STUDENTS IN ATTENDING STUDY GROUP: **KEY (Questions, alternatives, and bug reports to Discussion board)**

Use the handout of three tables (Vehicle, Own, Person) to answer these. Give SQL queries for each (give relational algebra versions ONLY if time remains).

STUDY GROUPS: FILL IN AS MANY ANSWERS AS YOU CAN AS A GROUP AND POST TO THE OAK DISCUSSION BOARD.

1. Assume that Vehicle only has the first two rows shown on the handout and that Person only has the first two rows shown on the handout. Show or otherwise describe the result on the natural join of Vehicle and Person. ***Natural join results in a pairing of rows in which a row of one table is paired with a row of another table if the two constituent rows are equal along all the same-named (and typed) attributes in the two tables. There are ZERO same-named attributes between Vehicle and Person, so all pairs of rows satisfy the constraint that ZERO same-named rows have the same value***

Vehicle ⋈ Person							
VRN	Ma	Mo	Color	SSN	Name	Addr	Phone
123	Honda	Hawk	Red	abc	Dave	Birch	xxx
123	Honda	Hawk	Red	bcd	Mary	Grove	yyy
234	Mazda	RX7	Blue	abc	Dave	Birch	xxx
234	Mazda	RX7	Blue	bcd	Mary	Grove	yyy

2. The reading/videos indicate that natural join is associative. So (Vehicle ⋈ Own) = (Own ⋈ Vehicle). Its also the case that ((Vehicle ⋈ Own) ⋈ Person) = ((Vehicle ⋈ (Own ⋈ Person)) = ((Vehicle ⋈ Person) ⋈ Own). Revisit question 1 again if your previous answer now causes you pause. What general property of natural join do these exercises suggest to you?

Answer: ***I wanted you to conclude that the natural join of two tables in which there was no same-named attributes was the cross-product of the two tables (and that the natural join in such a case was NOT the empty set)***

Give relational algebra and SQL expressions for each of the following queries specified in English (assuming the three tables, with all rows, of the handout). For RA expressions use the rename operator as appropriate. SQL statements, use

1. Write a query that returns the Name and Phone of all Persons owning VRN=123.

Style I like most (consistent use of row variables, such as O and P)

```
SELECT P.Name, P.Phone                    //order by presentation order
FROM Own O, Person P                      //any order
WHERE O.VRN = 123 AND O.SSN = P.SSN;
```

but other variants possible (recognizing where there is no confusion between attributes across tables in many cases), such as

```
SELECT Name, Phone
FROM Own, Person
WHERE Own.VRN = 123 AND Own.SSN = Person.SSN;
```

You might have missed the ';' at end – that's fine. Also, WHERE conditions can be listed in any order, though often I will put conditions involving only one table first.

2. Write a query that returns the Name and Phone of all Persons owning a Ford.

```
SELECT P.Name, P.Phone
FROM Vehicle V, Own O, Person P
WHERE V.Ma = 'Ford' AND V.VRN = O.VRN AND O.SSN = P.SSN;
```

3. Write a query that returns the Models and Colors of all vehicles owned by someone on Birch.

```
SELECT V.Mo, V.Color
FROM Vehicle V, Own O, Person P
WHERE P.Addr = 'Birch' AND P.SSN = O.SSN AND V.VRN = O.VRN;
```

4. Write a query to show the VRN and Mo of each owned Vehicle and the Name and Addr of the Person who owns it.

```
SELECT V.VRN, V.Mo, P.Name, P.Addr    // or O.VRN
FROM Vehicle V, Own O, Person P
WHERE V.VRN = O.VRN AND O.SSN = P.SSN;
```

4a. Write a query to show the VRN and Mo of each owned Vehicle BY SOMEONE ON 'Birch' and the Name of the Person who owns it. In SQL, write versions of the query that DO and do NOT use a subquery.

```
//Not using subquery:
SELECT V.VRN, V.Mo, P.Name
FROM Vehicle V, Own O, Person P
WHERE V.VRN = O.VRN AND O.SSN = P.SSN AND P.Addr = ‘Birch’;
```

--Using subqueries (but the above query is better – much simpler):

```
SELECT V.VRN, V.Mo, Temp.Name
FROM
    Vehicle V, Own O,
    (SELECT P.Name, P.SSN FROM Person P WHERE P.Addr = ‘Birch’)
           AS Temp
WHERE V.VRN = O.VRN AND O.SSN = Temp.SSN
```

Others possible – point of this exercise is that you can use subqueries to help break a problem up, solving a simplified problem, and then building on the solution of that simplified problem

5. Write a query that returns the VRN and Mo of any vehicle that isn't owned.

```
SELECT V.VRN, V.Mo
FROM Vehicle V
WHERE V.VRN NOT IN (SELECT O.VRN FROM Own O);
```

You could also probably adapt the EXCEPT operator

6. Write a query that returns the Name, Addr and Phone of any Person that doesn't own any Vehicle.

```
SELECT P.Name, P.Addr, P.Phone
FROM Person P
WHERE P.SSN NOT IN (SELECT O.SSN FROM Own O);
```

7. Write a query that returns pairs of SRNs for **(different)** Persons that live at the same Addr.

```
SELECT P1.SSN, P2.SSN
FROM Person P1, Person P2
WHERE P1.Addr = P2.Addr AND
          P1.SSN <> P2.SSN; //example of non-equality join condition
```

This query will list abc, efg AND efg, abc (for example)
If SSN of appropriate type, then following will list only one of these (e.g., abc,efg):

```
SELECT P1.SSN, P2.SSN
FROM Person P1, Person P2
WHERE P1.Addr = P2.Addr AND P1.SSN < P2.SSN;  // '<'
```

8. Write a query that returns pairs of Names for (different) Persons that live at the same Addr.

```
SELECT P1.Name, P2.Name     //names could be the same
FROM Person P1, Person P2
WHERE P1.Addr = P2.Addr AND P1.SSN < P2.SSN;
```

9. Make up your own query specification in English of a query that requires a "self-join" of Vehicle, then give the query on both relational algebra and SQL to your spec

…

10. Write a query that lists pairs of Vehicles by Ma and Mo that are owned by the same Person.

```
SELECT V1.Ma, V1.Mo, V2.Ma V2.Mo
FROM Vehicle V1, Vehicle V2, Own O1, Own O2, Person P
WHERE V1.VRN = O1.VRN AND O1.SSN = P.SSN AND
          P.SSN = O2.SSN AND O2.VRN = V2.VRN AND
          O1.VRN < O2.VRN // need this or other variation
```

11. Write a query that lists pairs of Persons by SSN and Name that own a Vehicle of the same Ma, Mo and Color

```
SELECT P1.SSN, P1.Name, P2.SSN, P2.Name
FROM Person P1, Person P2, Own O1, Own O2,
        Vehicle V1, Vehicle V2  //why two Vs and only one P in 10?
WHERE P1.SSN < P2.SSN // need this
    AND P1.SSN = O1.SSN AND O1.VRN = V1.VRN
    AND V1.VRN <> V2.VRN  //need this. Why not '<'?
    AND V1.Ma = V2.Ma AND V1.Mo = V2.Mo AND V1.Color = V2.Color
```

12. Suppose that there was a Price attribute on Vehicle. Write a query in which each car is listed (paired) with each other car that costs less than it does.

```
SELECT V1.VRN, V2.VRN
FROM Vehicle V1, Vehicle V2
WHERE V1.Price < V2.Price
```

Even if you didn't catch on with previous problems, this problem was to simply highlight that not all joins are equality joins