Advanced SQL Assignments (AdventureWorks)

1. List employees who have worked in more than one department.

```
SELECT BusinessEntityID
FROM HumanResources.EmployeeDepartmentHistory
GROUP BY BusinessEntityID
HAVING COUNT(DISTINCT DepartmentID) > 1;
```

2. Get the average hire year for each job title.

```
SELECT JobTitle, AVG(YEAR(HireDate)) AS AvgHireYear
FROM HumanResources.Employee
GROUP BY JobTitle;
```

3. Show the highest paid employee(s), including job title, department, and pay frequency.

```
SELECT TOP 1 WITH TIES e.BusinessEntityID, e.JobTitle, p.Rate,
p.PayFrequency, d.Name AS Department
FROM HumanResources.EmployeePayHistory p
JOIN HumanResources.Employee e ON p.BusinessEntityID =
e.BusinessEntityID
JOIN HumanResources.EmployeeDepartmentHistory edh ON
e.BusinessEntityID = edh.BusinessEntityID
JOIN HumanResources.Department d ON edh.DepartmentID =
d.DepartmentID
WHERE edh.EndDate IS NULL
ORDER BY p.Rate DESC;
```

4. Which sales territory has the highest total sales?

```
SELECT st.Name, SUM(soh.TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesTerritory st ON soh.TerritoryID = st.TerritoryID
```

```
GROUP BY st.Name
ORDER BY TotalSales DESC;
```

5.
6. List top 3 selling products for each year.
7.

```
WITH ProductSales AS (
  SELECT YEAR(soh.OrderDate) AS Year, p.Name, SUM(sod.OrderQty)
AS TotalSold
  FROM Sales.SalesOrderHeader soh
  JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
  JOIN Production.Product p ON sod.ProductID = p.ProductID
  GROUP BY YEAR(soh.OrderDate), p.Name
),
Ranked AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY Year ORDER BY
TotalSold DESC) AS rn
  FROM ProductSales
)
SELECT Year, Name, TotalSold FROM Ranked WHERE rn <= 3;
```

8. Show all customers who have never placed an order.

```
SELECT c.CustomerID
FROM Sales.Customer c
LEFT JOIN Sales.SalesOrderHeader soh ON c.CustomerID =
soh.CustomerID
WHERE soh.SalesOrderID IS NULL;
```

9. List employees who are also vendors.

```
SELECT DISTINCT p.FirstName, p.LastName
FROM Person.Person p
JOIN Purchasing.Vendor v ON p.BusinessEntityID =
v.BusinessEntityID
```

```
JOIN HumanResources.Employee e ON p.BusinessEntityID =
e.BusinessEntityID;
```

10. Find the total and average sales per product category.

```
SELECT c.Name AS Category, SUM(sod.LineTotal) AS TotalSales,
AVG(sod.LineTotal) AS AverageSales
FROM Sales.SalesOrderDetail sod
JOIN Production.Product p ON sod.ProductID = p.ProductID
JOIN Production.ProductSubcategory s ON p.ProductSubcategoryID =
s.ProductSubcategoryID
JOIN Production.ProductCategory c ON s.ProductCategoryID =
c.ProductCategoryID
GROUP BY c.Name;
```

11. Show the top 5 customers who buy the most expensive products.

```
SELECT TOP 5 soh.CustomerID, SUM(sod.UnitPrice) AS
ExpensivePurchase
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
GROUP BY soh.CustomerID
ORDER BY ExpensivePurchase DESC;
```

12. “Input: ProductID 750; Output: Quantity ordered in last 3 years, by year.”

```
SELECT YEAR(soh.OrderDate) AS Year, SUM(sod.OrderQty) AS
OrderedQty
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
```

```sql
WHERE sod.ProductID = 750 AND soh.OrderDate >= DATEADD(YEAR, -3,
GETDATE())
GROUP BY YEAR(soh.OrderDate);
```

13. Rank employees by how long they've worked in the company.

```sql
SELECT p.FirstName, p.LastName, e.HireDate, RANK() OVER (ORDER
BY e.HireDate) AS SeniorityRank
FROM HumanResources.Employee e
JOIN Person.Person p ON e.BusinessEntityID = p.BusinessEntityID;
```

14. Find the average order value for each customer.

```sql
SELECT CustomerID, AVG(TotalDue) AS AvgOrderValue
FROM Sales.SalesOrderHeader
GROUP BY CustomerID;
```

15. Show the month-over-month sales growth for 2014.

```sql
WITH MonthlySales AS (
  SELECT MONTH(OrderDate) AS Month, SUM(TotalDue) AS Sales
  FROM Sales.SalesOrderHeader
  WHERE YEAR(OrderDate) = 2014
  GROUP BY MONTH(OrderDate)
)
SELECT Month, Sales,
  LAG(Sales) OVER (ORDER BY Month) AS PrevMonthSales,
  (Sales - LAG(Sales) OVER (ORDER BY Month)) AS Growth
FROM MonthlySales;
```

16. Find customers who ordered the same product more than once in a year.

```sql
SELECT soh.CustomerID, sod.ProductID, YEAR(soh.OrderDate) AS
Year, COUNT(*) AS Orders
```

```
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
GROUP BY soh.CustomerID, sod.ProductID, YEAR(soh.OrderDate)
HAVING COUNT(*) > 1;
```

17. Find top 5 salespersons by their total sales in 2013.

```
SELECT TOP 5 soh.SalesPersonID, SUM(soh.TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader soh
WHERE YEAR(soh.OrderDate) = 2013 AND soh.SalesPersonID IS NOT
NULL
GROUP BY soh.SalesPersonID
ORDER BY TotalSales DESC;
```

18. Given: List of product names and list prices. Output: Show only those with list price > average. How to achieve?

```
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice > (SELECT AVG(ListPrice) FROM
Production.Product);
```

19. For each year, show number of new products introduced.

```
SELECT YEAR(SellStartDate) AS Year, COUNT(*) AS NewProducts
FROM Production.Product
GROUP BY YEAR(SellStartDate);
```

20. Show products with a price change history (list those with more than one price).

    ```
    SELECT ProductID
    ```

```
FROM Production.ProductListPriceHistory
GROUP BY ProductID
HAVING COUNT(DISTINCT ListPrice) > 1;
```

21. Which vendors supplied a product that was ordered more than 10 times?

```
SELECT DISTINCT v.Name
FROM Purchasing.ProductVendor pv
JOIN Purchasing.Vendor v ON pv.BusinessEntityID =
v.BusinessEntityID
JOIN Production.Product p ON pv.ProductID = p.ProductID
JOIN Sales.SalesOrderDetail sod ON p.ProductID = sod.ProductID
GROUP BY v.Name, sod.ProductID
HAVING SUM(sod.OrderQty) > 10;
```

22. Find products which were never discontinued.

```
SELECT Name FROM Production.Product WHERE SellEndDate IS NULL;
```

23. List all retired employees and their department at retirement.

```
SELECT p.FirstName, p.LastName, d.Name AS Department,
edh.EndDate
FROM HumanResources.EmployeeDepartmentHistory edh
JOIN HumanResources.Department d ON edh.DepartmentID =
d.DepartmentID
JOIN Person.Person p ON edh.BusinessEntityID =
p.BusinessEntityID
WHERE edh.EndDate IS NOT NULL;
```

24. Input: A table of ProductID and average order quantity per year. Output: List ProductIDs ordered on average more than 100 units/year. How to do this?

```
WITH ProductYearAvg AS (
  SELECT sod.ProductID, AVG(sod.OrderQty) AS AvgQty
```

```sql
  FROM Sales.SalesOrderDetail sod
  JOIN Sales.SalesOrderHeader soh ON sod.SalesOrderID =
soh.SalesOrderID
  GROUP BY sod.ProductID, YEAR(soh.OrderDate)
)
SELECT ProductID
FROM ProductYearAvg
GROUP BY ProductID
HAVING AVG(AvgQty) > 100;
```

25. Which department has the largest average tenure (years)?

```sql
SELECT d.Name, AVG(DATEDIFF(YEAR, edh.StartDate,
ISNULL(edh.EndDate, GETDATE()))) AS AvgTenure
FROM HumanResources.EmployeeDepartmentHistory edh
JOIN HumanResources.Department d ON edh.DepartmentID =
d.DepartmentID
GROUP BY d.Name
ORDER BY AvgTenure DESC;
```

26. For each customer, return the most expensive product they've ever ordered.

```sql
SELECT soh.CustomerID, MAX(sod.UnitPrice) AS MaxPrice
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
GROUP BY soh.CustomerID;
```

27. Show all products that had no price change since their introduction.

```sql
SELECT p.ProductID, p.Name
FROM Production.Product p
JOIN Production.ProductListPriceHistory plph ON p.ProductID =
plph.ProductID
```

```
GROUP BY p.ProductID, p.Name
HAVING COUNT(DISTINCT plph.ListPrice) = 1;
```

28. List vendors who supplied products more than 2 years ago but not in the last year.

```
SELECT v.Name
FROM Purchasing.PurchaseOrderHeader poh
JOIN Purchasing.Vendor v ON poh.VendorID = v.BusinessEntityID
WHERE poh.OrderDate < DATEADD(YEAR, -1, GETDATE())
AND poh.OrderDate >= DATEADD(YEAR, -2, GETDATE());
```

29. Show the quarter-over-quarter revenue growth for 2014.

```
WITH QuarterlySales AS (
  SELECT DATEPART(QUARTER, OrderDate) AS Quarter, SUM(TotalDue)
AS Revenue
  FROM Sales.SalesOrderHeader
  WHERE YEAR(OrderDate) = 2014
  GROUP BY DATEPART(QUARTER, OrderDate)
)
SELECT Quarter, Revenue, LAG(Revenue) OVER (ORDER BY Quarter) AS
PrevQuarter,
       (Revenue - LAG(Revenue) OVER (ORDER BY Quarter)) AS
Growth
FROM QuarterlySales;
```

30. Find the average number of days to ship an order per year.

```
SELECT YEAR(OrderDate) AS OrderYear, AVG(DATEDIFF(DAY,
OrderDate, ShipDate)) AS AvgDaysToShip
FROM Sales.SalesOrderHeader
WHERE ShipDate IS NOT NULL
GROUP BY YEAR(OrderDate);
```

31. For each region, count the number of distinct customers who ordered in 2015.

```
SELECT st.Name AS Region, COUNT(DISTINCT soh.CustomerID) AS
NumCustomers
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesTerritory st ON soh.TerritoryID = st.TerritoryID
WHERE YEAR(soh.OrderDate) = 2015
GROUP BY st.Name;
```

32. Show the longest streak of consecutive years a product was sold.
(This requires a recursive CTE / advanced logic beyond standard SQL and may not be supported by all SQL engines. For brevity, suggest an aggregate with MIN/MAX instead):

```
SELECT ProductID, MIN(YEAR(soh.OrderDate)) AS FirstYear,
MAX(YEAR(soh.OrderDate)) AS LastYear, (MAX(YEAR(soh.OrderDate))
- MIN(YEAR(soh.OrderDate)) + 1) AS YearsActive
FROM Sales.SalesOrderDetail sod
JOIN Sales.SalesOrderHeader soh ON sod.SalesOrderID =
soh.SalesOrderID
GROUP BY ProductID;
```

33. Find which employee has worked in the most different job titles.

```
SELECT TOP 1 e.BusinessEntityID, COUNT(DISTINCT edh.JobTitle) AS
NumTitles
FROM HumanResources.EmployeeDepartmentHistory edh
JOIN HumanResources.Employee e ON edh.BusinessEntityID =
e.BusinessEntityID
GROUP BY e.BusinessEntityID
ORDER BY NumTitles DESC;
```

34. Provide a monthly count of products added in 2023.

```
SELECT MONTH(SellStartDate) AS Month, COUNT(*) AS NewProducts
FROM Production.Product
WHERE YEAR(SellStartDate) = 2023
```

```
GROUP BY MONTH(SellStartDate);
```

35. List sales orders where the invoice and ship-to country are different.

```
SELECT SalesOrderID
FROM Sales.SalesOrderHeader
WHERE ShipToAddressID <> BillToAddressID;
```

36. Show all customers who ordered a product that was discontinued after their purchase.

```
SELECT DISTINCT soh.CustomerID
FROM Sales.SalesOrderHeader soh
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID =
sod.SalesOrderID
JOIN Production.Product p ON sod.ProductID = p.ProductID
WHERE p.SellEndDate IS NOT NULL AND p.SellEndDate >
soh.OrderDate;
```

37. “Input: Table with CustomerID, TotalDue. Output: Only those with total over $10,000. How do you achieve this?”

```
SELECT CustomerID, SUM(TotalDue) AS TotalAmount
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
```

38. `HAVING SUM(TotalDue) > 10000;`