

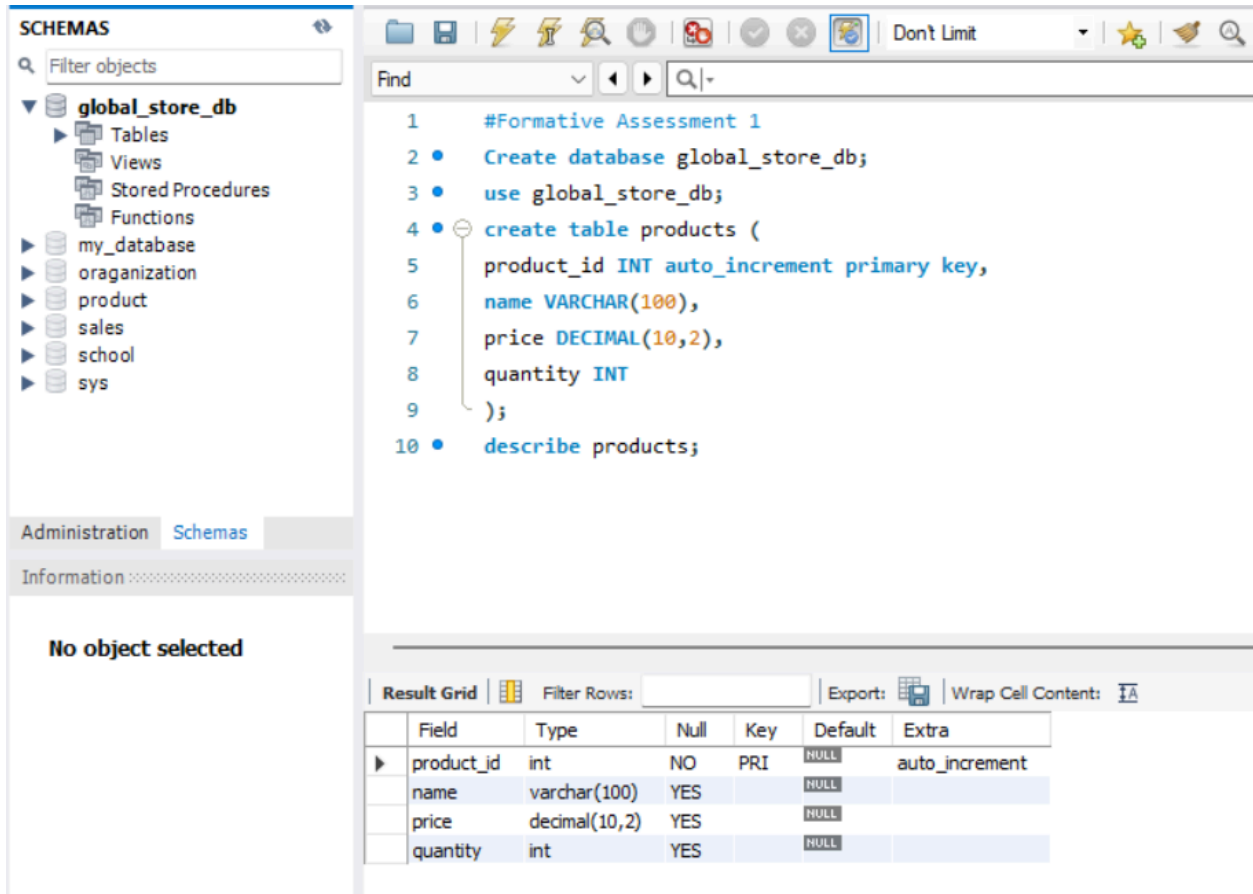
# Formative Assessment 1

Instructions:

Complete the following tasks in MySQL. Each task is assigned a specific score. Timely submission earns an additional 1 point.

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).



```
1 #Formative Assessment 1
2 Create database global_store_db;
3 use global_store_db;
4 create table products (
5   product_id INT auto_increment primary key,
6   name VARCHAR(100),
7   price DECIMAL(10,2),
8   quantity INT
9 );
10 describe products;
```

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	YES	YES	NULL	
price	decimal(10,2)	YES	YES	NULL	
quantity	int	YES	YES	NULL	

'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).

```

11
12 • create table orders (
13     order_id INT auto_increment primary key,
14     product_id INT ,
15     quantity_ordered INT,
16     order_date DATE,
17     FOREIGN KEY (product_id) REFERENCES products(product_id)
18 );
19 • describe orders;

```

as

id

Result Grid

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	<small>NULL</small>	auto_increment
product_id	int	YES	MUL	<small>NULL</small>	
quantity_ordered	int	YES		<small>NULL</small>	
order_date	date	YES		<small>NULL</small>	

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

```

20
21 • alter table products add category VARCHAR(50) after price ;
22 • describe products;

```

Result Grid

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	<small>NULL</small>	auto_increment
name	varchar(100)	YES		<small>NULL</small>	
price	decimal(10,2)	YES		<small>NULL</small>	
category	varchar(50)	YES		<small>NULL</small>	
quantity	int	YES		<small>NULL</small>	

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

```

24 • alter table products rename to inventory;
25 • describe products;
26 • describe inventory;

```

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	<b>NULL</b>	auto_increment
name	varchar(100)	YES		<b>NULL</b>	
price	decimal(10,2)	YES		<b>NULL</b>	
category	varchar(50)	YES		<b>NULL</b>	
quantity	int	YES		<b>NULL</b>	

Result 4 x

Output

#	Time	Action	Message
12	16:34:24	describe products	Error Code: 1146. Table 'global_store_db.products' doesn't exist
13	16:35:13	describe inventory	5 row(s) returned

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

Tables  
Views  
Stored Procedures  
Functions  
my\_database  
organization  
product  
sales  
school  
sys

Administration Schemas

Information

No object selected

```

26 • describe inventory;
27 • insert into inventory (name,price,category,quantity) values("soap",45.00,"stationary",100);
28 • insert into inventory (name,price,category,quantity) values("Sugar",45.00,"stationary",100);
29 • insert into inventory (name,price,category,quantity) values("Rice",55.00,"stationary",500);
30 • insert into inventory (name,price,category,quantity) values("Lipstick",545.00,"Cosmetic",250);
31 • insert into inventory (name,price,category,quantity) values("Snickers",10.00,"sweets",1000);
32 • insert into inventory (name,price,category,quantity) values("Bisuits",45.00,"bakery",500);
33 • insert into inventory (name,price,category,quantity) values("Dal",125.00,"stationary",100);
34 • insert into inventory (name,price,category,quantity) values("Ghee",845.00,"stationary",75);
35 • insert into inventory (name,price,category,quantity) values("Meat Masala",55.00,"stationary",100);
36 • insert into inventory (name,price,category,quantity) values("Chicken Masala",55.00,"stationary",100);
37 • select * from inventory;

```

product_id	name	price	category	quantity
1	soap	45.00	stationary	100
2	Rice	55.00	stationary	500
3	Lipstick	545.00	Cosmetic	250
4	Snickers	10.00	sweets	1000
5	Bisuits	45.00	bakery	500
6	Ghee	845.00	stationary	75
7	Meat Masala	55.00	stationary	100
8	Chicken Masala	55.00	stationary	100
9	Sugar	45.00	stationary	100
10	Dal	125.00	stationary	100

```

38 • insert into orders (product_id,quantity_ordered,order_date) values(1,10,"2024-06-10");
39 • insert into orders (product_id,quantity_ordered,order_date) values(2,10,"2024-06-10");
40 • insert into orders (product_id,quantity_ordered,order_date) values(4,1,"2024-06-10");
41 • insert into orders (product_id,quantity_ordered,order_date) values(5,5,"2024-06-15");
42 • insert into orders (product_id,quantity_ordered,order_date) values(6,1,"2024-06-19");
43 • insert into orders (product_id,quantity_ordered,order_date) values(7,10,"2024-06-20");
44 • insert into orders (product_id,quantity_ordered,order_date) values(1,10,"2024-06-28");
45 • insert into orders (product_id,quantity_ordered,order_date) values(2,10,"2024-06-29");
46 • insert into orders (product_id,quantity_ordered,order_date) values(4,1,"2024-06-30");
47 • insert into orders (product_id,quantity_ordered,order_date) values(5,5,"2024-06-16");
48 • insert into orders (product_id,quantity_ordered,order_date) values(6,1,"2024-06-21");
49 • insert into orders (product_id,quantity_ordered,order_date) values(7,10,"2024-06-22");
50 • select * from orders;

```

order_id	product_id	quantity_ordered	order_date
1	1	10	2024-06-10
2	2	10	2024-06-10
3	2	10	2024-06-10
4	4	1	2024-06-10
5	5	5	2024-06-15
6	6	1	2024-06-19
7	7	10	2024-06-20
8	1	10	2024-06-28

orders 13 x

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

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

```

52 • select distinct(category) from inventory;

```

category
stationary
Cosmetic
sweets
bakery

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

54 • `Select * from inventory order by price desc limit 5;`

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap
product_id	name	price	category	quantity
6	Ghee	845.00	stationary	75
3	Lipstick	545.00	Cosmetic	250
10	Dal	125.00	stationary	100
2	Rice	55.00	stationary	500
7	Meat Masala	55.00	stationary	100
NULL	NULL	NULL	NULL	NULL

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

54 • `select name,quantity from inventory where quantity> 10 ;`  
55

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
name	quantity		
Lipstick	250		
Snickers	1000		
Bisuits	500		
Ghee	75		
Meat Masala	100		
Chicken Masala	100		
Sugar	100		
Dal	100		

inventory 18 x

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

55 • `select sum(price) as Total_price from inventory ;`

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Total_price			
1825.00			

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

57

```
58 • select category,count(category) from inventory group by category;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
category	count(category)			
stationary	7			
Cosmetic	1			
sweets	1			
bakery	1			

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.

61

```
62 • insert into inventory (name,price,category,quantity) values("Tomato",120.00,"Vegeatble",0);
```

```
63 • select product_id,name,price from inventory where quantity <=0;
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
product_id	name	price			
11	Tomato	120.00			
NULL	NULL	NULL			

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)

70

```
67 • CREATE VIEW expensive_products AS
```

```
68 SELECT *
```

```
69 FROM inventory
```

```
70 WHERE price > (SELECT avg(price) from inventory);
```

71

```
72 • select * from inventory;
```

```
73 • select * from expensive_products;
```

74

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
product_id	name	price	category	quantity
3	Lipstick	545.00	Cosmetic	250
6	Ghee	845.00	stationary	75

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)

75 • `SELECT inventory.product_id,`  
76 `inventory.name,`  
77 `sum(orders.quantity_ordered) as total FROM inventory LEFT JOIN orders ON inventory.product_id=orders.product_id group by inventory.product_id having sum(orders.quantity_ordered)>8 ;`

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: `IF`

product_id	name	total
1	soap	20
2	Rice	30
4	Snickers	2
5	Biscuits	10
6	Ghee	2
7	Meat Masala	20

nt AI

Total Score : 10

## Queries Used

### #Formative Assessment 1

Create database global\_store\_db;

use global\_store\_db;

create table products (

product\_id INT auto\_increment primary key,

name VARCHAR(100),

price DECIMAL(10,2),

quantity INT

);

describe products;

create table orders (

order\_id INT auto\_increment primary key,

product\_id INT ,

quantity\_ordered INT,

order\_date DATE,

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

describe orders;

alter table products add category VARCHAR(50) after price ;

describe products;

alter table products rename to inventory;

```
describe products;
describe inventory;
insert into inventory (name,price,category,quantity)
values("soap",45.00,"stationary",100);
insert into inventory (name,price,category,quantity)
values("Sugar",45.00,"stationary",100);
insert into inventory (name,price,category,quantity)
values("Rice",55.00,"stationary",500);
insert into inventory (name,price,category,quantity)
values("Lipstick",545.00,"Cosmetic",250);
insert into inventory (name,price,category,quantity)
values("Snickers",10.00,"sweets",1000);
insert into inventory (name,price,category,quantity)
values("Bisuits",45.00,"bakery",500);
insert into inventory (name,price,category,quantity)
values("Dal",125.00,"stationary",100);
insert into inventory (name,price,category,quantity)
values("Ghee",845.00,"stationary",75);
insert into inventory (name,price,category,quantity) values("Meat
Masala",55.00,"stationary",100);
insert into inventory (name,price,category,quantity) values("Chicken
Masala",55.00,"stationary",100);
select * from inventory;
insert into orders (product_id,quantity_ordered,order_date)
values(1,10,"2024-06-10");
insert into orders (product_id,quantity_ordered,order_date)
values(2,10,"2024-06-10");
insert into orders (product_id,quantity_ordered,order_date)
values(4,1,"2024-06-10");
insert into orders (product_id,quantity_ordered,order_date)
values(5,5,"2024-06-15");
insert into orders (product_id,quantity_ordered,order_date)
values(6,1,"2024-06-19");
insert into orders (product_id,quantity_ordered,order_date)
values(7,10,"2024-06-20");
```



**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(1,10,"2024-06-28");**

**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(2,10,"2024-06-29");**

**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(4,1,"2024-06-30");**

**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(5,5,"2024-06-16");**

**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(6,1,"2024-06-21");**

**insert into orders (product\_id,quantity\_ordered,order\_date)  
values(7,10,"2024-06-22");**

**select \* from orders;**

**select distinct(category) from inventory;**

**Select \* from inventory order by price desc limit 5;**

**select name,quantity from inventory where quantity> 10 ;**

**select sum(price) as Total\_price from inventory ;**

**#Group products by their categories and display the count of products in each category.**

**select category,count(category) from inventory group by category;**

**#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.**

**insert into inventory (name,price,category,quantity)**

**values("Tomato",120.00,"Vegeatble",0);**

**select product\_id,name,price from inventory where quantity <=0;**

**Create view expensive\_products as SELECT name,price from inventory where price>=avg(price);**

**CREATE VIEW expensive\_products AS**

**SELECT \***

```
FROM inventory  
WHERE price > (SELECT avg(price) from inventory);
```

```
select * from inventory;  
select * from expensive_products;
```

```
SELECT inventory.product_id,  
       inventory.name,  
       sum(orders.quantity_ordered) as total FROM inventory LEFT JOIN  
orders ON inventory.product_id=orders.product_id group by  
inventory.product_id having sum(orders.quantity_ordered)>0 ;
```