SQL Project :

Analysis and Queries on AdventureWorks Database

PREPARED BY:- GAURAV KUMAR

Project Source Description

This SQL project is based on the **AdventureWorks** sample database provided by Microsoft, which is commonly used for demonstrating database concepts, SQL queries, and data analytics. The data has been sourced from the AdventureWorks2014 edition and is structured to simulate the operations of a fictitious manufacturing company.

OVERVIEW

The database is organized into five distinct schemas

HumanResources

Person

Production

Purchasing

Sales

each serving as a logical container for related tables, views, and other database objects to support efficient data management and organizational clarity.

WHAT IS SCHEMA?

A schema represents the logical layout of a database, describing how data is structured and connected. It includes specifications for tables, columns, data types, relationships, constraints, and indexes used to manage and organize the data.

Used **Star** and **Snowflake schema** structures in database design to optimize data analysis and reporting.

Functions and concepts are used while solving a problem in SQL Server :-

- **DATABASE:-** A database is an organized collection of data that is stored and accessed electronically. It is designed to efficiently store, manage, and retrieve information in the form of rows and columns.

- **FUNCTION USED :- COUNT() , DISTINCT(), MIN(), MAX(), ROUND(), AVG(),SUM(),CONCAT(),OFFSET()**

- **Window function USED :-** A window function in SQL performs a calculation across a set of rows related to the current row, without collapsing the rows into a single result. It allows operations like ranking, running totals, and moving averages while keeping each row in the output **ROW NUMBER(),RANK (),DENSE_RANK(), NTILE()**

- **Union all, Subqueries, JOIN, Group by, CTE(common table expression)**

- **Used AVG(),SUM() aggeregate function as window function.**

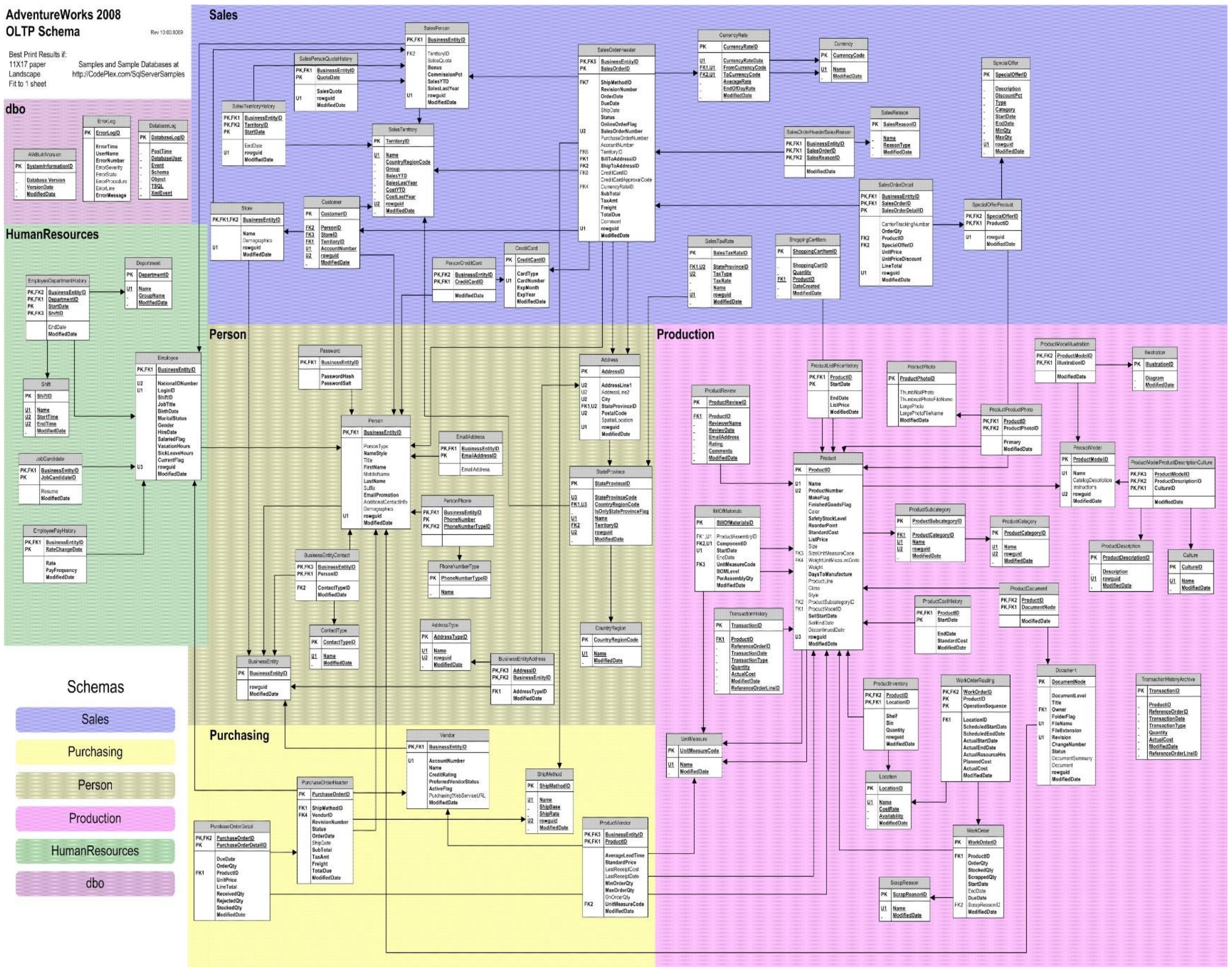
- **Used "CASE STATEMENT"**

- **USED YEAR() to extract year from date column.**

- **Foreign Key:-** A column or combination of columns that is used to establish and enforce a link between the data in two tables to control the data that can be stored in the foreign key table.

- **Primary Key:-** A primary key is a column (or combination of columns) that uniquely identifies each row in a table. It cannot contain NULL values and must hold unique values only.

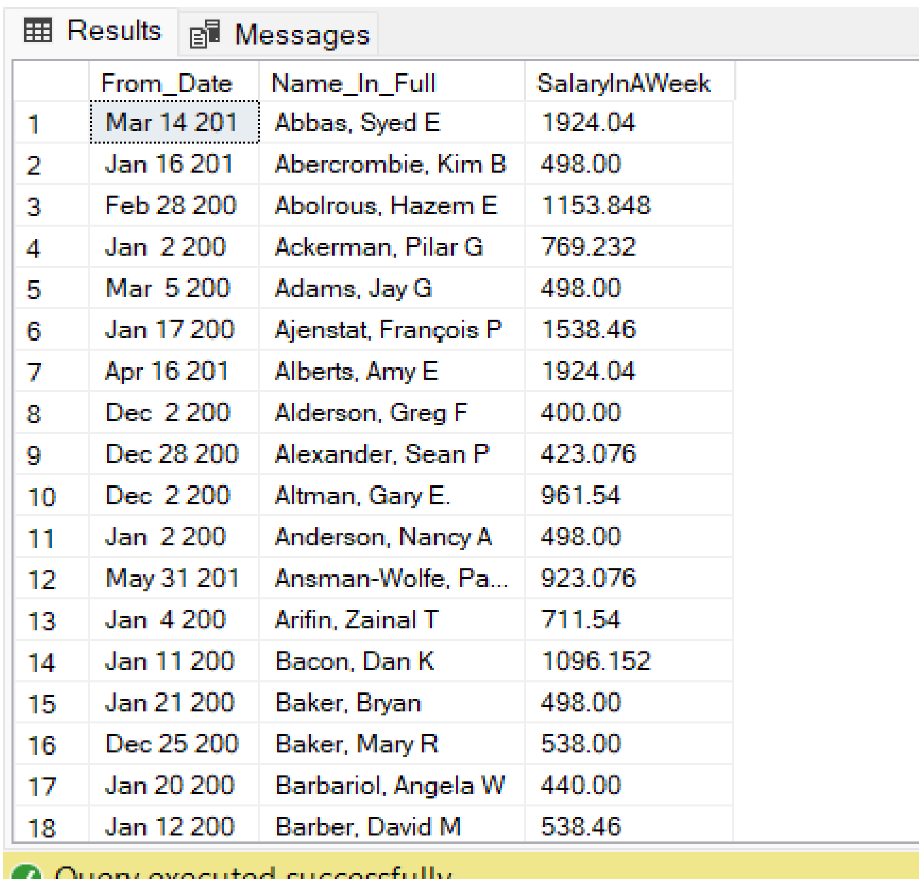
Database Diagram Including All Schemas and Tables:



NOTE:- All five database schemas, including diagrams and table structures, are documented in the project report (pages 18–22).

1:- From the table HumanResources.EmployeePayHistory, Person.Person write a query in SQL to retrieve the RateChangeDate, full name (first name, middle name and last name) and weekly salary (40 hours in a week) of employees. In the output the RateChangeDate should appears in date format. Sort the output in ascending order on NameInFull.

```
SELECT
   CAST(hur.RateChangeDate as VARCHAR(10)) AS From_Date,
   CONCAT(LastName, ', ', FirstName, ' ', MiddleName) AS Name_In_Full,
     (40 * hur.Rate) AS SalaryInAWeek
FROM Person.Person AS pp
    INNER JOIN HumanResources.EmployeePayHistory AS hur
        ON hur.BusinessEntityID = pp.BusinessEntityID
ORDER BY Name_In_Full
```

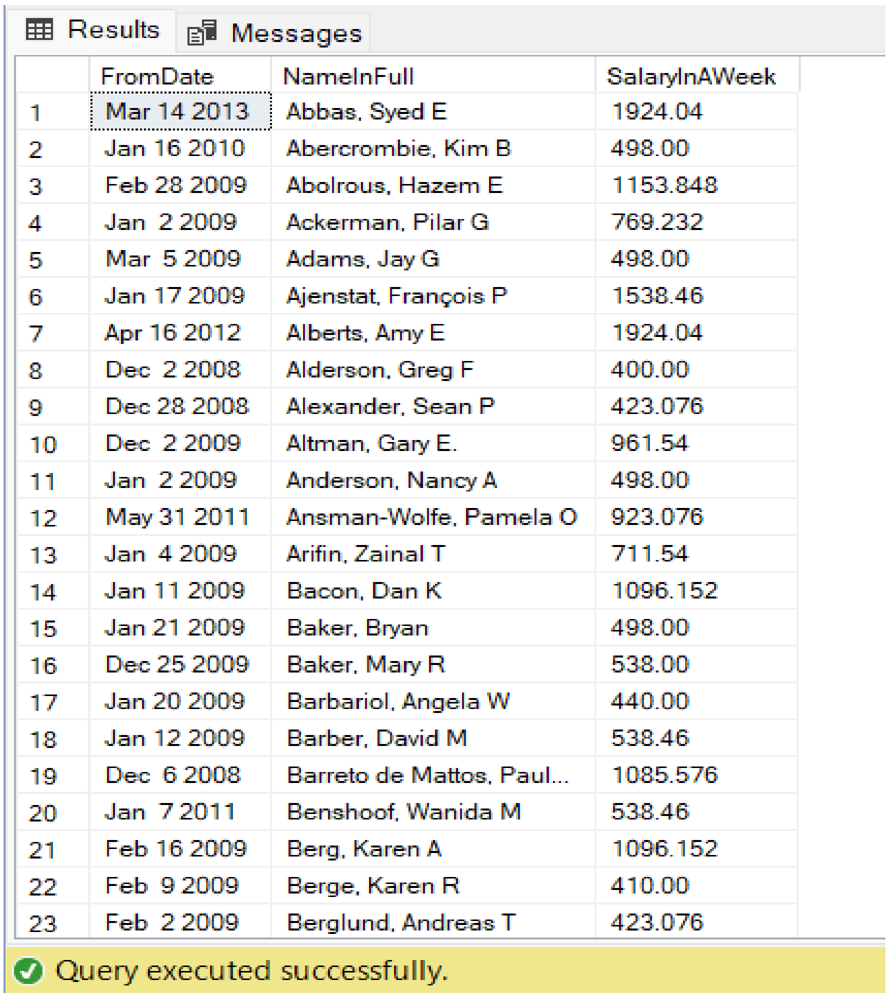


	From_Date	Name_In_Full	SalaryInAWeek
1	Mar 14 201	Abbas, Syed E	1924.04
2	Jan 16 201	Abercrombie, Kim B	498.00
3	Feb 28 200	Abolrous, Hazem E	1153.848
4	Jan 2 200	Ackerman, Pilar G	769.232
5	Mar 5 200	Adams, Jay G	498.00
6	Jan 17 200	Ajenstat, François P	1538.46
7	Apr 16 201	Alberts, Amy E	1924.04
8	Dec 2 200	Alderson, Greg F	400.00
9	Dec 28 200	Alexander, Sean P	423.076
10	Dec 2 200	Altman, Gary E.	961.54
11	Jan 2 200	Anderson, Nancy A	498.00
12	May 31 201	Ansman-Wolfe, Pa...	923.076
13	Jan 4 200	Arifin, Zainal T	711.54
14	Jan 11 200	Bacon, Dan K	1096.152
15	Jan 21 200	Baker, Bryan	498.00
16	Dec 25 200	Baker, Mary R	538.00
17	Jan 20 200	Barbariol, Angela W	440.00
18	Jan 12 200	Barber, David M	538.46

Query executed successfully.

2:-From the tables **Person.Person , HumanResources.EmployeePayHistory** write a query in SQL to calculate and display the **latest** weekly salary of each employee. Return RateChangeDate, full name (first name, middle name and last name) and weekly salary (40 hours in a week) of employees Sort the output in ascending order on NameInFull.

```sql
SELECT
    CAST(hur.RateChangeDate as VARCHAR(11)) AS FromDate,
    -- Concatenating the LastName, FirtName, and MiddleName columns and aliasing it as NameInFull
    CONCAT(LastName, ', ', FirstName, ' ', MiddleName) AS NameInFull,
    (40 * hur.Rate) AS SalaryInAWeek
FROM Person.Person AS pp
    INNER JOIN HumanResources.EmployeePayHistory AS hur
        ON hur.BusinessEntityID = pp.BusinessEntityID
WHERE hur.RateChangeDate = (
    SELECT MAX(RateChangeDate)
    FROM HumanResources.EmployeePayHistory
    WHERE BusinessEntityID = hur.BusinessEntityID
)
ORDER BY NameInFull
```



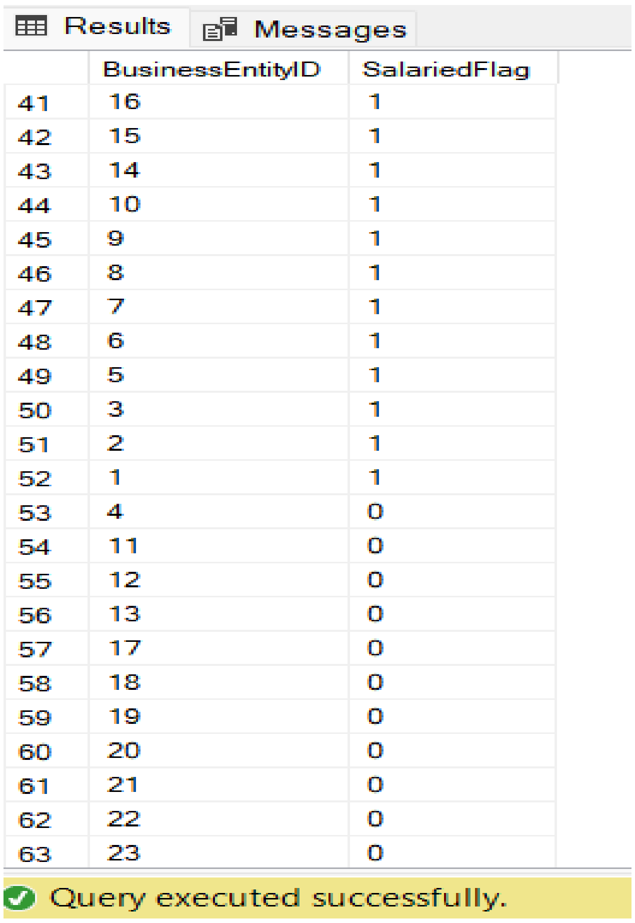
Results Messages

	FromDate	NameInFull	SalaryInAWeek
1	Mar 14 2013	Abbas, Syed E	1924.04
2	Jan 16 2010	Abercrombie, Kim B	498.00
3	Feb 28 2009	Abolrous, Hazem E	1153.848
4	Jan 2 2009	Ackerman, Pilar G	769.232
5	Mar 5 2009	Adams, Jay G	498.00
6	Jan 17 2009	Ajenstat, François P	1538.46
7	Apr 16 2012	Alberts, Amy E	1924.04
8	Dec 2 2008	Alderson, Greg F	400.00
9	Dec 28 2008	Alexander, Sean P	423.076
10	Dec 2 2009	Altman, Gary E.	961.54
11	Jan 2 2009	Anderson, Nancy A	498.00
12	May 31 2011	Ansman-Wolfe, Pamela O	923.076
13	Jan 4 2009	Arifin, Zainal T	711.54
14	Jan 11 2009	Bacon, Dan K	1096.152
15	Jan 21 2009	Baker, Bryan	498.00
16	Dec 25 2009	Baker, Mary R	538.00
17	Jan 20 2009	Barbariol, Angela W	440.00
18	Jan 12 2009	Barber, David M	538.46
19	Dec 6 2008	Barreto de Mattos, Paul...	1085.576
20	Jan 7 2011	Benshoof, Wanida M	538.46
21	Feb 16 2009	Berg, Karen A	1096.152
22	Feb 9 2009	Berge, Karen R	410.00
23	Feb 2 2009	Berglund, Andreas T	423.076

Query executed successfully.

3:- From the following table **HumanResources.Employee** write a query in SQL to ordered the BusinessEntityID column descendingly when SalariedFlag set to 'true' and BusinessEntityID in ascending order when SalariedFlag set to 'false'. Return BusinessEntityID, SalariedFlag columns

```
SELECT BusinessEntityID, SalariedFlag
FROM HumanResources.Employee
ORDER BY CASE SalariedFlag WHEN 'true' THEN BusinessEntityID END DESC
        ,CASE WHEN SalariedFlag ='false' THEN BusinessEntityID END;
```

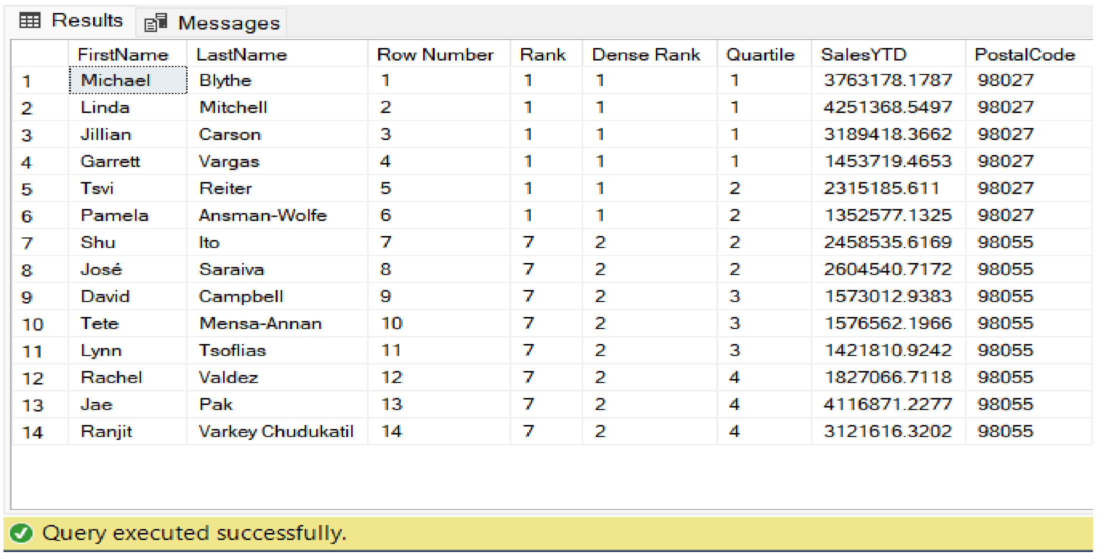


	BusinessEntityID	SalariedFlag
41	16	1
42	15	1
43	14	1
44	10	1
45	9	1
46	8	1
47	7	1
48	6	1
49	5	1
50	3	1
51	2	1
52	1	1
53	4	0
54	11	0
55	12	0
56	13	0
57	17	0
58	18	0
59	19	0
60	20	0
61	21	0
62	22	0
63	23	0

Query executed successfully.

4:- From the following table **Sales.SalesPerson, Person.Person, Person.Address** write a query in SQL to find those persons who lives in a territory and the value of salesytd except 0. Return first name, last name,row number as 'Row Number', 'Rank', 'Dense Rank' and NTILE as 'Quartile', salesytd and postalcode. Order the output on postalcode column.

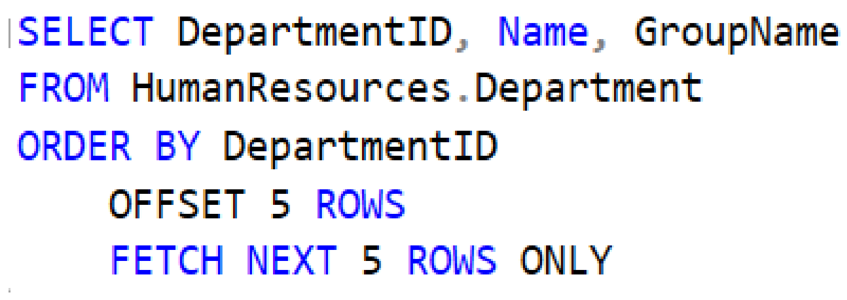
```
SELECT p.FirstName, p.LastName
    ,ROW_NUMBER() OVER (ORDER BY a.PostalCode) AS "Row Number"
    ,RANK() OVER (ORDER BY a.PostalCode) AS "Rank"
    ,DENSE_RANK() OVER (ORDER BY a.PostalCode) AS "Dense Rank"
    ,NTILE(4) OVER (ORDER BY a.PostalCode) AS "Quartile"
    ,s.SalesYTD, a.PostalCode
FROM Sales.SalesPerson AS s
    INNER JOIN Person.Person AS p
        ON s.BusinessEntityID = p.BusinessEntityID
    INNER JOIN Person.Address AS a
        ON a.AddressID = p.BusinessEntityID
WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0
```



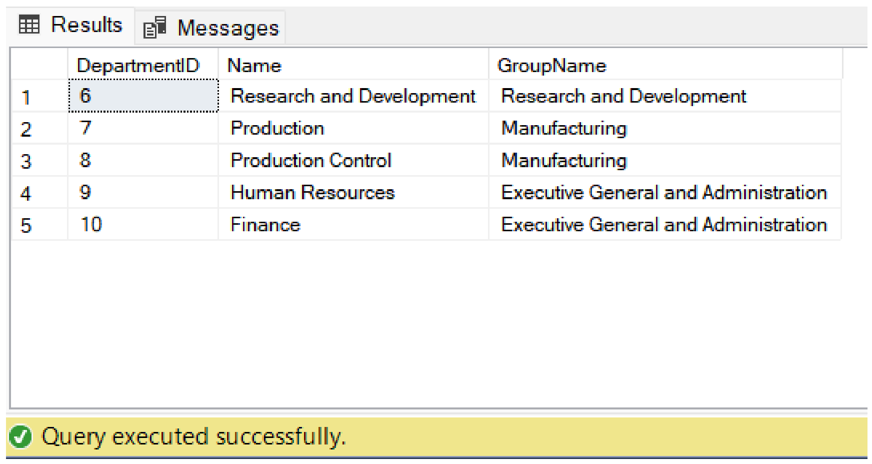
	FirstName	LastName	Row Number	Rank	Dense Rank	Quartile	SalesYTD	PostalCode
1	Michael	Blythe	1	1	1	1	3763178.1787	98027
2	Linda	Mitchell	2	1	1	1	4251368.5497	98027
3	Jillian	Carson	3	1	1	1	3189418.3662	98027
4	Garrett	Vargas	4	1	1	1	1453719.4653	98027
5	Tsvi	Reiter	5	1	1	2	2315185.611	98027
6	Pamela	Ansman-Wolfe	6	1	1	2	1352577.1325	98027
7	Shu	Ito	7	7	2	2	2458535.6169	98055
8	José	Saraiva	8	7	2	2	2604540.7172	98055
9	David	Campbell	9	7	2	3	1573012.9383	98055
10	Tete	Mensa-Annan	10	7	2	3	1576562.1966	98055
11	Lynn	Tsoflias	11	7	2	3	1421810.9242	98055
12	Rachel	Valdez	12	7	2	4	1827066.7118	98055
13	Jae	Pak	13	7	2	4	4116871.2277	98055
14	Ranjit	Varkey Chudukatil	14	7	2	4	3121616.3202	98055

Query executed successfully.

5:- From the following table **HumanResources.Department** write a query in SQL to skip the first 5 rows and return the next 5 rows from the sorted result set.



```sql
SELECT DepartmentID, Name, GroupName
FROM HumanResources.Department
ORDER BY DepartmentID
    OFFSET 5 ROWS
    FETCH NEXT 5 ROWS ONLY
```

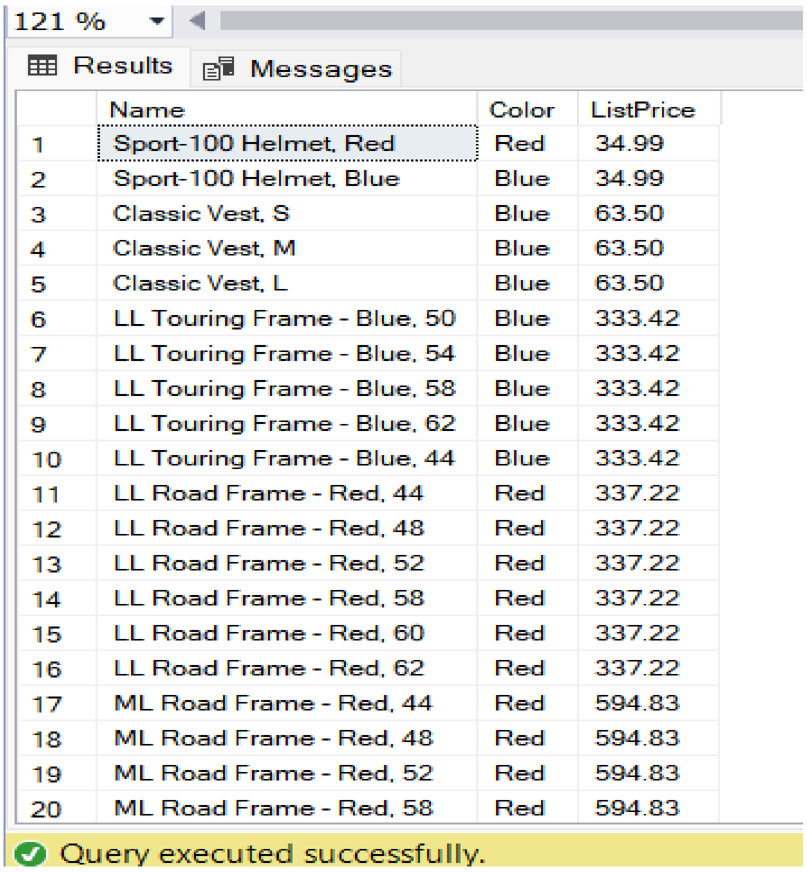


	DepartmentID	Name	GroupName
1	6	Research and Development	Research and Development
2	7	Production	Manufacturing
3	8	Production Control	Manufacturing
4	9	Human Resources	Executive General and Administration
5	10	Finance	Executive General and Administration

Query executed successfully.

6:-From the following table **Production.Product** write a query in SQL to list all the products that are Red or Blue in color. Return name, color and listprice.Sorts this result by the column listprice.

```
SELECT Name, Color, ListPrice
FROM Production.Product
WHERE Color = 'Red'
UNION ALL
SELECT Name, Color, ListPrice
FROM Production.Product
WHERE Color = 'Blue'
ORDER BY ListPrice ASC
```

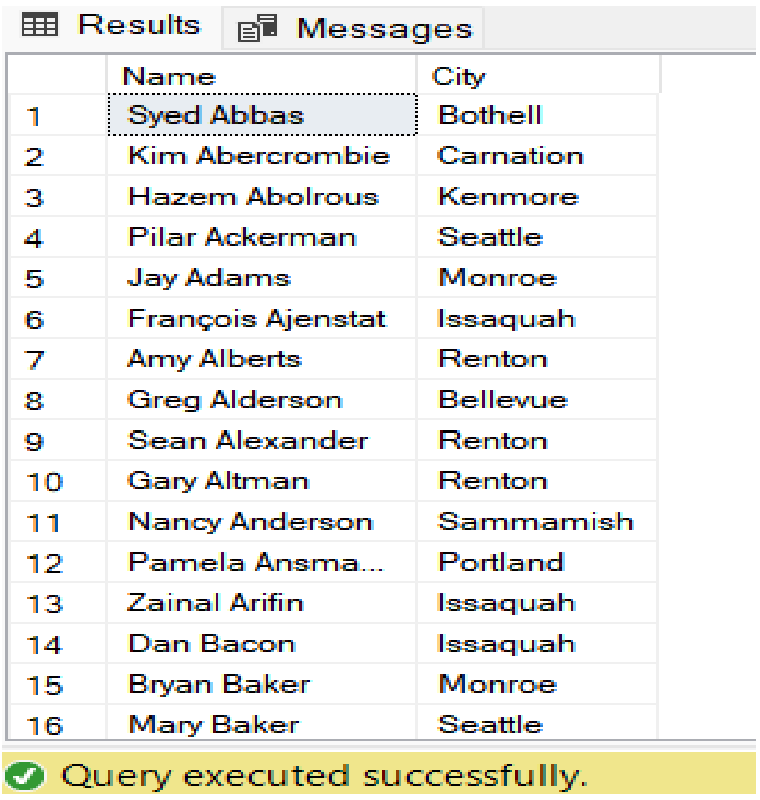


	Name	Color	ListPrice
1	Sport-100 Helmet, Red	Red	34.99
2	Sport-100 Helmet, Blue	Blue	34.99
3	Classic Vest, S	Blue	63.50
4	Classic Vest, M	Blue	63.50
5	Classic Vest, L	Blue	63.50
6	LL Touring Frame - Blue, 50	Blue	333.42
7	LL Touring Frame - Blue, 54	Blue	333.42
8	LL Touring Frame - Blue, 58	Blue	333.42
9	LL Touring Frame - Blue, 62	Blue	333.42
10	LL Touring Frame - Blue, 44	Blue	333.42
11	LL Road Frame - Red, 44	Red	337.22
12	LL Road Frame - Red, 48	Red	337.22
13	LL Road Frame - Red, 52	Red	337.22
14	LL Road Frame - Red, 58	Red	337.22
15	LL Road Frame - Red, 60	Red	337.22
16	LL Road Frame - Red, 62	Red	337.22
17	ML Road Frame - Red, 44	Red	594.83
18	ML Road Frame - Red, 48	Red	594.83
19	ML Road Frame - Red, 52	Red	594.83
20	ML Road Frame - Red, 58	Red	594.83

Query executed successfully.

7:-Write a query in SQL to find the employee's full name (firstname and lastname) and city from the following tables **Person.Person, HumanResources.Employee , Person.Address, Person.BusinessEntityAddress.** Order the result set on lastname then by firstname.

```
SELECT CONCAT(RTRIM(p.FirstName), ' ', LTRIM(p.LastName)) AS Name, d.City
FROM Person.Person AS p
INNER JOIN HumanResources.Employee e ON p.BusinessEntityID = e.BusinessEntityID
INNER JOIN
   (
    SELECT bea.BusinessEntityID, a.City
    FROM Person.Address AS a
    INNER JOIN Person.BusinessEntityAddress AS bea
    ON a.AddressID = bea.AddressID
   ) AS d
ON p.BusinessEntityID = d.BusinessEntityID
ORDER BY p.LastName, p.FirstName
```

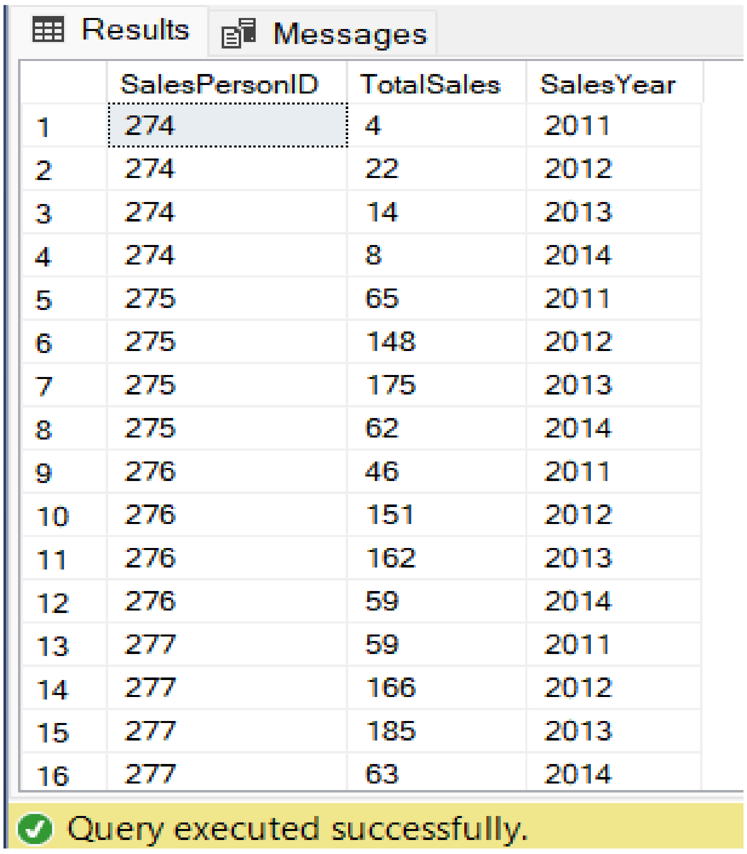


	Name	City
1	Syed Abbas	Bothell
2	Kim Abercrombie	Carnation
3	Hazem Abolrous	Kenmore
4	Pilar Ackerman	Seattle
5	Jay Adams	Monroe
6	François Ajenstat	Issaquah
7	Amy Alberts	Renton
8	Greg Alderson	Bellevue
9	Sean Alexander	Renton
10	Gary Altman	Renton
11	Nancy Anderson	Sammamish
12	Pamela Ansma...	Portland
13	Zainal Arifin	Issaquah
14	Dan Bacon	Issaquah
15	Bryan Baker	Monroe
16	Mary Baker	Seattle

Query executed successfully.

8:-Create a SQL query to display the total number of sales orders each sales representative receives annually. Sort the result set by SalesPersonID and then by the date component of the orderdate in ascending order. Return the year component of the OrderDate, SalesPersonID, and SalesOrderID.

```sql
WITH Sales_CTE (SalesPersonID, SalesOrderID, SalesYear)
AS
(
    SELECT SalesPersonID, SalesOrderID, year(OrderDate) AS SalesYear
    FROM Sales.SalesOrderHeader
    WHERE SalesPersonID IS NOT NULL
)
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
FROM Sales_CTE
GROUP BY SalesYear, SalesPersonID
ORDER BY SalesPersonID, SalesYear
```



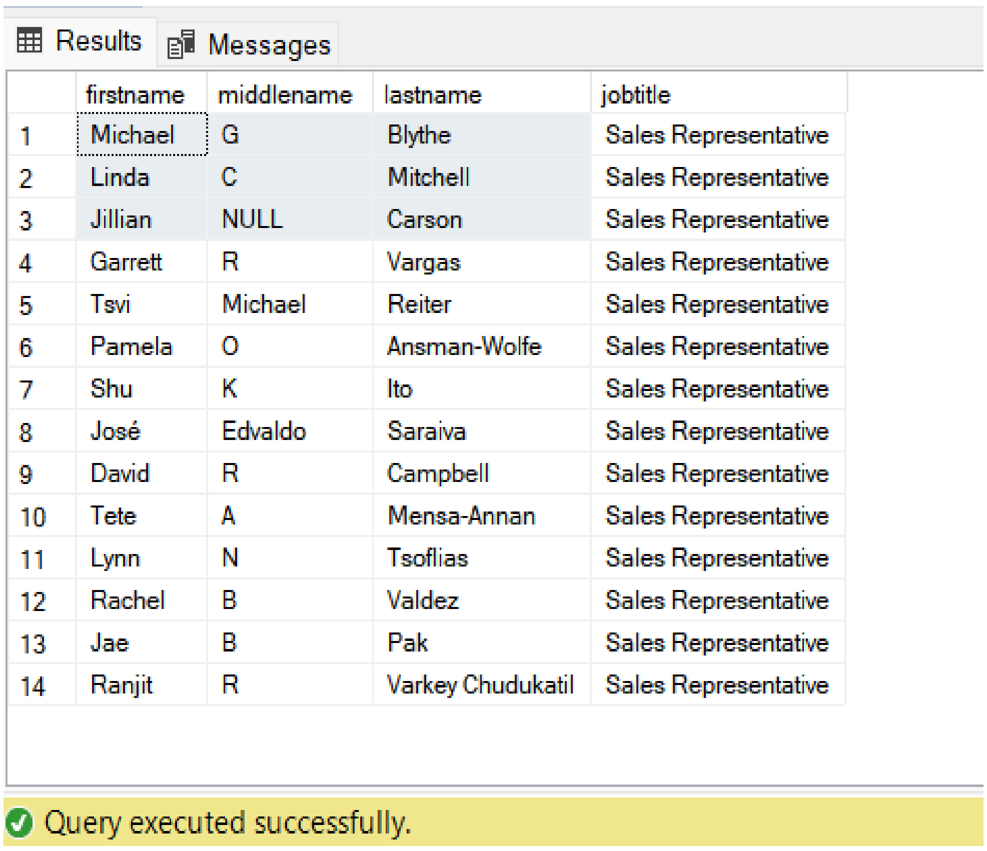
Results | Messages

	SalesPersonID	TotalSales	SalesYear
1	274	4	2011
2	274	22	2012
3	274	14	2013
4	274	8	2014
5	275	65	2011
6	275	148	2012
7	275	175	2013
8	275	62	2014
9	276	46	2011
10	276	151	2012
11	276	162	2013
12	276	59	2014
13	277	59	2011
14	277	166	2012
15	277	185	2013
16	277	63	2014

Query executed successfully.

9:-From the following table **Person.Person, HumanResources.Employee** write a SQL query to retrieve all the employees whose job titles begin with "Sales". Return firstname, middlename, lastname and jobtitle column

```sql
SELECT pepe.firstname, pepe.middlename, pepe.lastname, huem.jobtitle
FROM person.person pepe
INNER JOIN humanresources.employee huem
ON pepe.businessentityid = huem.businessentityid
WHERE SUBSTRING(huem.jobtitle, 1, 5) = 'Sales'
```



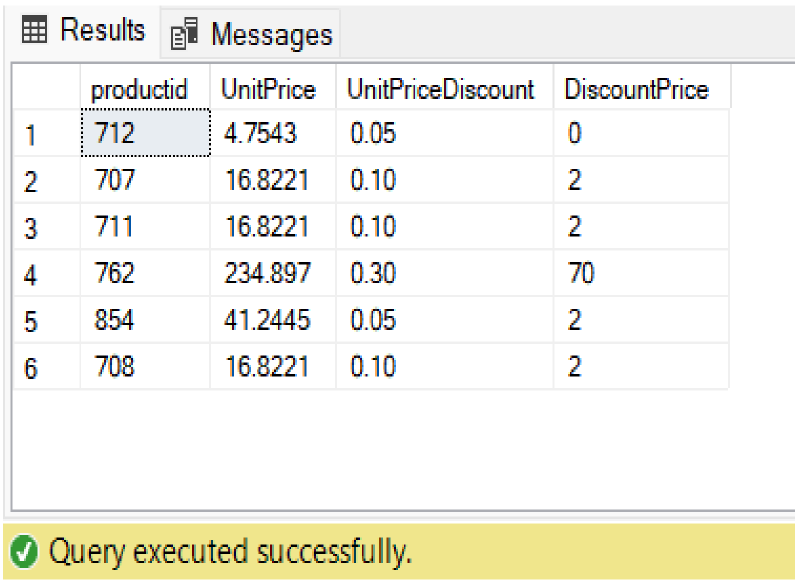
Results | Messages

	firstname	middlename	lastname	jobtitle
1	Michael	G	Blythe	Sales Representative
2	Linda	C	Mitchell	Sales Representative
3	Jillian	NULL	Carson	Sales Representative
4	Garrett	R	Vargas	Sales Representative
5	Tsvi	Michael	Reiter	Sales Representative
6	Pamela	O	Ansman-Wolfe	Sales Representative
7	Shu	K	Ito	Sales Representative
8	José	Edvaldo	Saraiva	Sales Representative
9	David	R	Campbell	Sales Representative
10	Tete	A	Mensa-Annan	Sales Representative
11	Lynn	N	Tsoflias	Sales Representative
12	Rachel	B	Valdez	Sales Representative
13	Jae	B	Pak	Sales Representative
14	Ranjit	R	Varkey Chudukatil	Sales Representative

Query executed successfully.

10:-From the following table **Sales.SalesOrderDetail** write a SQL query to determine the discount price for the salesorderid 46672. Calculate only those orders with discounts of more than.02 percent. Return productid, UnitPrice, UnitPriceDiscount, and DiscountPrice (UnitPrice*UnitPriceDiscount).

```
SELECT productid, UnitPrice, UnitPriceDiscount,
       CAST(ROUND(UnitPrice * UnitPriceDiscount, 0) AS int) AS DiscountPrice
FROM sales.salesorderdetail
WHERE SalesOrderid = 46672
      AND UnitPriceDiscount > .02
```

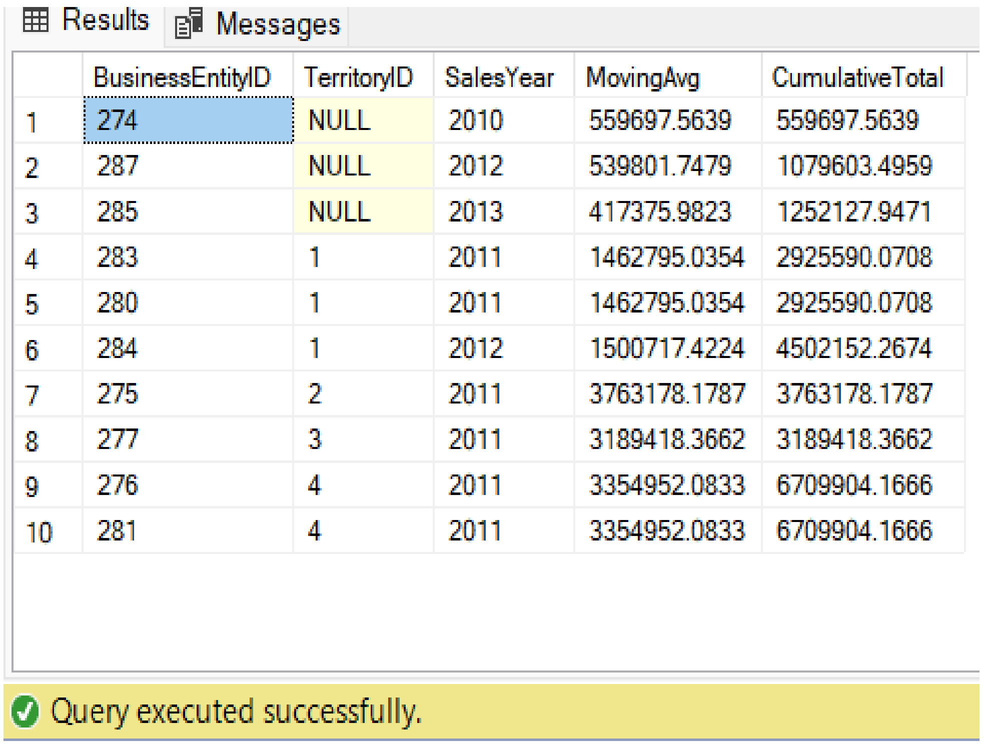


	productid	UnitPrice	UnitPriceDiscount	DiscountPrice
1	712	4.7543	0.05	0
2	707	16.8221	0.10	2
3	711	16.8221	0.10	2
4	762	234.897	0.30	70
5	854	41.2445	0.05	2
6	708	16.8221	0.10	2

Query executed successfully.

11:- From the following table **Sales.SalesPerson** write a query in SQL to return a moving average of yearly sales for each territory. Return BusinessEntityID, TerritoryID, SalesYear, SalesYTD, average SalesYTD as MovingAvg, and total SalesYTD as CumulativeTotal.

```sql
SELECT BusinessEntityID, TerritoryID
   ,year( ModifiedDate) AS SalesYear
   ,AVG(SalesYTD) OVER (PARTITION BY TerritoryID ORDER BY year( ModifiedDate)) AS MovingAvg
   ,SUM(SalesYTD) OVER (PARTITION BY TerritoryID ORDER BY year( ModifiedDate)) AS CumulativeTotal
FROM Sales.SalesPerson
WHERE TerritoryID IS NULL OR TerritoryID < 5
ORDER BY TerritoryID, SalesYear
```

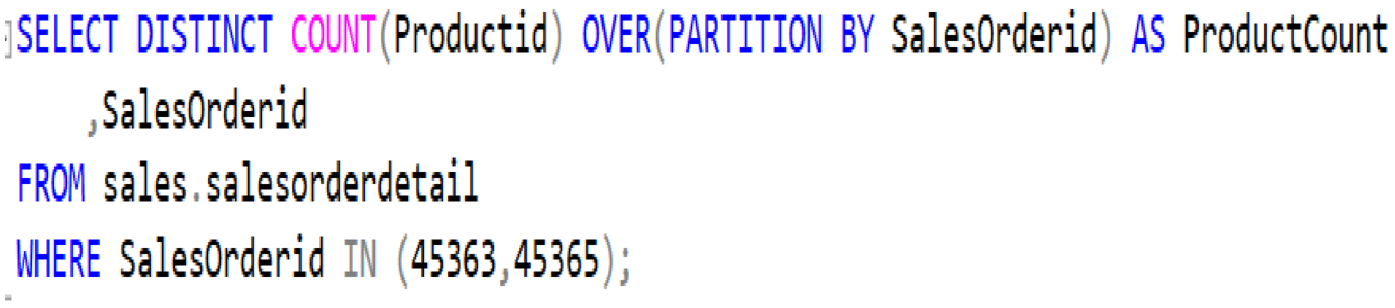


Results | Messages

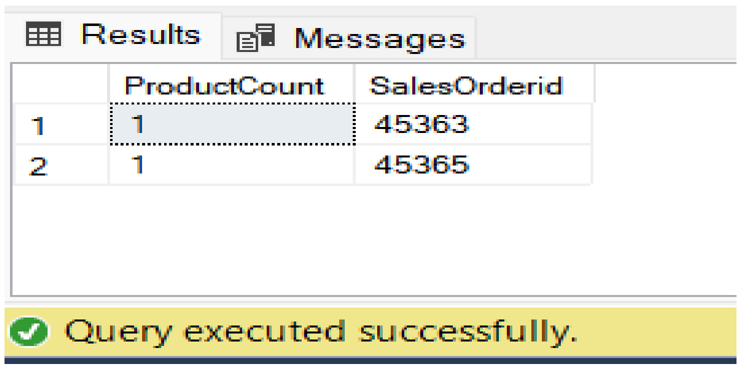
	BusinessEntityID	TerritoryID	SalesYear	MovingAvg	CumulativeTotal
1	274	NULL	2010	559697.5639	559697.5639
2	287	NULL	2012	539801.7479	1079603.4959
3	285	NULL	2013	417375.9823	1252127.9471
4	283	1	2011	1462795.0354	2925590.0708
5	280	1	2011	1462795.0354	2925590.0708
6	284	1	2012	1500717.4224	4502152.2674
7	275	2	2011	3763178.1787	3763178.1787
8	277	3	2011	3189418.3662	3189418.3662
9	276	4	2011	3354952.0833	6709904.1666
10	281	4	2011	3354952.0833	6709904.1666

Query executed successfully.

12:- From the following table **Sales.SalesOrderDetail** write a query in SQL to find the number of products that ordered in each of the specified sales orders.



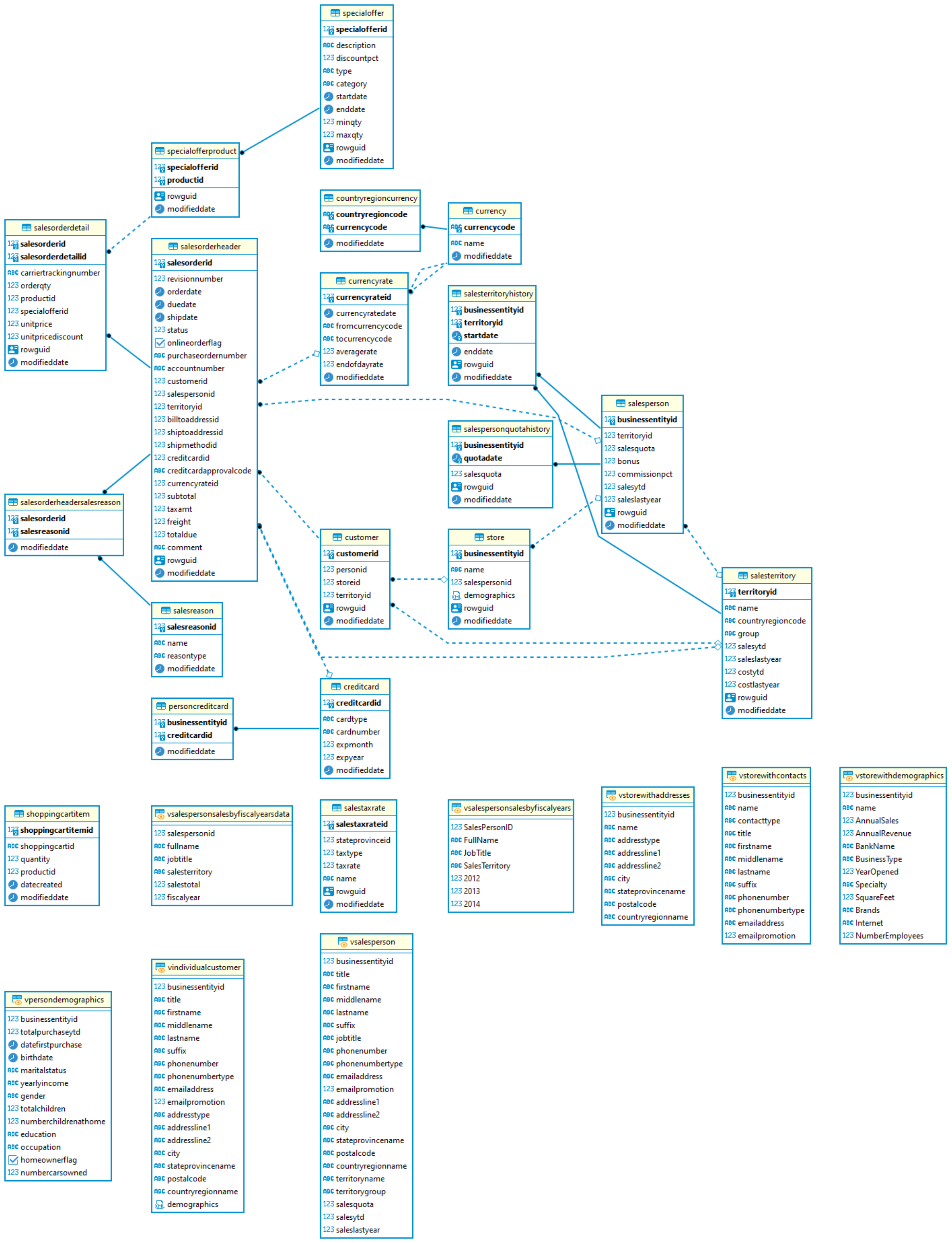
```
SELECT DISTINCT COUNT(Productid) OVER(PARTITION BY SalesOrderid) AS ProductCount
     ,SalesOrderid
FROM sales.salesorderdetail
WHERE SalesOrderid IN (45363,45365);
```



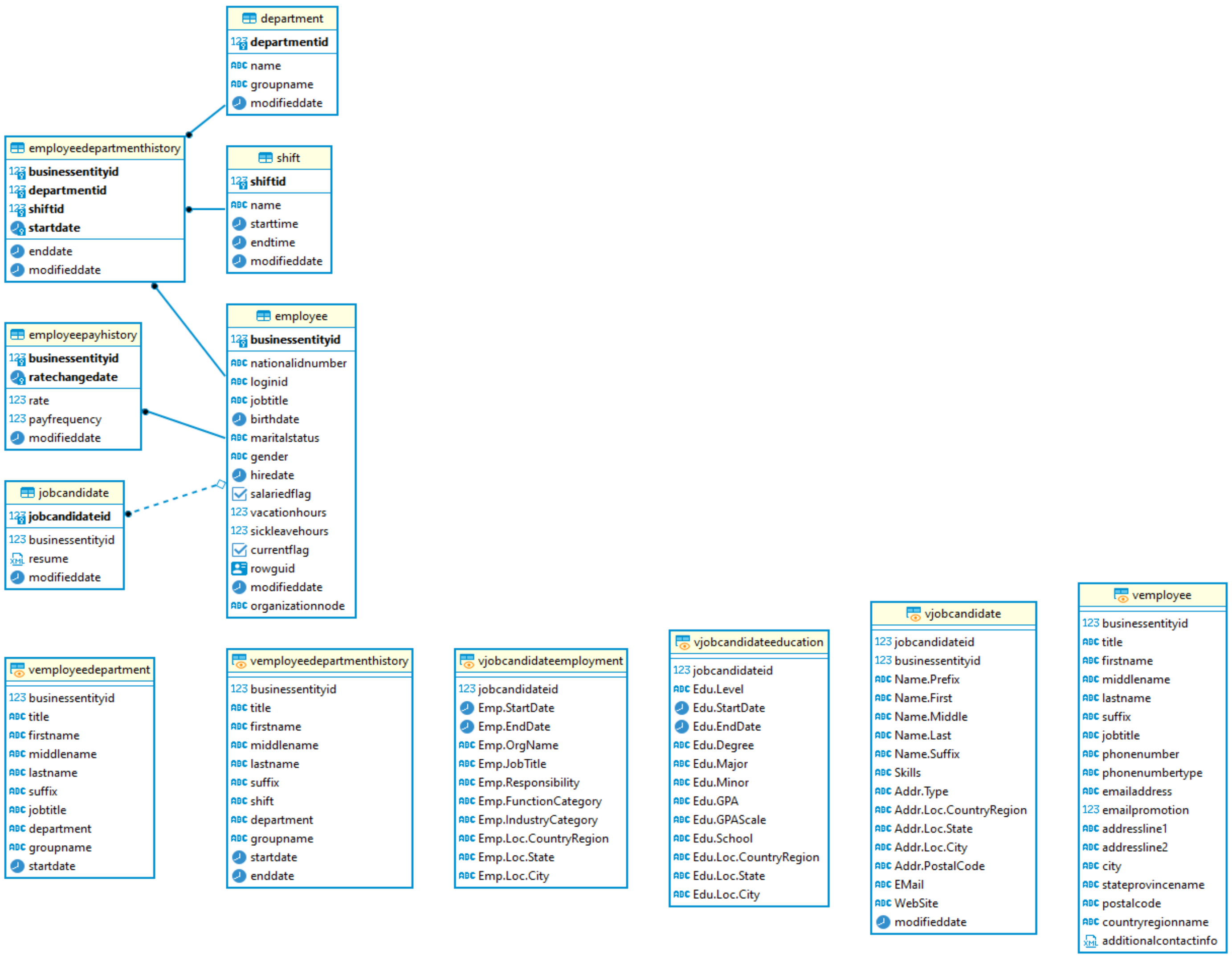
```
   ProductCount  SalesOrderid
1  1             45363
2  1             45365
```

Query executed successfully.

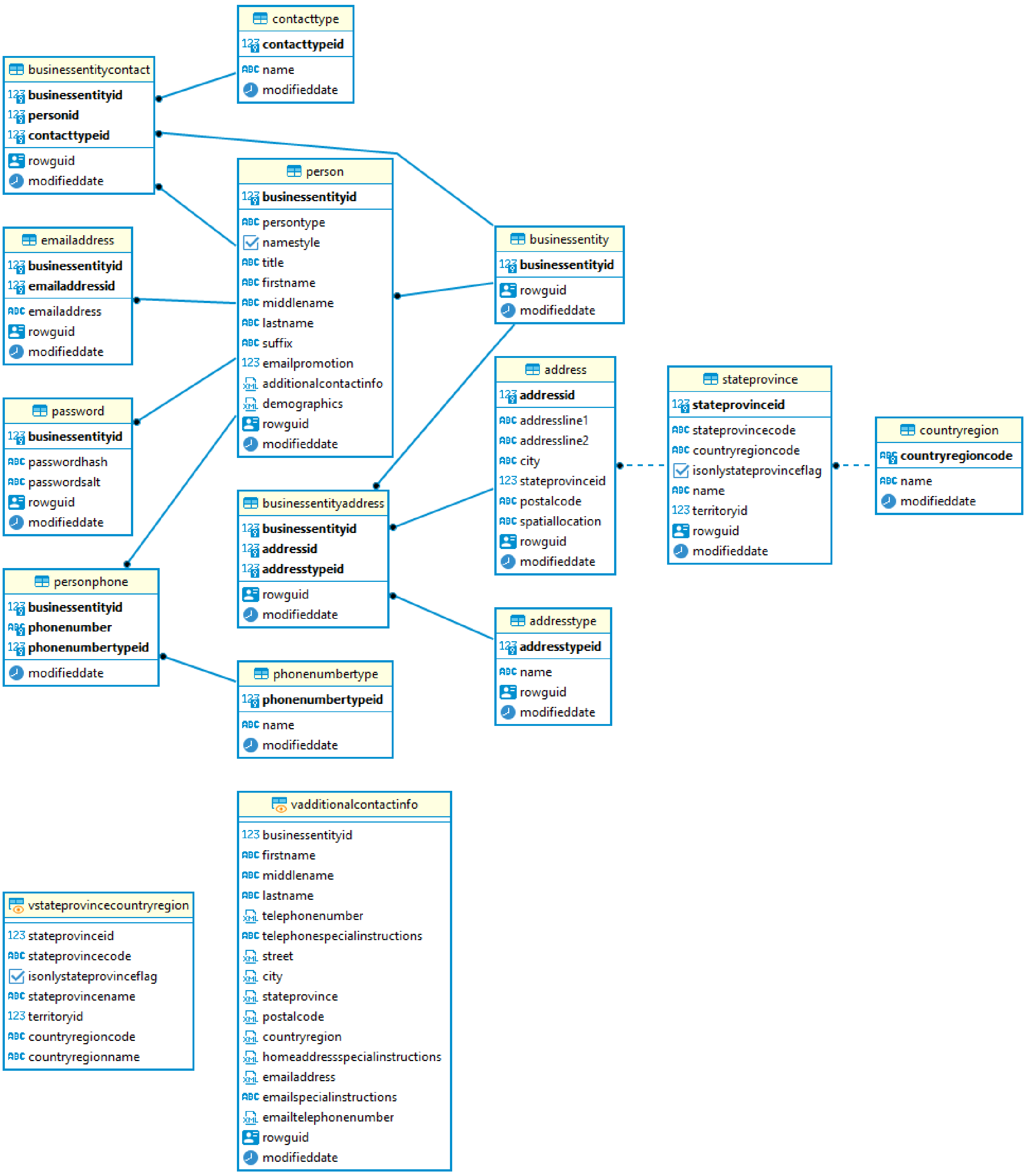
Sales Schema and Tables:



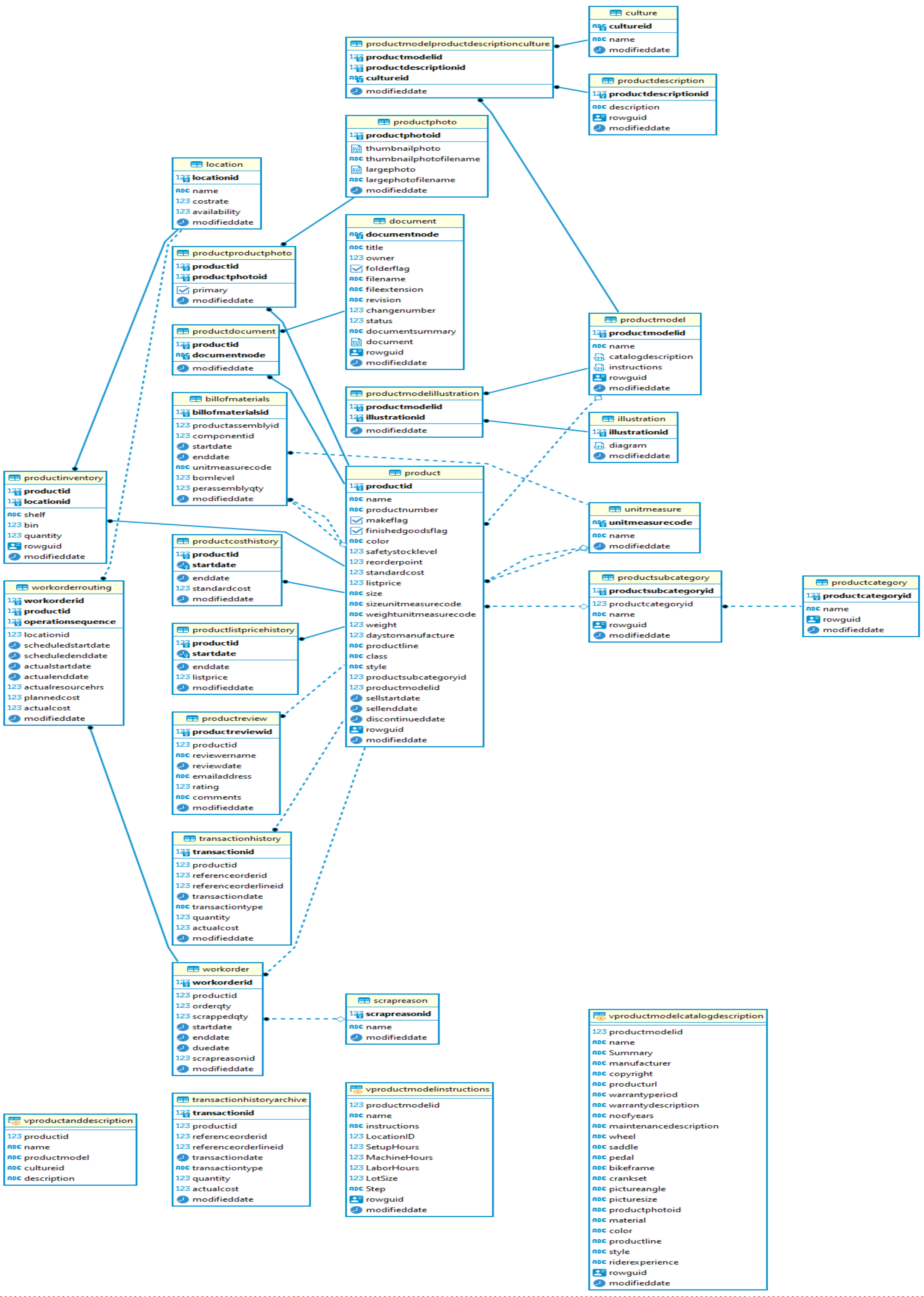
HumanResources Schema and Tables:



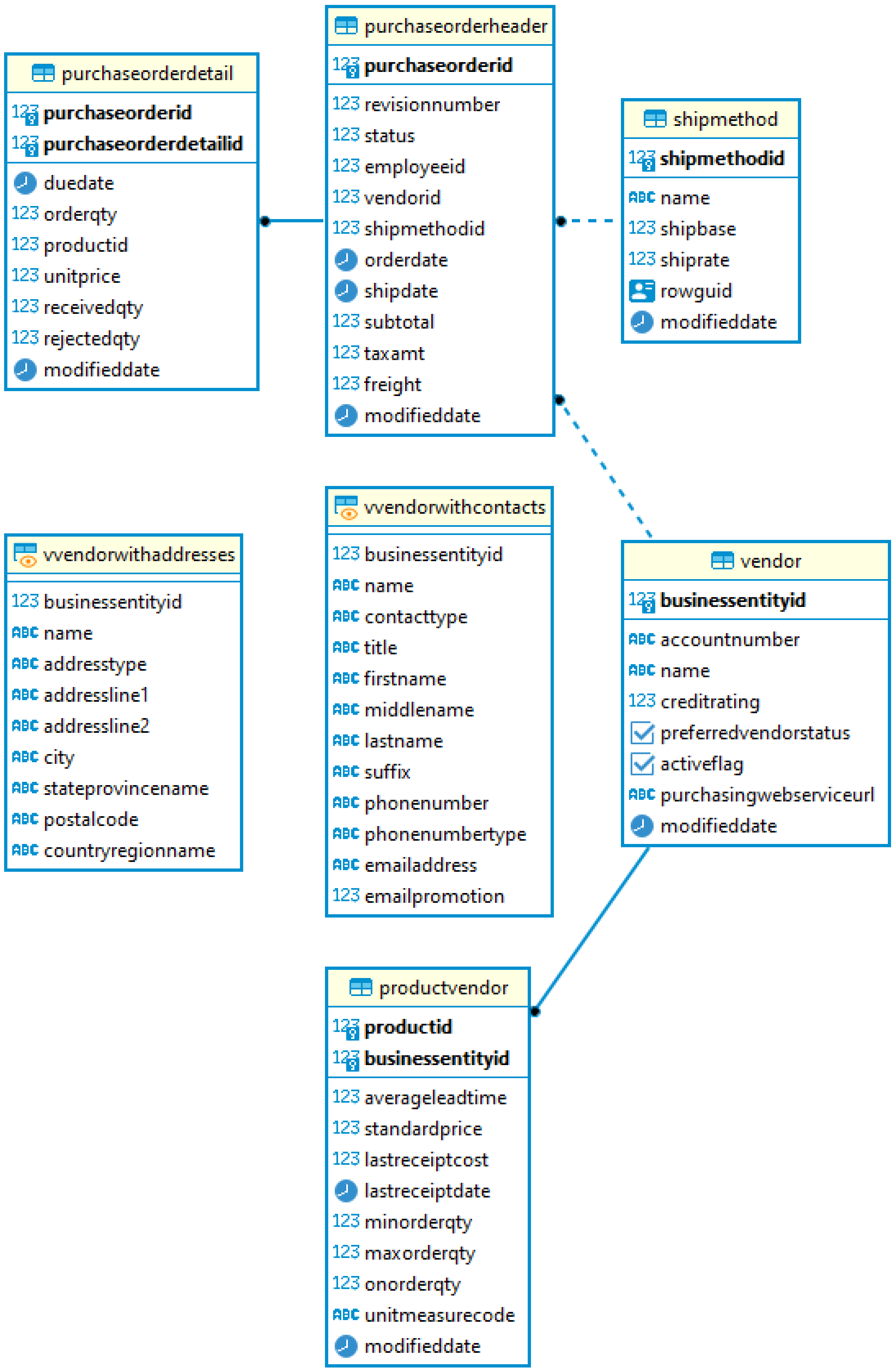
Person Schema and Tables:



Production Schema and Tables:



Purchase Schema and Tables:



<u>THANK YOU</u>