Data Wrangling with SQL

MOHAMMED DANISH MUSTAFA

1.)

SQL Query:

```
-- CTE: Common Table Expression
-- let's first group all orderline per orderID to compute the total for each order
-- regardless of the customer
with
individualOrderTotal as (
select Ol.OrderID as OrderID, sum(Ol.Quantity * Ol.UnitPrice) as orderTotal
from Sales.OrderLines  as Ol
group by Ol.OrderID
),
-- CTE: Common Table Expression
-- we do the same for the invoiceline and group it per invoiceID and compute the total
individualInvoiceTotal as (
select Il.InvoiceID as InvoiceID, sum(Il.Quantity * Il.UnitPrice) as invoiceTotal
from Sales.InvoiceLines  as Il
group by Il.InvoiceID
)
-- Now we can form the query Result set
select  O.CustomerID,
                    C.CustomerName ,
```

```
        COUNT(O.OrderID) as TotalNBOrders ,
        COUNT(I.OrderID) as TotalNBInvoices,
        SUM(orderTotal) as OrdersTotalValue,
        SUM(invoiceTotal) as InvoicesTotalValue,
        ABS(SUM(orderTotal) - SUM(invoiceTotal)) as AbsoluteValueDifference

from    Sales.Orders as O,
        individualOrderTotal as Ol,
        Sales.Invoices as I,
        individualInvoiceTotal as Il,
        Sales.Customers as C
where  O.CustomerID = C.CustomerID
        -- only order transformed into invoice are considered by this filter
        and O.OrderID = I.OrderID
        and O.OrderID = Ol.OrderID
        and I.InvoiceID = Il.InvoiceID

group by O.CustomerID, C.CustomerName
order by AbsoluteValueDifference desc, TotalNBOrders, C.CustomerName
```

ANSWER: TRUE

2.)

SQL Query:

```
-- Let's first identify for customerID 1060 the first invoice line of the first invoice
select  I.CustomerID, I.InvoiceID,Il.InvoiceLineID, Il.Quantity, Il.UnitPrice
from    Sales.Invoices as I,
                Sales.InvoiceLines as Il
where  I.InvoiceID = Il.InvoiceID
                and I.CustomerID = 1060
order by I.InvoiceID, Il.InvoiceLineID

-- this gives us the appropriate InvoiceLineID 225394 where to modify the UnitPrice from 240 to 260 as requested

update Sales.InvoiceLines
set UnitPrice = 260 where InvoiceLineID = 225394

-- rerunning the First query gives us the expected Result Set
```

ANSWER: TRUE

4.)

SQL Query:

```
-- CTE: Common Table Expression
-- first we aggregate the loss for all the customers
with LostCustomersDeals as (
	SELECT		o.CustomerID,
			c.CustomerName,
			cg.CustomerCategoryName,
			sum(ol.Quantity*ol.UnitPrice) as TotalLoss
	FROM
		Sales.Orders as o,
		Sales.OrderLines as ol,
		Sales.Customers as c,
		Sales.CustomerCategories as cg
	WHERE NOT EXISTS
		(
			SELECT *
			FROM Sales.Invoices as i
			WHERE
				o.OrderID = i.OrderID
		)
```

```
                and ol.OrderID = o.OrderID
                and c.CustomerID = o.CustomerID
                and c.CustomerCategoryID = cg.CustomerCategoryID

        group by o.CustomerID, c.CustomerName,cg.CustomerCategoryName
)
-- then we "auto-join" this table with the one where the max loss is identified
-- based on category grouping, this is to identify the max loss responsible customer
select  lc1.CustomerCategoryName,
                lc1.TotalLoss,
                lc1.CustomerName,
                lc1.CustomerID

from LostCustomersDeals as lc1
        join (
                select  lc.CustomerCategoryName,
                                max(lc.TotalLoss) as maxLoss
                from LostCustomersDeals as lc
                group by lc.CustomerCategoryName
                ) as lc2
                on lc1.CustomerCategoryName = lc2.CustomerCategoryName
                and lc1.TotalLoss = lc2.maxLoss
order by lc1.TotalLoss desc
```

ANSWER: TRUE

5.)

SQL Query:

```
-- Query to find customers who have purchased all products,
-- customers who have made purchases encompassing all products in the available product catalog.
select * from Customer as C
where not exists (
	select * from Product as Pr
	where not exists (
		select * from Purchase as Pu1
		where C.CustomerId = Pu1.CustomerId
		and Pr.ProductId = Pu1.ProductId
		-- sub Query to filter customer based on their overall purchases
		-- if we set purchase threshold to 11 instead of 50 we can also include Leo to Sebastien
		and ( select SUM(Pu2.Qty) from Purchase as Pu2
		where C.CustomerId = Pu2.CustomerId
		group by Pu2.CustomerId ) >= 50
		));
```

ANSWER: TRUE