## 1. BEGIN TRANSACTION:

```
BEGIN TRANSACTION
UPDATE aisles SET aisle=”new value 2” where aisle_id=2;
SAVE TRANSACTION t1
DELETE FROM aisles where aisle_id=2
ROLLBACK TRANSACTION t1
COMMIT
```

---

```
BEGIN TRANSACTION

INSERT INTO products VALUES (99234, 'Vitamin Johnson & Johnson', 16, 13);

INSERT INTO products VALUES (99235, 'Oil Johnson & Johnson', 20, 13);

ROLLBACK
```

## 2. BEGIN-END

```
BEGIN

        SELECT user_id, days_since_prior order

        FROM orders

        WHERE order_id=4;

END;
```

---

```
BEGIN

        SELECT product_id, product_name, aisle_id, department_id

        FROM   products

        WHERE product_id = 9235;

END;
```

---

## 3. IF-ELSE

```
DECLARE @NumAisles int;

SET @NumAisles =  (SELECT COUNT (DISTINCT aisle_id) FROM aisles)

IF @NumAisles > 100

    PRINT ‘There are more than 100 aisles’;

ELSE

    BEGIN

        IF @NumAisles < 50

            PRINT ‘There are less than 50 aisles’;

        ELSE

            PRINT ‘There are more than 50 but less than 100 aisles’;

    END;
```

---

```
DECLARE @order INT;

SET @order = (SELECT COUNT(order_id) FROM orders);

BEGIN

    IF @order < 50

        PRINT ‘Shipping on hold‘;

    ELSE

        PRINT 'Orders are ready for shipping';


END;
```

---

# 4. WHILE-BREAK CONTINUE

```
WHILE (SELECT  MAX(product_id) FROM products) < 50
    BEGIN
        UPDATE products
        SET product_name = “Less than 50”
        IF (SELECT COUNT(product_id) FROM products) >20
            BREAK
        ELSE
            CONTINUE
    END;
```

---

```
WHILE (SELECT COUNT(reorder) FROM order_products where reorder =0) > 0
    BEGIN
        UPDATE order_products SET reorder = 1;
        IF (SELECT COUNT(order_id) FROM orders where order_id>39) > 500
            BREAK
        ELSE
            CONTINUE
    END;
```

## 5. GOTO

```
DECLARE @VARIABLE1 INT;
SET @VARIABLE1 = 1;
WHILE @VARIABLE1 < 20
BEGIN
        SELECT @VARIABLE1
        SET @VARIABLE1 = @VARIABLE1 * 2
        IF @VARIABLE1 = 4 GOTO Branch One
        IF @VARIABLE1 = 8 GOTO Branch Two
END;
Branch One:
        SELECT ‘Jumping To Branch One’
Branch Two:
        SELECT ‘Jumping To Branch Two’
```

```
DECLARE @Counter int;
SET @Counter = 0;
BEGIN
  SELECT @Counter = COUNT(aisles_id) from aisles;
  SELECT @Counter;
  IF @Counter = 40 GOTO B_ONE
  IF @Counter = 15 GOTO B_TWO
END
B_ONE:
  PRINT 'Too many aisles';
B_TWO:
  PRINT 'Few aisles';
```

## 6. RETURN

```
IF (SELECT COUNT(*) FROM aisles) < 20
        PRINT ‘Less than 20’;
ELSE
        RETURN;
```

```
CREATE PROCEDURE productCount
AS
IF (SELECT COUNT(*) FROM Products) > 0
  RETURN 1;
ELSE
  RETURN 2;
```

## 7. CASE

```
SELECT product_id,
   CASE
     WHEN add_to_cart_order > 0 and add_to_cart_order < 30 THEN 'Low'
     WHEN add_to_cart_order > 29 and add_to_cart_order < 50 THEN 'Medium'
     ELSE 'High'
   END
FROM order_products;
```

```
SELECT   order_id, product_id, reorder=
    CASE
      WHEN reorder =  0 THEN 'no reorder'
      WHEN reorder =  1 THEN 'reorder'
      ELSE 'no information'
    END
FROM order_products;
```

## 8. WAITFOR

```
BEGIN
          WAITFOR TIME ’18:10’;
          EXECUTE procedure1;
END;
```

---

```
BEGIN
   WAITFOR DELAY '02:00';
   EXECUTE productCount ;
END;
```

## 9. TRY..CATCH

```
BEGIN TRY
          SELECT count(*) from products;
   THROW 1000, 'Error raised in TRY block.', 16;
END TRY
BEGIN CATCH
          SELECT count(product_id) from products;
END CATCH;
```

---

```
BEGIN TRANSACTION;
BEGIN TRY
  DELETE FROM products
  WHERE products_id = 39235;
END TRY
BEGIN CATCH
  SELECT
    ERROR_NUMBER() AS ErrorNumber
    ,ERROR_SEVERITY() AS ErrorSeverity
    ,ERROR_STATE() AS ErrorState
    ,ERROR_PROCEDURE() AS ErrorProcedure
    ,ERROR_LINE() AS ErrorLine
    ,ERROR_MESSAGE() AS ErrorMessage;
  IF @@TRANCOUNT > 0
    ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
  COMMIT TRANSACTION;
GO
```

## 10. THROW

```
BEGIN TRY
        DECLARE @A as int =10;
        SELECT @A/0
END TRY
BEGIN CATCH
        THROW;
        PRINT ‘AFTER THROW’;
END CATCH
```

```
BEGIN TRANSACTION;
BEGIN TRY
   DELETE FROM products
   WHERE products_id = 39235;
END TRY
BEGIN CATCH
   PRINT 'In catch block.';
   THROW;
END CATCH;
```

## 11. USER DEFINED FUNCTIONS:

```
CREATE FUNCTION getaislename
(
        @aisle_id_input int
)
returns varchar(30)
as
BEGIN
        DECLARE @output varchar(30);
        SET @output = (SELECT aisle FROM aisles WHERE aisle_id=@aisle_id_input);
        return @output;
END;

SELECT dbo.getaislename(4) as ‘Aisle’
```

```
CREATE FUNCTION GetProductDepartment (product_id INT)
  RETURNS INT
  AS
  BEGIN
    DECLARE pdi INT;
    SET pdi = (SELECT ProdPrice FROM V1Table WHERE Id = product_id);
  RETURN pdi;
 END
```

## 12. Stored Procedures

```
CREATE PROCEDURE spGetProductName
AS
BEGIN
        SELECT product_name FROM products;
END;

Exec spGetProductName
```

```
CREATE PROCEDURE SelectProducts (tbl nvarchar(100))
AS
Declare sqls varchar (40000)
Set sqls='SELECT * FROM ' + tbl
execute(sqls);
```

## 13. Triggers

```
Create trigger  tr_aisles_InsteadOfInsert
on aisles
Instead Of Delete
as
BEGIN
        UPDATE a
        SET a.product_id=1
        FROM pproducts a, deleted b
WHERE a.product_id=b.product_id
END;
```

```
CREATE TRIGGER reminder
ON products
AFTER INSERT, DELETE
AS RAISERROR ('Notify products Relations', 16, 10);
GO
```

## 14. Cursors

```
DECLARE @id int
DECLARE @name varchar(50)

DECLARE ProductCursor CURSOR FOR
SELECT product_id, product_name FROM products WHERE product_id <=200
Open ProductCursor
Fetch Next FROM ProductCursor into @id, @name
WHILE(@@FETCH_STATUS=0)
BEGIN
        PRINT ‘ID = ‘+Cast(@id as nvarchar(10)+ ‘ Name= ‘+@name
        Fetch Next FROM ProductCursor into @id, @name
END
Close ProductCursor
DEALLOCATE ProductCursor
```

---

```
DECLARE @ProductId INT
DECLARE MyCursor CURSOR FOR SELECT ProductId From products
OPEN MyCursor

FETCH NEXT FROM MyCursor INTO @ProductId

WHILE @@FETCH_STATUS = 0
BEGIN
        PRINT @ProductId
        FETCH NEXT FROM MyCursor INTO @ProductId
END
CLOSE @ProductId
DEALLOCATE MyCursor
```