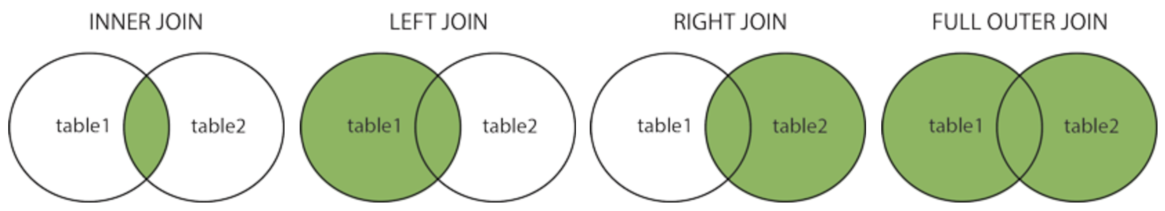
DS 100: Principles and Techniques of Data Science **Date: November 7, 2018**

Discussion #10 Solutions

Name:

SQL Joins



Note: You do not always have to use the JOIN keyword to join sql tables. The following are equivalent:

```
SELECT column1, column2
FROM table1, table2
WHERE table1.id = table2.id;

SELECT column1, column2
FROM table1 JOIN table2
ON table1.id = table2.id;
```

1. Describe which records are returned from each type of join.

Solution:

(INNER) JOIN: Returns records that have matching values in both tables
LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

SQL

2. Circle TRUE or FALSE.

(a) **TRUE** False SQL is a declarative language that specifies what to produce but not how to compute it.

> **Solution:** SQL is declarative programming language which specifies what the user wants to accomplish allowing the system to determine how to accomplish it.

(b) **TRUE** False The primary key of a relation is the column or set of columns that determine the values of the remaining column.

(c) True **FALSE** The schema of a table consists of the data stored in the table.

> **Solution:** The schema of a table consists of the column names, their types, and any constraints on those columns. The instance of a database is the data stored in the database.

(d) True **FALSE** The WHERE and HAVING clause can be used interchangeably as they perform the same operation.

> **Solution:** The having clause is used to filter out groups, while the where clause operates on individual rows.

Writing SQL Queries

Consider the following schema:

```
Clowns(cid int, cname text, booth text)
Balloons(bid int, bshape text, bcolor text)
Catalog(cid int, bid int, cost float)
```

Note: The Catalog table contains prices for Balloons sold by different Clowns standing at certain booths in a fair.

3. How may we query for the top 3 most expensive shapes sold by Whompers LeFou, ignoring the possibility that Whompers could be selling the same shape in different colors?

Solution:

```
SELECT bshape, cost
FROM Clowns, Balloons, Catalog
WHERE Clowns.cid=Catalog.cid
    AND Balloons.bid=Catalog.bid
    AND cname='Whompers LeFou'
ORDER BY cost DESC
LIMIT 3;
```

4. How may we query for the top 3 most expensive shapes sold by Whompers LeFou, taking into consideration the possibility that Whompers could be selling the same shape in different colors by using the highest-priced color of each shape?

Solution:

```
SELECT bshape, MAX(cost)
FROM Clowns, Balloons, Catalog
WHERE Clowns.cid=Catalog.cid
AND Balloons.bid=Catalog.bid
AND cname='Whompers LeFou'
GROUP BY bshape
ORDER BY cost DESC
LIMIT 3;
```

5. What is the average cost of a red balloon at booths that offer more than 3 red shapes per clown? Note that each clown at the booth does not necessarily have to be selling more than 3 shapes.

Solution:

```
SELECT booth, avg(cost)
FROM Clowns, Balloons, Catalog
WHERE Clowns.cid=Catalog.cid
    AND Balloons.cid=Catalog.cid
    AND bcolor='red'
GROUP BY booth
HAVING COUNT(DISTINCT bshape)/COUNT(DISTINCT Clowns.cid) > 3
```

You can play with a toy version of this schema at:

`https://tinyurl.com/ds100-clowns`

6. Consider the following real estate schema:

```
Homes(home_id int, city text, bedrooms int, bathrooms int,
area int)
Transactions(home_id int, buyer_id int, seller_id int,
transaction_date date, sale_price int)
Buyers(buyer_id int, name text)
Sellers(seller_id int, name text)
```

For the query language questions below, fill in the blanks in the answer to complete the query. For each SQL query and nested subquery, please start a new line when you reach a SQL keyword (SELECT, WHERE, AND, etc.). However, do not start a new line for aggregate functions (COUNT, SUM, etc.), and comparisons (LIKE, AS, IN, NOT IN, EXISTS, NOT EXISTS, ANY, or ALL.)

(a) Fill in the blanks in the SQL query to find the duplicate-free set of id's of all homes in Berkeley with at least 6 bedrooms and at least 2 bathrooms that were bought by "Bobby Tables."

```
SELECT DISTINCT H.home_id
FROM Homes H, Transactions T, Buyers B
WHERE H.home_id=T.home_id
AND T.buyer_id=B.buyer_id
AND H.city="Berkeley"
AND H.bedrooms>=6
AND H.bathrooms>=2
AND B.name='Bobby Tables';
```

(b) Fill in the blanks in the SQL query to find the id and selling price for each home in Berkeley. If the home has not ben sold yet, **the price should be NULL**.

```
SELECT H.home_id, T.sale_price
FROM Homes H
LEFT OUTER JOIN Transactions T
ON H.home_id = T.home_id
WHERE H.city = 'Berkeley';
```

Solution: An alternate solution was to use Transactions in the FROM clause and perform a RIGHT OUTER JOIN with Homes.

```
SELECT H.home_id, T.sale_price
FROM Transactions T
RIGHT OUTER JOIN Homes H
ON H.home_id=T.home_id
WHERE H.city = 'Berkeley'
```