-- These are additional examples
-- Run them on SQL Azure: https://xbn4slk6hn.database.windows.net, in your own database

<table>
<thead>
<tr>
<th>Purchase(pid, product, price, quantity)</th>
<th>--</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product(pid, pname, manufacturer)</td>
<td>--</td>
</tr>
</tbody>
</table>

-- in SQL Azure we must either have primary key or define an explicit clustered index
-- or, run this if no primary key: create clustered index purchasepid on Purchase(pid);
create table Purchase (pid int primary key, product varchar(20), price float, quantity int, month varchar(15));
insert into Purchase values(01, 'bagel', 1.99, 20, 'september');
insert into Purchase values(02, 'bagel', 2.50, 12, 'december');
insert into Purchase values(03, 'banana', 0.99, 9, 'september');
insert into Purchase values(04, 'banana', 1.59, 9, 'february');
insert into Purchase values(05, 'gizmo', 99.99, 5, 'february');
insert into Purchase values(06, 'gizmo', 99.99, 3, 'march');
insert into Purchase values(07, 'gizmo', 49.99, 3, 'april');
insert into Purchase values(08, 'gadget', 89.99, 3, 'january');
insert into Purchase values(09, 'gadget', 89.99, 3, 'february');
insert into Purchase values(10, 'gadget', 49.99, 3, 'march');

create table Product (pid int primary key, pname varchar(20), manufacturer varchar(50));
insert into product values(1, 'bagel', 'Sunshine Co.');
insert into product values(2, 'banana', 'BusyHands');
insert into product values(3, 'gizmo', 'GizmoWorks');
insert into product values(4, 'gadget', 'BusyHands');
insert into product values(5, 'powerGizmo', 'PowerWorks');

-- SUBQUERIES IN SQL:
-- can occur in three different places
-- 1. Subqueries in the SELECT clause

-- Query: for each month, compute the number of purchases with quantity > 5
select distinct x.month, (select count(*) from purchase y where x.month=y.month and y.quantity > 5) as total
from purchase x
where x.quantity > 5;

-- same as
select month, count(*)
from purchase
where quantity > 5
group by month;

-- Question1: why query is more efficient?
-- Question2: what happens if we change the first query to:
select distinct x.month, (select count(*) from purchase y where x.month=y.month and y.quantity > 5) as total
from purchase x
where x.quantity > 5;

-- Query: retrieve all products purchased, their prices, and their manufacturers:
select x.product, x.price, (select y.manufacturer from product y where x.product = y.pname)
from purchase x;

-- Question3: write an equivalent query without nested subqueries

-- Not all subqueries make sense in a SELECT clause.
-- Question4: why doesn't the following query work?
select y.manufacturer, (select x.product, x.price from purchase x where x.product=y.pname)
from product y;

-- 2. Subqueries in the FROM clause

-- find all months that had at least one purchase < $120.00;
-- of these retain only those months that also had at least one purchase > $12.00
select distinct x.month
from (select * from purchase y where y.price < 120.0) x
where x.price > 12.0;

-- Question 5: write an equivalent, unnested query

-- Application: finding witnesses
-- For each month, find the product that sold at the highest price.
--
-- Step 1: for each month find the highest price
select x.month, max(x.price) from purchase x group by x.month;

-- Step 2: using this as a subquery, find the "witness", i.e. the product that sold in that
-- month at that price
select y.month, z.price, z.product
from (select x.month, max(x.price) as maxprice from purchase x group by x.month) y,
purchase z
where y.month = z.month and y.maxprice = z.price;

-- 3. Subqueries in the WHERE clause
--
-- We need these especially in order to express universal quantifiers
--
-- Let's start with EXISTENTIAL QUANTIFIERS:

-- Query : find all products that sold at least once in a quantity >= 5
-- easy...
select distinct x.product
from purchase x
where x.quantity >= 5;

-- Contrieved way #1, using the quantifier EXISTS
select distinct x.pname
from product x
where exists (select * from purchase y where x.pname=y.product and y.quantity >= 5);

-- In general, exists(...some subquery...) does this:
-- it evaluates the subquery, and if the answer is non-empty, then returns TRUE, otherwise FALSE

-- Contrieved way #2, using the operator IN:
select distinct pname
from product
where pname in (select product from purchase where quantity >= 5);

-- In general, "value in (subquery)" checks if the value is among the answers returned by subquery

------------
-- Contrevied way #3, using the operator ANY:

select distinct x.manufacturer
from product x
where 5 <= any (select y.quantity from purchase y where x.pname = y.product);

-- SQLITE DOES NOT SUPPORT "ANY" and "ALL"!
-- make sure you use SQL Azure, or SQL Server

-- in general, "value op any (subquery)" checks if there exists a value returned by the subquery
-- that is in the relationship "op" with the value

-------------------------------------------
-- Next, let's do UNIVERSAL QUANTIFIERS:
-- Query: find all products that sold only in quantities >= 5
-- Equivalently, more verbose formaula:
-- find all products with the following property:
-- for all their purchases, quantity >= 5
-- Question 6: suppose a product did not sell at all, but it is still in "product"
-- Should we return it?
-- Similarly: suppose quantity=NULL for all purchases a product; Should we return it?
-- Writing this query is difficult; we do it in two steps.
-- Step 1: find THE WRONG products
-- i.e. who sold in some quantity < 5
-- we use one of the contrived queries above
-- Using alternative #1:

select distinct x.pname
from product x
where exists (select * from purchase y where x.pname = y.product and y.quantity < 5);

-- Step 2: negate!
-- modify the query to retrieve THE RIGHT manufacturers
-- i.e. who did not sell any product >= 5.00

select distinct x.pname
from product x
where not exists (select * from purchase y where x.pname = y.product and y.quantity < 5);

-- Using alternative #2:

select distinct x.pname
from product x
where x.pname in (select y.product from purchase y where y.quantity < 5);

-- Step 2: negate!

select distinct x.pname
from product x
where x.pname not in (select y.product from purchase y where y.quantity < 5);

-- Using alternative #3 (doesn't work on sqlite)

select distinct x.pname
from product x
where 5 > any (select y.quantity from purchase y where x.pname = y.product);
-- Step 2: negate !

select distinct x.pname
from product x
where 5 <= all (select y.quantity from purchase y where x.pname = y.product);