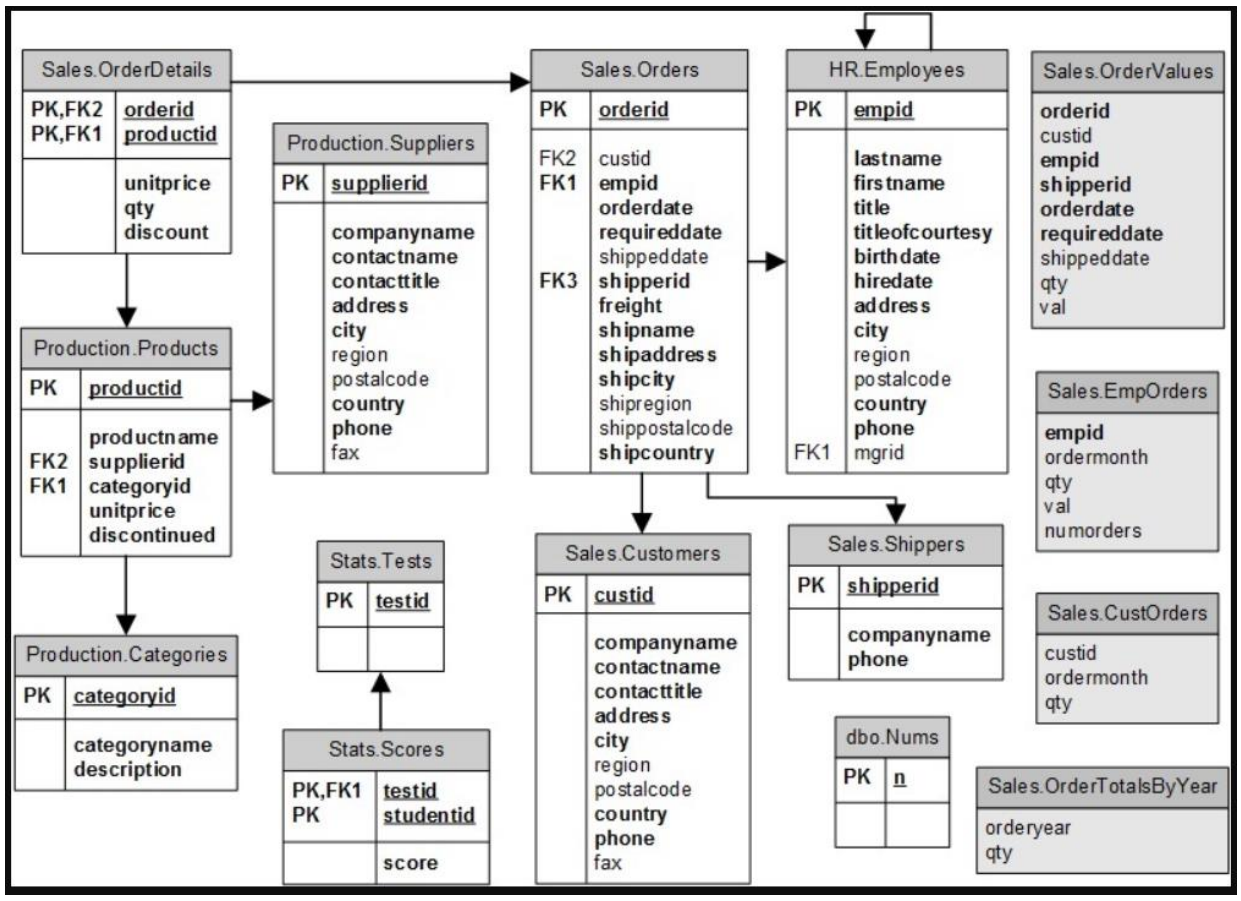
/*

Neba Nfonsang

Project 8: Queries that use variables, flow elements, cursors, temporary tables, UDFs, and stored procedures.

*/

```
USE TSQLV4
```



```
/*
Query # 1: Store results of a subquery in a
variable
*/

DECLARE @empname AS NVARCHAR(61);

SET @empname = (SELECT firstname + N' ' +
lastname
                FROM HR.Employees
                WHERE empid = 3);

SELECT @empname AS empname;
GO

/*
Query # 2: Using the SET command to assign one
variable at a time

*/
```

```
DECLARE @firstname AS NVARCHAR(20), @lastname AS
NVARCHAR(40); -- It is better to declare these
on separate lines for readability

SET @firstname = (SELECT firstname
                  FROM HR.Employees
                  WHERE empid = 3);
SET @lastname = (SELECT lastname
                  FROM HR.Employees
                  WHERE empid = 3);

SELECT @firstname AS firstname, @lastname AS
lastname;
GO
```

**/*
Query # 3: Using the SELECT command to assign multiple variables in the same statement
Using the SELECT command to assign multiple variables in the same statement
*/**

```
DECLARE @firstname AS NVARCHAR(20), @lastname AS
NVARCHAR(40); -- It is better to declare these
on separate lines for readability

SELECT
  @firstname = firstname,
  @lastname  = lastname
FROM HR.Employees
WHERE empid = 3;

SELECT @firstname AS firstname, @lastname AS
lastname;
GO

/*
Query # 4: Batches and Variables
*/

---variable
DECLARE @i AS INT;
SET @i = 10;
-- batch
PRINT @i;
GO
```

```
/*
Query # 5: The GO n Option
*/


-- Create T1 with identity column
DROP TABLE IF EXISTS dbo.T1;
CREATE TABLE dbo.T1(col1 INT IDENTITY CONSTRAINT
PK_T1 PRIMARY KEY);
GO

-- Suppress insert messages
SET NOCOUNT ON;
GO

-- Execute batch 100 times
INSERT INTO dbo.T1 DEFAULT VALUES;
GO 100
```

```
SELECT col1 FROM dbo.T1;
```

```
/*
Query # 6: The IF ... ELSE Flow Element
*/

IF YEAR(SYSDATETIME()) <> YEAR(DATEADD(day, 1,
SYSDATETIME()))
  PRINT 'Today is the last day of the year.';
ELSE
  PRINT 'Today is not the last day of the
year.';
GO

/*
Query # 7: IF ELSE IF
*/

IF YEAR(SYSDATETIME()) <> YEAR(DATEADD(day, 1,
SYSDATETIME()))
  PRINT 'Today is the last day of the year.';
```

```
ELSE
  IF MONTH(SYSDATETIME()) <> MONTH(DATEADD(day,
1, SYSDATETIME()))
    PRINT 'Today is the last day of the month
but not the last day of the year.';
  ELSE
    PRINT 'Today is not the last day of the
month.';
GO

/*
Query # 8: Statement Block
*/

IF DAY(SYSDATETIME()) = 1
BEGIN
  PRINT 'Today is the first day of the month.';
  PRINT 'Starting first-of-month-day process.';
  /* ... process code goes here ... */
  PRINT 'Finished first-of-month-day database
process.';
END;
ELSE
BEGIN
```

```
  PRINT 'Today is not the first day of the
month.';
  PRINT 'Starting non-first-of-month-day
process.';
  /* ... process code goes here ... */
  PRINT 'Finished non-first-of-month-day
process.';
END;
GO
```

**/*
Query # 9:
*/**

```
-- The WHILE Flow Element
DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
  PRINT @i;
  SET @i = @i + 1;
END;
GO
```

```
-- BREAK
DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
  IF @i = 6 BREAK;
  PRINT @i;
  SET @i = @i + 1;
END;
GO
```

**/*
Query # 10: A cursor code that calculates the running total quantity for each customer and month from the Sales.CustOrders view:
*/**

```
SET NOCOUNT ON;

DECLARE @Result AS TABLE
(
  custid     INT,
  ordermonth DATE,
  qty        INT,
  runqty     INT,
```

```
  PRIMARY KEY(custid, ordermonth)
);

DECLARE
  @custid     AS INT,
  @prvcustid  AS INT,
  @ordermonth AS DATE,
  @qty        AS INT,
  @runqty     AS INT;

DECLARE C CURSOR FAST_FORWARD /* read only,
forward only */ FOR
  SELECT custid, ordermonth, qty
  FROM Sales.CustOrders
  ORDER BY custid, ordermonth;

OPEN C;

FETCH NEXT FROM C INTO @custid, @ordermonth,
@qty;

SELECT @prvcustid = @custid, @runqty = 0;

WHILE @@FETCH_STATUS = 0
```

```
BEGIN
  IF @custid <> @prvcustid
    SELECT @prvcustid = @custid, @runqty = 0;

  SET @runqty = @runqty + @qty;

  INSERT INTO @Result VALUES(@custid,
@ordermonth, @qty, @runqty);

  FETCH NEXT FROM C INTO @custid, @ordermonth,
@qty;
END;

CLOSE C;

DEALLOCATE C;

SELECT
  custid,
  CONVERT(VARCHAR(7), ordermonth, 121) AS
ordermonth,
  qty,
  runqty
FROM @Result
```

```
ORDER BY custid, ordermonth;
```

```
/*
Query # 11: Using a local temporary table
*/

DROP TABLE IF EXISTS #MyOrderTotalsByYear;
GO

CREATE TABLE #MyOrderTotalsByYear
(
  orderyear INT NOT NULL PRIMARY KEY,
  qty       INT NOT NULL
);

INSERT INTO #MyOrderTotalsByYear(orderyear, qty)
  SELECT
    YEAR(O.orderdate) AS orderyear,
    SUM(OD.qty) AS qty
  FROM Sales.Orders AS O
    INNER JOIN Sales.OrderDetails AS OD
```

```
      ON OD.orderid = O.orderid
  GROUP BY YEAR(orderdate);

SELECT Cur.orderyear, Cur.qty AS curyearqty,
Prv.qty AS prvyearqty
FROM #MyOrderTotalsByYear AS Cur
  LEFT OUTER JOIN #MyOrderTotalsByYear AS Prv
    ON Cur.orderyear = Prv.orderyear + 1;


/*
Query # 12: Dynamic SQL
*/

DECLARE @sql AS VARCHAR(100);
SET @sql = 'PRINT ''A dynamic SQL batch
message.'';';
EXEC(@sql);

/*
Query # 13: Create a User-Defined Function
*/

DROP FUNCTION IF EXISTS dbo.GetAge;
```

```
GO

CREATE FUNCTION dbo.GetAge
(
  @birthdate AS DATE,
  @eventdate AS DATE
)
RETURNS INT
AS
BEGIN
  RETURN
    DATEDIFF(year, @birthdate, @eventdate)
    - CASE WHEN 100 * MONTH(@eventdate) +
DAY(@eventdate)
               < 100 * MONTH(@birthdate) +
DAY(@birthdate)
           THEN 1 ELSE 0
      END;
END;
GO

-- call the function to calculate the age of
each employee today:
SELECT
```

```
  empid, firstname, lastname, birthdate,
  dbo.GetAge(birthdate, SYSDATETIME()) AS age
FROM HR.Employees;


/*
Query # 14: creates a stored procedure called
Sales.GetCustomerOrders.
*/

DROP PROC IF EXISTS Sales.GetCustomerOrders;
GO

CREATE PROC Sales.GetCustomerOrders
  @custid   AS INT,
  @fromdate AS DATETIME = '19000101',
  @todate   AS DATETIME = '99991231',
  @numrows  AS INT OUTPUT
AS
SET NOCOUNT ON;

SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE custid = @custid
```

```
  AND orderdate >= @fromdate
  AND orderdate < @todate;

SET @numrows = @@rowcount;
GO

DECLARE @rc AS INT;

EXEC Sales.GetCustomerOrders
  @custid   = 1,
  @fromdate = '20150101',
  @todate   = '20160101',
  @numrows  = @rc OUTPUT;

SELECT @rc AS numrows;
```