```sql
-- There are no quiz questions that use sub-queries.
-- you will need subqueries to accomplish the
-- "ItemsAvailable" procedure for the final project
--        (this is the most difficult procedure in the final project)
-- So there will also be additional practice in the pair assignment


/* TOPIC 1:  "IN" conditional accepts a comma-separated list
*/
Select * from Class where ID  IN   (1,2,3,4,5,6, -1000);
Select * from Class where Code IN   ('MA030','COMM113','ABCXYZ123');

-- (you can also use a subquery of 1 column and many rows - see
below)



--  Now for SubQueries!
/*  Some general comments:
There are many ways to use subqueries!

This review will show you a variety of ways to use them.
Each way that you apply them has distinct limitations.

This walk through  will show some of the common ways of using
subqueries

This assumes that you have some experience with how aggregations
and groupings will behave.

This also assumes that you have an understanding of rows vs
columns in a result set.

A subquery is helpful when you need to get the answer
to a compound question - where the answer to a one question
relies on the answer to another.

Note: "Compound" is not an industry term for this, it's just the way
I'm attempting to describe the situation.

*/
```

```sql
/* Topic 2:  Sub-Query in the WHERE clause,
             With 1 column and many ROWS in the result set:

for this scenario, you need to use the IN conditional operator
Instead of an “=” sign
(because you have many rows)

The IN conditional works for the question below, because we only have
one column.

The rows with this one column subquery are going to be
evaluated as if we had provided a comma-separated list.

*/



-- Question: What classes have students signed up?
Select * from Class
where ID   IN   (select ClassID from ClassStudent);

-- Limitation: only 1 column can be included the sub-select
-- Limitation: do not try to give this table an alias - it’s just
-- not allowed when the subquery is in the WHERE clause






/* Topic 3:  Sub-Query in the WHERE clause,
             With 1 column and 1 row in the result set:

for this scenario, you can use the equal ( = ) operator

Note: 1 row and 1 column subqueries are referred to as a “scalar
subquery”
*/

-- start with a basic question:
-- What was the most recent “SignUpDate” for any class?


-- we can use this aggregation query  to answer that question
```

```
select max(SignUpDate) as maxsignupdate
from ClassStudent;





--  Now I will expand my question to make it more complex:
-- "Which Student was the last to sign up for a class?"





-- Think -Pair-Share if time allows:
--    Why won't a simple query work? (why doesn't  a GROUP BY work?)



-- Let's Try it:

select max(SignUpDate), StudentID as maxsignupdate
from ClassStudent
GROUP BY StudentID;






-- To code this in SQL, you'll learn to break such questions into 2
-- distinct parts. Primarily because the answer to the second
-- question is dependent on answer to the first .

--  Question 1:  What is the most recent SignUpDate?
--  Question 2:  Who was that person?


-- Find the answer to question 1 first:
select max(SignUpDate)  from ClassStudent;




-- Put that into a subquery to get Answer 2:
```

```
Select *
FROM ClassStudent cs
WHERE cs.SignUpDate = (select max(SignUpDate)  from ClassStudent);
-- Ok, you may see the same time on all of your records due to how
-- we added many records at a time... this is the
-- drawback to not having  "real data"
/*
-- you can create variation by running this script
Update ClassStudent
SET SignUpDate = date_add(SignUpDate ,Interval ID HOUR )

*/


-- Limitations:  only 1 value can be returned , and only one row

-- though we didn’t include the student name, it’s
-- apparent that we can see the relevant data, and now that
-- I see the StudentID for just one record… I could JOIN
-- to get the other details..


-- Personal Challenge: Try joining this result to the Student table,
--    include “Student.FirstName” in the result set
--    to complete the answer.




/* Topic 4:   Sub-Query as a result set that can be JOINED
*/
-- Ok, so now we go yet another step further, and ask:
-- “Who was the last person to sign up for EACH class”


-- Write our 2 questions - since this seems like a compound question:
-- Question 1:  What is the most recent SignUpDate for each class
-- Question 2:  Who were those students ?

-- Answer 1: (most recent for each class)
select max(SignUpDate) as maxsignupdate
     , ClassID
```

```
from ClassStudent
GROUP BY ClassID;

-- Answer 2 is going to be more complex this time
-- I can no longer use a WHERE conditional… it just doesn’t work.
-- why??  Because there are now 2 columns.

-- Personal Challenge: try it out anyway.  What error do you get?



Select cs.*
FROM (SELECT
                max(SignUpDate) as maxsignupdate
                , ClassID
           FROM ClassStudent
           GROUP BY ClassID
 )  AS  csmax_Answer1   -- ”table” alias is required
JOIN  ClassStudent cs
     On cs.SignUpDate = csmax_Answer1.maxsignupdate
     And cs.ClassID = csmax_Answer1.ClassID;


-- Limitations: a table alias is required for the sub-select
-- Limitations: Alias the column names if you are using a function
-- I can use now the subquery like any other table

-- Personal Challenge:  what errors would you get if you
--                    don’t alias the columns with functions?

-- ALTERNATE 1:  another way to answer the same questions:


/* Topic 5:   Sub-Query with a correlated column in the WHERE clause
*/

Select cs.*
FROM ClassStudent cs
WHERE cs.SignUpDate = (SELECT max(SignUpDate)
                                FROM ClassStudent csMax
                                Where csMax.ClassID = cs.ClassID
                -- correlated!
                -- Now this runs on each ClassStudent “cs” record!
 )  ;



-- Limitations: only 1 column can be returned
-- Limitations: correlation always happens inside the subquery



-- Think-Pair-Share:   Why doesn't this query need or want a GROUP BY
clause?



-- ALTERNATE 2:  another way to answer the same questions:
/* Topic 6:   Sub-Query with a correlated column in the SELECT


Hey look: this is a scalar value per every ClassStudent "cs" record
*/
Select DISTINCT c.ClassID
      ,          (SELECT max(SignUpDate)
                      FROM ClassStudent csMax
                      Where csMax.ClassID = cs.ClassID  --
                correlated!
                 ) as maxSignUpDate
FROM ClassStudent cs;

-- Limitations: only 1 column can be returned
-- Limitations: correlation always happens inside the subquery
-- Limitations: only 1 value per correlation
--             (i.e. 1 value per row in the outer table)


-- Personal challenge: what happens if I have multiple rows return
from a sub-select in the SELECT clause?  Can you think of an example
and code to review the error?
```