

Date : 25/07/2024

Create Customer table

```
SQLQuery2.sql - PT...ODA\yuvraj.b (67) Day2.sql - PTSQLE...ODA\yuvraj.b (69)*
--Create Customer table

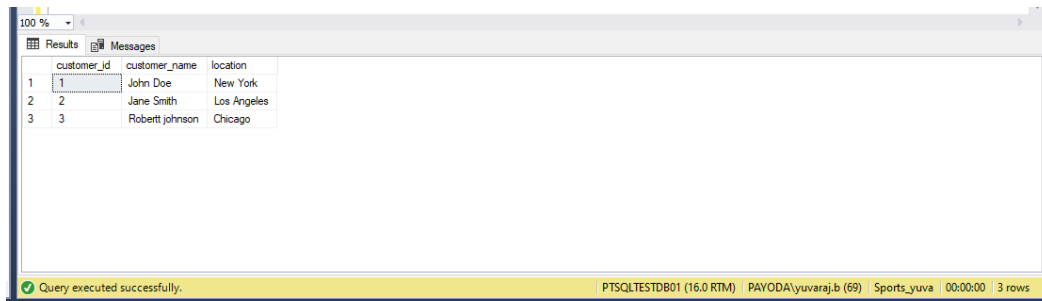
create table customer(
    customer_id int primary key,
    customer_name varchar(100),
    location varchar(100));

--insert sample data into customer

insert into customer values
(1,'John Doe','New York'),
(2,'Jane Smith','Los Angeles'),
(3,'Robertt johnson','Chicago');

select * from customer;
```

Output:



The screenshot shows the SQL Developer interface with the 'Results' tab selected. It displays a table with three columns: customer_id, customer_name, and location. The data is as follows:

	customer_id	customer_name	location
1	1	John Doe	New York
2	2	Jane Smith	Los Angeles
3	3	Robertt johnson	Chicago

At the bottom, a status bar indicates 'Query executed successfully.' and '3 rows'.

Create product table

```
SQLQuery2.sql - PT...ODA\yuvraj.b (67) Day2.sql - PTSQLE...ODA\yuvraj.b (69)*
--create product table

create table product(product_id int primary key,
    product_name varchar(100),
    category varchar(100),
    price int);

--insert sample data into product

insert into product values(1,'Mobile Phone','Electronics',500),
(2,'Bluetooth Speaker','Electronics',150),
(3,'Laptop','Computers',1200);

select * from product;
```

Output:



The screenshot shows the SQL Developer interface with the 'Results' tab selected. It displays a table with four columns: product_id, product_name, category, and price. The data is as follows:

	product_id	product_name	category	price
1	1	Mobile Phone	Electronics	500
2	2	Bluetooth Speaker	Electronics	150
3	3	Laptop	Computers	1200

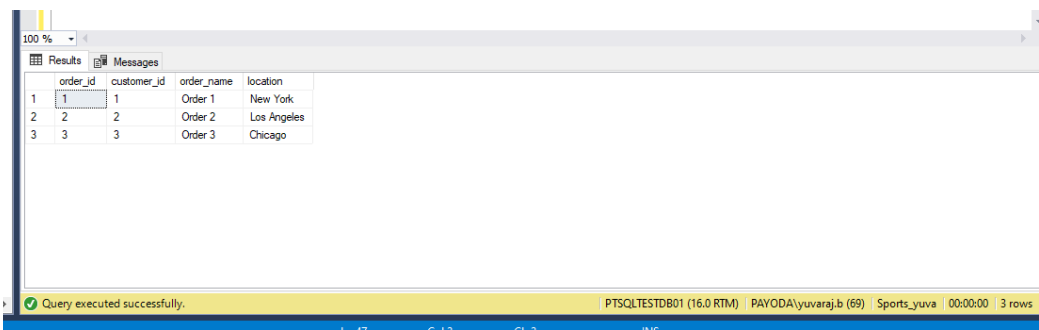
Create orders table

```
SQLQuery2.sql - PT...ODA\yuvvaraj.b (67) Day2.sql - PTSQLE...ODA\yuvvaraj.b (69)
--create order table
create table orders(order_id int primary key,
customer_id int,
order_name varchar(100),
location varchar(100),
foreign key(customer_id) references customer(customer_id));

--insert sample data into orders
insert into orders values(1,1,'Order 1','New York'),
(2,2,'Order 2','Los Angeles'),
(3,3,'Order 3','Chicago');

select * from orders;
```

Output:



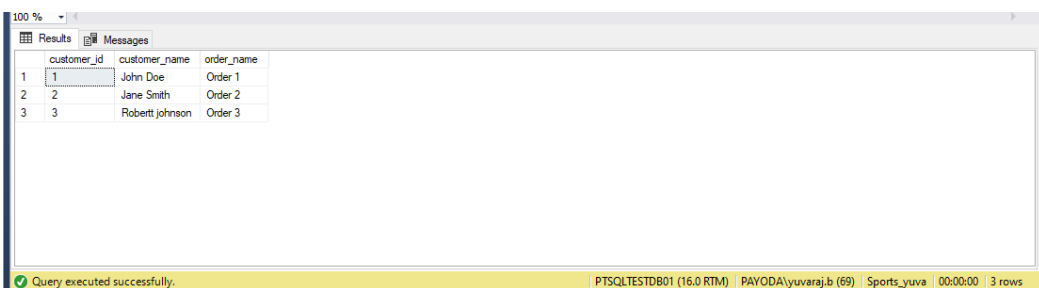
	order_id	customer_id	order_name	location
1	1	1	Order 1	New York
2	2	2	Order 2	Los Angeles
3	3	3	Order 3	Chicago

Query executed successfully. PTSQLESTDB01 (16.0 RTM) PAYODA\yuvvaraj.b (69) Sports_yuva 00:00:00 3 rows

1.Fetch all customer and display if there are any order names

```
--fetch all customer and display if there are any order names
select c.customer_id,c.customer_name,o.order_name from customer c left join orders o
on c.customer_id=o.customer_id;
```

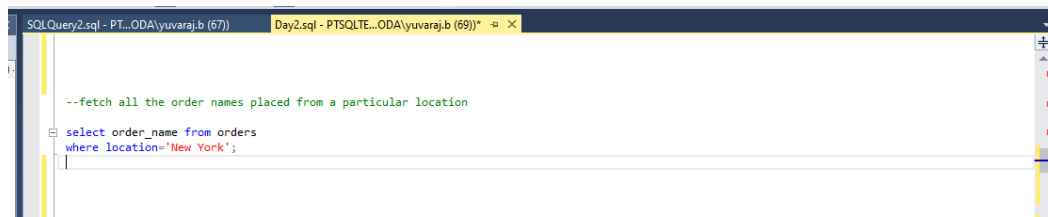
Output:



	customer_id	customer_name	order_name
1	1	John Doe	Order 1
2	2	Jane Smith	Order 2
3	3	Robert johnson	Order 3

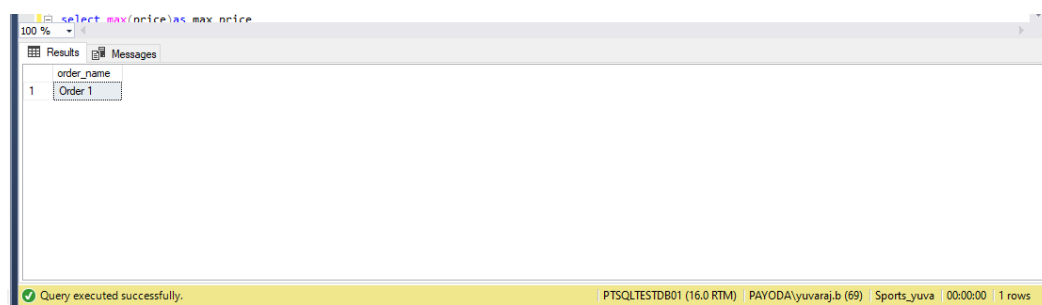
Query executed successfully. PTSQLESTDB01 (16.0 RTM) PAYODA\yuvvaraj.b (69) Sports_yuva 00:00:00 3 rows

2.Fetch all the order names placed from a particular location



```
--fetch all the order names placed from a particular location
select order_name from orders
where location='New York';
```

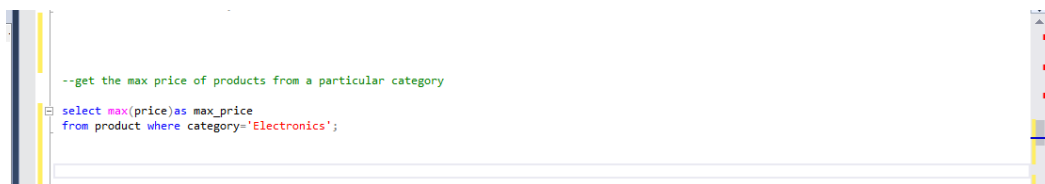
Output:



	order_name
1	Order 1

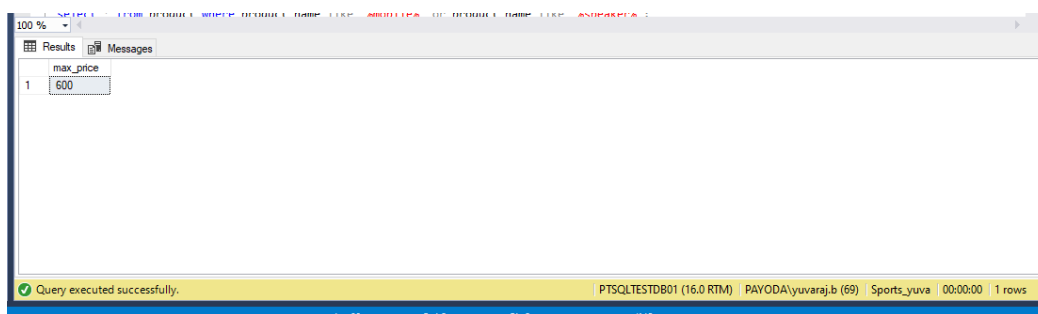
Query executed successfully. PTSQLESTDB01 (16.0 RTM) PAYODA\yuvvaraj.b (69) Sports_yuva 00:00:00 1 rows

3.Get the max price of products from a particular category



```
--get the max price of products from a particular category
select max(price)as max_price
from product where category='Electronics';
```

Output:



	max_price
1	600

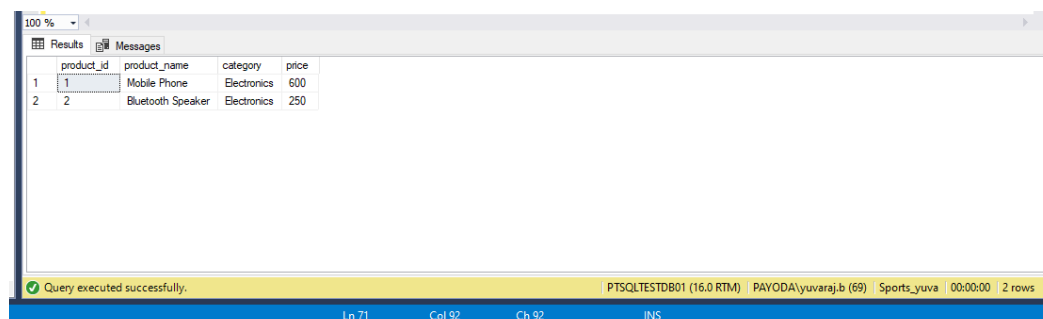
Query executed successfully. PTSQLESTDB01 (16.0 RTM) PAYODA\yuvvaraj.b (69) Sports_yuva 00:00:00 1 rows

4.Display any product with the product name like mobile, speaker



```
--display any product with the product name like mobile,speaker
select * from product where product_name like '%mobile%' or product_name like '%speaker%';
go
```

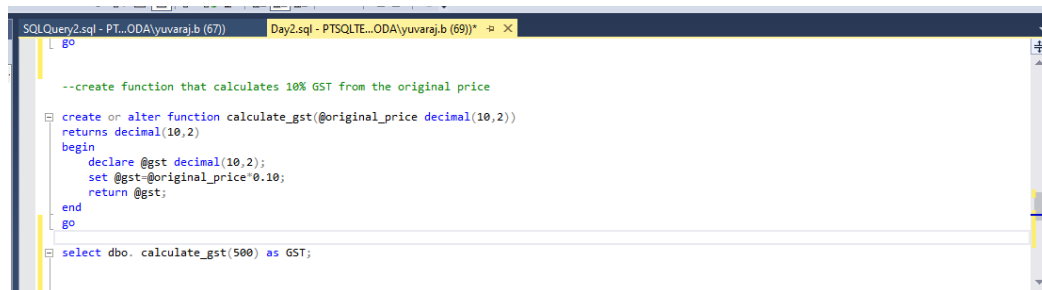
Output:



	product_id	product_name	category	price
1	1	Mobile Phone	Electronics	600
2	2	Bluetooth Speaker	Electronics	250

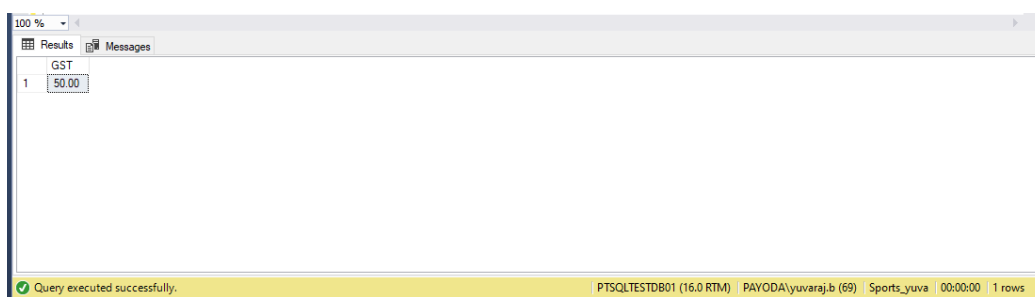
Query executed successfully. PTSQLESTDB01 (16.0 RTM) | PAYODA\yuvraj.b (69) | Sports_yuva | 00:00:00 | 2 rows

5.Create function that calculates 10% GST from the original price



```
--create function that calculates 10% GST from the original price
create or alter function calculate_gst(@original_price decimal(10,2))
returns decimal(10,2)
begin
    declare @gst decimal(10,2);
    set @gst=@original_price*0.10;
    return @gst;
end
go
select dbo. calculate_gst(500) as GST;
```

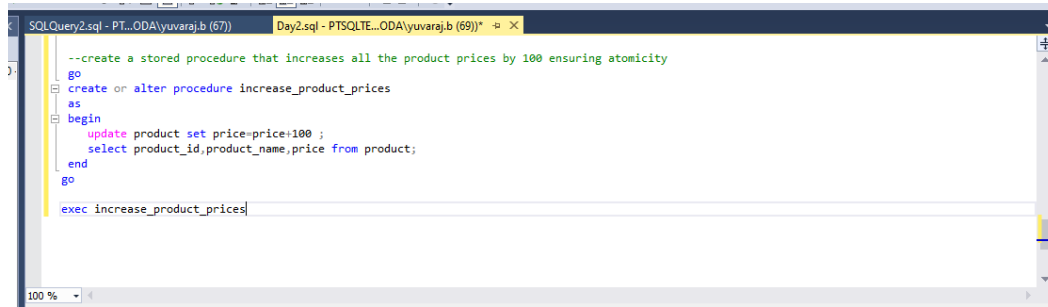
Output:



	GST
1	50.00

Query executed successfully. PTSQLESTDB01 (16.0 RTM) | PAYODA\yuvraj.b (69) | Sports_yuva | 00:00:00 | 1 rows

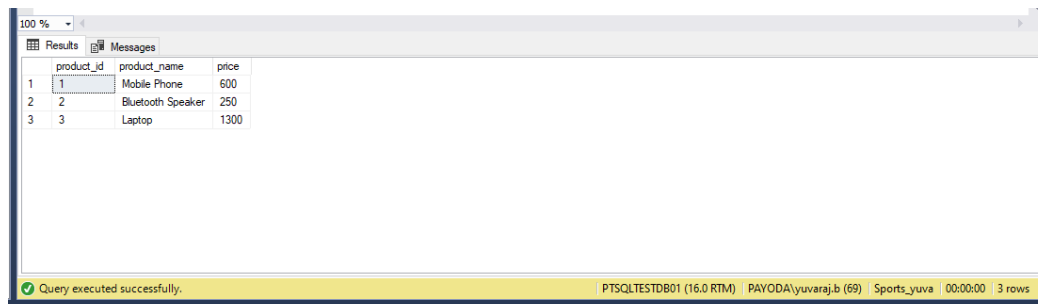
6. Create a stored procedure that increases all the product prices by 100 ensuring atomicity



The screenshot shows a SQL Server Enterprise Manager window with a query editor. The query is as follows:

```
--create a stored procedure that increases all the product prices by 100 ensuring atomicity
go
create or alter procedure increase_product_prices
as
begin
    update product set price=price+100 ;
    select product_id,product_name,price from product;
end
go
exec increase_product_prices
```

Output:



The screenshot shows the output of the stored procedure in the Results tab. The output is a table with 3 rows and 3 columns: product_id, product_name, and price.

	product_id	product_name	price
1	1	Mobile Phone	600
2	2	Bluetooth Speaker	250
3	3	Laptop	1300

At the bottom of the window, a status bar indicates: "Query executed successfully. PTSQLESTDB01 (16.0 RTM) PAYODA\yuvaraj.b (69) Sports_yuva 00:00:00 3 rows".