FUNCTION Practice Questions

```
CREATE TABLE Customers (
   CustomerId INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(100),
   Status VARCHAR(20)
);
-- 1. Function: Get Customer Full Name
CREATE OR REPLACE FUNCTION GetCustomerFullName(p_CustomerId INT)
RETURNS VARCHAR AS
$$
DECLARE
   FullName VARCHAR;
BEGIN
   SELECT FirstName || ' ' || LastName INTO FullName
   FROM Customers
   WHERE Customers.CustomerId = p_CustomerId;

   RETURN FullName;
END;
$$ LANGUAGE plpgsql;

INSERT INTO Customers (CustomerId, FirstName, LastName, Email, Status) VALUES
(1, 'sunil ', 'saini', 'sunil123@gmail.com', 'Active'),
(2, 'Mudit', 'pandit', 'Mudit24@gmail.com', 'Inactive'),
(3, 'Yesh', 'sharma', 'yesh45gmail.com', 'Active'),
(4, 'John', 'Doe', 'john.doe@example.com', 'Active'),
(5, 'Ravi', 'kumar', 'ravi.smith@example.com', 'Inactive'),
(6, 'Raja', 'sethi', 'raja.johnson@example.com', 'Active');
SELECT GetCustomerFullName(1);
```

	getcustomerfullname character varying
1	sunil saini

```
-- 2. Create Products table
CREATE TABLE Products (
   ProductId INT PRIMARY KEY,
   ProductName VARCHAR(100),
   Price NUMERIC(10, 2)
);

-- Sample Inserts (optional)
INSERT INTO Products VALUES
(101, 'Laptop', 50000.00),
(102, 'Mouse', 500.00),
(103, 'Keyboard', 1000.00),
(104, 'Monitor', 7000.00),
(105, 'Printer', 8500.00),
(106, 'USB Cable', 250.00);

-- 3. Create Orders table
CREATE TABLE Orders (
   OrderId INT PRIMARY KEY,
   CustomerId INT REFERENCES Customers(CustomerId),
   OrderDate DATE
);

INSERT INTO Orders VALUES
(201, 1, '2024-12-01'),
(202, 2, '2024-12-05'),
(203, 1, '2025-01-10'),
(204, 3, '2025-02-20'),
(205, 5, '2025-03-15'),
(206, 2, '2025-04-01');

-- 4. Create OrderDetails table
CREATE TABLE OrderDetails (
   OrderDetailId INT PRIMARY KEY,
   OrderId INT REFERENCES Orders(OrderId),
```

```
    ProductId INT REFERENCES Products(ProductId),
    Quantity INT
);

INSERT INTO OrderDetails VALUES
(301, 201, 101, 1),
(302, 201, 102, 2),
(303, 202, 104, 1),
(304, 203, 103, 1),
(305, 204, 105, 1),
(306, 205, 106, 3);

-- 2. Function to calculate the square of a number
CREATE OR REPLACE FUNCTION SquareNumber(numeric)
RETURNS numeric AS
$$
BEGIN
    RETURN $1 * $1;
END;
$$ LANGUAGE plpgsql;

SELECT SquareNumber(9);
```

	squarenumber numeric
1	81

```
-- 3. Function to return price after adding 18% GST
CREATE OR REPLACE FUNCTION PriceAfterGST(price NUMERIC)
RETURNS NUMERIC AS
$$
BEGIN
    RETURN price * 1.18;  -- Adding 18%
END;
$$ LANGUAGE plpgsql;
```

```
SELECT PriceAfterGST(5000);
```

	priceaftergst numeric
1	5900.00

```
-- 4. Function to count total orders for a given CustomerId
CREATE OR REPLACE FUNCTION TotalOrdersForCustomer(p_CustomerId INT)
RETURNS INT AS
$$
DECLARE
    order_count INT;
BEGIN
    SELECT COUNT(*) INTO order_count
    FROM Orders
    WHERE CustomerId = p_CustomerId;

    RETURN order_count;
END;
$$ LANGUAGE plpgsql;

SELECT TotalOrdersForCustomer(3);
```

	totalordersforcustomer integer
1	1

```
-- 5. Table-valued function to return all orders of a specific customer
CREATE OR REPLACE FUNCTION GetOrdersByCustomer(p_CustomerId INT)
RETURNS TABLE (OrderId INT, OrderDate DATE) AS
$$
BEGIN
    RETURN QUERY
    SELECT o.OrderId, o.OrderDate
```

```
    FROM Orders o
    WHERE o.CustomerId = p_CustomerId;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM GetOrdersByCustomer(3);
```

	orderid integer	orderdate date
1	204	2025-02-20

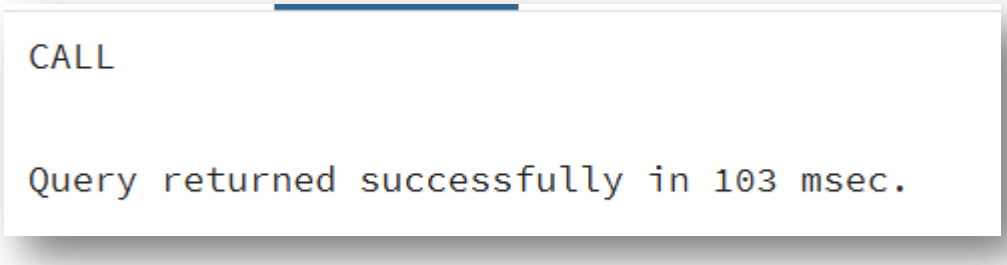
```
-- 6. Function to get the latest order date for a customer
CREATE OR REPLACE FUNCTION GetLatestOrderDate(p_CustomerId INT)
RETURNS DATE AS
$$
DECLARE
    latest_date DATE;
BEGIN
    SELECT MAX(OrderDate) INTO latest_date
    FROM Orders
    WHERE CustomerId = p_CustomerId;

    RETURN latest_date;
END;
$$ LANGUAGE plpgsql;
SELECT GetLatestOrderDate(1);
```

	getlatestorderdate date
1	2025-01-10

-

SP Practice Questions

```
-- 1. Insert a new product with name and price
CREATE OR REPLACE PROCEDURE InsertProduct(p_ProductName VARCHAR, p_Price NUMERIC)
LANGUAGE plpgsql
AS $$
BEGIN
   INSERT INTO Products(ProductId, ProductName, Price)
   VALUES (
      (SELECT COALESCE(MAX(ProductId), 100) + 1 FROM Products),
      p_ProductName,
      p_Price
   );
END;
$$;

CALL InsertProduct('Webcam', 1500.00);
```




```
-- 2. Get all products that cost more than a given amount

CREATE OR REPLACE PROCEDURE GetProductsAbovePrice(p_MinPrice NUMERIC)
LANGUAGE plpgsql
AS $$
DECLARE
   rec RECORD;
BEGIN
   RAISE NOTICE 'ProductId | ProductName | Price';
   FOR rec IN
      SELECT ProductId, ProductName, Price FROM Products WHERE Price > p_MinPrice
   LOOP
```

```
    RAISE NOTICE '% | % | %', rec.ProductId, rec.ProductName, rec.Price;
  END LOOP;
END;
$$;

CALL GetProductsAbovePrice(1000);
```

```
NOTICE:  ProductId | ProductName | Price
NOTICE:  101 | Laptop | 50000.00
NOTICE:  104 | Monitor | 7000.00
NOTICE:  105 | Printer | 8500.00
NOTICE:  107 | Webcam | 1500.00
NOTICE:  108 | Webcam | 1500.00
CALL

Query returned successfully in 74 msec.
```

```
-- 3. Return all active customers

CREATE OR REPLACE PROCEDURE GetActiveCustomers()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;  -- □ Declare the loop variable as a RECORD
BEGIN
  RAISE NOTICE 'CustomerId | Full Name | Email';

  FOR rec IN
    SELECT CustomerId, FirstName || ' ' || LastName AS FullName, Email
    FROM Customers
    WHERE Status = 'Active'
  LOOP
    RAISE NOTICE '% | % | %', rec.CustomerId, rec.FullName, rec.Email;
  END LOOP;
```

```
END;
$$;

CALL GetActiveCustomers();
```

```
NOTICE:  CustomerId | Full Name | Email
NOTICE:  3 | Yesh sharma | yesh45gmail.com
NOTICE:  4 | John Doe | john.doe@example.com
NOTICE:  6 | Raja sethi | raja.johnson@example.com
NOTICE:  1 | sunil  saini | new.sunil@gmail.com
CALL

Query returned successfully in 69 msec.
```

```
-- 4. Procedure with OUTPUT parameter to return total quantity of a product sold
CREATE OR REPLACE PROCEDURE GetTotalQuantitySold(p_ProductId INT, OUT total_qty INT)
LANGUAGE plpgsql
AS $$
BEGIN
   SELECT COALESCE(SUM(Quantity), 0) INTO total_qty
   FROM OrderDetails
   WHERE ProductId = p_ProductId;
END;
$$;

CALL GetTotalQuantitySold(101, NULL); -- Or use a DO block to display
```

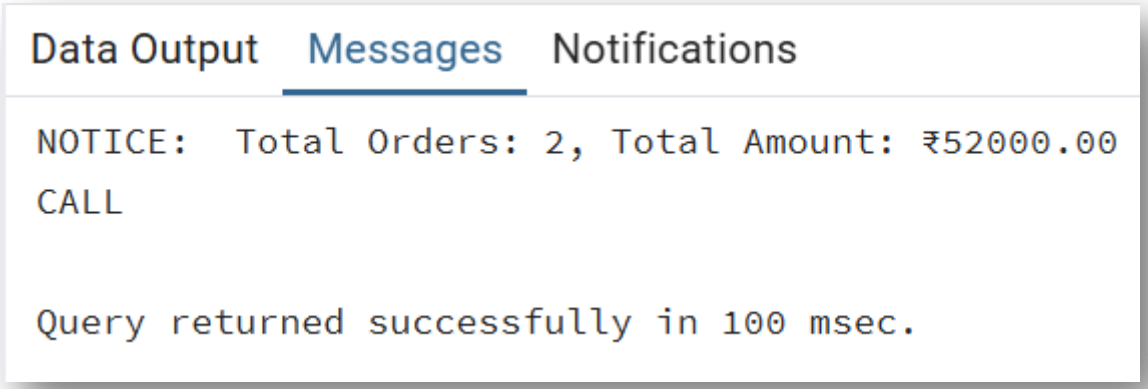
	total_qty integer
1	1

```
-- 5. Update the email of a customer using CustomerId
CREATE OR REPLACE PROCEDURE UpdateCustomerEmail(p_CustomerId INT, p_NewEmail VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
   UPDATE Customers
   SET Email = p_NewEmail
   WHERE CustomerId = p_CustomerId;

   IF NOT FOUND THEN
      RAISE NOTICE 'Customer ID % not found.', p_CustomerId;
   ELSE
      RAISE NOTICE 'Email updated for Customer ID %.', p_CustomerId;
   END IF;
END;
$$;

CALL UpdateCustomerEmail(1, 'new.sunil@gmail.com');
SELECT * FROM Customers
ORDER BY customerid;
```

	customerid [PK] integer	firstname character varying (50)	lastname character varying (50)	email character varying (100)	status character varying (20)
1	1	sunil	saini	new.sunil@gmail.com	Active
2	2	Mudit	pandit	Mudit24@gmail.com	Inactive
3	3	Yesh	sharma	yesh45gmail.com	Active
4	4	John	Doe	john.doe@example.com	Active
5	5	Ravi	kumar	ravi.smith@example.com	Inactive
6	6	Raja	sethi	raja.johnson@example.com	Active

```
-- 6. Return total orders and total amount for a customer

CREATE OR REPLACE PROCEDURE GetCustomerOrderSummary(p_CustomerId INT)
LANGUAGE plpgsql
AS $$
DECLARE
   total_orders INT;
```

```
    total_amount NUMERIC;
BEGIN
    SELECT COUNT(*) INTO total_orders
    FROM Orders
    WHERE CustomerId = p_CustomerId;

    SELECT COALESCE(SUM(od.Quantity * p.Price), 0) INTO total_amount
    FROM Orders o
    JOIN OrderDetails od ON o.OrderId = od.OrderId
    JOIN Products p ON od.ProductId = p.ProductId
    WHERE o.CustomerId = p_CustomerId;

    RAISE NOTICE 'Total Orders: %, Total Amount: ₹%', total_orders, total_amount;
END;
$$;

CALL GetCustomerOrderSummary(1);
```




```
-- finally end this code.....
```