

## CUSTOMER TABLE(MASTER/PARENT TABLE):

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane is open, showing the 'AMAZON\_SALES' database. The 'Tables (Filtered)' folder is expanded, and the 'CUSTOMER' table is selected. The main window displays the 'Query Result' for the 'CUSTOMER' table. The query is 'SELECT \* FROM CUSTOMER;'. The result shows 5 rows of customer data.

CUSTOMER_ID	CUSTOMER_NAME	CITY
1	Ravi Kumar	Hyderabad
2	Anjali Sharma	Mumbai
3	Amit Singh	Delhi
4	Priya Verma	Bangalore
5	Rahul Jain	Chennai

## ORDERS TABLE(CHILD TABLE):

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane is open, showing the 'AMAZON\_SALES' database. The 'Tables (Filtered)' folder is expanded, and the 'ORDERS' table is selected. The main window displays the 'Query Result' for the 'ORDERS' table. The query is 'SELECT \* FROM ORDERS;'. The result shows 20 rows of order data.

ORDER_ID	CUSTOMER_ID	ORDER_DATE	ORDER_AMOUNT
1	101	101-01-25	1500.5
2	102	105-01-25	2000
3	103	110-01-25	3500
4	104	207-01-25	900
5	105	212-01-25	1900.75
6	106	215-01-25	2600
7	107	308-01-25	4200
8	108	318-01-25	5800
9	109	322-01-25	1200.5
10	110	402-01-25	300
11	111	405-01-25	750.25
12	112	409-01-25	900
13	113	411-01-25	1340.75
14	114	503-01-25	2500
15	115	506-01-25	4000
16	116	514-01-25	1200.5
17	117	517-01-25	1800
18	118	220-01-25	3900.75
19	119	325-01-25	2800
20	120	130-01-25	5000

-----

-- SIMPLE SUBQUERY (IN WHERE clause)

-- Get customers who placed at least one order

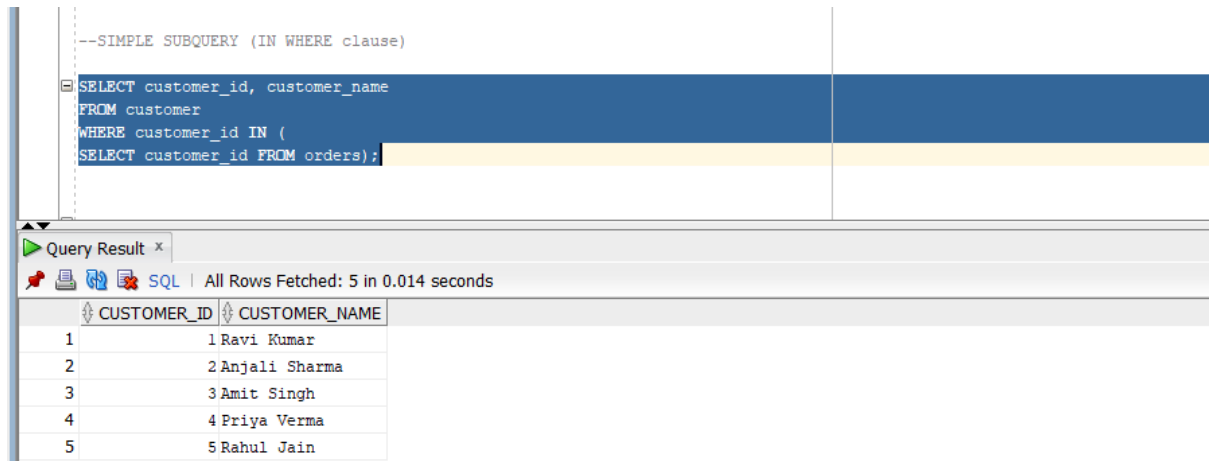
-----

SELECT customer\_id, customer\_name

FROM customer

WHERE customer\_id IN (

```
SELECT customer_id
FROM orders
);
```



The screenshot shows a SQL IDE interface. At the top, a query is written in a text editor: `--SIMPLE SUBQUERY (IN WHERE clause)`, followed by `SELECT customer_id, customer_name`, `FROM customer`, `WHERE customer_id IN (`, and `SELECT customer_id FROM orders);`. Below the editor, a 'Query Result' tab is active, displaying the results of the query. The status bar indicates 'All Rows Fetched: 5 in 0.014 seconds'. The results are shown in a table with two columns: 'CUSTOMER\_ID' and 'CUSTOMER\_NAME'.

CUSTOMER_ID	CUSTOMER_NAME
1	1 Ravi Kumar
2	2 Anjali Sharma
3	3 Amit Singh
4	4 Priya Verma
5	5 Rahul Jain

-----

```
-- NESTED SUBQUERY (Multiple levels)
-- Find customers with order amount greater than avg of all orders
```

-----

```
SELECT customer_id, customer_name
FROM customer
WHERE customer_id IN (
    SELECT customer_id
    FROM orders
    WHERE order_amount > (
        SELECT AVG(order_amount)
        FROM orders
    )
);
```

<pre>--NESTED SUBQUERY SELECT customer_id, customer_name FROM customer WHERE customer_id IN (SELECT customer_id FROM orders WHERE order_amount &gt; (SELECT AVG(order_amount) FROM orders));</pre>	
<div>Query Result x</div> <div>SQL   All Rows Fetched: 4 in 0.017 seconds</div>	
CUSTOMER_ID	CUSTOMER_NAME
1	1 Ravi Kumar
2	2 Anjali Sharma
3	5 Rahul Jain
4	3 Amit Singh

-----

-- SCALAR SUBQUERY (In SELECT)

-- Display each order with total company revenue

-----

SELECT

order\_id,

customer\_id,

order\_amount,

(SELECT SUM(order\_amount) FROM orders) AS total\_sales

FROM orders;

```
--SCALAR SUBQUERY (In SELECT)
SELECT order_id,customer_id,order_amount,
(SELECT SUM(order_amount) FROM orders) AS total_sales
FROM orders;
```

Query Result x

SQL | All Rows Fetched: 20 in 0.009 seconds

	ORDER_ID	CUSTOMER_ID	ORDER_AMOUNT	TOTAL_SALES
1	101	1	1500.5	47994
2	102	1	2000	47994
3	103	1	3500	47994
4	104	2	800	47994
5	105	2	1900.75	47994
6	106	2	2600	47994
7	107	3	4200	47994
8	108	3	5800	47994
9	109	3	1200.5	47994
10	110	4	300	47994
11	111	4	750.25	47994
12	112	4	900	47994
13	113	4	1340.75	47994
14	114	5	2500	47994
15	115	5	4000	47994
16	116	5	1200.5	47994
17	117	5	1800	47994
18	118	2	3900.75	47994
19	119	3	2800	47994
20	120	1	5000	47994

-----

-----

<pre>--INLINE VIEW SELECT * FROM (SELECT order_id, customer_id, order_amount FROM orders ORDER BY order_amount DESC ) top_orders WHERE ROWNUM &lt;= 5;</pre>			
<div> <div>Query Result x</div> <div> <div>SQL</div> <div>All Rows Fetched: 5 in 0.008 seconds</div> </div> </div>			
ORDER_ID	CUSTOMER_ID	ORDER_AMOUNT	
1	108	3	5800
2	120	1	5000
3	107	3	4200
4	115	5	4000
5	118	2	3900.75

-----

--CORRELATED SUBQUERY (Depends on outer query)

-- Find orders greater than customer's average order amount

-----

```
SELECT
    o.order_id,
    o.customer_id,
    o.order_amount
FROM
    orders o
WHERE
    o.order_amount > (
        SELECT AVG(order_amount)
        FROM orders
        WHERE customer_id = o.customer_id
    );
```

The screenshot shows the SQL Developer interface. On the left, the 'Tables (Filtered)' tree is expanded, showing 'AMAZON\_SALES\_REPORT', 'CUSTOMER', and 'ORDERS'. The main window displays a SQL query using a correlated subquery to find orders with amounts greater than the average for each customer.

```
--CORRELATED SUBQUERY
SELECT o.order_id,o.customer_id,o.order_amount
FROM orders o
WHERE o.order_amount > (
    SELECT AVG(order_amount)
    FROM orders
    WHERE customer_id = o.customer_id);
```

The 'Query Result' window shows 10 rows fetched in 0.025 seconds. The results are as follows:

ORDER_ID	CUSTOMER_ID	ORDER_AMOUNT
1	103	3500
2	106	2600
3	107	4200
4	108	5800
5	112	900
6	113	1340.75
7	114	2500
8	115	4000
9	118	3900.75
10	120	5000

The screenshot shows the SQL Developer interface. On the left, the 'Tables (Filtered)' tree is expanded, showing 'AMAZON\_SALES\_REPORT', 'CUSTOMER', and 'ORDERS'. The main window displays a SQL query using the EXISTS operator to find customers who have at least one order.

```
--USING EXISTS
SELECT customer_id, customer_name
FROM customer c
WHERE EXISTS (
    SELECT 1
    FROM orders o
    WHERE o.customer_id = c.customer_id
);
```

The 'Query Result' window shows 5 rows fetched in 0.008 seconds. The results are as follows:

CUSTOMER_ID	CUSTOMER_NAME
1	1 Ravi Kumar
2	2 Anjali Sharma
3	3 Amit Singh
4	4 Priya Verma
5	5 Rahul Jain

## USING EXISTS

-- Show customer names who have at least one order

SELECT customer\_id, customer\_name

FROM customer c

WHERE EXISTS (

SELECT 1

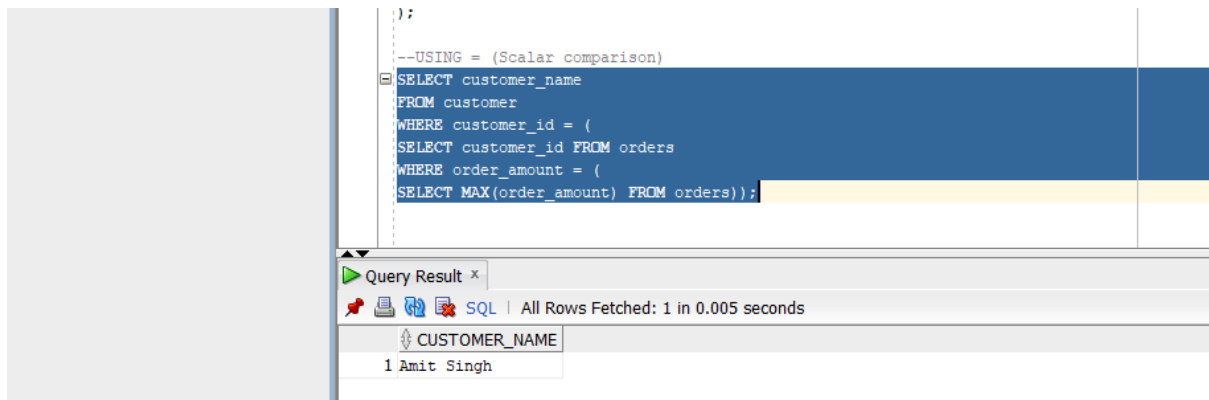
FROM orders o

WHERE o.customer\_id = c.customer\_id

);

-----  
-- USING = (Scalar comparison)  
-- Show customer having maximum order amount  
-----

```
SELECT customer_name
FROM customer
WHERE customer_id = (
    SELECT customer_id
    FROM orders
    WHERE order_amount = (
        SELECT MAX(order_amount) FROM orders
    )
);
```



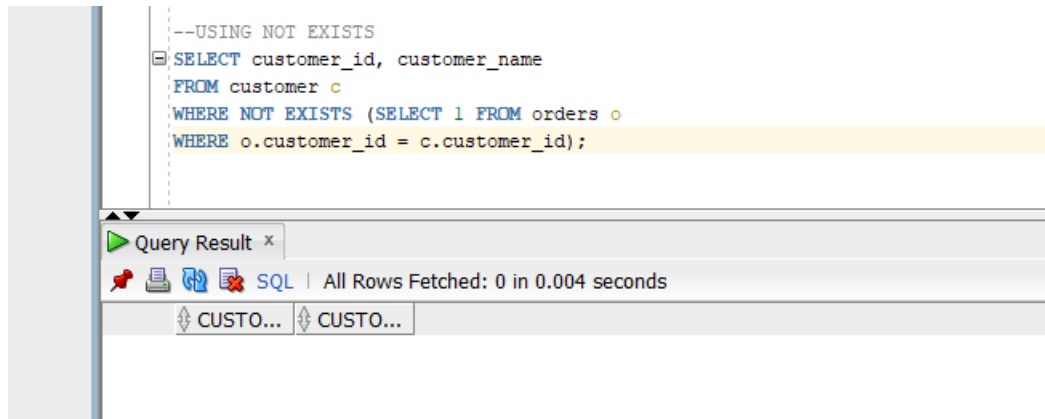
-----  
-- USING NOT EXISTS  
-- Customers who have NOT placed any orders  
-----

```
SELECT customer_id, customer_name
```

```

FROM customer c
WHERE NOT EXISTS (
    SELECT 1
    FROM orders o
    WHERE o.customer_id = c.customer_id
);

```



-----

--INLINE VIEW + GROUPING

-- Top customer by total purchase using inline view

-----

```

SELECT * FROM (
    SELECT
        customer_id,
        SUM(order_amount) AS total_revenue
    FROM orders
    GROUP BY customer_id
    ORDER BY total_revenue DESC
) revenue_table
WHERE ROWNUM = 1;

```



Connections

- Oracle Connections
  - TNS
    - AMAZON\_SALES
      - Tables (Filtered)
        - AMAZON\_SALES\_REPORT
        - CUSTOMER
        - ORDERS
      - Views
      - Indexes
      - Packages
      - Procedures
      - Functions
      - Operators
      - Queues
      - Queues Tables
      - Triggers
      - Types
      - Sequences

Worksheet Query Builder

```
WHERE c.customer_id = c.customer_id);

--INLINE VIEW + GROUPING
SELECT * FROM (SELECT customer_id,
SUM(order_amount) AS total_revenue
FROM orders
GROUP BY customer_id
ORDER BY total_revenue DESC
) revenue_table
WHERE ROWNUM = 1;
```

Query Result x

SQL | All Rows Fetched: 1 in 0.012 seconds

CUSTOMER_ID	TOTAL_REVENUE
1	14000.5