#### -- overview database

# SELECT TOP 10 \* FROM [Sales].[SalesOrderDetail]

100 %	00 % 🔻 🜓											
■ Results   Messages												
	SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	rowguid		
1	43659	1	4911-403C-98	1	776	1	2024.994	0.00	2024.994000	B207C96D-D9E6-402B-8		
2	43659	2	4911-403C-98	3	777	1	2024.994	0.00	6074.982000	7ABB600D-1E77-41BE-9		
3	43659	3	4911-403C-98	1	778	1	2024.994	0.00	2024.994000	475CF8C6-49F6-486E-B		
4	43659	4	4911-403C-98	1	771	1	2039.994	0.00	2039.994000	04C4DE91-5815-45D6-8		
5	43659	5	4911-403C-98	1	772	1	2039.994	0.00	2039.994000	5A74C7D2-E641-438E-A		
6	43659	6	4911-403C-98	2	773	1	2039.994	0.00	4079.988000	CE472532-A4C0-45BA-8		
7	43659	7	4911-403C-98	1	774	1	2039.994	0.00	2039.994000	80667840-F962-4EE3-96		
8	43659	8	4911-403C-98	3	714	1	28.8404	0.00	86.521200	E9D54907-E7B7-4969-80		

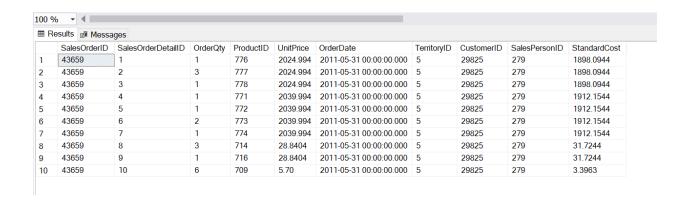
# SELECT TOP 10 \* FROM [Sales].[SalesOrderHeader]

⊞ Results № Messages											
	SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	Purchase		
3	43661	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43661	PO18473		
4	43662	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43662	PO18444		
5	43663	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43663	PO18009		
6	43664	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43664	PO16617		
7	43665	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43665	PO16588		
8	43666	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43666	PO16008		
9	43667	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43667	PO15428		
10	43668	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	SO43668	PO14732		

# SELECT TOP 10\* FROM [Production].[ProductCostHistory]

<					
	ProductID	StartDate	EndDate	StandardCost	ModifiedDate
3	707	2013-05-30 00:00:00.000	NULL	13.0863	2013-05-16 00:00:00.000
4	708	2011-05-31 00:00:00.000	2012-05-29 00:00:00.000	12.0278	2012-05-29 00:00:00.000
5	708	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	13.8782	2013-05-29 00:00:00.000
6	708	2013-05-30 00:00:00.000	NULL	13.0863	2013-05-16 00:00:00.000
7	709	2011-05-31 00:00:00.000	2012-05-29 00:00:00.000	3.3963	2012-05-29 00:00:00.000
8	710	2011-05-31 00:00:00.000	2012-05-29 00:00:00.000	3.3963	2012-05-29 00:00:00.000
9	711	2011-05-31 00:00:00.000	2012-05-29 00:00:00.000	12.0278	2012-05-29 00:00:00.000
10	711	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	13.8782	2013-05-29 00:00:00.000

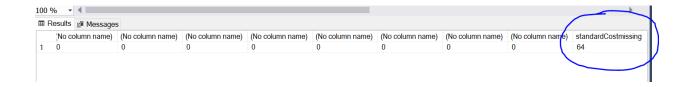
```
-- create a temp table #SALE
SELECT
       saleD.SalesOrderID,
          saleD.SalesOrderDetailID,
          saleD.OrderQty,saleD.ProductID,
          saleD.UnitPrice,saleH.OrderDate,
          saleH.TerritoryID,
          saleH.CustomerID,
          saleH.SalesPersonID,
         c.StandardCost
         INTO #SALE
  FROM [Sales].[SalesOrderDetail] as saleD
  JOIN [Sales].[SalesOrderHeader] as saleH
   ON saleD.SalesOrderID = saleH.SalesOrderID
  LEFT JOIN [Production].[ProductCostHistory] as c
   ON saleD.ProductID=c.ProductID
          AND saleH.OrderDate >=c.StartDate
         AND saleH.OrderDate <= COALESCE(c.EndDate, GETDATE());
SELECT TOP 10 *
  FROM #SALE
```



#### -- check missing value

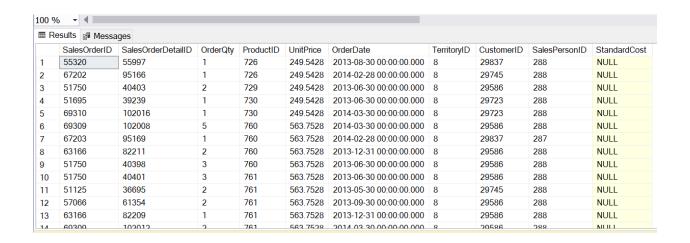
```
SELECT count(*) - Count(SalesOrderID),
count(*) - Count(SalesOrderDetailID),
count(*) - Count(OrderQty),
count(*) - Count(OrderDate),
count(*) - Count(ProductID),
count(*) - Count(UnitPrice),
count(*) - Count(TerritoryID),
count(*) - Count(CustomerID),
count(*) - Count(SalesOrderID),
```

count(\*) - Count(StandardCost) AS standardCostmissing
FROM #SALE;



-- show the missing value

SELECT \*
 FROM #SALE
WHERE StandardCost is NULL
ORDER BY ProductID;



- -- The reason why StandardCost return to NULL for some rows is in the ProductCostHistory the EndDate is less than the OrderDate in SaleHeader table.
- -- So the solution is create a table called MaxCost to get the lasted cost for each ProductID and after that join into the #SALE table
- -- correct the null value
- -- Create a table with the last price for each productid

with lastchange AS

```
(SELECT
          ProductID, MAx(ModifiedDate) AS lastchangedate
FROM [Production].[ProductCostHistory]
GRoup by ProductID)

SELECT 1.ProductID,p.StandardCost

INTO #MaxCost

FROM [Production].[ProductCostHistory] as p
JOIN lastchange as 1
ON     1.ProductID = p.ProductID
AND     1.lastchangedate =p.ModifiedDate
ORDER BY 1.ProductID
```

## SELECT TOp 10 \* FROM #MaxCost

≣ Re	■ Results							
	Prod	uctID	StandardCost					
1	999		343.6496					
2	998		343.6496					
3	997		343.6496					
4	996		53.9416					
5	995		44.9506					
6	994		23.9716					
7	993		294.5797					
8	992		294.5797					
9	991		294.5797					
10	990		294.5797					

-- create #SALEFINAL table hen fix teh null value of #SALE table

#### **SELECT**

```
SalesOrderID,
SalesOrderDetailID,
OrderQty,
#SALE.ProductID,
UnitPrice,
OrderDate,
TerritoryID,
CustomerID,
SalesPersonID,
```

```
coalesce(#SALE.StandardCost, #MaxCost.StandardCost) AS Cost
INTO #SALEFINAL
FROM #SALE
JOIN #MaxCost
ON #SALE.ProductID = #MaxCost.ProductID;
-- show first 100 rows of SALEFINAL
-- Check the missing value of #SALEFINAL
SELECT count(*) - Count(SalesOrderID),
count(*) - Count(SalesOrderDetailID),
count(*) - Count(OrderQty),
count(*) - Count(OrderDate),
count(*) - Count(ProductID),
count(*) - Count(UnitPrice),
count(*) - Count(TerritoryID),
count(*) - Count(CustomerID),
count(*) - Count(SalesOrderID),
count(*) - Count(Cost)
FROM #SALEFINAL;
100 % - 4
 ■ Results  Messages
          (No column name) (No co
       0
-- checking the duplicate data
SELECT SalesOrderID, SalesOrderDetailID
     FROM #SALEFINAL
     GROUP BY SalesOrderID, SalesOrderDetailID
    HAVING COUNT(*) >1
SELECT SalesOrderID,SalesOrderDetailID,OrderQty
     FROM #SALEFINAL
     GROUP BY SalesOrderID,SalesOrderDetailID,OrderQty
     HAVING COUNT(*) >1
SELECT SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate
     FROM #SALEFINAL
     GROUP BY SalesOrderID,SalesOrderDetailID,OrderQty,OrderDate
    HAVING COUNT(*) >1
SELECT SalesOrderID,SalesOrderDetailID,OrderQty,OrderDate,ProductID
     FROM #SALEFINAL
     GROUP BY SalesOrderID,SalesOrderDetailID,OrderQty,OrderDate,ProductID
    HAVING COUNT(*) >1
SELECT SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice
     FROM #SALEFINAL
```

```
GROUP BY SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice
  HAVING COUNT(*) >1
SELECT SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID
  FROM #SALEFINAL
  GROUP BY
SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID
  HAVING COUNT(*) >1
SELECT
SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
  FROM #SALEFINAL
  GROUP BY
SalesOrderID, SalesOrderDetailID, OrderOty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
  HAVING COUNT(*) >1
SELECT
SalesOrderID, SalesOrderDetailID, OrderOty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
erID, SalesOrderID
  FROM #SALEFINAL
  GROUP BY
SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
erID,SalesOrderID
  HAVING COUNT(*) >1
SELECT
SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
erID,SalesOrderID,Cost
  FROM #SALEFINAL
  GROUP BY
SalesOrderID, SalesOrderDetailID, OrderQty, OrderDate, ProductID, UnitPrice, TerritoryID, Custom
erID,SalesOrderID,Cost
  HAVING COUNT(*) >1
■ Results Messages
    SalesOrderID SalesOrderDetailID OrderQty OrderDate ProductID UnitPrice TerritoryID CustomerID SalesOrderID Cost
-- cheking the invalid data
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(SalesOrderID) = 0;
SELECT *
```

```
FROM #SALEFINAL
WHERE ISNUMERIC(SalesOrderDetailID) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(OrderQty) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISDATE(OrderDate) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(ProductID) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(UnitPrice) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(TerritoryID) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(CustomerID) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(SalesOrderID) = 0;
SELECT *
FROM #SALEFINAL
WHERE ISNUMERIC(Cost) = 0;
00 % ▼ 4

■ Results 
■ Messages

   SalesOrderID SalesOrderDetailID OrderQty ProductID UnitPrice OrderDate TerritoryID CustomerID SalesPersonID Cost
```

```
-- create Product table

SELECT

pro.ProductID,
    subca.ProductCategoryID,
    subca.ProductSubcategoryID,
    proca.Name as ProductCategoryName,
    subca.Name AS ProductSubCategory,
    pro.Name AS ProductName

INTO #PRODUCT
```

```
FROM [Production].[ProductCategory] as proca
JOIN [Production].[ProductSubcategory] as subca
ON proca.ProductCategoryID=subca.ProductCategoryID
JOIN [Production].[Product] as pro
ON pro.ProductSubcategoryID = subca.ProductSubcategoryID
```

### SELECT \* FROM #PRODUCT

⊞ K	esults 📠 Me	essages				
	ProductID	ProductCategoryID	ProductSubcategoryID	ProductCategoryName	ProductSubCategory	ProductName
1	680	2	14	Components	Road Frames	HL Road Frame - Black, 58
2	706	2	14	Components	Road Frames	HL Road Frame - Red, 58
3	707	4	31	Accessories	Helmets	Sport-100 Helmet, Red
4	708	4	31	Accessories	Helmets	Sport-100 Helmet, Black
5	709	3	23	Clothing	Socks	Mountain Bike Socks, M
6	710	3	23	Clothing	Socks	Mountain Bike Socks, L
7	711	4	31	Accessories	Helmets	Sport-100 Helmet, Blue
8	712	3	19	Clothing	Caps	AWC Logo Cap
9	713	3	21	Clothing	Jerseys	Long-Sleeve Logo Jersey, S
10	714	3	21	Clothing	Jerseys	Long-Sleeve Logo Jersey, M
11	715	3	21	Clothing	Jerseys	Long-Sleeve Logo Jersey, L
12	716	3	21	Clothing	Jerseys	Long-Sleeve Logo Jersey, XL
13	717	2	14	Components	Road Frames	HL Road Frame - Red, 62
1/	71Ω	2	1/	Components	Dood Framos	HI Pood Framo Pod 44

# SELECT \* INTO #TERRITORY FROM [Sales].[SalesTerritory]

#### SELECT \*

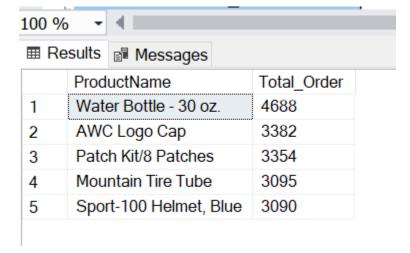
FROM #TERRITORY



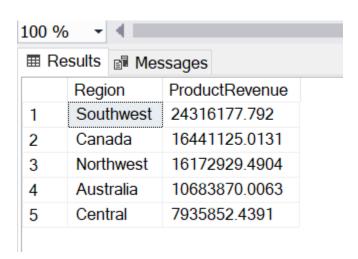


- 2. top 5 product have a highest orders ?

```
SELECT top 5 p.ProductName, COUNT(distinct s.SalesOrderID) as Total_Order
FROM #SALE as s
JOIN #PRODUCT as p
ON s.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY Total_Order DESC;
```



# --3. Region has highest revenue ? SELECT top 5 t.Name AS Region, SUM(s.UnitPrice\*s.OrderQty) AS ProductRevenue FROM #SALEFINAL as s JOIN #TERRITORY as t ON s.TerritoryID=t.TerritoryID GROUP BY t.Name ORDER BY ProductRevenue DESC;



```
-- 4.Top 5 Product have higest orders

SELECT TOP 5 s.ProductID,p.ProductName,COUNT(*) AS NumberOfOrders

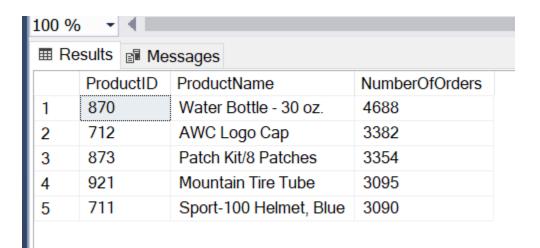
FROM #SALEFINAL as s

JOIN #PRODUCT as p

ON s.ProductID = p.ProductID

GROUP BY s.ProductID,p.ProductName

ORDER BY COUNT(*) DESC;
```



```
-- 6. Show the profit margin of each Product Category
SELECT p.ProductCategoryName,SUM(s.UnitPrice*s.OrderQty) AS
Revenue,SUM(s.OrderQty*s.Cost) AS Cost,(SUM(s.UnitPrice*s.OrderQty)-
SUM(s.OrderQty*s.Cost)) / SUM(s.UnitPrice*s.OrderQty) AS ProfitMargin
FROM #SALEFINAL as s
JOIN #PRODUCT as p
ON s.ProductID = p.ProductID
GROUP BY p.ProductCategoryName;
```

# 100 % ▼ ◀ ■ Results ■ Messages

	ProductCategoryName	Revenue	Cost	ProfitMargin
1	Accessories	1278760.9125	637601.0385	0.5013
2	Bikes	95145813.3519	84131887.7555	0.1157
3	Clothing	2141507.0245	1750636.2931	0.1825
4	Components	11807808.0245	10768475.7137	0.088