

Coding Challenge 6

Ecommerce

Table Creation:

```
1  /* coding challenge 6 */
2  * Create database Ecom;
3  * use Ecom;
4
5  * CREATE TABLE customers (
6      customer_id INT PRIMARY KEY,
7      name VARCHAR(255),
8      email VARCHAR(255),
9      password VARCHAR(255)
10 );
11
12 * CREATE TABLE products (
13     product_id INT PRIMARY KEY,
14     name VARCHAR(255),
15     price DECIMAL(10, 2),
16     description TEXT,
17     stockQuantity INT
18 );
19
20 * CREATE TABLE cart (
21     cart_id INT PRIMARY KEY,
22     customer_id INT,
23     product_id INT,
24     quantity INT,
25     FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
26     FOREIGN KEY (product_id) REFERENCES products(product_id)
27 );
28
29 * CREATE TABLE orders (
30     order_id INT PRIMARY KEY,
31     customer_id INT,
32     order_date DATE,
33     total_price DECIMAL(10, 2),
34     shipping_address TEXT,
35     FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
36 );
37
38 * CREATE TABLE order_items (
39     order_item_id INT PRIMARY KEY,
40     order_id INT,
41     product_id INT,
42     quantity INT,
43     FOREIGN KEY (order_id) REFERENCES orders(order_id),
44     FOREIGN KEY (product_id) REFERENCES products(product_id)
45 );
46
```

Data Insertion:

```
47 * INSERT INTO products (product_id, name, description, price, stockQuantity) VALUES
48 (1, 'Laptop', 'High-performance laptop', 899.99, 10),
49 (2, 'Smartphone', 'Latest smartphone', 499.99, 15),
50 (3, 'Tablet', 'Portable tablet', 299.99, 20),
51 (4, 'Headphones', 'Noise-canceling', 149.99, 30),
52 (5, 'TV', '4K Smart TV', 599.99, 5),
53 (6, 'Coffee Maker', 'Automatic coffee maker', 79.99, 25),
54 (7, 'Refrigerator', 'Energy-efficient', 399.99, 10),
55 (8, 'Microwave Oven', 'Countertop microwave', 89.99, 15),
56 (9, 'Blender', 'High-speed blender', 49.99, 20),
57 (10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 129.99, 12);
58
59
60 * ALTER TABLE customers ADD COLUMN address text;
61 * INSERT INTO customers (customer_id, name, email, password, address) VALUES
62 (1, 'John Doe', 'john.doe@example.com', 'password1', '123 Main St, City'),
63 (2, 'Jane Smith', 'jane.smith@example.com', 'password2', '456 Elm St, Town'),
64 (3, 'Robert Johnson', 'robert.j@example.com', 'password3', '789 Oak St, Village'),
65 (4, 'Sarah Brown', 'sarah.b@example.com', 'password4', '101 Pine St, Suburb'),
66 (5, 'David Lee', 'david.l@example.com', 'password5', '114 Cedar St, District'),
67 (6, 'Laura Hall', 'laura.h@example.com', 'password6', '147 Birch St, County'),
68 (7, 'Michael Davis', 'michael.d@example.com', 'password7', '189 Maple St, State'),
69 (8, 'Anna Wilson', 'anna.w@example.com', 'password8', '111 Redwood St, Country'),
70 (9, 'William Taylor', 'william.t@example.com', 'password9', '432 Spruce St, Province'),
71 (10, 'Olivia Adams', 'olivia.a@example.com', 'password10', '765 Fir St, Territory');
72
73
74 * INSERT INTO orders (order_id, customer_id, order_date, total_price, shipping_address) VALUES
75 (1, 1, '2023-01-05', 1299.98, '123 Main St, City, India'),
76 (2, 2, '2023-02-10', 999.98, '456 Elm St, Town, India'),
77 (3, 3, '2023-03-15', 599.98, '789 Oak St, Village, India'),
78 (4, 4, '2023-04-18', 159.98, '101 Pine St, Suburb, India'),
79 (5, 5, '2023-05-23', 1899.98, '114 Cedar St, District, India'),
80 (6, 6, '2023-06-30', 499.98, '147 Birch St, County, India'),
81 (7, 7, '2023-07-05', 799.98, '189 Maple St, State, India'),
82 (8, 8, '2023-08-10', 169.98, '111 Redwood St, Country, India'),
83 (9, 9, '2023-09-15', 149.98, '432 Spruce St, Province, India'),
84 (10, 10, '2023-10-18', 1499.98, '765 Fir St, Territory, India');
85
86 * ALTER TABLE order_items ADD COLUMN item_amount DECIMAL(10,2);
87 * INSERT INTO order_items (order_item_id, order_id, product_id, quantity, item_amount) VALUES
88 (1, 1, 1, 1, 1000.00),
89 (2, 1, 3, 1, 299.98),
90 (3, 2, 2, 2, 1000.00),
91 (4, 3, 5, 2, 1000.00),
92 (5, 4, 4, 4, 600.00),
93 (6, 4, 6, 1, 59.99),
94 (7, 5, 1, 1, 899.99),
95 (8, 5, 2, 2, 1000.00),
96 (9, 6, 10, 2, 100.00),
97 (10, 4, 9, 3, 150.00);
98
99
100 * INSERT INTO cart (cart_id, customer_id, product_id, quantity) VALUES
101 (1, 1, 1, 2),
102 (2, 1, 3, 1),
103 (3, 2, 2, 3),
104 (4, 3, 4, 4),
105 (5, 3, 5, 1),
106 (6, 4, 6, 1),
107 (7, 5, 1, 1),
108 (8, 6, 10, 2),
109 (9, 6, 9, 3),
110 (10, 7, 7, 1);
111
```

Queries:

1.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL statement:

```
110  
111 /*1*/  
112 * UPDATE products SET price = 800.00 WHERE product_id = 7;  
113
```

The output pane shows the execution results:

Time	Action	Message	Duration / Fetch
1 13:25:12	UPDATE products SET price = 800.00 WHERE product_id = 7	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec

2.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL statement:

```
113  
114 /*2*/  
115 * DELETE FROM cart WHERE customer_id = 7;  
116
```

The output pane shows the execution results:

Time	Action	Message	Duration / Fetch
1 13:26:21	DELETE FROM cart WHERE customer_id = 7	1 row(s) affected	0.000 sec

3.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL statement:

```
117  
118 /*3*/  
119 * SELECT * FROM products WHERE price < 100.00;  
120  
121
```

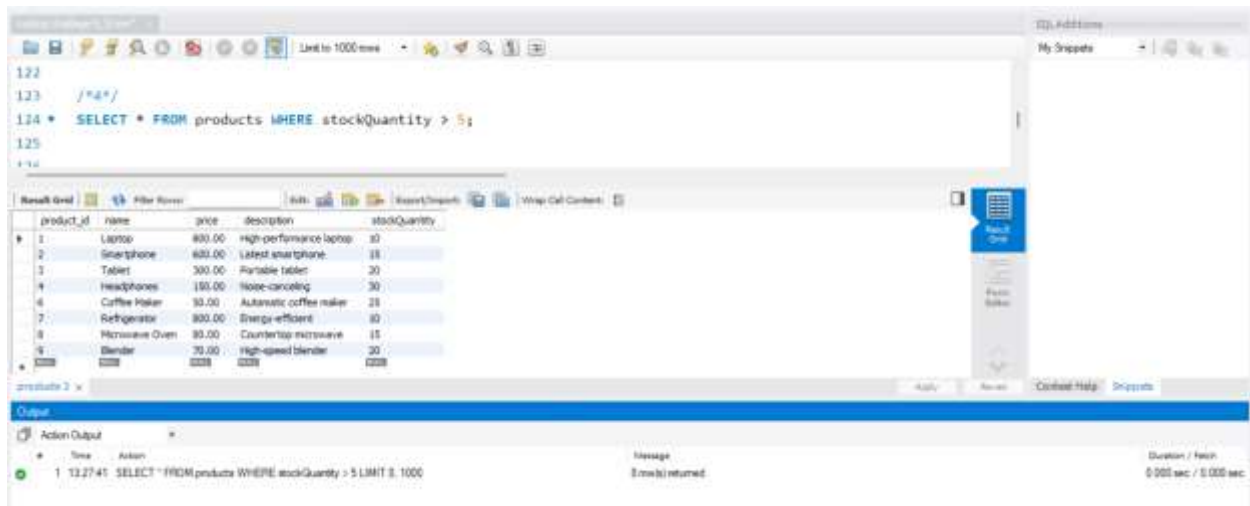
The result grid displays the following data:

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

The output pane shows the execution results:

Time	Action	Message	Duration / Fetch
1 13:26:51	SELECT * FROM products WHERE price < 100.00 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

4.

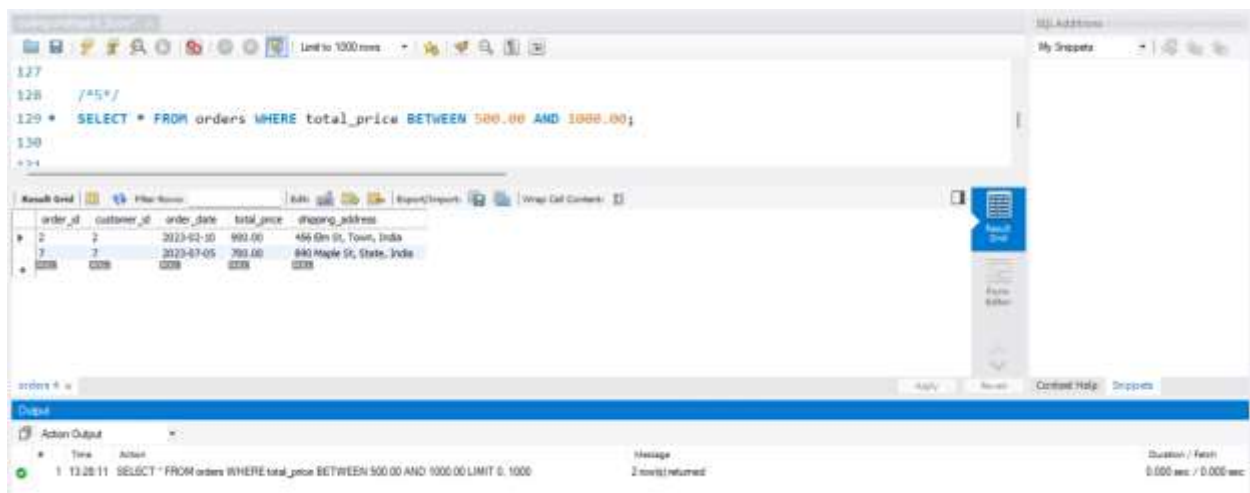


The screenshot shows a SQL query editor with a query that filters products by stock quantity. The results table displays the following data:

product_id	name	price	description	stockQuantity
1	Laptop	800.00	high-performance laptop	30
2	Smartphone	600.00	latest smartphone	18
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-canceling	30
4	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

The output pane shows the execution of the query, indicating that 8 rows were returned.

5.



The screenshot shows a SQL query editor with a query that filters orders by total price. The results table displays the following data:

order_id	customer_id	order_date	total_price	shipping_address
2	2	2023-02-10	900.00	456 Elm St, Town, India
7	7	2023-07-05	790.00	890 Maple St, State, India

The output pane shows the execution of the query, indicating that 2 rows were returned.

6.

The screenshot shows a SQL IDE interface. The query editor at the top contains the following SQL query:

```
130  
131  
132 /* */  
133 * SELECT * FROM products WHERE name LIKE '%-'  
134
```

Below the query editor, the 'Result Grid' displays the results of the query. The grid has five columns: product_id, name, price, description, and stockQuantity. The results are as follows:

product_id	name	price	description	stockQuantity
4	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	1

At the bottom, the 'Output' pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13.28.51	SELECT * FROM products WHERE name LIKE '%-' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

7.

The screenshot shows a SQL IDE interface. The query editor at the top contains the following SQL query:

```
135  
136 /* */  
137 * SELECT * FROM cart WHERE customer_id = 5  
138  
139
```

Below the query editor, the 'Result Grid' displays the results of the query. The grid has four columns: cart_id, customer_id, product_id, and quantity. The results are as follows:

cart_id	customer_id	product_id	quantity
7	5	1	1

At the bottom, the 'Output' pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13.29.33	SELECT * FROM cart WHERE customer_id = 5 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

8.

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

```
140 /*B*/
141 * SELECT DISTINCT c.* FROM customers c
142 JOIN orders o ON c.customer_id = o.customer_id
143 WHERE EXTRACT(YEAR FROM o.order_date) = 2023;
```

The results are displayed in a table with the following columns: customer_id, name, email, password, and address. The table contains 9 rows of data.

customer_id	name	email	password	address
1	John Doe	john.doe@example.com	password1	111 Main St, City
2	Jane Smith	jane.smith@example.com	password2	456 Elm St, Town
3	Robert Johnson	robert@example.com	password3	789 Oak St, Village
4	Sarah Brown	sarah@example.com	password4	101 Pine St, Suburb
5	David Lee	david@example.com	password5	234 Cedar St, District
6	Laura Hall	laura@example.com	password6	567 Birch St, County
7	Michael Davis	michael@example.com	password7	890 Maple St, State
8	Emma Wilson	emma@example.com	password8	321 Redwood St, Country
9	William Taylor	william@example.com	password9	432 Spruce St, Province

The bottom panel shows the Action Output with the following message: "13:30:55 SELECT DISTINCT c.* FROM customers c JOIN orders o ON c.customer_id = o.customer_id WHERE EXTRACT(YEAR FROM o.order_date) = 2023; 13 rows returned". The duration is 0.000 sec / 0.000 sec.

9.

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

```
145 /*B*/
146 * SELECT product_id, MIN(stockQuantity) AS min_stock
147 FROM products
148 GROUP BY product_id;
```

The results are displayed in a table with the following columns: product_id and min_stock. The table contains 9 rows of data.

product_id	min_stock
1	10
2	15
3	20
4	30
5	5
6	25
7	30
8	15
9	20

The bottom panel shows the Action Output with the following message: "13:30:30 SELECT product_id, MIN(stockQuantity) AS min_stock FROM products GROUP BY product_id LIMIT 0, 1000; 10 rows returned". The duration is 0.000 sec / 0.000 sec.

10.

The screenshot shows a SQL query editor with the following code:

```
150 /*10*/
151 * SELECT customer_id, SUM(total_price) AS total_spent
152 FROM orders
153 GROUP BY customer_id;
```

The results grid displays the following data:

customer_id	total_spent
1	1200.00
2	900.00
3	300.00
4	150.00
5	1800.00
6	400.00
7	700.00
8	180.00
9	140.00

The output pane shows the execution message: "SELECT customer_id, SUM(total_price) AS total_spent FROM orders GROUP BY customer_id LIMIT 0, 1000" with 10 rows returned in 0.000 sec.

11.

The screenshot shows a SQL query editor with the following code:

```
155 /*11*/
156 * SELECT customer_id, AVG(total_price) AS avg_order_amount
157 FROM orders
158 GROUP BY customer_id;
```

The results grid displays the following data:

customer_id	avg_order_amount
1	1200.000000
2	900.000000
3	300.000000
4	150.000000
5	1800.000000
6	400.000000
7	700.000000
8	180.000000
9	140.000000

The output pane shows the execution message: "SELECT customer_id, AVG(total_price) AS avg_order_amount FROM orders GROUP BY customer_id LIMIT 0, 1..." with 10 rows returned in 0.000 sec.

12.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
160 /*12*/  
161 * SELECT customer_id, COUNT(order_id) AS order_count  
162 FROM orders  
163 GROUP BY customer_id;
```

The results 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

The output pane shows the execution details:

Time	Action	Message	Duration / Refresh
13:31:59	SELECT customer_id, COUNT(order_id) AS order_count FROM orders GROUP BY customer_id LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

13.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
165 /*13*/  
166 * SELECT customer_id, MAX(total_price) AS max_order_amount  
167 FROM orders  
168 GROUP BY customer_id;
```

The results grid displays the following data:

customer_id	max_order_amount
1	1200.00
2	3000.00
3	3000.00
4	1500.00
5	1800.00
6	4000.00
7	7000.00
8	1800.00
9	1400.00

The output pane shows the execution details:

Time	Action	Message	Duration / Refresh
13:32:20	SELECT customer_id, MAX(total_price) AS max_order_amount FROM orders GROUP BY customer_id LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

14.

The screenshot shows the SQL Assistant interface with a query editor at the top containing the following SQL code:

```
170 /*14*/  
171 * SELECT c.* FROM customers c  
172 JOIN orders o ON c.customer_id = o.customer_id  
173 WHERE o.total_price > 1000.00;
```

Below the editor, the 'Result Grid' displays the following data:

customer_id	name	email	password	address
1	John Doe	john.doe@example.com	password1	123 Main St, City
5	David Lee	david@example.com	password5	234 Cedar St, District
10	Olivia Adams	olivia@example.com	password10	567 Pine St, Territory

The 'Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13.32.44	SELECT c.* FROM customers c JOIN orders o ON c.customer_id = o.customer_id WHERE o.total_price > 1000...	3 row(s) returned	0.000 sec / 0.000 sec

15.

The screenshot shows the SQL Assistant interface with a query editor at the top containing the following SQL code:

```
174  
175 /*15*/  
176 * SELECT * FROM products  
177 WHERE product_id NOT IN (SELECT product_id FROM cart);
```

Below the editor, the 'Result Grid' displays the following data:

product_id	name	price	description	stockQuantity
7	HalfGinger	80.00	Irving-efficient	12
8	Microwave Oven	80.00	Countertop microwave	15

The 'Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13.33.09	SELECT * FROM products WHERE product_id NOT IN (SELECT product_id FROM cart) LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

16.

The screenshot shows the SQL Assistant interface with a query editor at the top containing the following SQL code:

```
178  
179 /*16*/  
180 * SELECT * FROM customers  
181 WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders);
```

Below the editor, the 'Result Grid' displays the following data:

customer_id	name	email	password	address
-------------	------	-------	----------	---------

The 'Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13.33.44	SELECT * FROM customers WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders) LIMIT...	0 row(s) returned	0.000 sec / 0.000 sec

17.

The screenshot shows a SQL query in a text editor and its results in a grid. The query calculates the revenue percentage for each product by dividing the sum of item amounts by the sum of total prices from orders, multiplied by 100. The results are grouped by product ID.

```

183 /*17*/
184 * SELECT product_id, (SUM(item_amount) / (SELECT SUM(total_price) FROM orders)) * 100 AS revenue_percentage
185 FROM order_items
186 GROUP BY product_id;

```

product_id	revenue_percentage
1	33.566434
2	41.998042
3	4.199804
4	8.391608
5	25.174625
6	6.690361
8	2.937063
10	3.39643

The output pane shows the execution of the query, indicating that 8 rows were returned.

18.

The screenshot shows a SQL query in a text editor and its results in a grid. The query filters products where the stock quantity is less than the average stock quantity of all products. The results are limited to 10 rows.

```

187
188 /*18*/
189 * SELECT * FROM products
190 WHERE stockQuantity < (SELECT AVG(stockQuantity) FROM products);

```

product_id	name	price	description	stockQuantity
1	Laptop	800.00	High performance laptop	10
2	Smartphone	600.00	Latest smartphone	25
3	TV	300.00	4K Smart TV	5
7	Refrigerator	800.00	Energy-efficient	10
8	Microwave Oven	90.00	Countertop microwave	25
10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	1

The output pane shows the execution of the query, indicating that 6 rows were returned.

19.

The screenshot shows a SQL IDE interface. The top pane contains a SQL query:

```
191
192 /*19*/
193 * SELECT * FROM customers
194 WHERE customer_id IN (SELECT customer_id FROM orders WHERE total_price > 1000.00);
```

The bottom pane displays the results of the query in a table format. The table has five columns: customer_id, name, email, password, and address. The results are as follows:

customer_id	name	email	password	address
1	John Doe	johndoe@example.com	password1	123 Main St, City
5	David Lee	davidl@example.com	password5	234 Cedar St, District
15	Olivia Adams	oliviaa@example.com	password10	765 Pine St, Territory

The bottom pane also shows the execution log with the following entry:

Time	Action	Message	Duration / Pacing
11:37:58	SELECT * FROM customers WHERE customer_id IN (SELECT customer_id FROM orders WHERE total_price > 1000.00);	3 rows returned	0.000 sec / 0.000 sec