**Practical Questions**

1. Write a scalar user-defined function to calculate the factorial of a given integer.

create or replace function Factorial(n INT)

returns bigint as $$

declare

result bigint :=1;

begin

while n>1 loop

result:=result\*n;

n:=n-1;

end loop;

return result;

end;

$$ language plpgsql;

select Factorial(5) as FactorialResult;

1. Create a table-valued function that returns all employees who have joined in the last 30 days.

create or replace function recent\_employees()

returns table(emp\_id int, emp\_name varchar,join\_date date )

as $$

begin

return query

select emp\_id, emp\_name, join\_date

from emp

where join\_date >= current\_date - interval '30 days';

end;

$$ language plpgsql;

select \* from recent\_employees();

1. Write a user-defined function that takes a string as input and returns the reverse of that string.

create or replace function rev\_string(input\_string varchar)

returns varchar as $$

declare

reversed\_string varchar :='';

i int;

begin

for i in reverse length(input\_string)..1 loop

reversed\_string:=reversed\_string || substring(input\_string from i for 1);

end loop;

return reversed\_string;

end;

$$ language plpgsql;

select rev\_string('Hello') as reversedString;

1. Implement a user-defined function to calculate the total sales for a specific product ID using data from a Sales table.

create or replace function total\_sales(product\_id int)

returns decimal as $$

declare

total decimal:=0;

begin

select coalesce(sum(quantity\*price),0) into total

from sales

where product\_id=product\_id;

return total;

end;

$$ language plpgsql;

select total\_sales(101) as TotalSalesAmount;

1. Write an inline table-valued function to return a list of all customers who have placed orders above a specific amount, where the amount is passed as a parameter.

create or replace function high\_value\_cust(min\_amount decimal)

returns table(customer\_id int, customer\_name varchar) as $$

begin

return query

select distinct c.customer\_id,c.customer\_name

from customers c

join orders o on c.customer\_id = o.customer\_id

where o.total\_amount > min\_amount;

end;

$$ language plpgsql;

select \* from high\_value\_cust(500);

**Scenario-Based Questions**

1. A company wants to standardize date formatting across its reports. Write a user-defined function to accept a date as input and return it in the format DD-MMM-YYYY.
2. Suppose you need to calculate the age of employees based on their birthdate. Write a scalar-valued function for this task.
3. In a database with a Products table, create a multi-statement table-valued function that categorizes products into low, medium, and high price ranges based on predefined thresholds.
4. Discuss the impact of user-defined functions on query performance. Suggest best practices for optimizing UDFs in SQL Server.
5. Write a user-defined function that calculates the number of weekdays (Monday-Friday) between two given dates.

**Advanced Questions**

1. Can a user-defined function call another user-defined function? Demonstrate this with an example.
2. Explain how you would handle errors in a user-defined function. Provide an example where you handle a divide-by-zero error in a scalar-valued function.
3. Write a multi-statement table-valued function that returns the Fibonacci series up to a given number.