6.Stored Procedures in MS-SQL Server.

❑ A SQL Server stored procedure groups one or more Transact-SQL statements into a logical unit and is stored as an object in the Database Server.

❑ When a stored procedure is called at the first time, SQL Server creates an execution plan and stores it in the plan cache.

❑ In the subsequent executions of the stored procedure, SQL Server reuses the plan so that the stored procedure can execute very fast with reliable performance.

Section 1: Getting started with SQL Server Stored Procedures:

❑To Create

❑To Execute

❑To Modify

❑To Drop

Creating a simple stored procedure:

The following SELECT statement returns a list of products from the products table in the BikeStores

```
SELECT
product_name,
list_price
FROM
production.products
ORDER BY
product_name;
```

```
CREATE PROCEDURE uspProductList
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    ORDER BY
        product_name;
END;
```

Execute a stored procedure:

```
EXECUTE sp_name;
```

```
EXEC sp_name;
```

```
EXEC uspProductList;
```

Modify a stored procedure:

```
ALTER PROCEDURE uspProductList
    AS
    BEGIN
        SELECT
            product_name,
            list_price
        FROM
            production.products
        ORDER BY
            list_price
    END;
```

Deleting a stored procedure:

To delete a stored procedure, you use the DROP PROCEDURE or DROP PROC statement

```
DROP PROCEDURE sp_name;
```

Creating a stored procedure with one parameter

```
CREATE PROCEDURE uspFindProducts(@min_list_price AS DECIMAL)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price
    ORDER BY
        list_price;
END;
```

Executing a stored procedure with one parameter

```
EXEC uspFindProducts 100;
```

Creating a stored procedure with multiple parameters:

The following statement modifies the uspFindProducts stored procedure by adding one more parameter named @max_list_price to it:

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL
    ,@max_list_price AS DECIMAL
)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        list_price <= @max_list_price
    ORDER BY
        list_price;
END;
```

```
EXECUTE uspFindProducts 900, 1000;
```

Using named parameters

```
EXECUTE uspFindProducts
    @min_list_price = 900,
    @max_list_price = 1000;
```

Creating optional parameters

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL = 0
    ,@max_list_price AS DECIMAL = 999999
    ,@name AS VARCHAR(max)
)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        list_price <= @max_list_price AND
        product_name LIKE '%' + @name + '%'
    ORDER BY
        list_price;
END;
```

```
EXECUTE uspFindProducts
    @name = 'Trek';
```

Creating output parameters

```
parameter_name data_type OUTPUT
```

```
CREATE PROCEDURE uspFindProductByModel (
    @model_year SMALLINT,
    @product_count INT OUTPUT
) AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        model_year = @model_year;

    SELECT @product_count = @@ROWCOUNT;
END;
```

```sql
DECLARE @count INT;

EXEC uspFindProductByModel
    @model_year = 2018,
    @product_count = @count OUTPUT;

SELECT @count AS 'Number of products found';
```

Section 2. Handling Exceptions

```sql
BEGIN TRY
    --- statements that may cause exceptions
END TRY
BEGIN CATCH
    -- statements to handle exception
    BEGIN TRY
        --- nested TRY block
    END TRY
    BEGIN CATCH
        --- nested CATCH block
    END CATCH
END CATCH
```

```
CREATE PROC usp_divide(
    @a decimal,
    @b decimal,
    @c decimal output
) AS
BEGIN
    BEGIN TRY
        SET @c = @a / @b;
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
GO

```

```
DECLARE @r decimal;
EXEC usp_divide 10, 2, @r output;
PRINT @r;
```