**SUBQUERIES EXAMPLES**

1. **Single Row SubQueries - use operators =, >, <, <>**

1) a) Display full name of an employee (like Ortiz, Daisy), job title and hire date, if they were hired After the last hired Stock Clerk. Sort the output by earlier hire dates. Headings should be called *Full Name, Job and Hire Date.*

SELECT SUBSTR(last\_name || ', ' || first\_name,1,20) "Full Name",

SUBSTR(job\_title,1,15) "Job", hire\_date "Hire Date"

FROM employees

WHERE hire\_date > (SELECT MAX(hire\_date) FROM employees

WHERE lower(job\_title) = 'stock clerk')

ORDER by hire\_date;

**Full Name Job Hire Date**

Ortiz, Daisy Sales Representative 15-DEC-16

Palmer, Abigail Shipping Clerk 19-DEC-16

Price, Frederick Purchasing Clerk 24-DEC-16

1) b) Can you comment how the output changed if we use condition “ hired After or On day when the last Stock Clerk was hired”?

SELECT SUBSTR(last\_name || ', ' || first\_name,1,20) "Full Name",

SUBSTR(job\_title,1,15) "Job", hire\_date "Hire Date"

FROM employees

WHERE hire\_date >= (SELECT MAX(hire\_date) FROM employees

WHERE lower(job\_title) = 'stock clerk')

ORDER by hire\_date;

**Full Name Job Hire Date**

Bryant, Felix Stock Clerk 12-DEC-16 🡪 this is the last Stock Clerk hired

Ortiz, Daisy Sales Represent 15-DEC-16

Palmer, Abigail Shipping Clerk 19-DEC-16

Price, Frederick Purchasing Cler 24-DEC-16

2) Display product Id , Name and list Price of those products that belong to category CPU and that cost less than $700. Sort by Id ascending. Headings should be called *ProdId, Product Name and LPrice.*

SELECT product\_id "ProdId",

SUBSTR(product\_name,1,20) "Product name", list\_price "LPrice"

FROM products

WHERE category\_id = (SELECT category\_id FROM product\_categories

WHERE UPPER(category\_name) = 'CPU')

AND list\_price < 700

ORDER BY product\_id ;

**ProdId Product name LPrice**

8 Intel Xeon E5-1650 V 601.99

69 Intel Core i7-7820X 678.75

71 Intel Core i7-3930K 660

72 Intel Xeon E5-2630 V 589.99

75 Intel Core i7-4930K 624.04

76 Intel Xeon E5-2630 V 629.99

80 Intel Xeon E5-1650 V 564.89

106 Intel Xeon E5-2640 V 608.95

124 Intel Xeon E5-1650 V 594.99

132 Intel Core i7-5930K 554.99

155 Intel Xeon E5-2630 V 588.95

198 Intel Core i7-980 699.99

199 Intel Xeon E5-2630 V 647.99

200 Intel Core i7-4790K 620.95

14 rows selected.

3) a) Display Lowest list Price for each category. Sort the output by the Price descending.

SELECT category\_id, MIN(list\_price)

FROM products

GROUP BY category\_id

ORDER BY 2 desc ;

**CATEGORY\_ID MIN(LIST\_PRICE)**

2 739.99 🡪 Highest Minimal Price Amount per Category

1. 554.99

4 279.99

5 15.55

3) b) Display Aveerage list Price for each category rounded to 2 decimals. Sort the output by the Price descending.

SELECT category\_id, round( avg(list\_price),2) "Avg Price per Cat"

FROM products

GROUP BY category\_id

ORDER BY 2 desc;

**CATEGORY\_ID Avg Price per Cat**

2 1406.10 🡪 greater than $740

1 1386.97 🡪 greater than $740

5 635.22

4 402.29

4) ***Variation of 3 a and 3 b***  Display Average list Price for each category rounded to 2 decimals, but only if that Average amount is greater than the Highest minimal price amount per each category. Sort the output by Id ascending.

SELECT category\_id, ROUND(AVG(list\_Price,2) "AVG Price"

FROM products

GROUP BY category\_id

HAVING AVG(list\_Price) >

(SELECT MAX(min(list\_price))

FROM products

GROUP BY category\_id)

ORDER BY category\_id ;

**CATEGORY\_ID AVG Price**

1 1386.96

2 1407.00

5) Same as 4) just display category Name as well.

SELECT category\_id, SUBSTR(category\_name,1,12) "Cat name",

AVG(list\_Price) "AVG Price"

FROM products JOIN product\_categories

USING (category\_id)

GROUP BY category\_id, category\_name

HAVING AVG(list\_Price) >

(SELECT MAX(min(list\_price))

FROM products

GROUP BY category\_id)

ORDER BY category\_id ;

**CATEGORY\_ID Cat name AVG Price**

1 CPU 1386.96643

2 Video Card 1406.098

SQL> desc order\_items

**Name Null? Type**

ORDER\_ID NOT NULL NUMBER(12)

ITEM\_ID NOT NULL NUMBER(12)

PRODUCT\_ID NOT NULL NUMBER(12)

QUANTITY NOT NULL NUMBER(8,2)

UNIT\_PRICE NOT NULL NUMBER(8,2)

**II) Multiple Row SubQueries – use operators IN,ANY,ALL**

1) Display product Id , Name and list Price of those products that belong to category starting on C or V and that cost less than $700. Sort by Id ascending. Headings should be called *ProdId, Product Name and LPrice.*

SELECT product\_id "ProdId", SUBSTR(product\_name,1,20) "Product name",

list\_price "Lprice"

FROM products

WHERE category\_id IN (SELECT category\_id FROM product\_categories

WHERE SUBSTR(category\_name,1,1) IN ('C','V') )

AND list\_price < 700

ORDER BY product\_id ;

**ProdId Product name LPrice**

8 Intel Xeon E5-1650 V 601.99

69 Intel Core i7-7820X 678.75

71 Intel Core i7-3930K 660

72 Intel Xeon E5-2630 V 589.99

75 Intel Core i7-4930K 624.04

76 Intel Xeon E5-2630 V 629.99

80 Intel Xeon E5-1650 V 564.89

106 Intel Xeon E5-2640 V 608.95

124 Intel Xeon E5-1650 V 594.99

132 Intel Core i7-5930K 554.99

155 Intel Xeon E5-2630 V 588.95

198 Intel Core i7-980 699.99

199 Intel Xeon E5-2630 V 647.99

200 Intel Core i7-4790K 620.95

14 rows selected**.**

2) Display product Id , Name and list Price of those products that belong to category starting on C and that cost less than $700, but only for those products that have sold less than 100 units in total. Sort by Id ascending. Headings should be called *ProdId, Product Name and LPrice.*

SELECT product\_id “ProdId”,

SUBSTR(product\_name,1,20) "Product name", list\_price "LPrice"

FROM products

WHERE category\_id = (SELECT category\_id FROM product\_categories

WHERE category\_name LIKE 'C%')

AND list\_price < 700

AND product\_id IN (SELECT product\_id FROM order\_items

GROUP BY product\_id

HAVING SUM(quantity) < 100)

ORDER BY product\_id ;

**ProdId Product name LPrice**

8 Intel Xeon E5-1650 V 601.99

72 Intel Xeon E5-2630 V 589.99

132 Intel Core i7-5930K 554.99

Instead IN operator in the second SubQuery, we could use =ANY like shown here;

AND product\_id =ANY (SELECT product\_id FROM order\_items

IF there were TWO or more categories starting on C, we would get the following error

ERROR at line 3:

ORA-01427: single-row subquery returns more than one row

To fix this problem, just replace = with IN in the first SubQuery like shown here:

WHERE category\_id IN (SELECT category\_id FROM product\_categories

3) Display product Id , Name and list Price of those products that belong to category starting on C or V, and that cost less than ANY lowest product price per category. Sort by Id ascending.

SELECT product\_id "ProdId", SUBSTR(product\_name,1,20) "Product name",

list\_price "Lprice"

FROM products

WHERE category\_id IN (SELECT category\_id FROM product\_categories

WHERE SUBSTR(category\_name,1,1) IN ('C','V') )

AND list\_price < ANY (SELECT MIN(list\_price) FROM products

GROUP by category\_id)

ORDER BY product\_id ;

**ProdId Product name LPrice**

8 Intel Xeon E5-1650 V 601.99

69 Intel Core i7-7820X 678.75

70 Intel Xeon E5-2687W 710.99

71 Intel Core i7-3930K 660

72 Intel Xeon E5-2630 V 589.99

74 Intel Xeon E5-2680 V 701.95

75 Intel Core i7-4930K 624.04

76 Intel Xeon E5-2630 V 629.99

80 Intel Xeon E5-1650 V 564.89

106 Intel Xeon E5-2640 V 608.95

124 Intel Xeon E5-1650 V 594.99

132 Intel Core i7-5930K 554.99

155 Intel Xeon E5-2630 V 588.95

198 Intel Core i7-980 699.99

199 Intel Xeon E5-2630 V 647.99

200 Intel Core i7-4790K 620.95

16 rows selected.

*Note: < ANY means < than the highest value of those in SubQuery list of values. Check 3a) and see that value returned by this Multiple Row SubQuery is $740.*

4) VARIATION of 3) Display product Id , Name and list Price of those products that belong to category starting on C or V, and that cost more than ANY lowest product price per category. Sort by Id ascending.

SELECT product\_id "ProdId", SUBSTR(product\_name,1,20) "Product name",

list\_price "Lprice"

FROM products

WHERE category\_id IN (SELECT category\_id FROM product\_categories

WHERE SUBSTR(category\_name,1,1) IN ('C','V') )

AND list\_price > ANY (SELECT MIN(list\_price) FROM products

GROUP by category\_id)

ORDER BY product\_id ;

**ProdId Product name LPrice**

2 Intel Xeon E5-2697 V 2554.99

3 Corsair CB-9060011-W 799.99

4 AMD 100-505989 2699.99

5 PNY VCQK6000-PB 2290.79

6 Zotac ZT-P10810A-10P 849.99

8 Intel Xeon E5-1650 V 601.99

9 Intel Xeon E5-2640 V 899.99

11 PNY VCQP5000-PB 2015.11

12 Gigabyte GV-N108TAOR 824.98

16 Intel Core i7-6900K 889.89

19 Intel Core i7-6950X 1704.37

ETC …

270 Gigabyte GV-N1070WF2 769.99

281 Asus ROG-POSEIDON-GT 864.98

120 rows selected.

*Note: > ANY means > than the lowest value of those in SubQuery list of values. Check 3a) and see that value returned by this Multiple Row SubQuery is $15.5*

5) VARIATION of 3) Display product Id , Name and list Price of those products that belong to category starting on C or V, and that cost matches ANY amount of lowest product price per category. Sort by Id ascending.

SELECT product\_id "ProdId", SUBSTR(product\_name,1,20) "Product name",

list\_price "Lprice"

FROM products

WHERE category\_id IN (SELECT category\_id FROM product\_categories

WHERE SUBSTR(category\_name,1,1) IN ('C','V') )

AND list\_price =ANY (SELECT MIN(list\_price) FROM products

GROUP by category\_id)

ORDER BY product\_id ;

**ProdId Product name LPrice**

132 Intel Core i7-5930K 554.99

239 MSI GTX 1080 Ti DUKE 739.99

*Note: = ANY means = to any value of those in SubQuery list of values. Check 3a) and see that we have 4 values total in that list, but here we get only two values for two categories starting on C and V (categories 1 and 2).*