

Ecommerce – SQL

create database SQLAssignment;

use SQLAssignment;

Customer's Table:

```
create table customers(customer_id int primary key,first_name  
varchar(60),last_name varchar(60),email varchar(60),address  
varchar(60),password varchar(60));
```

```
insert into customers values(1,"John","Doe","johndoe@example.com","123  
mainstret","1234");
```

```
insert into customers values (2, "Jane", "Smith", "janesmith@example.com", "456  
Elm St, Town", 2345);
```

```
insert into customers values (3, "Robert", "Johnson", "robert@example.com",  
"789 Oak St, Village", 3456);
```

```
insert into customers values (4, "Sarah", "Brown", "sarah@example.com", "101  
Pine St, Suburb", 4567);
```

```
insert into customers values (5, "David", "Lee", "david@example.com", "234  
Cedar St, District", 5678);
```

```
insert into customers values (6, "Laura", "Hall", "laura@example.com", "567 Birch  
St, County", 6789);
```

```
insert into customers values (7, "Michael", "Davis", "michael@example.com",  
"890 Maple St, State", 7890);
```

```
insert into customers values (8, "Emma", "Wilson", "emma@example.com", "321  
Redwood St, Country", 8901);
```

```
insert into customers values (9, "William", "Taylor", "william@example.com", "432  
Spruce St, Province", 9012);
```

```
insert into customers values (10, "Olivia", "Adams", "olivia@example.com", "765  
Fir St, Territory", 1235);
```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
90 • select * from product_table where product_id not in (select product_id from cart);
91
92 • select * from customers where customer_id not in (select distinct customer_id from orders);
93
94 • select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by
95
96 • select * from product_table where stock_quan < 10;
97
98 • select * from customers where customer_id in (select customer_id from orders where total_price > 1000);
99
100 • select * from customers;
```

Results Grid:

customer_id	first_name	last_name	email	address	password
1	John	Doe	johndoe@example.com	123 mainstret	1234
2	Jane	Smith	janesmith@example.com	456 Elm St, Town	2345
3	Robert	Johnson	robert@example.com	789 Oak St, Village	3456
4	Sarah	Brown	sarah@example.com	101 Pine St, Suburb	4567
5	David	Lee	david@example.com	234 Cedar St, District	5678
6	Laura	Hall	laura@example.com	567 Birch St, County	6789
7	Michael	Davis	michael@example.com	890 Maple St, State	7890
8	Emma	Wilson	emma@example.com	321 Redwood St, Country	8901
9	William	Taylor	william@example.com	432 Spruce St, Province	9012
10	Olivia	Adams	olivia@example.com	765 Fir St, Territory	1235

Product Table:

```
create table product_table(product_id int primary key,prod_name  
varchar(60),price decimal(10,2),description varchar(60),stock_quan int);  
insert into product_table values (1, "Laptop", 800.00, "High-performance laptop",  
10);  
insert into product_table values (2, "Smartphone", 600.00, "Latest smartphone",  
15);  
insert into product_table values (3, "Tablet", 300.00, "Portable tablet", 20);  
insert into product_table values (4, "Headphones", 150.00, "Noise-canceling",  
30);  
insert into product_table values (5, "TV", 900.00, "4K Smart TV", 5);  
insert into product_table values (6, "Coffee Maker", 50.00, "Automatic coffee  
maker", 25);  
insert into product_table values (7, "Refrigerator", 700.00, "Energy-efficient", 10);  
insert into product_table values (8, "Microwave Oven", 80.00, "Countertop  
microwave", 15);  
insert into product_table values (9, "Blender", 70.00, "High-speed blender", 20);  
insert into product_table values (10, "Vacuum Cleaner", 120.00, "Bagless  
vacuum cleaner", 10);
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following SQL code:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101
```

The results grid displays the data from the product_table:

product_id	prod_name	price	description	stock_quan
1	Laptop	800.00	High-performance laptop	10
2	Smartphone	600.00	Latest smartphone	15
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-canceling	30
5	TV	900.00	4K Smart TV	5
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	700.00	Energy-efficient	10
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20
10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL

Orders Table:

```
create table orders (order_id int primary key, customer_id int, order_date date, total_price decimal(10,2), foreign key (customer_id) references customers(customer_id));
```

```
insert into orders values (1, 1, '2023-01-05', 1200.00);
```

```
insert into orders values (2, 2, '2023-02-10', 900.00);
```

```
insert into orders values (3, 3, '2023-03-15', 300.00);
```

```
insert into orders values (4, 4, '2023-04-20', 150.00);
```

```
insert into orders values (5, 5, '2023-05-25', 1800.00);
```

```
insert into orders values (6, 6, '2023-06-30', 400.00);
```

```
insert into orders values (7, 7, '2023-07-05', 700.00);
```

```
insert into orders values (8, 8, '2023-08-10', 160.00);
```

```
insert into orders values (9, 9, '2023-09-15', 140.00);
```

```
insert into orders values (10, 10, '2023-10-20', 1400.00);
```

The screenshot shows the MySQL Workbench interface. The top part displays a SQL query in the Query 1 tab:

```
93 •   select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by
94 •
95 •   select * from product_table where stock_quan < 10;
96 •
97 •   select * from customers where customer_id in (select customer_id from orders where total_price > 1000);
98 •
99 •   select * from customers;
100 •   select * from product_table;
101 •   select * from order_items;
102 •   select * from orders;
```

The bottom part shows the Result Grid with the following data:

order_id	customer_id	order_date	total_price
1	1	2023-01-05	1200.00
2	2	2023-02-10	900.00
3	3	2023-03-15	300.00
4	4	2023-04-20	150.00
5	5	2023-05-25	1800.00
6	6	2023-06-30	400.00
7	7	2023-07-05	700.00
8	8	2023-08-10	160.00
9	9	2023-09-15	140.00
10	10	2023-10-20	1400.00
NULL	NULL	NULL	NULL

Order Items Table:

```
create table order_items (order_item_id int primary key,order_id int,product_id int,quantity int,item_amount decimal(10,2),foreign key (order_id) references orders(order_id),foreign key (product_id) references product_table(product_id));  
insert into order_items values (1, 1, 1, 2, 1600.00);  
insert into order_items values (2, 1, 3, 1, 300.00);  
insert into order_items values (3, 2, 2, 3, 1800.00);  
insert into order_items values (4, 3, 5, 2, 1800.00);  
insert into order_items values (5, 4, 4, 4, 600.00);  
insert into order_items values (6, 4, 6, 1, 50.00);  
insert into order_items values (7, 5, 1, 1, 800.00);  
insert into order_items values (8, 5, 2, 2, 1200.00);  
insert into order_items values (9, 6, 10, 2, 240.00);  
insert into order_items values (10, 6, 9, 3, 210.00);
```

The screenshot shows the MySQL Workbench interface. The top half displays a query editor window with several SQL statements. The bottom half shows a results grid for the 'order_items' table.

Query Editor (Query 1):

```
92 • select * from customers where customer_id not in (select distinct customer_id from orders);  
93  
94 • select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by  
95  
96 • select * from product_table where stock_quan < 10;  
97  
98 • select * from customers where customer_id in (select customer_id from orders where total_price > 1000);  
99  
100 • select * from customers;  
101 • select * from product_table;  
102 • select * from order_items;  
103
```

Results Grid:

order_item_id	order_id	product_id	quantity	item_amount
1	1	1	2	1600.00
2	1	3	1	300.00
3	2	3	3	1800.00
4	3	2	2	1800.00
5	4	4	4	600.00
6	4	6	1	50.00
7	5	1	1	800.00
8	5	2	2	1200.00
9	6	10	2	240.00
10	6	9	3	210.00

Cart Table:

```
create table cart (cart_id int primary key, customer_id int, product_id int, quantity int, foreign key (customer_id) references customers(customer_id), foreign key (product_id) references product_table(product_id));
```

```
insert into cart values (1, 1, 1, 2);
insert into cart values (2, 1, 3, 1);
insert into cart values (3, 2, 2, 3);
insert into cart values (4, 3, 4, 4);
insert into cart values (5, 3, 5, 2);
insert into cart values (6, 4, 6, 1);
insert into cart values (7, 5, 1, 1);
insert into cart values (8, 6, 10, 2);
insert into cart values (9, 6, 9, 3);
insert into cart values (10, 7, 7, 2);
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Menu:** new, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Schemas pane showing databases: gamingzone, paypert, sys.
- Query Editor:** SQL tab containing the creation script for the Cart table and its data insertion statements.
- Result Grid:** Shows the data inserted into the Cart table, with 10 rows displayed.
- Object Info:** Shows the Cart table's structure with columns: cart_id, customer_id, product_id, and quantity.
- System Tray:** Shows system icons like Search, Task View, File Explorer, Control Panel, Taskbar, and a status bar indicating ENG US, 12:34 PM, and 25-Jun-25.

cart_id	customer_id	product_id	quantity
1	1	1	2
2	1	3	1
3	2	2	3
4	3	4	4
5	3	5	2
6	4	6	1
7	5	1	1
8	6	10	2
9	6	9	3
10	7	7	2

1.Update refrigerator product price to 800.

->update product_table set price = 800 where product_id = 7;

The screenshot shows the MySQL Workbench interface. In the top pane, there is a SQL editor window containing the following code:

```
54 • insert into cart values (7, 5, 1, 1);
55 • insert into cart values (8, 6, 10, 2);
56 • insert into cart values (9, 6, 9, 3);
57 • insert into cart values (10, 7, 7, 2);
58
59 • update product_table set price = 800 where product_id = 7;
60
61 • select * from product_table;
62
63 • delete from cart where cart_id in (select cart_id from cart where customer_id = 5);
64 • select * from cart;
65
```

Below the SQL editor is a Result Grid showing the contents of the product_table:

product_id	prod_name	price	description	stock_quan
1	Laptop	800.00	High-performance laptop	10
2	Smartphone	600.00	Latest smartphone	15
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-cancelling	30
5	TV	900.00	4K Smart TV	5
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	800.00	Energy-efficient	10
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20
10	Vacuum Cleaner	120.00	Eggless vacuum cleaner	10

The bottom pane shows the cart table:

cart_id	customer_id	product_id	quantity
1	1	2	
2	1	3	1
3	2	2	3
4	3	4	4
5	3	5	2
6	4	6	1
7	5	1	1
8	6	10	2
9	6	9	3
10	7	7	2

2.Remove all cart items for a specific customer

->delete from cart where cart_id in (select cart_id from cart where customer_id = 5);

The screenshot shows the MySQL Workbench interface. In the top pane, there is a SQL editor window containing the following code:

```
56 • insert into cart values (9, 6, 9, 3);
57 • insert into cart values (10, 7, 7, 2);
58
59 • update product_table set price = 800 where product_id = 7;
60
61 • select * from product_table;
62
63 • delete from cart where cart_id in (select cart_id from cart where customer_id = 5);
64 • select * from cart;
65
66 • select * from product_table where price < 100;
67
```

Below the SQL editor is a Result Grid showing the contents of the cart table after the delete operation:

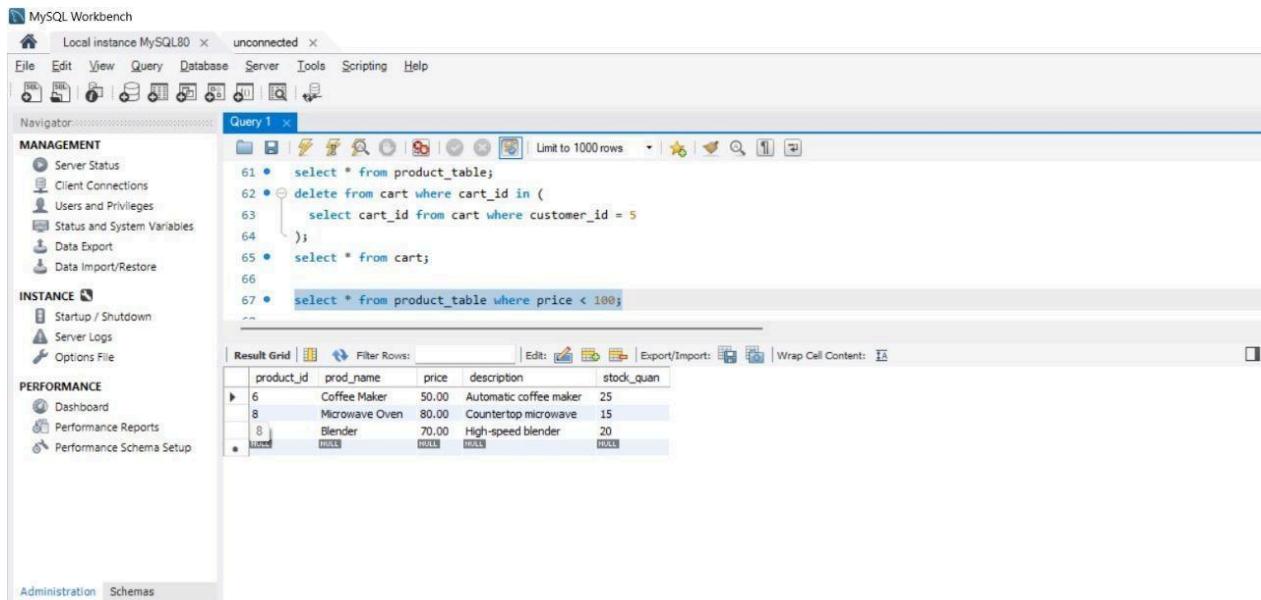
cart_id	customer_id	product_id	quantity
1	1	2	
2	1	3	1
3	2	2	3
4	3	4	4
5	3	5	2
6	4	6	1
7	5	1	1
8	6	10	2
9	6	9	3
10	7	7	2

The bottom pane shows the product_table:

product_id	prod_name	price	description	stock_quan
1	Laptop	800.00	High-performance laptop	10
2	Smartphone	600.00	Latest smartphone	15
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-cancelling	30
5	TV	900.00	4K Smart TV	5
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	800.00	Energy-efficient	10
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20
10	Vacuum Cleaner	120.00	Eggless vacuum cleaner	10

3.Retrieve Products Priced Below \$100.

->select * from product_table where price < 100;



The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

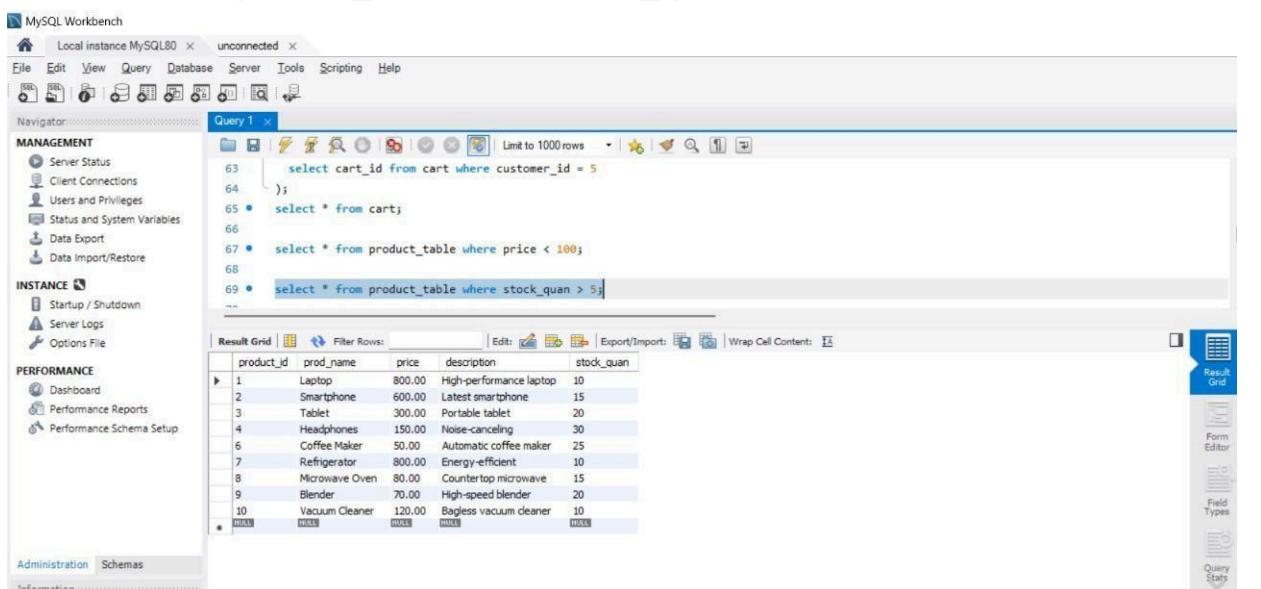
```
61 •    select * from product_table;
62 •    delete from cart where cart_id in (
63     select cart_id from cart where customer_id = 5
64 );
65 •    select * from cart;
66
67 •    select * from product_table where price < 100;
```

The result grid displays the following data:

product_id	prod_name	price	description	stock_quan
6	Coffee Maker	50.00	Automatic coffee maker	25
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20
10	HULL	HULL	HULL	HULL

4.Find Products with Stock Quantity Greater Than 5.

->select * from product_table where stock_quan > 5;



The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

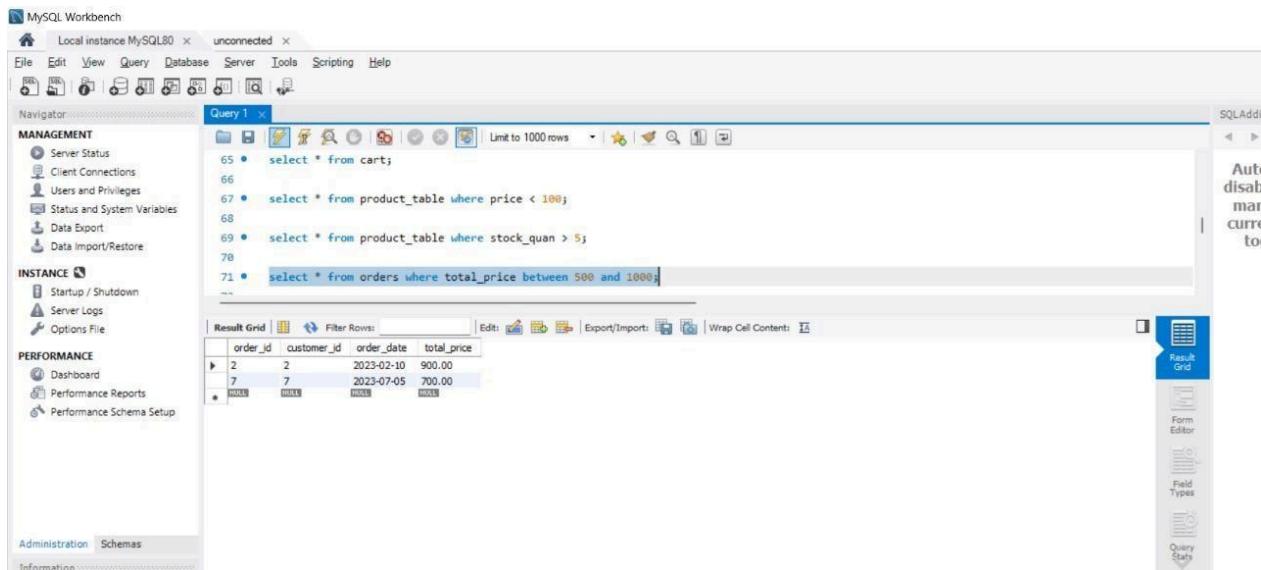
```
63 |    select cart_id from cart where customer_id = 5
64 |
65 •    select * from cart;
66
67 •    select * from product_table where price < 100;
68
69 •    select * from product_table where stock_quan > 5;
```

The result grid displays the following data:

product_id	prod_name	price	description	stock_quan
1	Laptop	800.00	High-performance laptop	10
2	Smartphone	600.00	Latest smartphone	15
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-cancelling	30
5	Coffee Maker	50.00	Automatic coffee maker	25
6	Refrigerator	800.00	Energy-efficient	10
7	Microwave Oven	80.00	Countertop microwave	15
8	Blender	70.00	High-speed blender	20
9	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10
10	HULL	HULL	HULL	HULL

5) Retrieve Orders with Total Amount Between \$500 and \$1000.

->select * from orders where total_price between 500 and 1000;



The screenshot shows the MySQL Workbench interface with the following details:

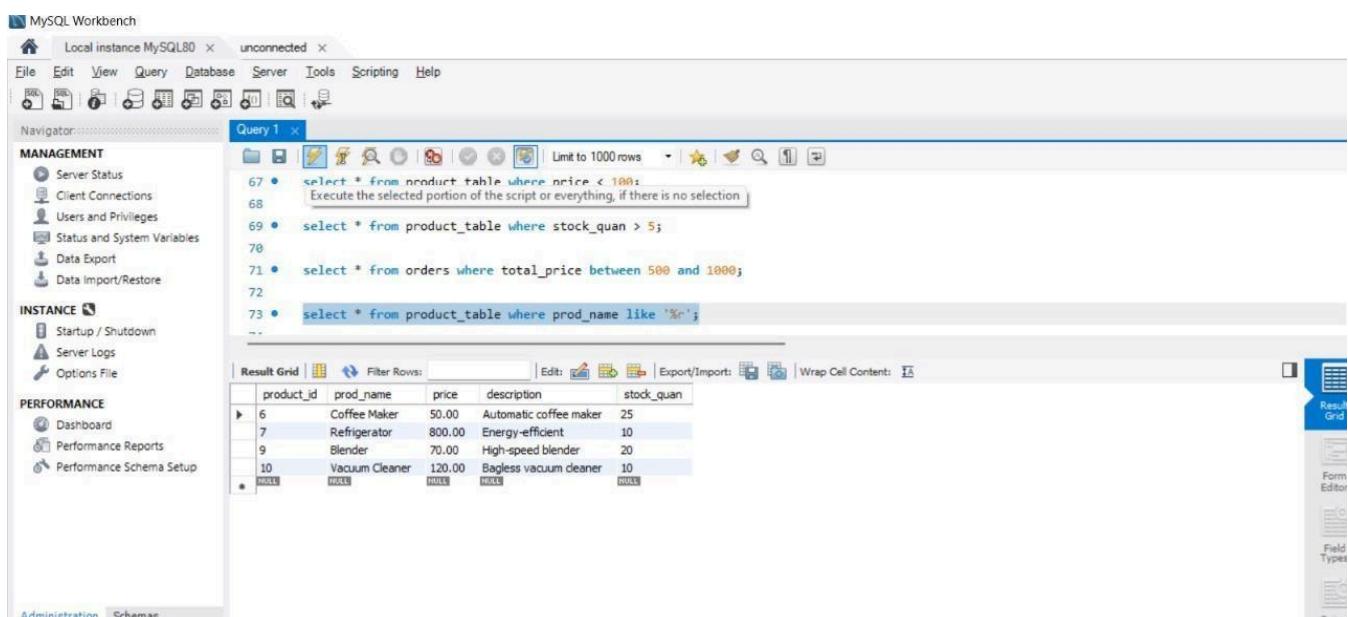
- Navigator:** Shows various database management options like Server Status, Client Connections, and Performance Reports.
- Query Editor:** Displays the following SQL query:

```
65 • select * from cart;
66
67 • select * from product_table where price < 100;
68
69 • select * from product_table where stock_quan > 5;
70
71 • select * from orders where total_price between 500 and 1000;
--
```
- Result Grid:** Shows the results of the last query, which are two rows of data from the 'orders' table:

order_id	customer_id	order_date	total_price
2	2	2023-02-10	900.00
7	7	2023-07-05	700.00
- Right Panel:** Contains icons for Result Grid, Form Editor, Field Types, and Query Stats.

6) Find Products whose name ends with the letter 'r'.

->select * from product_table where prod_name like '%r';



The screenshot shows the MySQL Workbench interface with the following details:

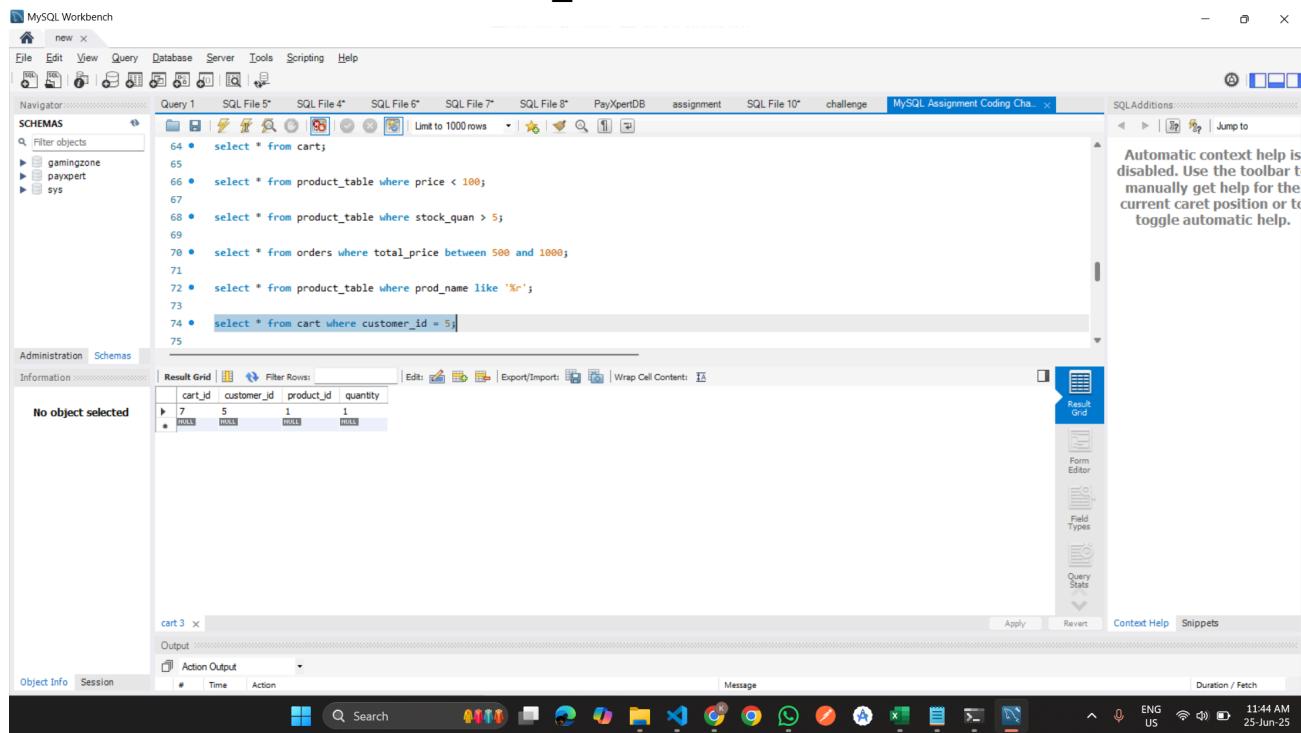
- Navigator:** Shows various database management options like Server Status, Client Connections, and Performance Reports.
- Query Editor:** Displays the following SQL query:

```
67 • select * from product_table where price < 100;
Execute the selected portion of the script or everything, if there is no selection.
68
69 • select * from product_table where stock_quan > 5;
70
71 • select * from orders where total_price between 500 and 1000;
72
73 • select * from product_table where prod_name like '%r';
--
```
- Result Grid:** Shows the results of the last query, which are four rows of data from the 'product_table' table:

product_id	prod_name	price	description	stock_quan
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	800.00	Energy-efficient	10
9	Blender	70.00	High-speed blender	20
10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10
- Right Panel:** Contains icons for Result Grid, Form Editor, Field Types, and Query Stats.

7) Retrieve Cart Items for Customer 5.

->select * from cart where customer_id = 5;



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

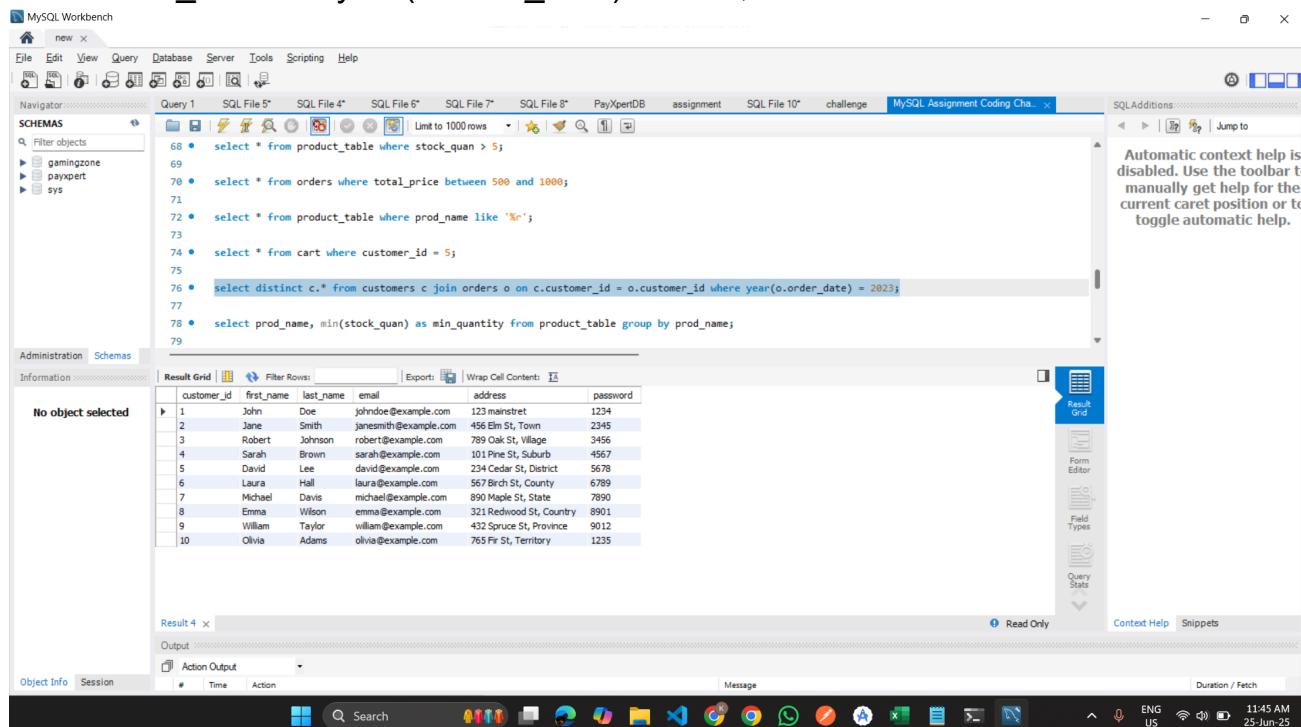
```
64 • select * from cart;
65
66 • select * from product_table where price < 100;
67
68 • select * from product_table where stock_quan > 5;
69
70 • select * from orders where total_price between 500 and 1000;
71
72 • select * from product_table where prod_name like '%r';
73
74 • select * from cart where customer_id = 5;
75
```

The results pane displays a single row of data from the 'cart' table:

cart_id	customer_id	product_id	quantity
7	5	1	1

8) Find Customers Who Placed Orders in 2023.

->select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
68 • select * from product_table where stock_quan > 5;
69
70 • select * from orders where total_price between 500 and 1000;
71
72 • select * from product_table where prod_name like '%r';
73
74 • select * from cart where customer_id = 5;
75
76 • select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;
77
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
```

The results pane displays 10 rows of data from the 'customers' table:

customer_id	first_name	last_name	email	address	password
1	John	Doe	john doe@example.com	123 mainstreet	1234
2	Jane	Smith	jane smith@example.com	456 Elm St, Town	2345
3	Robert	Johnson	robert@ example.com	789 Oak St, Village	3456
4	Sarah	Brown	sarah@example.com	101 Pine St, Suburb	4567
5	David	Lee	david@example.com	234 Cedar St, District	5678
6	Laura	Hall	laura@example.com	567 Birch St, County	6789
7	Michael	Davis	michael@example.com	890 Maple St, State	7890
8	Emma	Wilson	emma@example.com	321 Redwood St, Country	8901
9	William	Taylor	william@example.com	432 Spruce St, Province	9012
10	Olivia	Adams	olivia@example.com	765 Fir St, Territory	1235

9)Determine the Minimum Stock Quantity for Each Product Category.

->select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
68 • select * from product_table where stock_quan > 5;
69
70 • select * from orders where total_price between 500 and 1000;
71
72 • select * from product_table where prod_name like '%r';
73
74 • select * from cart where customer_id = 5;
75
76 • select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;
77
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
```

The results grid displays the following data:

prod_name	min_quantity
Laptop	10
Smartphone	15
Tablet	20
Headphones	30
TV	5
Coffee Maker	25
Refrigerator	10
Microwave Oven	15
Blender	20
Vacuum Cleaner	10

10)Calculate the Total Amount Spent by Each Customer.

->select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
71
72 • select * from product_table where prod_name like '%n';
73
74 • select * from cart where customer_id = 5;
75
76 • select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;
77
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
80 • select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
81
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
```

The results grid displays the following data:

customer_id	first_name	total_spent
1	John	1200.00
2	Jane	900.00
3	Robert	300.00
4	Sarah	150.00
5	David	1800.00
6	Laura	400.00
7	Michael	700.00
8	Emma	160.00
9	William	140.00
10	Olivia	1400.00

11)Find the Average Order Amount for Each Customer.

->select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
73
74 • select * from cart where customer_id = 5;
75
76 • select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;
77
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
80 • select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
81
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
83
84 • select customer_id, count(*) as order_count from orders group by customer_id;
```

The result grid displays the following data:

customer_id	avg_order_amount
1	1200.000000
2	900.000000
3	300.000 900.000000
4	150.000000
5	1800.000000
6	400.000000
7	700.000000
8	160.000000
9	140.000000
10	1400.000000

12)Count the Number of Orders Placed by Each Customer.

->select customer_id, count(*) as order_count from orders group by customer_id;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
89
90 • select c.customer_id, avg(o.total_price) as avg_order_amount
91   from customers c
92   join orders o on c.customer_id = o.customer_id
93   group by c.customer_id;
94
95 • select customer_id, count(*) as order_count
```

The result grid displays the following data:

customer_id	order_count
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

13)Find the Maximum Order Amount for Each Customer.

->select customer_id, max(total_price) as max_order from orders group by customer_id;

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
76 • select distinct c.* from customers c join orders o on c.customer_id = o.customer_id where year(o.order_date) = 2023;
77
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
80 • select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
81
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
83
84 • select customer_id, count(*) as order_count from orders group by customer_id;
85
86 • select customer_id, max(total_price) as max_order from orders group by customer_id;
87
```

The results grid displays the following data:

customer_id	max_order
1	1200.00
2	900.00
3	300.00
4	150.00
5	1800.00
6	400.00
7	700.00
8	160.00
9	140.00
10	1400.00

14)Get Customers Who Placed Orders Totaling Over \$1000.

->select customer_id from orders group by customer_id having sum(total_price) > 1000;

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
78 • select prod_name, min(stock_quan) as min_quantity from product_table group by prod_name;
79
80 • select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
81
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
83
84 • select customer_id, count(*) as order_count from orders group by customer_id;
85
86 • select customer_id, max(total_price) as max_order from orders group by customer_id;
87
88 • select customer_id from orders group by customer_id having sum(total_price) > 1000;
89
```

The results grid displays the following data:

customer_id
1
5
10

15) Subquery to Find Products Not in the Cart.

->select * from product_table where product_id not in (select product_id from cart);

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
80 • select c.customer_id, c.first_name, sum(o.total_price) as total_spent from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
81
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
83
84 • select customer_id, count(*) as order_count from orders group by customer_id;
85
86 • select customer_id, max(total_price) as max_order from orders group by customer_id;
87
88 • select customer_id from orders group by customer_id having sum(total_price) > 1000;
89
90 • select * from product_table where product_id not in (select product_id from cart);
91
```

The results grid displays one row of data from the product_table:

product_id	prod_name	price	description	stock_quan
8	Microwave Oven	80.00	Countertop microwave	15

16) Subquery to Find Customers Who Haven't Placed Orders.

->select * from customers where customer_id not in (select distinct customer_id from orders);

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
82 • select c.customer_id, avg(o.total_price) as avg_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id;
83
84 • select customer_id, count(*) as order_count from orders group by customer_id;
85
86 • select customer_id, max(total_price) as max_order from orders group by customer_id;
87
88 • select customer_id from orders group by customer_id having sum(total_price) > 1000;
89
90 • select * from product_table where product_id not in (select product_id from cart);
91
92 • select * from customers where customer_id not in (select distinct customer_id from orders);
93
```

The results grid displays one row of data from the customers table:

customer_id	first_name	last_name	email	address	password
1	John	Doe	john.doe@example.com	123 Main St	password123

17) Subquery to Calculate the Percentage of Total Revenue for a Product.

->select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by product_id;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
85 •    select customer_id, max(total_price) as max_order from orders group by customer_id;
86 •    select customer_id from orders group by customer_id having sum(total_price) > 1000;
87
88 •    select * from product_table where product_id not in (select product_id from cart);
89
90 •    select * from customers where customer_id not in (select distinct customer_id from orders);
91
92 •    select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by
93
94 •    select * from product_table where stock_quan < 10;
```

The results grid displays the following data:

product_id	revenue_percentage
1	27.91
2	34.88
3	3.49
4	6.98
5	20.93
6	0.58
9	2.44
10	2.79

18) Subquery to Find Products with Low Stock.

->select * from product_table where stock_quan < 10;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

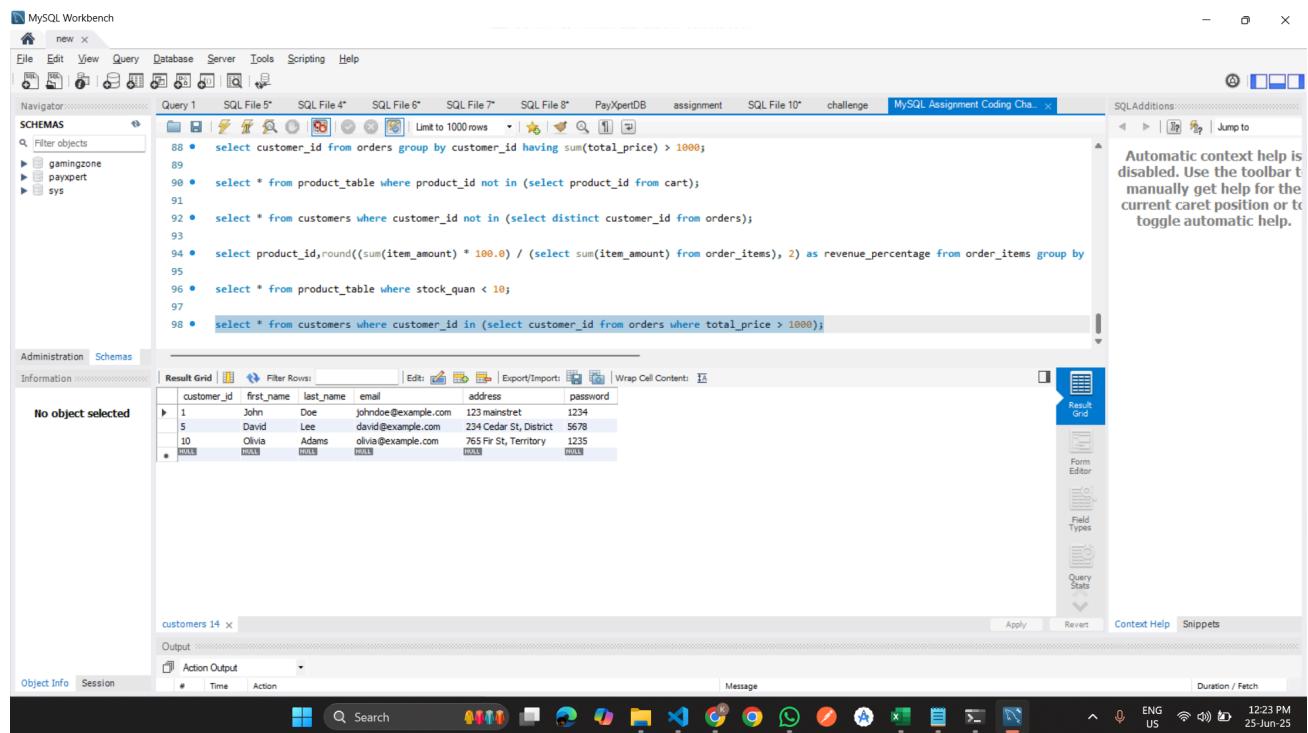
```
85 •    select customer_id, max(total_price) as max_order from orders group by customer_id;
86 •    select customer_id from orders group by customer_id having sum(total_price) > 1000;
87
88 •    select * from product_table where product_id not in (select product_id from cart);
89
90 •    select * from customers where customer_id not in (select distinct customer_id from orders);
91
92 •    select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by
93
94 •    select * from product_table where stock_quan < 10;
```

The results grid displays the following data:

product_id	prod_name	price	description	stock_quan
5	TV	900.00	4K Smo 4K Smart TV	NULL
• NULL	NULL	NULL	NULL	NULL

19) Subquery to Find Customers Who Placed High-Value Orders.

->select * from customers where customer_id in (select customer_id from orders where total_price > 1000);



The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has a Navigator section with Schemas (gamingzone, paypert, sys). The main area contains a SQL editor with the following code:

```
88 • select customer_id from orders group by customer_id having sum(total_price) > 1000;
89
90 • select * from product_table where product_id not in (select product_id from cart);
91
92 • select * from customers where customer_id not in (select distinct customer_id from orders);
93
94 • select product_id,round((sum(item_amount) * 100.0) / (select sum(item_amount) from order_items), 2) as revenue_percentage from order_items group by
95
96 • select * from product_table where stock_quan < 10;
97
98 • select * from customers where customer_id in (select customer_id from orders where total_price > 1000);
```

Below the SQL editor is a Results Grid showing the output of the query:

customer_id	first_name	last_name	email	address	password
1	John	Doe	john doe@example.com	123 mainstreet	1234
5	David	Lee	david@example.com	234 Cedar St, District	5678
10	Olivia	Adams	olivia@example.com	765 Fir St, Territory	1235
*	NULL	NULL	NULL	NULL	NULL

The bottom status bar shows the session details: customers 14 x, Action Output, Object Info, Session, 12:23 PM, ENG US, 25-Jun-25.

