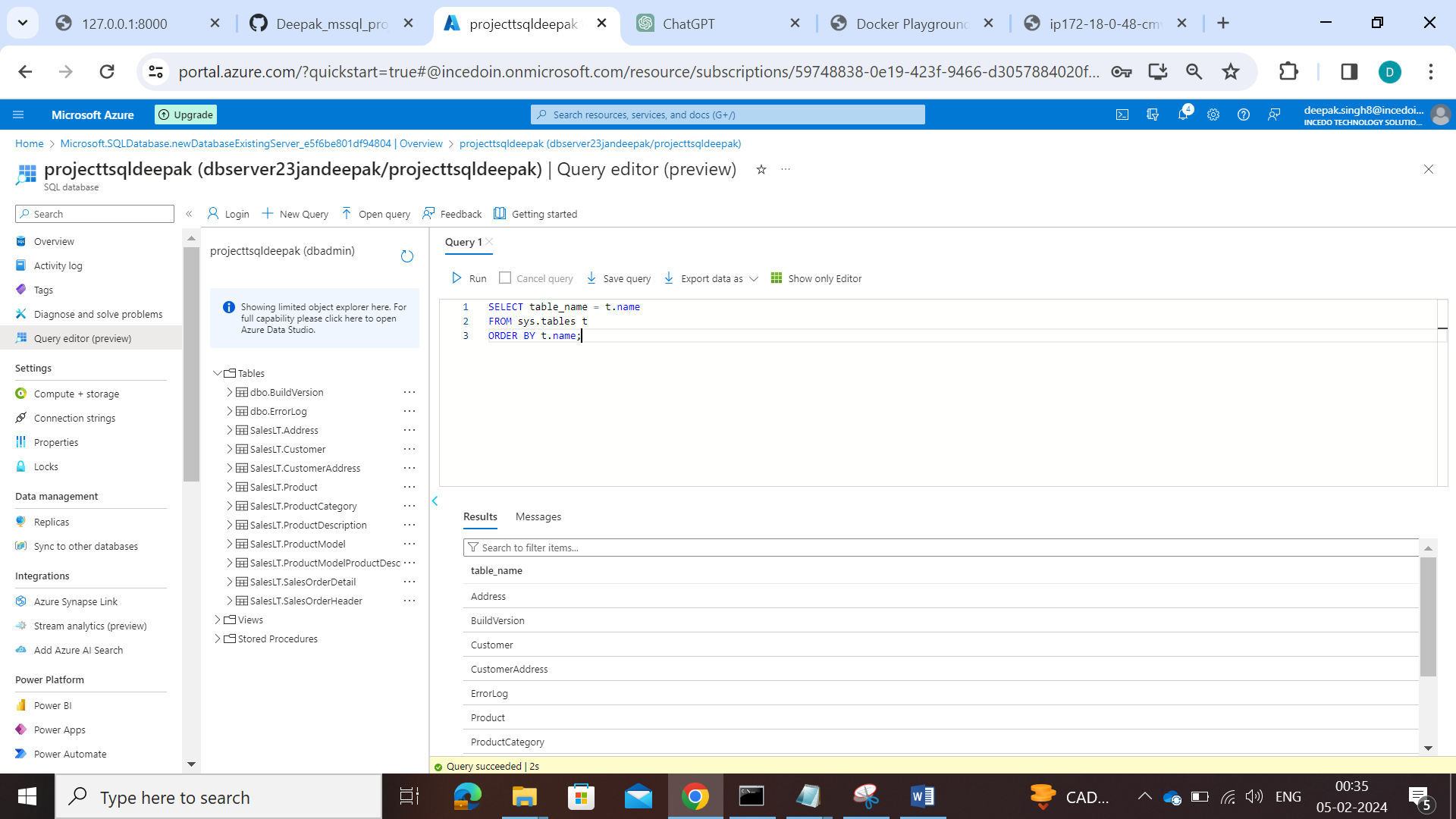
**MODULE 1: AdventureWorks sample database in Azure SQL Database-**



**MODULE 2: T-SQL Operations/Queries-**

**Below given queries are written considering all tables exist in the database, for queries according to the “Ref. AdventureWorks Schema Design” diagram in project 2 find the queries at the end of this document.**

1. **Retrieve a list of customers along with their total order amounts.**

SELECT

c.FirstName,

c.LastName,

SUM(o.TotalAmount) AS TotalOrderAmount

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

LEFT JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

GROUP BY

c.FirstName, c.LastName;

1. **Display product information along with the number of units sold for each product.**

SELECT

p.ProductID,

p.Name AS ProductName,

SUM(od.OrderQty) AS UnitsSold

FROM

Production.Product AS p

LEFT JOIN

Sales.SalesOrderDetail AS od

ON

p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.Name;

1. **Find employees who have the same manager.**

SELECT

e1.FirstName AS EmployeeName,

e2.FirstName AS ManagerName

FROM

HumanResources.Employee AS e1

INNER JOIN

HumanResources.Employee AS e2

ON

e1.ManagerID = e2.BusinessEntityID;

1. **List all customers who have never placed an order.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

LEFT JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

WHERE

cust.CustomerID IS NULL;

1. **Retrieve the total sales amount for each product category.**

SELECT

pc.Name AS CategoryName,

SUM(od.LineTotal) AS TotalSalesAmount

FROM

Production.ProductCategory AS pc

LEFT JOIN

Production.Product AS p

ON

pc.ProductCategoryID = p.ProductCategoryID

LEFT JOIN

Sales.SalesOrderDetail AS od

ON

p.ProductID = od.ProductID

GROUP BY

pc.Name;

1. **Display the names of employees and their direct managers.**

SELECT

e1.FirstName AS EmployeeName,

e2.FirstName AS ManagerName

FROM

HumanResources.Employee AS e1

LEFT JOIN

HumanResources.Employee AS e2

ON

e1.ManagerID = e2.BusinessEntityID;

1. **Show the order details with product names for a specific customer.**

SELECT

o.SalesOrderID,

p.Name AS ProductName,

od.OrderQty,

od.UnitPrice,

od.LineTotal

FROM

Sales.SalesOrder AS o

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

WHERE

o.CustomerID = (SELECT CustomerID FROM Sales.Customer WHERE AccountNumber = 'your\_account\_number\_here');

1. **List customers who have made purchases in the last 30 days.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

WHERE

o.OrderDate >= DATEADD(day, -30, GETDATE())

GROUP BY

c.FirstName, c.LastName;

1. **Find employees who do not have any direct reports.**

SELECT

e.FirstName,

e.LastName

FROM

HumanResources.Employee AS e

LEFT JOIN

HumanResources.Employee AS e2

ON

e.BusinessEntityID = e2.ManagerID

WHERE

e2.BusinessEntityID IS NULL;

1. **Retrieve all products along with their average selling prices.**

SELECT

p.ProductID,

p.Name AS ProductName,

AVG(od.UnitPrice) AS AverageSellingPrice

FROM

Production.Product AS p

LEFT JOIN

Sales.SalesOrderDetail AS od

ON

p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.Name;

1. **Find the order with the highest total amount.**

SELECT TOP 1

SalesOrderID

FROM

Sales.SalesOrder

ORDER BY

TotalAmount DESC;

1. **Display customers who have placed orders with a total amount greater than the average.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

WHERE

o.TotalAmount > (SELECT AVG(TotalAmount) FROM Sales.SalesOrder);

1. **List products with prices higher than the average product price.**

SELECT

p.ProductID,

p.Name AS ProductName,

p.ListPrice

FROM

Production.Product AS p

WHERE

p.ListPrice > (SELECT AVG(ListPrice) FROM Production.Product);

1. **Retrieve orders placed by employees who have a specific job title.**

SELECT

o.SalesOrderID,

e.FirstName AS EmployeeFirstName,

e.LastName AS EmployeeLastName

FROM

Sales.SalesOrder AS o

INNER JOIN

HumanResources.Employee AS e

ON

o.EmployeeID = e.BusinessEntityID

WHERE

e.JobTitle = 'your\_specific\_job\_title\_here';

1. **Display customers who have placed orders for a specific product category.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

INNER JOIN

Production.ProductCategory AS pc

ON

p.ProductCategoryID = pc.ProductCategoryID

WHERE

pc.Name = 'your\_specific\_category\_name\_here';

1. **Find employees with salaries greater than the average salary in their department.**

SELECT

e.FirstName,

e.LastName

FROM

HumanResources.Employee AS e

WHERE

e.Salary > (

SELECT AVG(Salary)

FROM HumanResources.Employee

WHERE DepartmentID = e.DepartmentID

);

1. **List customers who have placed orders before a specific date.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

WHERE

o.OrderDate < 'your\_specific\_date\_here';

1. **Retrieve the order with the highest quantity of a specific product.**

SELECT TOP 1

od.SalesOrderID,

od.ProductID,

od.OrderQty

FROM

Sales.SalesOrderDetail AS od

WHERE

od.ProductID = 'your\_specific\_product\_id\_here'

ORDER BY

od.OrderQty DESC;

1. **Display products with prices lower than the lowest product price in a specific category.**

SELECT

p.ProductID,

p.Name AS ProductName,

p.ListPrice

FROM

Production.Product AS p

WHERE

p.ListPrice < (

SELECT MIN(ListPrice)

FROM Production.Product

WHERE ProductCategoryID = 'your\_specific\_category\_id\_here'

);

1. **Find employees who have the same job title as their manager.**

SELECT

e.FirstName AS EmployeeName,

e.JobTitle

FROM

HumanResources.Employee AS e

INNER JOIN

HumanResources.Employee AS manager

ON

e.ManagerID = manager.BusinessEntityID

WHERE

e.JobTitle = manager.JobTitle;

1. **Combine results from two queries to get a list of unique customer and employee names.**

SELECT FirstName, LastName FROM Person.Person

UNION

SELECT FirstName, LastName FROM HumanResources.Employee;

1. **Retrieve product names that are common in two different product categories.**

SELECT p1.Name AS ProductName

FROM Production.Product AS p1

INNER JOIN Production.ProductCategory AS pc1 ON p1.ProductCategoryID = pc1.ProductCategoryID

WHERE pc1.Name = 'Category1'

INTERSECT

SELECT p2.Name AS ProductName

FROM Production.Product AS p2

INNER JOIN Production.ProductCategory AS pc2 ON p2.ProductCategoryID = pc2.ProductCategoryID

WHERE pc2.Name = 'Category2';

1. **Display the names of employees and customers in a single result set.**

SELECT FirstName, LastName FROM Person.Person

UNION ALL

SELECT FirstName, LastName FROM HumanResources.Employee;

1. **List products that are in stock or have been discontinued.**

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

WHERE p.DiscontinuedFlag = 1

UNION

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Production.ProductInventory AS pi ON p.ProductID = pi.ProductID

WHERE pi.QuantityOnHand > 0;

1. **Combine the results of two queries to find unique products ordered by a specific customer.**

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Sales.SalesOrderDetail AS od ON p.ProductID = od.ProductID

WHERE od.SalesOrderID IN (

SELECT SalesOrderID

FROM Sales.SalesOrder

WHERE CustomerID = 'your\_specific\_customer\_id\_here'

)

UNION

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Sales.SalesOrder AS o ON p.ProductID = o.CustomerID

WHERE o.CustomerID = 'your\_specific\_customer\_id\_here';

1. **Retrieve orders placed by customers and employees in a single result set.**

SELECT 'Customer' AS Source, FirstName, LastName

FROM Person.Person

UNION ALL

SELECT 'Employee' AS Source, FirstName, LastName

FROM HumanResources.Employee;

1. **Display products that are either in a specific category or have a specific safety stock level.**

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Production.ProductCategory AS pc ON p.ProductCategoryID = pc.ProductCategoryID

WHERE pc.Name = 'your\_specific\_category\_name\_here'

UNION

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Production.ProductInventory AS pi ON p.ProductID = pi.ProductID

WHERE pi.SafetyStockLevel = 'your\_specific\_safety\_stock\_level\_here';

1. **List customers who have placed orders and employees who have direct reports in a single result set.**

SELECT FirstName, LastName

FROM Person.Person AS c

INNER JOIN Sales.SalesOrder AS o ON c.BusinessEntityID = o.CustomerID

UNION ALL

SELECT FirstName, LastName

FROM HumanResources.Employee AS e

INNER JOIN HumanResources.Employee AS e2 ON e.BusinessEntityID = e2.ManagerID;

1. **Retrieve products that are in stock in one location and out of stock in another.**

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Production.ProductInventory AS pi1 ON p.ProductID = pi1.ProductID

INNER JOIN Production.Location AS l1 ON pi1.LocationID = l1.LocationID

WHERE l1.Name = 'Location1' AND pi1.QuantityOnHand > 0

INTERSECT

SELECT p.ProductID, p.Name AS ProductName

FROM Production.Product AS p

INNER JOIN Production.ProductInventory AS pi2 ON p.ProductID = pi2.ProductID

INNER JOIN Production.Location AS l2 ON pi2.LocationID = l2.LocationID

WHERE l2.Name = 'Location2' AND pi2.QuantityOnHand = 0;

1. **Combine information about employees who are managers and employees who have managers.**

SELECT FirstName, LastName, 'Manager' AS Role

FROM HumanResources.Employee AS e

WHERE EXISTS (

SELECT 1

FROM HumanResources.Employee AS e2

WHERE e.BusinessEntityID = e2.ManagerID

)

UNION

SELECT FirstName, LastName, 'Employee' AS Role

FROM HumanResources.Employee AS e

WHERE e.ManagerID IS NOT NULL;

**31. Retrieve a list of customers along with the names of the products they have purchased.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName,

p.Name AS ProductName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID;

**32. Display employees who have the same manager, including indirect reports.**

WITH RecursiveManagerCTE AS (

SELECT

BusinessEntityID,

FirstName,

LastName,

ManagerID

FROM

HumanResources.Employee

WHERE

ManagerID IS NULL

UNION ALL

SELECT

e.BusinessEntityID,

e.FirstName,

e.LastName,

e.ManagerID

FROM

HumanResources.Employee AS e

INNER JOIN

RecursiveManagerCTE AS cte

ON

e.ManagerID = cte.BusinessEntityID

)

SELECT

EmployeeID,

FirstName,

LastName,

ManagerID

FROM

RecursiveManagerCTE;

**33. Find orders with multiple products and display the product names.**

SELECT

o.SalesOrderID,

p.Name AS ProductName

FROM

Sales.SalesOrder AS o

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

GROUP BY

o.SalesOrderID, p.Name

HAVING

COUNT(od.ProductID) > 1;

**34. List customers along with the names of the salespeople who handled their orders.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName,

e.FirstName AS SalespersonFirstName,

e.LastName AS SalespersonLastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

HumanResources.Employee AS e

ON

o.EmployeeID = e.BusinessEntityID;

**35. Retrieve a list of products along with the names of suppliers.**

SELECT

p.ProductID,

p.Name AS ProductName,

s.Name AS SupplierName

FROM

Production.Product AS p

LEFT JOIN

Purchasing.ProductVendor AS pv

ON

p.ProductID = pv.ProductID

LEFT JOIN

Purchasing.Vendor AS v

ON

pv.BusinessEntityID = v.BusinessEntityID;

**36. Display customers who have placed orders and the products they have purchased, including product details.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName,

o.SalesOrderID,

p.ProductID,

p.Name AS ProductName,

od.OrderQty,

od.UnitPrice,

od.LineTotal

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID;

**37. Find orders where multiple employees were involved, showing the employee names.**

SELECT

o.SalesOrderID,

e.FirstName AS EmployeeFirstName,

e.LastName AS EmployeeLastName

FROM

Sales.SalesOrder AS o

INNER JOIN

HumanResources.Employee AS e

ON

o.EmployeeID = e.BusinessEntityID

WHERE

o.SalesOrderID IN (

SELECT SalesOrderID

FROM Sales.SalesOrder

GROUP BY SalesOrderID

HAVING COUNT(DISTINCT EmployeeID) > 1

);

**38. List products that have similar names but belong to different categories.**

SELECT

p1.ProductID AS ProductID1,

p1.Name AS ProductName1,

pc1.Name AS CategoryName1,

p2.ProductID AS ProductID2,

p2.Name AS ProductName2,

pc2.Name AS CategoryName2

FROM

Production.Product AS p1

INNER JOIN

Production.ProductCategory AS pc1

ON

p1.ProductCategoryID = pc1.ProductCategoryID

INNER JOIN

Production.Product AS p2

ON

p1.ProductID <> p2.ProductID

INNER JOIN

Production.ProductCategory AS pc2

ON

p2.ProductCategoryID = pc2.ProductCategoryID

WHERE

p1.Name LIKE '%' + LEFT(p2.Name, 3) + '%'

AND

p1.ProductID < p2.ProductID;

**39. Retrieve a list of employees along with their training courses and training dates.**

SELECT

e.FirstName AS EmployeeFirstName,

e.LastName AS EmployeeLastName,

tc.CourseName,

et.TrainingDate

FROM

HumanResources.Employee AS e

INNER JOIN

HumanResources.EmployeeTraining AS et

ON

e.BusinessEntityID = et.BusinessEntityID

INNER JOIN

HumanResources.TrainingCourse AS tc

ON

et.CourseID = tc.CourseID;

**40. Display customers who have placed orders and the total quantity of each product ordered.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName,

p.ProductID,

p.Name AS ProductName,

SUM(od.OrderQty) AS TotalQuantityOrdered

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

GROUP BY

c.FirstName, c.LastName, p.ProductID, p.Name;

**41. Find customers who have made more purchases than the average number of purchases.**

SELECT

c.FirstName,

c.LastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN (

SELECT

CustomerID,

COUNT(SalesOrderID) AS OrderCount

FROM

Sales.SalesOrder

GROUP BY

CustomerID

) AS subq

ON

cust.CustomerID = subq.CustomerID

WHERE

subq.OrderCount > (

SELECT AVG(OrderCount) FROM (

SELECT

CustomerID,

COUNT(SalesOrderID) AS OrderCount

FROM

Sales.SalesOrder

GROUP BY

CustomerID

) AS avg\_orders

);

**42. Display products that have been ordered more than the average number of times.**

SELECT

p.ProductID,

p.Name AS ProductName

FROM

Production.Product AS p

INNER JOIN (

SELECT

ProductID,

COUNT(SalesOrderID) AS OrderCount

FROM

Sales.SalesOrderDetail

GROUP BY

ProductID

) AS subq

ON

p.ProductID = subq.ProductID

WHERE

subq.OrderCount > (

SELECT AVG(OrderCount) FROM (

SELECT

ProductID,

COUNT(SalesOrderID) AS OrderCount

FROM

Sales.SalesOrderDetail

GROUP BY

ProductID

) AS avg\_orders

);

**43. Retrieve orders placed by employees who have completed a specific training course.**

SELECT

o.SalesOrderID,

e.FirstName AS EmployeeFirstName,

e.LastName AS EmployeeLastName

FROM

Sales.SalesOrder AS o

INNER JOIN

HumanResources.Employee AS e

ON

o.EmployeeID = e.BusinessEntityID

WHERE

o.EmployeeID IN (

SELECT BusinessEntityID

FROM HumanResources.EmployeeTraining

WHERE CourseID = 'your\_specific\_course\_id\_here'

);

**44. List employees who have a higher salary than at least one employee in another department.**

SELECT

e1.FirstName AS EmployeeFirstName,

e1.LastName AS EmployeeLastName,

e1.Salary AS EmployeeSalary,

e1.DepartmentID AS EmployeeDepartmentID,

e2.FirstName AS OtherEmployeeFirstName,

e2.LastName AS OtherEmployeeLastName,

e2.Salary AS OtherEmployeeSalary,

e2.DepartmentID AS OtherEmployeeDepartmentID

FROM

HumanResources.Employee AS e1

INNER JOIN

HumanResources.Employee AS e2

ON

e1.Salary > e2.Salary

AND

e1.DepartmentID <> e2.DepartmentID;

**45. Display products that have not been ordered in the last 60 days.**

SELECT

p.ProductID,

p.Name AS ProductName

FROM

Production.Product AS p

WHERE

p.ProductID NOT IN (

SELECT DISTINCT od.ProductID

FROM Sales.SalesOrderDetail AS od

WHERE od.OrderDate >= DATEADD(day, -60, GETDATE())

);

**46. Find employees who have the same job title as the employee with the highest salary.**

SELECT

e1.FirstName AS EmployeeFirstName,

e1.LastName AS EmployeeLastName,

e1.JobTitle AS EmployeeJobTitle

FROM

HumanResources.Employee AS e1

WHERE

e1.JobTitle = (

SELECT TOP 1 JobTitle

FROM HumanResources.Employee

ORDER BY Salary DESC

);

**47. List customers who have placed orders with a total amount greater than the total amount of a specific order.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o1

ON

cust.CustomerID = o1.CustomerID

WHERE

o1.TotalAmount > (

SELECT TotalAmount

FROM Sales.SalesOrder

WHERE SalesOrderID = 'your\_specific\_order\_id\_here'

);

**48. Retrieve products that have been ordered by customers with the same shipping address.**

SELECT

p.ProductID,

p.Name AS ProductName

FROM

Production.Product AS p

INNER JOIN

Sales.SalesOrderDetail AS od

ON

p.ProductID = od.ProductID

INNER JOIN (

SELECT

CustomerID,

ShipToAddressID

FROM

Sales.SalesOrder

GROUP BY

CustomerID, ShipToAddressID

HAVING

COUNT(DISTINCT SalesOrderID) > 1

) AS subq

ON

od.CustomerID = subq.CustomerID

AND

od.ShipToAddressID = subq.ShipToAddressID;

**49. Display orders with quantities higher than the average quantity for a specific product.**

SELECT

o.SalesOrderID,

p.Name AS ProductName,

od.OrderQty

FROM

Sales.SalesOrder AS o

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

WHERE

od.OrderQty > (

SELECT AVG(OrderQty)

FROM Sales.SalesOrderDetail

WHERE ProductID = 'your\_specific\_product\_id\_here'

);

**50. Find customers who have placed orders for products that have not been ordered by any other customer.**

SELECT

c.FirstName AS CustomerFirstName,

c.LastName AS CustomerLastName

FROM

Person.Person AS c

INNER JOIN

Sales.Customer AS cust

ON

c.BusinessEntityID = cust.PersonID

INNER JOIN

Sales.SalesOrder AS o

ON

cust.CustomerID = o.CustomerID

INNER JOIN

Sales.SalesOrderDetail AS od

ON

o.SalesOrderID = od.SalesOrderID

INNER JOIN

Production.Product AS p

ON

od.ProductID = p.ProductID

WHERE

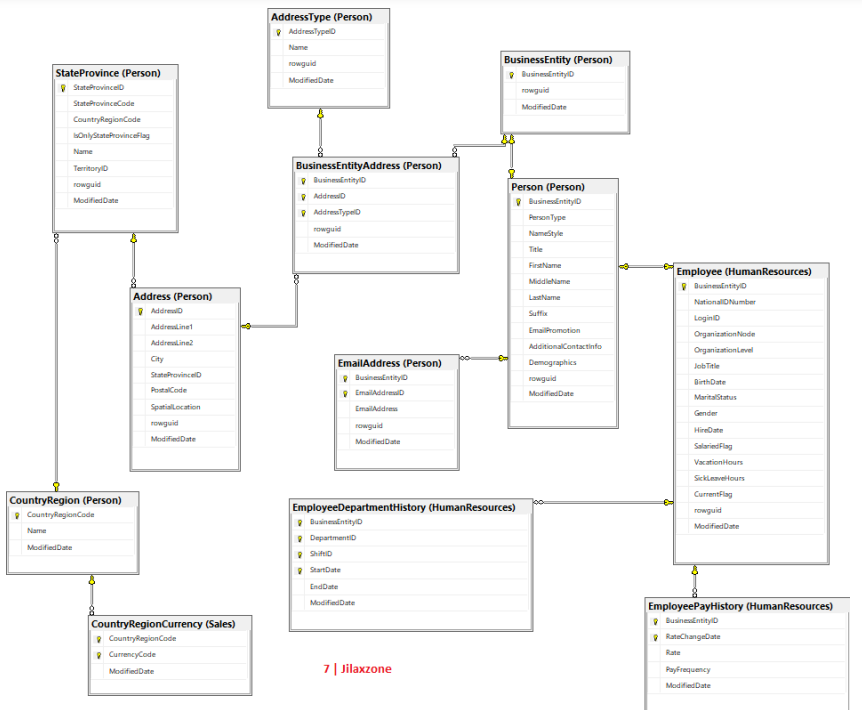
p.ProductID NOT IN (

SELECT DISTINCT od2.ProductID

FROM Sales.SalesOrderDetail AS od2

WHERE od2.SalesOrderID <> o.SalesOrderID

);



**According to this DB schema only the below given queries can be done.**

* **Find employees with salaries greater than the average salary in their department (Rate= Salary, OrganizationNode= Departments).**

SELECT e.BusinessEntityID, e.JobTitle, eph.Rate AS Salary

FROM Employee e

INNER JOIN EmployeePayHistory eph ON e.BusinessEntityID = eph.BusinessEntityID

WHERE eph.Rate > (

SELECT AVG(eph2.Rate)

FROM EmployeePayHistory eph2

INNER JOIN Employee e2 ON eph2.BusinessEntityID = e2.BusinessEntityID

WHERE e2.OrganizationNode = e.OrganizationNode

);

* **Combine results from two queries to get a list of unique customer and employee names.**

SELECT FirstName + ' ' + LastName AS Name

FROM Person

WHERE PersonType = 'EMPLOYEES'

UNION

SELECT FirstName + ' ' + LastName

FROM Person

WHERE PersonType = 'CUSTOMERS' ;