Subquery in SQL

What is Subquery in SQL

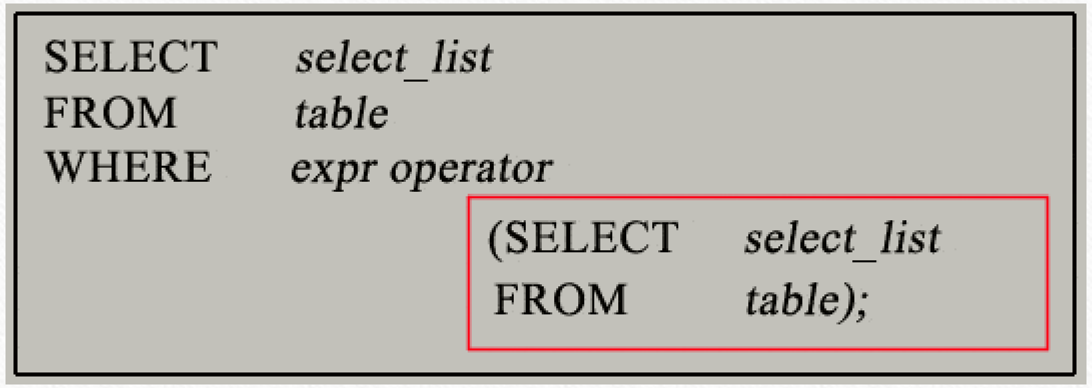
- A subquery is a SQL query within a query.
- Subqueries are nested queries that provide data to the enclosing query.
- Subqueries can return individual values or a list of records
- Subqueries must be enclosed with parenthesis.
- The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.

- You can use the comparison operators, such as >, <, or =.
- The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.

Where to use?

- Compare an expression to the result of the query.
- Determine if an expression is included in the results of the query.
- Check whether the query selects any rows.

Syntax



```
SELECT    select_list
FROM      table
WHERE     expr operator
                        (SELECT   select_list
                         FROM     table);
```

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

Example:

Write an sql query to display the name of student who is getting maximum marks

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

Example:

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

Write an sql query to display the name of student who is getting maximum marks

Select max(marks) from student;

Example:

Write an sql query to display the name of student who is getting maximum marks

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

```
Select name from student
where  marks=( Select max(marks) from student);
```

Example:

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

Write an sql query to display the name of student who is getting second highest marks

```
Select max(marks) from student);
```

Example:

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

Write an sql query to display the name of student who is getting second highest marks

```
Select max(marks) from
student where  marks<>(Select max(marks) from student);
```

Example:

Student id	name	marks
1	Roy	66
2	Joseph	77
3	Shreya	88
4	Roy	95
5	Alex	99

Write an sql query to display the name of student who is getting second highest marks

```
Select name  from student
where marks=(Select max(marks) from
                 student where  marks<>( Select max(marks) from student));
```

Example

- Create table student (student_id varchar(10),name varchar(10));
- Create table marks (student_id varchar(10), tot_marks int);

- Insert into student (student_id,name) values ('V001','Abe'), ('V002','Abhay'), ('V003', 'Acelin'), ('V004', 'Adelphos');
- Insert into marks (student_id,tot_marks) values ('V001',95), ('V002',80), ('V003', 74), ('V004',81);

student_id	name
V001	Abe
V002	Abhay
V003	Acelin
V004	Adelphos

student_id	tot_marks
V001	95
V002	80
V003	74
V004	81

Query:

Identify all students who get better marks than that of the student whose student_id is 'V002', but we do not know the marks of 'V002'.

Solution 1:

- Step 1: Returns the marks (stored in tot_marks field) of 'V002'

```
SELECT *  FROM marks  WHERE student_id = 'V002';
```

Student_id	tot_marks
V002	80

- Step 2: Identifies the students who get better marks than the result of the first query.

```
SELECT a.student_id, a.name, b.tot_marks FROM student a, marks b WHERE a.student_id
= b.student_id AND b.tot_marks >80;
```

student_id	name	tot_marks
V001	Abe	95
V004	Adelphos	81

Query:
Identify all students who get better marks than that of the student whose student_id is 'V002', but we do not know the marks of 'V002'.

Solution 2: With the help of sub query

- SELECT a.student_id, a.name, b.tot_marksFROM student a, marks b

  WHERE a.student_id = b.student_id AND b.tot_marks >

  (SELECT tot_marks FROM marks WHERE student_id = 'V002');

student_id	name	tot_marks
V001	Abe	95
V004	Adelphos	81

Subqueries with INSERT statement

INSERT statement can be used with subqueries

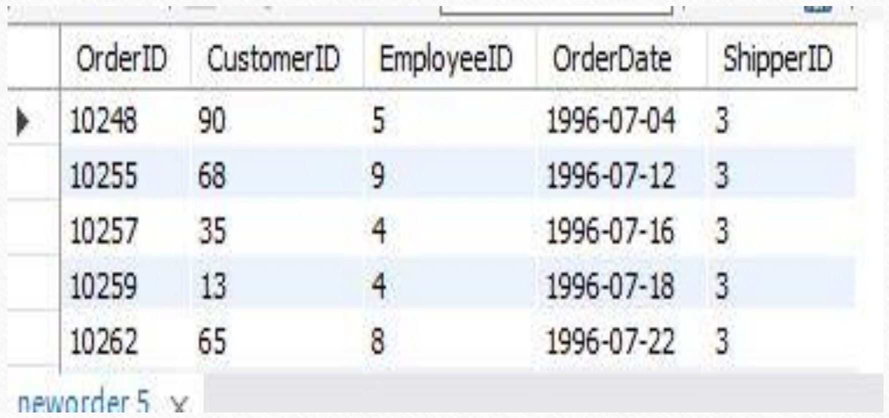
Syntax:

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
SELECT [ *|column1 [, column2 ]
FROM table1 [, table2 ]
[ WHERE VALUE OPERATOR ];
```

Subqueries with INSERT statement Example

- If we want to insert those orders from 'orders' table which have the SHIPPERID from 3 to 5 into 'neworder' table the following SQL can be used:

```
INSERT INTO neworder
SELECT * FROM  orders
WHERE SHIPPERID between 3     and
5;
select * from neworder;
```



	OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
▶	10248	90	5	1996-07-04	3
	10255	68	9	1996-07-12	3
	10257	35	4	1996-07-16	3
	10259	13	4	1996-07-18	3
	10262	65	8	1996-07-22	3

Subqueries with UPDATE statement

In a UPDATE statement, you can set new column value equal to the result returned by a single row subquery.

Syntax:

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
 (SELECT COLUMN_NAME FROM TABLE_NAME) [ WHERE) ]
```

Subqueries with UPDATE statement Example

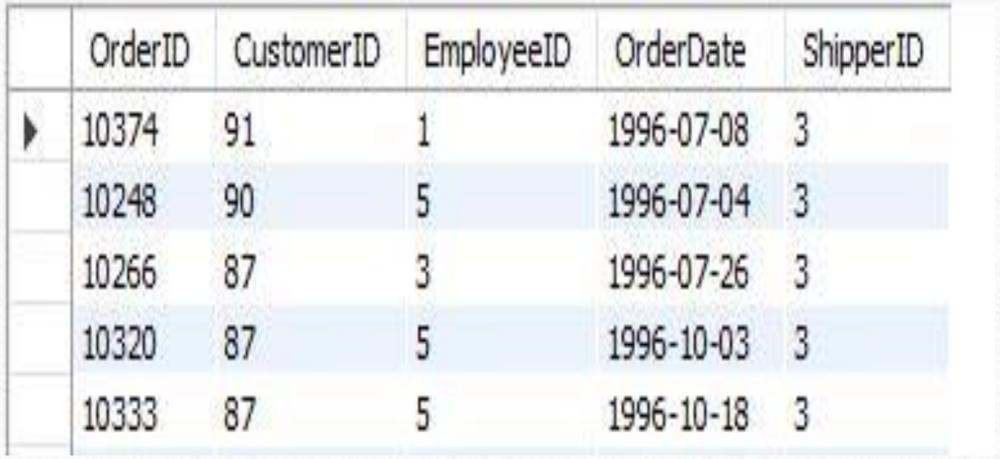
- `SELECT max(CustomerID) FROM orders`

  `Output: 91`

- `UPDATE neworder`

  `SET OrderDate ='1996-07-08’`

  `WHERE CustomerID = (SELECT max(CustomerID) FROM orders);`



	OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
▶	10374	91	1	1996-07-08	3
	10248	90	5	1996-07-04	3
	10266	87	3	1996-07-26	3
	10320	87	5	1996-10-03	3
	10333	87	5	1996-10-18	3

Subqueries with DELETE statement

Syntax:

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
(SELECT COLUMN_NAME
FROM TABLE_NAME) [ WHERE) ]
```

Example:

```
DELETE FROM neworder WHERE
EmployeeID <

(SELECT avg(EmployeeID) FROM
orders);
```

Type of Subqueries

- Single row subquery : Returns zero or one row.
- Multiple row subquery : Returns one or more rows.
- Multiple column subqueries : Returns one or more columns.
- Correlated subqueries : Reference one or more columns in the outer SQL statement. The subquery is known as a correlated subquery because the subquery is related to the outer SQL statement.
- Nested subqueries : Subqueries are placed within another subquery.

Single Row Subqueries

- A single row subquery returns zero or one row to the outer SQL statement.
- You can place a subquery in a WHERE clause, a HAVING clause, or a FROM clause of a SELECT statement.

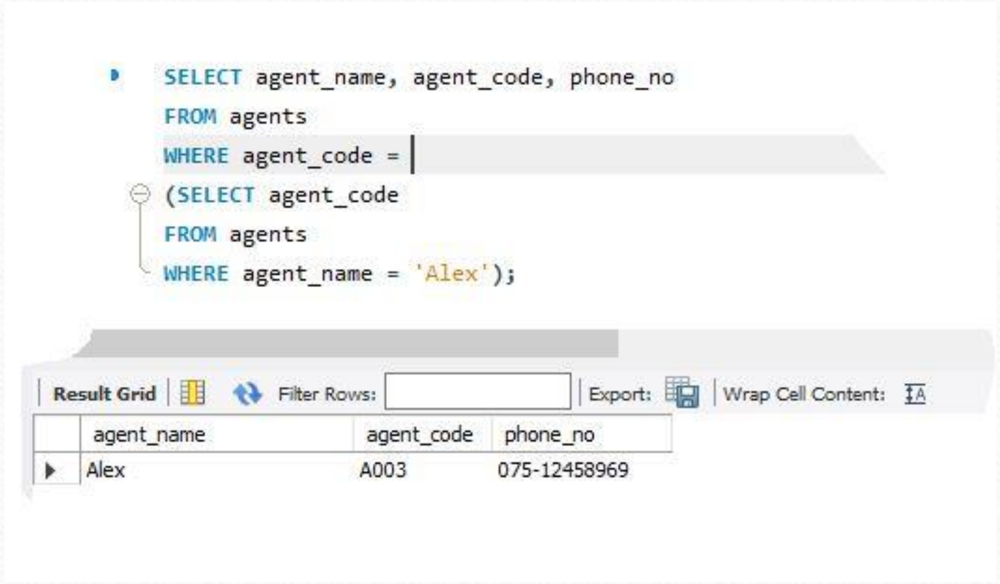
Single row subquery in a WHERE clause

- SELECT agent_name, agent_code, phone_no

  FROM agents

  WHERE agent_code =

    (SELECT agent_code FROM agentsWHERE agent_name = 'Alex');



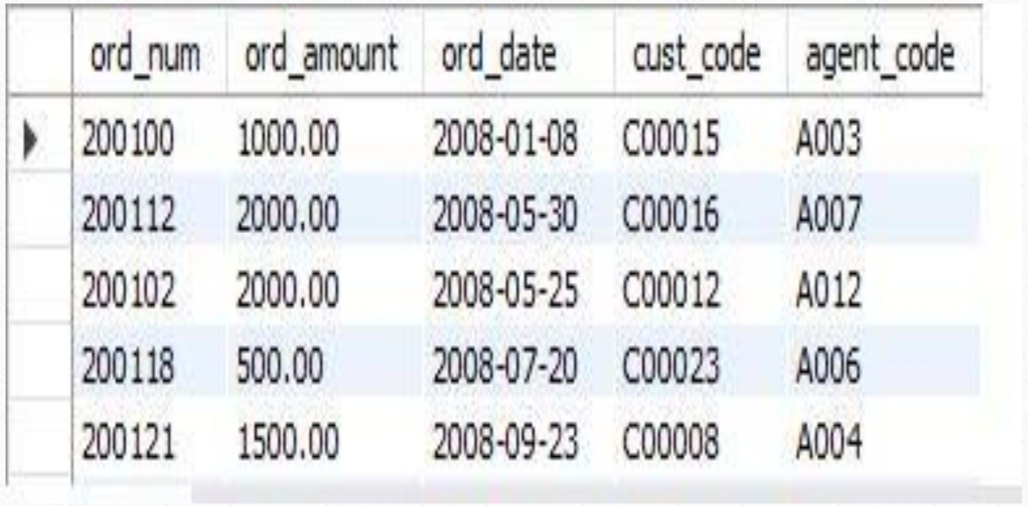
```
SELECT agent_name, agent_code, phone_no
FROM agents
WHERE agent_code =
(SELECT agent_code
FROM agents
WHERE agent_name = 'Alex');
```

```
agent_name    agent_code    phone_no
Alex          A003          075-12458969
```

Using comparison operators in Single Row subquery in where clause

```
SELECT
ord_num,ord_amount,ord_dat
e,cust_code, agent_code
FROM orders

WHERE ord_amount < (SELECT
AVG(ord_amount) FROM
orders);
```



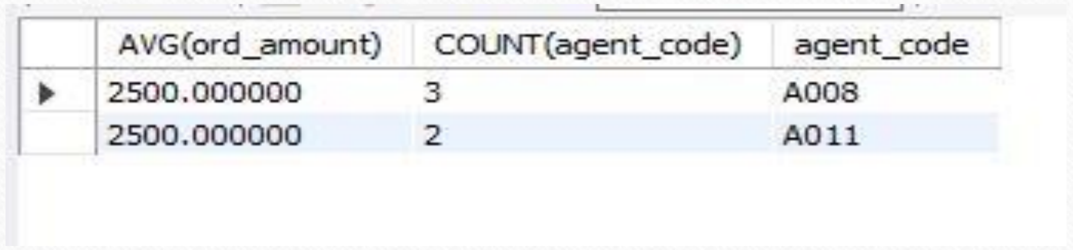
ord_num	ord_amount	ord_date	cust_code	agent_code
200100	1000.00	2008-01-08	C00015	A003
200112	2000.00	2008-05-30	C00016	A007
200102	2000.00	2008-05-25	C00012	A012
200118	500.00	2008-07-20	C00023	A006
200121	1500.00	2008-09-23	C00008	A004

Single row subqueries in a HAVING clause

- HAVING clause is used to filter groups of rows.
- You may place a subquery in HAVING clause in an outer query.
- This allows you to filter groups of rows based on the result returned by your subquery.

Single row subqueries in a HAVING clause example

-
  ```
  SELECT AVG(ord_amount),COUNT(agent_code),agent_code
   FROM orders
   GROUP BY agent_code
   HAVING AVG(ord_amount)= (SELECT  AVG(ord_amount) FROM
   orders WHERE  agent_code='A008');
  ```



AVG(ord_amount)	COUNT(agent_code)	agent_code
2500.000000	3	A008
2500.000000	2	A011

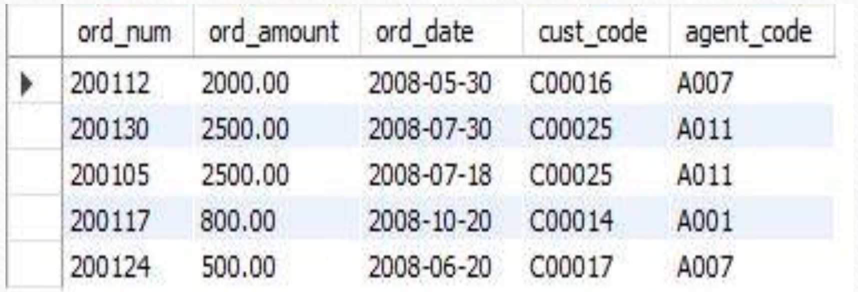
Multiple row subquery

- Multiple row subquery returns one or more rows to the outer SQL statement.
- You may use the IN, ANY, or ALL operator in outer query to handle a subquery that returns multiple rows

Using IN operator with a Multiple Row Subquery

- IN operator is used to checking a value within a set of values.
-
```
SELECT ord_num,ord_amount,ord_date, cust_code,
agent_code

FROM orders WHERE agent_code IN( SELECT agent_code
FROM agents WHERE  working_area='Bangalore');
```



ord_num	ord_amount	ord_date	cust_code	agent_code
200112	2000.00	2008-05-30	C00016	A007
200130	2500.00	2008-07-30	C00025	A011
200105	2500.00	2008-07-18	C00025	A011
200117	800.00	2008-10-20	C00014	A001
200124	500.00	2008-06-20	C00017	A007

Using any operator with a Multiple Row Subquery

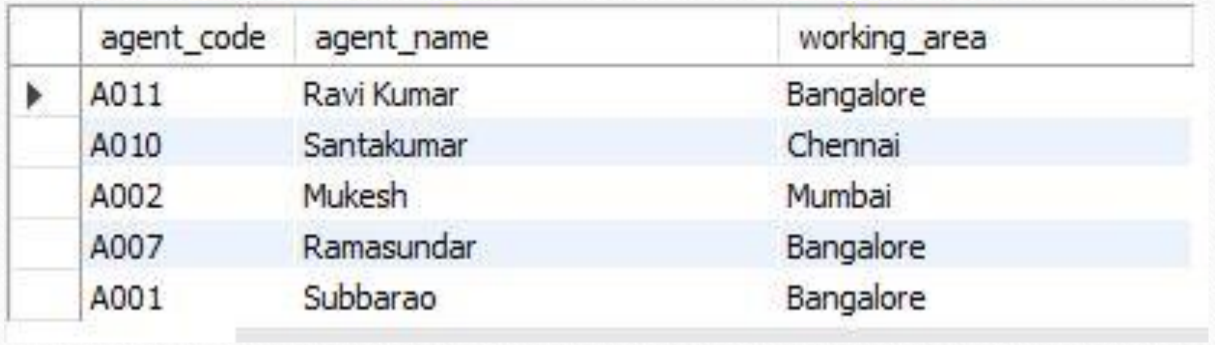
- You can use the ANY operator to compare a value with any value in a list.
- You must place an =, <>, >, <, <= or >= operator before ANY in your query.

Using any operator with a Multiple Row Subquery example

- SELECT agent_code, agent_name, working_area

  FROM agents

  WHERE agent_code= ANY(SELECT agent_code FROM customer WHERE cust_country = 'India');



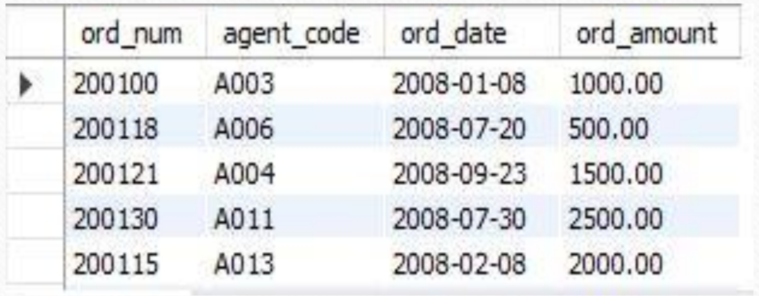
	agent_code	agent_name	working_area
▶	A011	Ravi Kumar	Bangalore
	A010	Santakumar	Chennai
	A002	Mukesh	Mumbai
	A007	Ramasundar	Bangalore
	A001	Subbarao	Bangalore

Multiple Column Subqueries

- select ord_num, agent_code, ord_date, ord_amount

  from orders where (agent_code, ord_amount)

  IN(SELECT agent_code, MIN(ord_amount) FROM orders GROUP BY agent_code);



ord_num	agent_code	ord_date	ord_amount
200100	A003	2008-01-08	1000.00
200118	A006	2008-07-20	500.00
200121	A004	2008-09-23	1500.00
200130	A011	2008-07-30	2500.00
200115	A013	2008-02-08	2000.00

Correlated Subqueries

- Correlated Subqueries are used to select data from a table referenced in the outer query.
- The subquery is known as a correlated because the subquery is related to the outer query.
- In this type of queries, a table alias (also called a correlation name) must be used to specify which table reference is to be used.

Correlated subquery example Contd…

- Display the employee_id, manager_id, first_name and last_name of those employees who manage other employees.

```
SELECT employee_id, manager_id, first_name, last_name

FROM employees a

WHERE EXISTS (SELECT employee_id FROM employees b WHERE
b.manager_id = a.employee_id)
```

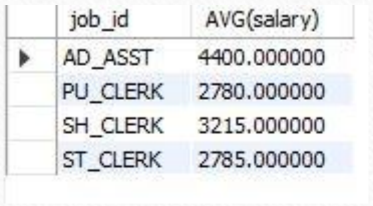
Nested Subqueries

- A subquery can be nested inside other subqueries.
- SQL has an ability to nest queries within one another.
- A subquery is a SELECT statement that is nested within another SELECT statement and which return intermediate results.
- SQL executes innermost subquery first, then next level.

Nested subquery example

- Retrieve job_id and its average salary from the employees table which have a salary is smaller than (averages of min_salary of job_id from the jobs table which job_id are in the list, picking from (the job_history table which is within the department_id 50 and 100)) (use hr database)

-
  ```
  SELECT job_id, AVG(salary)
   FROM employees GROUP BY job_id
   HAVING AVG(salary)<
   (SELECT AVG(min_salary) FROM jobs WHERE job_id IN
   (SELECT job_id FROM job_history WHERE department_id BETWEEN 50 AND 100));
  ```



job_id	AVG(salary)
AD_ASST	4400.000000
PU_CLERK	2780.000000
SH_CLERK	3215.000000
ST_CLERK	2785.000000