

```
For Record 1 (1001, 2000)

    SELECT      I1.ItemID, I1.ItemCost
    FROM        ItemSample I1
    WHERE       3 >=                                  COUNT(*) = 5
       (SELECT COUNT(*)
        FROM ItemSample I2
        WHERE I1.ItemCost <= I2.ItemCost);
                          ↑
                          2000
```

ITEMSAMPLE I1

ItemID	ItemCost
~~1001~~	~~2000~~
1002	5000
1003	10000
1004	6000
~~1005~~	~~3000~~

ITEMSAMPLE I2

ItemID	ItemCost
1001	2000
~~1002~~	5000
1003	10000
1004	6000
1005	3000

NOTE:
For this illustration, let's say we have 5 instances/records only

For Record 2 (1002, 5000)

```
SELECT      I1.ItemID, I1.ItemCost
FROM        ItemSample I1
WHERE       3 >=
   (SELECT COUNT(*)                    COUNT(*) = 3
    FROM ItemSample I2
    WHERE I1.ItemCost <= I2.ItemCost);
```

5000