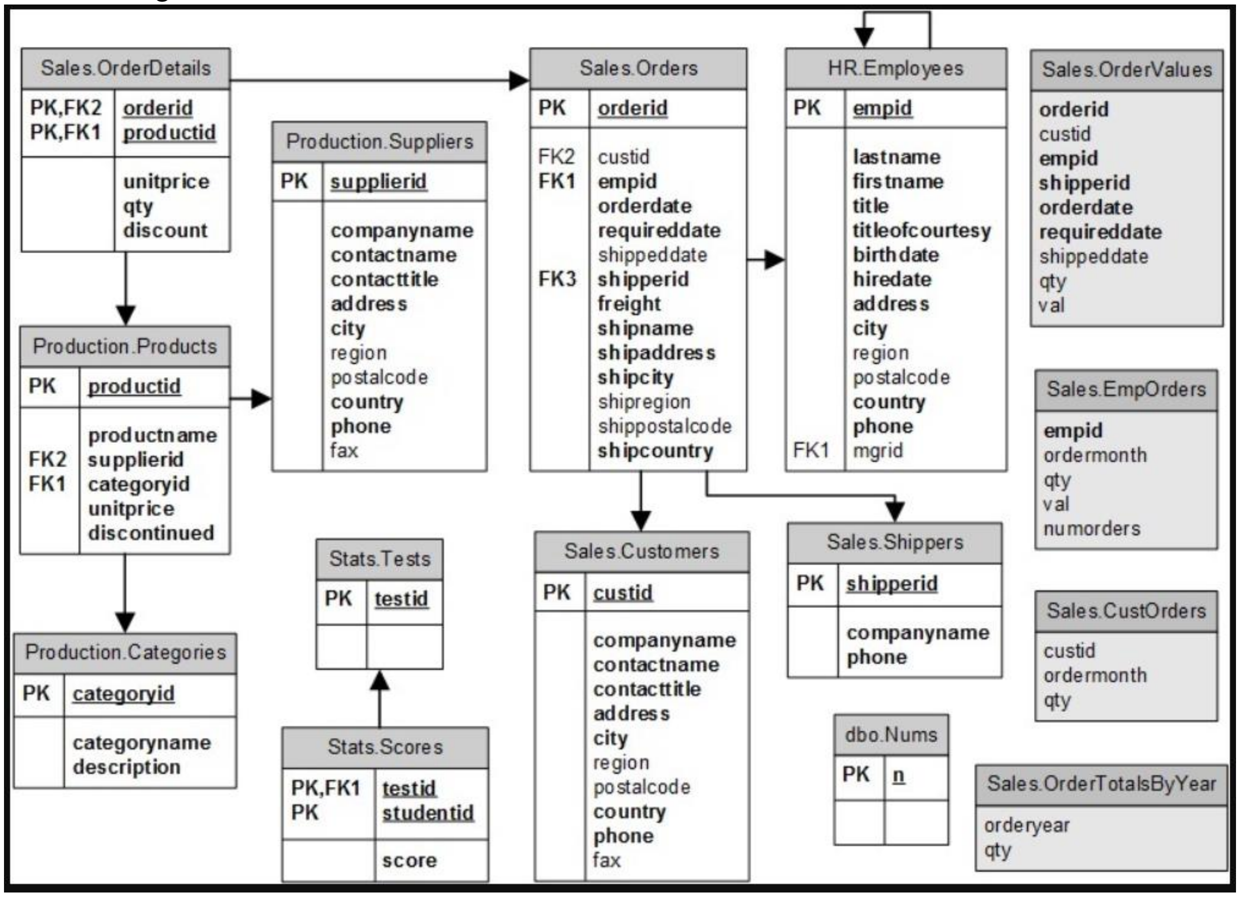
```
/*
Neba Nfonsang
Project 3: 20 different queries that use table
expressions or use views
*/

USE TSQLV4
```



```
/*
Query # 1: Count unique customers for each year
*/

SELECT orderyear, COUNT(DISTINCT custid) AS numcust
FROM
    (SELECT YEAR(orderdate) AS orderyear, custid
    FROM Sales.Orders)
AS CustomerCount
GROUP BY orderyear;

/*
Query # 2: Using table expressions with local
variable or argument. Count the number of distinct
customers by year, served by employee with emplid=3
*/

DECLARE @empid AS INT = 3;

SELECT orderyear, COUNT(DISTINCT custid) AS numcust
FROM
    (SELECT YEAR(orderdate) AS orderyear, custid
    FROM Sales.Orders
    WHERE empid = @empid)
AS CustomerCount
GROUP BY orderyear;

/*
Query # 3: Nesting:
*/
SELECT orderyear, numcusts
FROM
    (SELECT orderyear, COUNT(DISTINCT custid) AS
numcusts
```

```
    FROM
        (SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS D1 -- to assign alias
to order year
    GROUP BY orderyear) AS D2 -- to count and assign
to numcust
WHERE numcusts > 50;
```

/*
Query # 4: Calculate difference between number of customers handled in the current and previous years
***/**

```
SELECT Cur.orderyear,
    Cur.numcusts AS curnumcusts, Prv.numcusts AS
prvnumcusts,
    Cur.numcusts - Prv.numcusts AS growth
FROM
    (SELECT YEAR(orderdate) AS orderyear,
    COUNT(DISTINCT custid) AS numcusts
        FROM Sales.Orders
        GROUP BY YEAR(orderdate)) AS Cur
     LEFT OUTER JOIN
        (SELECT YEAR(orderdate) AS orderyear,
            COUNT (DISTINCT custid) AS numcusts
        FROM Sales.Orders
        GROUP BY YEAR(orderdate)) AS Prv
        ON Cur.orderyear = Prv.orderyear + 1;
```

```
/*
Query # 5: Common table expression
*/

WITH USAcusts AS
    (
    SELECT custid, companyname
    FROM Sales.Customers
    WHERE country = N'USA'
    )
    SELECT custid, companyname
    FROM USACusts;

/*
Query # 6: common table expression
*/

WITH CustomerCount AS
    (
    SELECT YEAR(orderdate) AS orderyear, custid
    FROM Sales.Orders
    )
SELECT orderyear,COUNT(DISTINCT custid) AS numcusts
FROM CustomerCount
GROUP BY orderyear;
```

```
/*
Query # 7: Count number of transactions handled by
employees
*/

WITH CustomerCount(orderyear, empid) AS
    (
    SELECT YEAR(orderdate) AS orderyear, empid
    FROM Sales.Orders
    )
SELECT orderyear,COUNT(empid) AS numtransaction
FROM CustomerCount
GROUP BY orderyear;


/*
Query # 8: Number of transactions handled by
employee with empid=5, each year
*/

DECLARE @employeeid AS INT = 5;

WITH CustomerCount(orderyear, empid) AS
    (
    SELECT YEAR(orderdate) AS orderyear, empid
    FROM Sales.Orders
    WHERE empid = @employeeid
    )
SELECT orderyear,COUNT(empid) AS numtransaction
FROM CustomerCount
GROUP BY orderyear;
```

```
/*
Query # 9: # avoiding nesting by using multiple CTEs
*/

WITH C1 AS
(
    SELECT YEAR(orderdate) AS orderyear, custid
    FROM Sales.Orders
),
C2 AS
(
    SELECT orderyear, COUNT(DISTINCT custid) AS
numcusts
    FROM C1
    GROUP BY orderyear
)
SELECT orderyear, numcusts
FROM C2
WHERE numcusts > 50;


/*
Query # 10: Multiple Refrences in CTEs
*/

WITH YearlyCount AS
(
    SELECT YEAR(orderdate) AS orderyear,
        COUNT(DISTINCT custid) AS numcusts
    FROM Sales.Orders
    GROUP BY YEAR(orderdate)
)
SELECT Cur.orderyear,
    Cur.numcusts AS Curnumcusts, Prv.numcusts AS
prvnumcusts,
    Cur.numcusts - Prv.numcusts AS growth
FROM YearlyCount AS Cur
        LEFT OUTER JOIN YearlyCount AS Prv
            ON Cur.orderyear = Prv.orderyear + 1;


/*
Query # 11: Recursive CTEs
*/

WITH EmpsCTE AS
(
    SELECT empid, mgrid, firstname, lastname
    FROM HR.Employees
    WHERE empid = 2

    UNION ALL

    SELECT C.empid, C.mgrid, C.firstname, C.lastname
    FROM EmpsCTE AS P
        INNER JOIN HR.Employees AS C
        ON C.mgrid = P.empid
    )
    SELECT empid, mgrid, firstname, lastname
    FROM EmpsCTE;
```

```
/*
Query # 12: views
*/

DROP VIEW IF EXISTS Sales.USACusts;
GO
CREATE VIEW Sales.USACusts
AS
SELECT  custid, companyname, contactname,
    contacttitle, address,  city, region,
    postalcode, country, phone, fax
FROM Sales.Customers
WHERE country = N'USA';
GO

SELECT custid, companyname
FROM Sales.USACusts;

/*
Query # 13: ORDER BY should be used in the outer
clause OR should be used with the TOP or OFFSET-
FETCH for a view
*/

ALTER VIEW Sales.USACusts
AS

SELECT TOP(20) PERCENT
    custid, companyname, contactname,
    contacttitle, address,  city, region,
    postalcode, country, phone, fax
FROM Sales.Customers
WHERE country = N'USA'
ORDER BY region;
```

```sql
GO

SELECT custid, companyname, region
FROM Sales.USACusts;

-- this works but the order is not guaranteed
--since ORDER BY is not in the outer query


/*
Query # 14: USing OFFSET-FETCH with views
*/
ALTER VIEW Sales.USACusts
AS

SELECT
    custid, companyname, contactname,
    contacttitle, address,  city, region,
    postalcode, country, phone, fax
FROM Sales.Customers
WHERE country = N'USA'
ORDER BY region
OFFSET 2 ROWS
FETCH NEXT 5 ROWS ONLY;
GO

SELECT custid, companyname, region
FROM Sales.USACusts;

-- drop views to clean up

DELETE FROM Sales.Customers
WHERE custid > 91;
DROP VIEW IF EXISTS Sales.USACusts;
```

```
/*
Query # 15: Inline Table-Valued Functions
*/

-- A function is created with parameter @cid to take
customer id as input
--The function returns a table where customer id is
equivalent to the input customer id

DROP FUNCTION IF EXISTS dbo.GetCustOrders;
GO
CREATE FUNCTION dbo.GetCustOrders
    (@cid AS INT) RETURNS TABLE
AS
RETURN
    SELECT orderid, custid, empid, orderdate,
requireddate,
    shippeddate, shipperid, freight, shipname,
shipaddress, shipcity
    shipregion, shippostalcode, shipcountry
    FROM Sales.Orders
    WHERE custid = @cid;
GO

-- query the function to request all orders that
were place by customer with custid = 2

SELECT orderid, custid
FROM dbo.GetCustOrders(2) AS O;
```

```
/*
Query # 16: Using inline TVF as part of a join
*/

SELECT O.orderid, O.custid, OD.productid, OD.qty
    FROM dbo.GetCustOrders(2) AS O
        INNER JOIN Sales.OrderDetails AS OD
            ON O.orderid = OD.orderid;

-- drop the function to clean up

DROP FUNCTION IF EXISTS dbo.GetCustOrders;

/*
Query # 17:
*/

SELECT S.shipperid, E.empid
FROM Sales.Shippers AS S
  CROSS JOIN HR.Employees AS E;
SELECT S.shipperid, E.empid
FROM Sales.Shippers AS S
  CROSS APPLY HR.Employees AS E;

/*
Query # 18:
*/
SELECT C.custid, A.orderid, A.orderdate
FROM Sales.Customers AS C
  CROSS APPLY    (SELECT TOP (3) orderid, empid,
orderdate, requireddate
     FROM Sales.Orders AS O
     WHERE O.custid = C.custid
     ORDER BY orderdate DESC, orderid DESC) AS A;
```

```
/*
Query # 19:
*/

SELECT C.custid, A.orderid, A.orderdate
FROM Sales.Customers AS C
  CROSS APPLY    (SELECT orderid, empid, orderdate,
requireddate
     FROM Sales.Orders AS O
     WHERE O.custid = C.custid
     ORDER BY orderdate DESC, orderid DESC
     OFFSET 0 ROWS FETCH NEXT 3 ROWS ONLY) AS A;

/*
Query # 20:
*/
DROP FUNCTION IF EXISTS dbo.TopOrders;
GO
CREATE FUNCTION dbo.TopOrders
    (@custid AS INT, @n AS INT)
    RETURNS TABLE
AS
RETURN
    SELECT TOP (@n) orderid, empid, orderdate,
requireddate
    FROM Sales.Orders
    WHERE custid = @custid
    ORDER BY orderdate DESC, orderid DESC
GO
SELECT
    C.custid, C.companyname,
    A.orderid, A.empid, A.orderdate, A.requireddate
FROM Sales.Customers AS C
    CROSS APPLY dbo.TopORDERS(C.custid, 3) AS A;
```