1. تحليل إنفاق العملاء

```
DECLARE @CustomerId INT = 1;
DECLARE @TotalSpent DECIMAL(10,2);

SELECT @TotalSpent = ISNULL(SUM(TotalAmount),0)
FROM Orders
WHERE CustomerID = @CustomerId;

IF @TotalSpent > 5000
   PRINT 'العميل مميز'
ELSE
   PRINT 'العميل عادي';
```

2. تقرير عتبة سعر المنتج

```
DECLARE @MinPrice DECIMAL(10,2) = 1500;
DECLARE @ProductCount INT;

SELECT @ProductCount = COUNT(*)
FROM Products
WHERE Price > @MinPrice;

PRINT 'Minimum Price: ' + CAST(@MinPrice AS VARCHAR)
   + ' | Products Count: ' + CAST(@ProductCount AS VARCHAR);
```

3. حاسبة أداء الموظفين

```
DECLARE @EmployeeId INT = 2;
DECLARE @Year INT = 2017;
DECLARE @TotalSales DECIMAL(10,2);

SELECT @TotalSales = ISNULL(SUM(TotalAmount),0)
FROM Orders
```

```
WHERE EmployeeID = @EmployeeId
AND YEAR(OrderDate) = @Year;

PRINT 'Total Sales: ' + CAST(@TotalSales AS VARCHAR);
```

4. المتغيرات العامة

```
PRINT @@SERVERNAME;
PRINT @@VERSION;
PRINT @@ROWCOUNT;
```

5. التحقق من المخزون

```
DECLARE @Quantity INT;

SELECT @Quantity = Quantity
FROM Inventory
WHERE ProductID = 1 AND StoreID = 1;

IF @Quantity > 20
   PRINT 'مخزون جيد'
ELSE IF @Quantity BETWEEN 10 AND 20
   PRINT 'مخزون متوسط'
ELSE
   PRINT 'المخزون منخفض - يلزم إعادة الطلب';
```

6. WHILE Loop لتحديث المخزون

```
DECLARE @Counter INT = 0;

WHILE @Counter < 3
BEGIN
   UPDATE TOP (3) Inventory
   SET Quantity = Quantity + 10
```

```
    WHERE Quantity < 5;


    PRINT 'تم تحديث دفعة';
    SET @Counter = @Counter + 1;
END;
```

7. تصنيف أسعار المنتجات

```
SELECT ProductName,
CASE
    WHEN Price < 300 THEN 'ميزانية'
    WHEN Price BETWEEN 300 AND 800 THEN 'متوسط'
    WHEN Price BETWEEN 801 AND 2000 THEN 'ممتاز'
    ELSE 'رفاهية'
END AS PriceCategory
FROM Products;
```

8. التحقق من وجود العميل

```
IF EXISTS (SELECT 1 FROM Customers WHERE CustomerID = 5)
BEGIN
    SELECT COUNT(*) AS OrdersCount
    FROM Orders
    WHERE CustomerID = 5;
END
ELSE
    PRINT 'العميل غير موجود';
```

9. دالة حساب الشحن

```
CREATE FUNCTION CalculateShipping (@Total DECIMAL(10,2))
RETURNS DECIMAL(10,2)
AS
BEGIN
    RETURN
```

```
   CASE
      WHEN @Total > 100 THEN 0
      WHEN @Total BETWEEN 50 AND 99 THEN 5.99
      ELSE 12.99
   END
END;
GO
```

10. دالة جدولية لنطاق السعر

```
CREATE FUNCTION GetProductsByPriceRange
(@MinPrice DECIMAL(10,2), @MaxPrice DECIMAL(10,2))
RETURNS TABLE
AS
RETURN
(
   SELECT ProductName, Price, Brand, Category
   FROM Products
   WHERE Price BETWEEN @MinPrice AND @MaxPrice
);
GO
```

11. ملخص مبيعات العميل السنوية

```
CREATE FUNCTION GetCustomerYearlySummary (@CustomerId INT)
RETURNS @Result TABLE
(
   OrderYear INT,
   TotalOrders INT,
   TotalSpent DECIMAL(10,2),
   AvgOrderValue DECIMAL(10,2)
)
AS
```

```
BEGIN
    INSERT INTO @Result
    SELECT YEAR(OrderDate),
           COUNT(*),
           SUM(TotalAmount),
           AVG(TotalAmount)
    FROM Orders
    WHERE CustomerID = @CustomerId
    GROUP BY YEAR(OrderDate);

    RETURN;
END;
GO
```

12. دالة حساب الخصم

```
CREATE FUNCTION CalculateBulkDiscount (@Quantity INT)
RETURNS INT
AS
BEGIN
    RETURN
    CASE
        WHEN @Quantity >= 10 THEN 15
        WHEN @Quantity >= 6 THEN 10
        WHEN @Quantity >= 3 THEN 5
        ELSE 0
    END
END;
GO
```

13. إجراء سجل طلبات العميل

```
CREATE PROCEDURE sp_GetCustomerOrderHistory
```

```
@CustomerId INT,
@StartDate DATE = NULL,
@EndDate DATE = NULL
AS
BEGIN
   SELECT *
   FROM Orders
   WHERE CustomerID = @CustomerId
   AND (@StartDate IS NULL OR OrderDate >= @StartDate)
   AND (@EndDate IS NULL OR OrderDate <= @EndDate);
END;
GO
```

14. إجراء إعادة تخزين

```
CREATE PROCEDURE sp_RestockProduct
@StoreId INT,
@ProductId INT,
@RestockQty INT,
@OldQty INT OUTPUT,
@NewQty INT OUTPUT
AS
BEGIN
   SELECT @OldQty = Quantity
   FROM Inventory
   WHERE StoreID = @StoreId AND ProductID = @ProductId;

   UPDATE Inventory
   SET Quantity = Quantity + @RestockQty
   WHERE StoreID = @StoreId AND ProductID = @ProductId;
```

```
    SELECT @NewQty = Quantity
    FROM Inventory
    WHERE StoreID = @StoreId AND ProductID = @ProductId;
END;
GO
```

15. إجراء معالجة طلب (مبسط)

```
CREATE PROCEDURE sp_ProcessNewOrder
@CustomerId INT,
@ProductId INT,
@Quantity INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN;

        INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
        VALUES (@CustomerId, GETDATE(), @Quantity * 100);

        COMMIT;
    END TRY
    BEGIN CATCH
        ROLLBACK;
        PRINT 'حدث خطأ أثناء تنفيذ الطلب';
    END CATCH
END;
GO
```