

CUSTOMER TABLE(MASTER/PARENT TABLE):

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows a connection to 'AMAZON_SALES'. In the center, the Worksheet pane contains the query: 'SELECT * FROM CUSTOMER;'. Below it, the Query Result pane displays the following data:

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

ORDERS TABLE(CHILD TABLE):

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows a connection to 'AMAZON_SALES'. In the center, the Worksheet pane contains the queries: 'SELECT * FROM CUSTOMER;' and 'SELECT * FROM ORDERS;'. Below it, the Query Result pane displays the following data:

	ORDER_ID	CUSTOMER_ID	ORDER_DATE	ORDER_AMOUNT
1	101	1	01-01-25	1500.5
2	102	1	05-01-25	2000
3	103	1	10-01-25	3500
4	104	1	20-01-25	800
5	105	2	12-01-25	1900.75
6	106	2	15-01-25	2600
7	107	2	08-01-25	4200
8	108	2	18-01-25	5800
9	109	2	22-01-25	1200.5
10	110	2	02-01-25	300
11	111	2	05-01-25	750.25
12	112	2	09-01-25	900
13	113	2	11-01-25	1340.75
14	114	2	03-01-25	2500
15	115	2	06-01-25	4000
16	116	2	14-01-25	1200.5
17	117	2	17-01-25	1800
18	118	2	20-01-25	3900.75
19	119	2	25-01-25	2800
20	120	2	30-01-25	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 MySQL Workbench interface. The SQL editor pane contains the following code:

```
--SIMPLE SUBQUERY (IN WHERE clause)  
SELECT customer_id, customer_name  
FROM customer  
WHERE customer_id IN (  
    SELECT customer_id FROM orders);
```

The results pane, titled "Query Result", displays the output of the query:

CUSTOMER_ID	CUSTOMER_NAME
1	Ravi Kumar
2	Anjali Sharma
3	Amit Singh
4	Priya Verma
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  
    )  
);
```

```
--NESTED SUBQUERY
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));
```

Query Result | SQL | All Rows Fetched: 4 in 0.017 seconds

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 | All Rows Fetched: 20 in 0.009 seconds

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

-- 4 ONLINE VIEW (Subquery in FROM clause)

-- Show top 5 highest order amounts only

```
SELECT *
FROM (
    SELECT order_id, customer_id, order_amount
    FROM orders
    ORDER BY order_amount DESC
) top_orders
WHERE ROWNUM <= 5;
```

--INLINE VIEW

```

SELECT *
FROM (SELECT order_id, customer_id, order_amount
      FROM orders
     ORDER BY order_amount DESC
     ) top_orders
   WHERE ROWNUM <= 5;
  
```

Query Result

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
);
  
```

```
--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);
```

Query Result

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

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

Query Result

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

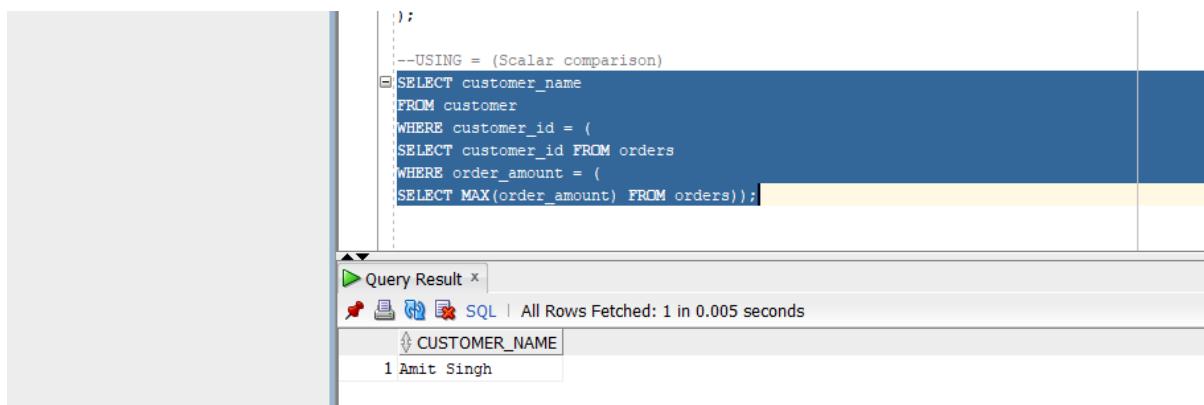
FROM orders o

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
    )
);
```



The screenshot shows the Oracle SQL Developer interface. The top pane displays the SQL code, which includes a scalar subquery using the equality operator (=). The bottom pane shows the 'Query Result' window with one row of data: 'Amit Singh'.

```
;) ;
--USING = (Scalar comparison)
SELECT customer_name
FROM customer
WHERE customer_id = (
    SELECT customer_id
    FROM orders
    WHERE order_amount = (
        SELECT MAX(order_amount) FROM orders));
```

CUSTOMER_NAME
Amit Singh

```
-- 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
);

```

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a code editor with the following SQL query:

```

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

```

In the bottom pane, there is a "Query Result" window. The title bar says "Query Result x". Below it, there are icons for Refresh, Print, and Stop, followed by "SQL | All Rows Fetched: 0 in 0.004 seconds". The result grid has two columns labeled "CUSTO..." and "CUSTO...".

--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;

```

Screenshot of Oracle SQL Developer interface showing a connection to the 'AMAZON_SALES' database.

Connections pane:

- Oracle Connections
- TNS
- AMAZON_SALES
 - Tables (Filtered)
 - AMAZON_SALES_REPORT
 - CUSTOMER
 - ORDERS
 - Views
 - Indexes
 - Packages
 - Functions
 - Operators
 - Queues
 - Queues Tables
 - Triggers
 - Types
 - Sequences

Worksheet pane:

```
WHERE o.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 pane:

CUSTOMER_ID	TOTAL_REVENUE	
1	3	14000.5

All Rows Fetched: 1 in 0.012 seconds