1. In SQL Server, assuming you can find the result by using both joins and subqueries, which one would you prefer to use and why?
   I prefer subqueries with CTE, it is more readable and better constructed.

2. What is CTE and when to use it?
   CTE = Common Table Expressions
   Use to create recursive query, to construct complex queries.

3. What are Table Variables? What is their scope and where are they created in SQL Server?
   Table Variable is a special type of the local variable that helps to store data temporarily, like the temp table but better performance with small data set.
   Well scoped. The lifetime of the table variable only lasts for the duration of the batch, function, or stored procedure.

4. What is the difference between DELETE and TRUNCATE? Which one will have better performance and why?
   DELETE removes rows one at a time.
   TRUNCATE removes all rows in a table by deallocating the pages that are used to store the table data.

5. What is Identity column? How does DELETE and TRUNCATE affect it?
   Identity column is the column in a table that is made up of values generated by the database.
   Delete retains the identity and does not reset it to the seed value.
   Truncate command will reset the identity to its seed value.

6. What is difference between "delete from table_name" and "truncate table table_name"?

   Truncate is faster compared to delete as it makes less use of the transaction log
   Truncate removes all records and does not fire triggers while delete has a condition statement.
   Truncate will reset the identity.

```
--1
select E.City
from Employees E
Intersect
select C.City
from Customers C

----2a
select C.city
from Customers C
where C.city not in (select E.city from Employees E)
```

```
--2b
select C.City
from Customers C
except
select E.City
from Employees E


--3
with orderCountCTE
as
(
	select Count(orderid) as "totalcount", ProductID from [Order Details] group by
ProductID
)
select p.ProductID, p.ProductName, ct.totalcount from Products p left join orderCountCTE
ct
on p.ProductID = ct.ProductID



--4
with productCountCTE
as
(
	select Count(ProductID) as "totalcount", OrderID from [Order Details] group by
OrderID
), customerIDByOrder
as
(
	select o.CustomerID, pc."totalcount"
	from Orders o, productCountCTE pc
	where o.OrderID = pc.OrderID
)
select c.City, ct.totalcount from Customers c left join customerIDByOrder ct
on c.CustomerID = ct.CustomerID
order by c.City

--5a


--5b
select ct.city, ct.totalcustomers from
(select count(CustomerID) totalcustomers, City from Customers group by City) ct
where ct.totalcustomers > 2

--6
with productCountCTE
as
(
	select Count(ProductID) as "totalcount", OrderID from [Order Details] group by
OrderID
), customerIDByOrder
as
(
	select o.CustomerID, pc."totalcount"
	from Orders o, productCountCTE pc
	where o.OrderID = pc.OrderID
```

```sql
), productCountByCity
as
(
	select c.City, ct.totalcount from Customers c left join customerIDByOrder ct
	on c.CustomerID = ct.CustomerID
)
select pc.city, pc.totalcount
from productCountByCity pc
where pc.totalcount > 1

--7
select distinct c.CustomerID, City from Customers c
left outer join (select o.CustomerID, o.ShipCity from Orders o) ct
on c.CustomerID = ct.CustomerID

--8


--9a
select city from Employees where
city = (Select City from Customers as c left outer join Orders as o on c.CustomerID =
o.CustomerID)

--9b
select city from Employees
Union
Select City from Customers as c left outer join Orders as o on c.CustomerID =
o.CustomerID



--recursion

Create table #mymap(dest int, dist int);
insert into #mymap values (1, 0);
insert into #mymap values (2, 20);
insert into #mymap values (3, 50);
insert into #mymap values (4, 70);
insert into #mymap values (5, 85);


with mapHierarchyCTE
as
(
select dest, dist, 0 as gap
from #mymap
where dest = 1
union all
select m.dest, m.dist, m.dist - mapHierarchyCTE.dist as gap
from #mymap m
join mapHierarchyCTE on m.dest = mapHierarchyCTE.dest + 1
)
select dest, gap from mapHierarchyCTE;
```