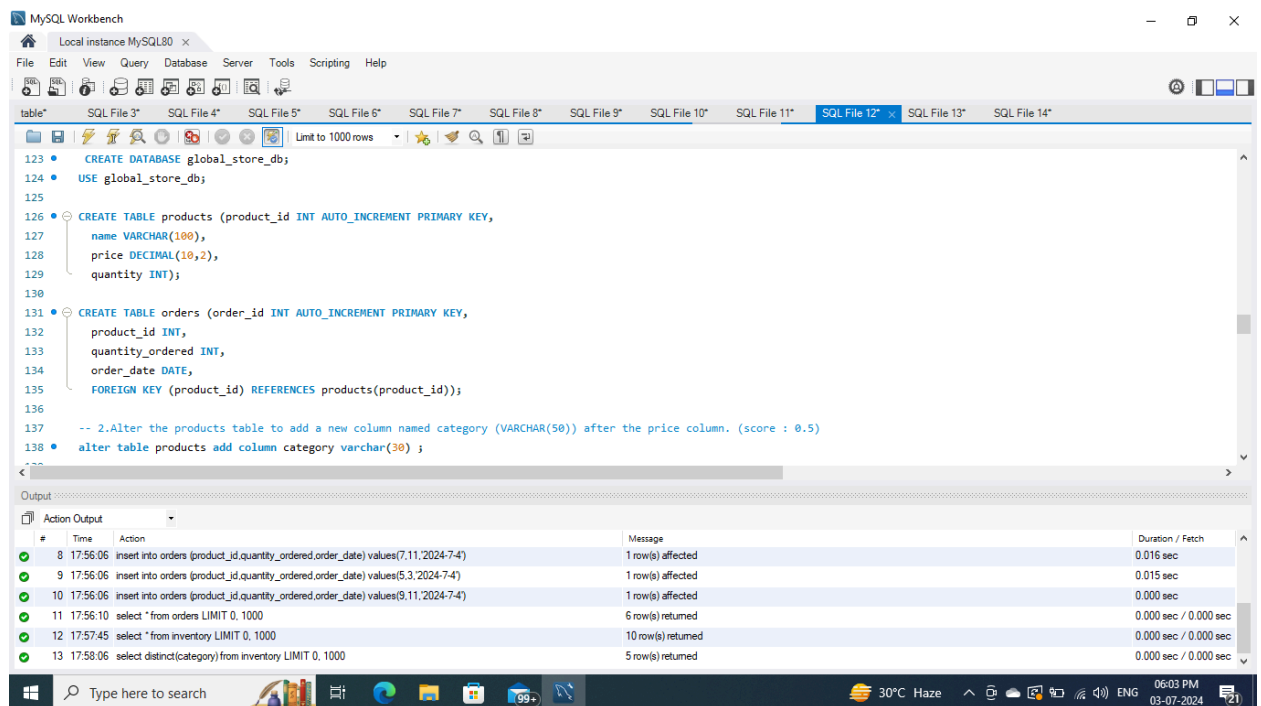


1. Create the following tables inside the database 'global\_store\_db'.(Score :2)  
'products' with columns:

- product\_id (INT, auto\_increment, primary key),
- name (VARCHAR(100)),
- price (DECIMAL(10,2)),
- quantity (INT).

'orders' with columns:

- order\_id (INT, auto\_increment, primary key),
- product\_id (INT, foreign key referencing product\_id in the inventory table),
- quantity\_ordered (INT)
- order\_date (DATE).



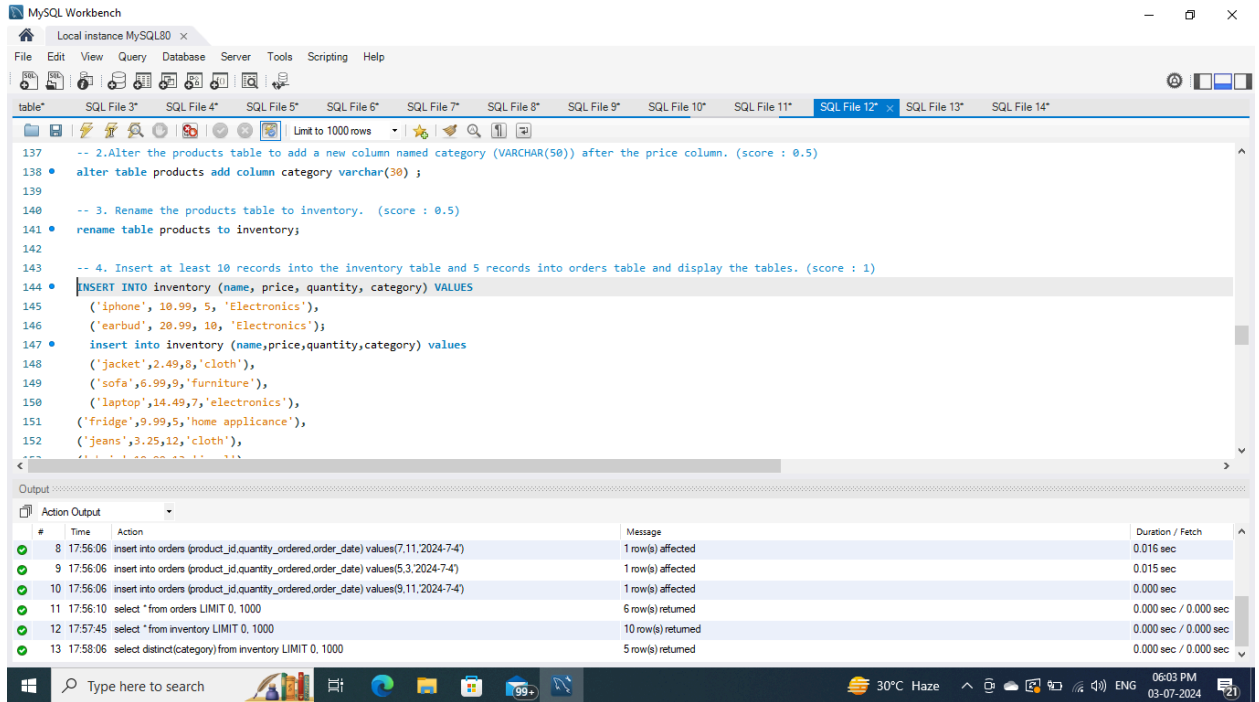
The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```
123 • CREATE DATABASE global_store_db;
124 • USE global_store_db;
125
126 • CREATE TABLE products (product_id INT AUTO_INCREMENT PRIMARY KEY,
127   name VARCHAR(100),
128   price DECIMAL(10,2),
129   quantity INT);
130
131 • CREATE TABLE orders (order_id INT AUTO_INCREMENT PRIMARY KEY,
132   product_id INT,
133   quantity_ordered INT,
134   order_date DATE,
135   FOREIGN KEY (product_id) REFERENCES products(product_id));
136
137 -- 2.Alter the products table to add a new column named category (VARCHAR(50)) after the price column. (score : 0.5)
138 • alter table products add column category varchar(30) ;
```

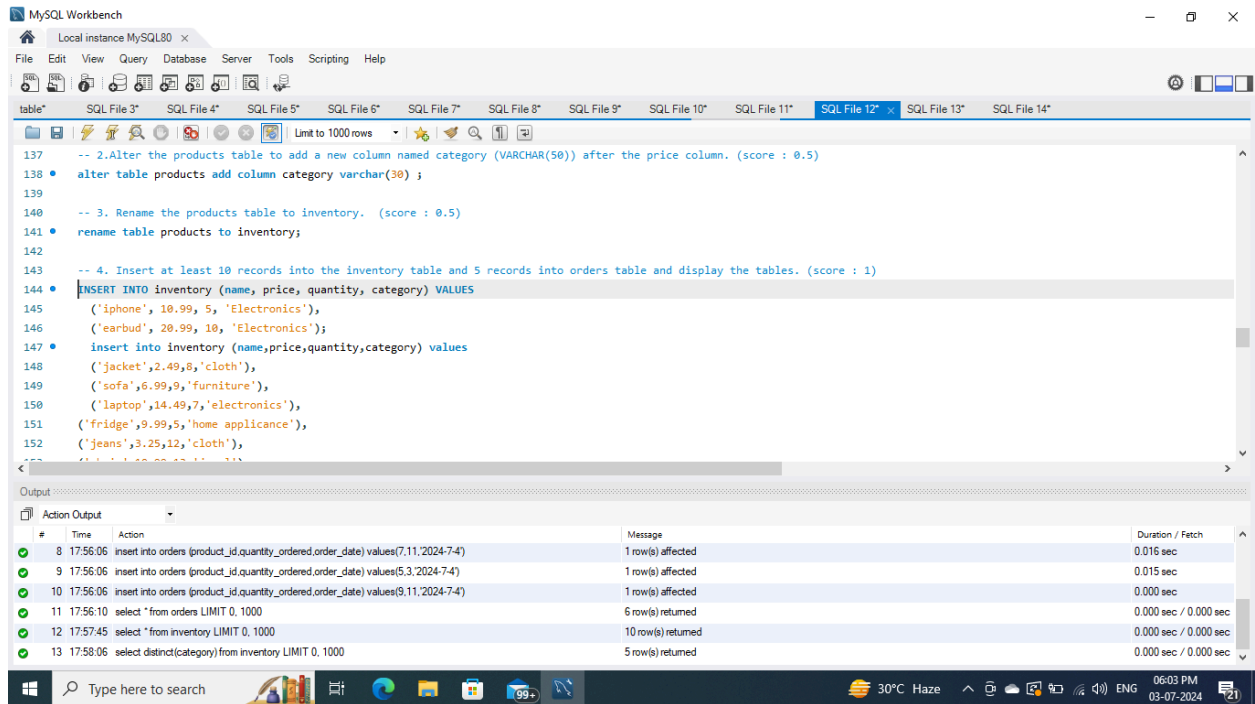
The Output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
8	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(7,11,2024-7-4)	1 row(s) affected	0.016 sec
9	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(5,3,2024-7-4)	1 row(s) affected	0.015 sec
10	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(9,11,2024-7-4)	1 row(s) affected	0.000 sec
11	17:56:10	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
12	17:57:45	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
13	17:58:06	select distinct(category) from inventory LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

2.Alter the products table to add a new column named category (VARCHAR(50)) after the price column. (score : 0.5)



### 3. Rename the products table to inventory. (score : 0.5)



### 4. Insert at least 10 records into the inventory table and 5 records into orders table and display the tables. (score : 1)

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

table\* SQL File 3\* SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQL File 12\* x SQL File 13\* SQL File 14\*

Limit to 1000 rows

```

141 • rename table products to inventory;
142
143 -- 4. Insert at least 10 records into the inventory table and 5 records into orders table and display the tables. (score : 1)
144 • INSERT INTO inventory (name, price, quantity, category) VALUES
145 ('iphone', 10.99, 5, 'Electronics'),
146 ('earbud', 20.99, 10, 'Electronics');
147 • insert into inventory (name,price,quantity,category) values
148 ('jacket',2.49,8,'cloth'),
149 ('sofa',6.99,9,'furniture'),
150 ('laptop',14.49,7,'electronics'),
151 ('fridge',9.99,5,'home appliance'),
152 ('jeans',3.25,12,'cloth'),
153 ('chain',10.99,13,'jewel'),
154 ('bangles',9.99,16,'jewel'),
155 ('filter',8.99,7,'home appliance');

```

Result Grid

product_id	name	price	quantity	category
1	iphone	10.99	5	Electronics
2	earbud	20.99	10	Electronics
3	jacket	2.49	8	cloth
4	sofa	6.99	9	furniture
5	laptop	14.49	7	electronics
6	fridge	9.99	5	home appliance
7	jeans	3.25	12	cloth
8	chain	10.99	13	jewel
9	bangles	9.99	16	jewel
10	filter	8.99	7	home appliance

Inventory 8 x

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

table\* SQL File 3\* SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQL File 12\* x SQL File 13\* SQL File 14\*

Limit to 1000 rows

```

156 • insert into orders (order_id,quantity_ordered,order_date) values(101,2,curentdate());
157 • update orders set product_id=1 where order_id = 101;
158 • insert into orders (product_id,quantity_ordered,order_date) values(2,2,'2024-5-4');
159 • insert into orders (product_id,quantity_ordered,order_date) values(3,4,'2024-7-4');
160 • insert into orders (product_id,quantity_ordered,order_date) values(7,11,'2024-7-4');
161 • insert into orders (product_id,quantity_ordered,order_date) values(5,3,'2024-7-4');
162 • insert into orders (product_id,quantity_ordered,order_date) values(9,11,'2024-7-4');
163
164 • select * from inventory;
165 • select * from orders;
166 -- 5. Write queries for the following : (Score :3)
167 -- a) Write a query to display distinct categories from the inventory table.
168 • select distinct(category) from inventory;
169 -- b) Select the top 5 products by their prices in descending order from the inventory table.
170 c) Display the names of products with a quantity greater than 10 from the inventory table.

```

Result Grid

order_id	product_id	quantity_ordered	order_date
101	1	2	2024-07-01
102	2	2	2024-05-04
103	3	4	2024-07-04
104	7	11	2024-07-04
105	5	3	2024-07-04
106	9	11	2024-07-04

orders 9 x

5. Write queries for the following : (Score :3)

a) Write a query to display distinct categories from the inventory table.

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

```

165
166 • select * from inventory;
167 • select * from orders;
168 -- 5. Write queries for the following : (Score :3)
169 -- a) Write a query to display distinct categories from the inventory table.
170 • select distinct(category) from inventory;
171
172
173

```

The Result Grid shows the output of the query:

category
Electronics
cloth
furniture
home appliance
jewel

The Action Output shows the execution details:

#	Time	Action	Message	Duration / Fetch
8	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(7,11,'2024-7-4');	1 row(s) affected	0.016 sec
9	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(5,3,'2024-7-4');	1 row(s) affected	0.015 sec
10	17:56:06	insert into orders (product_id,quantity_ordered,order_date) values(9,11,'2024-7-4');	1 row(s) affected	0.000 sec
11	17:56:10	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
12	17:57:45	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
13	17:58:06	select distinct(category) from inventory LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

b) Select the top 5 products by their prices in descending order from the inventory table.

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

```

162 • insert into orders (product_id,quantity_ordered,order_date) values(9,11,'2024-7-4');
163
164 • select * from inventory;
165 • select * from orders;
166 -- 5. Write queries for the following : (Score :3)
167 -- a) Write a query to display distinct categories from the inventory table.
168 • select distinct(category) from inventory;
169
170 -- b) Select the top 5 products by their prices in descending order from the inventory table.
171 • select name ,price from inventory order by price desc limit 5;
172
173
174
175
176

```

The Result Grid shows the output of the query:

name	price
earbud	20.99
laptop	14.49
phone	10.99
chain	10.99
fridge	9.99

c) Display the names of products with a quantity greater than 10 from the inventory table.

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

```
-- 5. Write queries for the following : (Score :3)
-- a) Write a query to display distinct categories from the inventory table.
select distinct(category) from inventory;

-- b) Select the top 5 products by their prices in descending order from the inventory table.
select name ,price from inventory order by price desc limit 5;

-- c) Display the names of products with a quantity greater than 10 from the inventory table.
select name, quantity from inventory where quantity > 10;
```

The result grid for the third query is displayed, showing the following data:

name	quantity
jeans	12
chain	13
bangles	16

d) Use the SUM() function to calculate the total price of all products in the inventory table.

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

```
-- 5. Write queries for the following : (Score :3)
-- a) Write a query to display distinct categories from the inventory table.
select distinct(category) from inventory;

-- b) Select the top 5 products by their prices in descending order from the inventory table.
select name ,price from inventory order by price desc limit 5;

-- c) Display the names of products with a quantity greater than 10 from the inventory table.
select name, quantity from inventory where quantity > 10;

-- d) Use the SUM() function to calculate the total price of all products in the inventory table.
select sum(price) 'tot price' from inventory;
```

The result grid for the fourth query is displayed, showing the following data:

tot price
99.16

e) Group products by their categories and display the count of products in each category.

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

```

-- d) Use the SUM() function to calculate the total price of all products in the inventory table.
177 • select sum(price) 'tot price' from inventory;

-- e) Group products by their categories and display the count of products in each category.
180 • select category, count(name) 'type of products in this category' from inventory group by category ;

181
182 f) Write a query to identify products that are currently out of stock (i.e., quantity is zero). Display the product details including the product name and price.
183
184 6. Create a view named expensive_products that displays the details of products with a price above the average price of all products. (score : 1)
185
186 7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables. (score :
187
188 -- Create tables
189
  
```

The Result Grid shows the output of the query in step 180:

category	type of products in this category
Electronics	3
cloth	2
furniture	1
home appliance	2
jewel	2

f) Write a query to identify products that are currently out of stock (i.e., quantity is zero). Display the product details including the product name and price.

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

```

182 -- f) Write a query to identify products that are currently out of stock (i.e., quantity is zero). Display the product details including the product name and price.
183 • update inventory set quantity = 0 where product_id =1 ;
184 • update inventory set quantity = 0 where product_id =6 ;
185 • select name, price from inventory where quantity = 0;

186
187 6. Create a view named expensive_products that displays the details of products with a price above the average price of all products. (score : 1)
188
189 7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables. (score :
  
```

The Result Grid shows the output of the query in step 185:

name	price
iphone	10.99
fridge	9.99

The Action Output log shows the execution of the queries:

#	Time	Action	Message	Duration / Fetch
33	18:22:57	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
34	18:23:22	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
35	18:25:43	update inventory set quantity = 0 where product_id =0	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
36	18:26:01	update inventory set quantity = 0 where product_id =1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
37	18:26:19	update inventory set quantity = 0 where product_id =6	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
38	18:27:29	select name, price from inventory where quantity = 0 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

6. Create a view named expensive\_products that displays the details of products with a price above the average price of all products. (score : 1)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

table\* SQL File 3\* SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQL File 12\* x SQL File 13\* SQL File 14\*

Limit to 1000 rows

```

183 • update inventory set quantity = 0 where product_id =1 ;
184 • update inventory set quantity = 0 where product_id =6 ;
185 • select name,price from inventory where quantity = 0;
186
187 -- 6. Create a view named expensive_products that displays the details of products with a price above the average price of all products. (score : 1)
188 • create view expensive_products
189 as
190 select * from inventory where price > (select avg(price) from inventory);
191
192 • select * from expensive_products;
193
194
195 -- 7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables. (score :
196 • select i.name,o.quantity_ordered from inventory i right join orders o on i.product_id = o.product_id;
197

```

Result Grid

product_id	name	price	quantity	category
1	iphone	10.99	0	Electronics
2	earbud	20.99	10	Electronics
5	laptop	14.49	7	electronics
6	fridge	9.99	0	home appliance
8	chain	10.99	13	jewel
9	bangles	9.99	16	jewel

expensive\_products 38 x

Read Only

Very humid 06:54 PM 03-07-2024

7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables. (score : 1)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

table\* SQL File 3\* SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\* SQL File 11\* SQL File 12\* x SQL File 13\* SQL File 14\*

Limit to 1000 rows

```

183 • update inventory set quantity = 0 where product_id =1 ;
184 • update inventory set quantity = 0 where product_id =6 ;
185 • select name,price from inventory where quantity = 0;
186
187 -- 6. Create a view named expensive_products that displays the details of products with a price above the average price of all products. (score : 1)
188 • create view expensive_products
189 as
190 select * from inventory where price > (select avg(price) from inventory);
191
192 • select * from expensive_products;
193
194 -- 7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables. (score :
195 • select i.name,o.quantity_ordered from inventory i right join orders o on i.product_id = o.product_id;
196
197

```

Result Grid

name	quantity_ordered
iphone	2
earbud	2
jacket	4
jeans	11
laptop	3
bangles	11

Result 37 x

Read Only

29°C Haze 06:44 PM 03-07-2024