**SQL SUBQUIRES :**

<https://www.w3resource.com/sql-exercises/subqueries/index.php#SQLEDITOR>

**Write a query to find all the orders issued against the salesman who may ­works for customer whose id is 3007­­**

SELECT \* FROM orders WHERE salesman\_id = (SELECT DISTINCT salesman\_id

FROM orders WHERE customer\_id =3007);

**Write a query to find the name and numbers of all salesmen who had more than one customer.**

SELECT salesman\_id,name FROM salesman a WHERE 1 < (SELECT COUNT(\*)

FROM customer WHERE salesman\_id=a.salesman\_id);

=== or

select name, salesman\_id from salesman where salesman\_id in (select salesman\_id from customer group by salesman\_id having count(\*)>1)

**Write a query to count the customers with grades above New York's average.**

SELECT grade, COUNT (\*) FROM customer GROUP BY grade HAVING grade >

(SELECT AVG(grade) FROM customer WHERE city = 'New York');

**Write a query to find all orders with order amounts which are above-average amounts for their customers.**

SELECT \* FROM orders a WHERE purch\_amt >(SELECT AVG(purch\_amt) FROM orders b WHERE b.customer\_id = a.customer\_id);

**Write a query to find the sums of the amounts from the orders table, grouped by date, eliminating all those dates where the sum was not at least 1000.00 above the maximum order amount for that date.**

SELECT ord\_date, SUM (purch\_amt) FROM orders a GROUP BY ord\_date

HAVING SUM (purch\_amt) > (SELECT 1000.00 + MAX(purch\_amt)

FROM orders b WHERE a.ord\_date = b.ord\_date);

**Write a query to extract the data from the customer table if and only if one or more of the customers in the customer table are located in London.**

SELECT customer\_id,cust\_name, city FROM customer WHERE EXISTS

(SELECT \* FROM customer WHERE city='London');

**Write a query to find all the salesmen who worked for only one customer.**

SELECT \* FROM salesman WHERE salesman\_id IN (SELECT DISTINCT salesman\_id

FROM customer a WHERE NOT EXISTS (SELECT \* FROM customer b

WHERE a.salesman\_id=b.salesman\_id AND a.cust\_name<>b.cust\_name));

== or

select \* from salesman where salesman\_id in (select salesman\_id from customer group by salesman\_id having count(\*) =1)

**Write a query that extract the rows of all salesmen who have customers with more than one orders**.

SELECT \* FROM salesman a WHERE EXISTS (SELECT \* FROM customer b

WHERE a.salesman\_id=b.salesman\_id AND 1<(SELECT COUNT (\*)

FROM orders WHERE orders.customer\_id = b.customer\_id));

== or

select \* from salesman where salesman\_id in(select salesman\_id from customer where customer\_id in (select customer\_id from orders group by customer\_id having count(\*)>1))

**Write a query to display only those customers whose grade are, in fact, higher than every customer in New York.**

SELECT \* FROM customer WHERE grade > ALL (SELECT grade

FROM customer WHERE city='New York');

**Write a query to get all the information for those customers whose grade is not as the grade of customer who belongs to the city London.**

SELECT \* FROM customer WHERE grade <> ANY (SELECT grade FROM customer

WHERE city='London');

**Write a SQL query to find the average price of each manufacturer's products along with their name.**

SELECT AVG(pro\_price) AS "Average Price", company\_mast.com\_name As "Company" FROM item\_mast, company\_mast WHERE item\_mast.pro\_com= company\_mast.com\_id GROUP BY company\_mast.com\_name;

**Write a SQL query to display the average price of the products which is more than or equal to 350 along with their names.**

SELECT AVG(pro\_price) AS "Average Price", company\_mast.com\_name AS "Company" FROM item\_mast, company\_mast WHERE item\_mast.pro\_com= company\_mast.com\_id GROUP BY company\_mast.com\_name

HAVING AVG(pro\_price) >= 350;

**Write a SQL query to display the name of each company, price for their most expensive product along with their Name.**

SELECT P.pro\_name AS "Product Name", P.pro\_price AS "Price", C.com\_name AS "Company" FROM item\_mast P, company\_mast C

WHERE P.pro\_com = C.com\_id AND P.pro\_price =

(SELECT MAX(P.pro\_price) FROM item\_mast P WHERE P.pro\_com = C.com\_id );

**Write a query in SQL to display the first name and last name of employees working for the department which allotment amount is more than Rs.50000.**

SELECT emp\_fname, emp\_lname FROM emp\_details WHERE emp\_dept IN

(SELECT dpt\_code FROM emp\_department WHERE dpt\_allotment > 50000);

**Or ==**

select ed.emp\_fname,ed.emp\_lname from emp\_details ed inner join emp\_department e on e.dpt\_code=ed.emp\_dept where e.dpt\_allotment>50000;

**Write a query in SQL to find the departments which sanction amount is larger than the average sanction amount of all the departments.**

SELECT \* FROM emp\_department WHERE dpt\_allotment > (SELECT AVG(dpt\_allotment) FROM emp\_department);

**Write a query in SQL to find the names of departments with more than two employees are working.**

SELECT dpt\_name FROM emp\_department WHERE dpt\_code IN (SELECT emp\_dept FROM emp\_details GROUP BY emp\_dept HAVING COUNT(\*) >2 );

**Write a query in SQL to find the first name and last name of employees working for departments which sanction amount is second lowest.**

SELECT emp\_fname, emp\_lname FROM emp\_details WHERE emp\_dept IN (

SELECT dpt\_code FROM emp\_department WHERE dpt\_allotment= (

SELECT MIN(dpt\_allotment) FROM emp\_department WHERE dpt\_allotment >(SELECT MIN(dpt\_allotment) FROM emp\_department )));

Write a query to display all the orders which values are greater than the average order value for 10th October 2012

SELECT \* FROM orders WHERE purch\_amt > (SELECT AVG(purch\_amt)

FROM orders WHERE ord\_date ='10/10/2012');

Write a query to display all the customers whose id is 2001 bellow the salesman ID of Mc Lyon

SELECT \* FROM customer WHERE customer\_id = (SELECT salesman\_id -2001 FROM salesman WHERE name = 'Mc Lyon');

Write a query to find the salesmen who have multiple customers.

SELECT \* FROM salesman WHERE salesman\_id IN (

SELECT DISTINCT salesman\_id FROM customer a WHERE EXISTS

(SELECT \* FROM customer b WHERE b.salesman\_id=a.salesman\_id

AND b.cust\_name<>a.cust\_name));

Write a query to find all the salesmen for whom there are customers that follow them.

SELECT \* FROM salesman WHERE city IN (SELECT city FROM customer);

Write a query to find salesmen with all information who lives in the city where any of the customers lives.

SELECT \* FROM salesman WHERE city=ANY (SELECT city FROM customer);

Write a query to display the salesmen which name are alphabetically lower than the name of the customers.

SELECT \* FROM salesman a WHERE EXISTS (SELECT \* FROM CUSTOMER b WHERE a.name < b.cust\_name);

Write a query to display the customers who have a greater gradation than any customer who belongs to the alphabetically lower than the city New York

SELECT \* FROM customer WHERE grade > ANY (SELECT grade FROM CUSTOMER

WHERE city < 'New York');

Write a query to display all the orders that had amounts that were greater than at least one of the orders on September 10th 2012.

SELECT \* FROM Orders WHERE purch\_amt > ANY (SELECT purch\_amt

FROM orders WHERE ord\_date='2012/09/10');

Write a query to find all orders with an amount smaller than any amount for a customer in London.

SELECT \* FROM orders WHERE purch\_amt < ANY (SELECT purch\_amt

FROM orders a, customer b WHERE a.customer\_id=b.customer\_id

AND b.city='London');

Write a query to display all orders with an amount smaller than the amount for a customer in London.

SELECT \* FROM orders WHERE purch\_amt < (SELECT MAX (purch\_amt)

FROM orders a, customer b WHERE a.customer\_id=b.customer\_id

AND b.city='London');

Write a query to find only those customers whose grade are, higher than every customer to the city New York.

SELECT \* FROM customer WHERE grade > ALL (SELECT grade FROM customer WHERE city = 'New York')

Write a query to find all those customers whose grade are not as the grade, belongs to the city Paris.

SELECT \* FROM customer WHERE grade NOT IN (SELECT grade FROM customer

WHERE city='Paris');

Write a query to find all those customers who hold a different grade than any customer of the city Dallas.

SELECT \* FROM customer WHERE NOT grade = ANY (SELECT grade

FROM customer WHERE city='Dallas');

Write a query in SQL to find all the details of employees whose last name is Gabriel or Dosio

SELECT \* FROM emp\_details WHERE emp\_lname IN ('Gabriel', 'Dosio');

To list all the customers who have total fixed deposit of amount less than the sum of all their loans

**Select cust\_id,sum(fd\_amount) from fd\_account group by cust\_id having sum(fd\_amount)< (select sum(loan\_amount) from loan\_account where loan\_account.cust\_id = fd\_account.cust\_id group by cust\_id)**

List all the items which are sold

Select \* from item where exists(select \* from lineitem where item.itemcode=lineitem.itemcode)

List all the items which are not been sold

Select \* from item where not exists(select \* from lineitem where item.itemcode=lineitem.itemcode)

**The following SQL statement lists the employees that have registered more than 10 orders:**

Example

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM (Orders INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID) GROUP BY LastName HAVING COUNT(Orders.OrderID) > 10;

**The following SQL statement lists if the employees "Davolio" or "Fuller" have registered more than 25 orders:**

Example

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM Orders INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID WHERE LastName = 'Davolio' OR LastName = 'Fuller'  
GROUP BY LastName HAVING COUNT(Orders.OrderID) > 25;

**IMPORTANT sql::**

STUDENTS (Sno: int, Sname: varchar(64), Gender: ‘F’ or ‘M’, Age: int),  
COURSES (Cno: int, Cname: varchar(32)),  
ENROLLMENT (Sno: int, Cno:int, Grade: int).

1. Find the names of the students who enroll in all the courses.

4. Find the names of the students who enroll more than 3 courses.

1. Find the name and the average grade for each course.

# 3

1. SELECT S.Sname
2. FROM Students AS S INNER JOIN Enrollment AS E ON S.Sno = E.Sno
3. GROUP BY S.Sno, S.Sname
4. HAVING COUNT(\*) = (SELECT COUNT(\*) FROM Courses)

# 4

1. SELECT S.Sname
2. FROM Students AS S INNER JOIN Enrollment AS E ON E.Sno = S.Sno
3. GROUP BY S.Sno, S.Sname
4. HAVING COUNT(\*) > 3

# 5

1. SELECT C.CName, AVG(E.Grade) AS AvgGrade
2. FROM Courses AS C INNER JOIN Enrollment AS E ON C.CNo = E.CNo
3. GROUP BY C.Cno, C.CName