# Joins - 2/20

## Announcements

- Lab 2 grades will be released by the end of this week.
- Lab 3 setup guide will be available by the end of the week, we'll be using Python so you need to install it on your own machines.

## Reading Quiz

- Q1: The result table is a virtual table that is stored in memory, not on disk.
- Q2: A query can have multiple joins.
- Q3: You can perform a join on tables that have different column names.
- Q4: Group join is not a type of join.
- Q5: Joins are expensive operations, especially on large tables.

## Joins

### Types

- `INNER JOIN` : Output contains all records that match between both tables.
- `LEFT OUTER JOIN` : Output contains all records that match plus the left tables records even where there is no match.
- `RIGHT OUTER JOIN` : Output contains all records that match plus the right tables records even where there is no match.
- `FULL OUTER JOIN` : Output contains all records that match plus the records from both tables even where there is no match.

Note: For outer joins, the records that do not have a match will be filled with empty values.

For the **outer join examples in the slides** the number of rows that will be returned are as follows:

- Left: 4
- Right: 5
- Full: 6

## Practice Problem 1

Use a `JOIN` between the Event, Category, Date, and Venue tables. Then use a `WHERE` clause to match only specific columns. We use `ORDER BY` to order how you want the results to be returned. We `SELECT` the columns that we want to be returned as well.

```
SELECT e.eventname, c.catname, v.venuecity, v.venuename, d.calcdate, e.starttime
FROM Event e
JOIN Category c ON e.catid = c.catid
JOIN Date d ON e.dateid = d.dateid
JOIN Venue v ON e.venueid = v.venueid
WHERE c.catgroup = 'Concerts'
AND d.month = 'MAR'
AND v.venuecity IN ('Austin', 'Dallas', 'Houston')
ORDER BY d.calcdate, e.eventname, v.venuecity;
```

## Practice Problem 2

Use an `OUTER JOIN` between the Users and Sales tables. Then look for entries which have empty values in the Sales table fields.

```
SELECT u.userid, u.firstname, u.lastname, u.email, u.city, u.state
FROM Users u
LEFT OUTER JOIN Sales s ON u.userid = s.buyerid
WHERE s.buyerid IS NULL
ORDER BY u.userid;
```

Some people might not have bought tickets because they are only ticket sellers. What if we wanted to find people who are not a buyer nor seller?

We would use the INTERSECT keyword.

```
SELECT u.userid, u.firstname, u.lastname, u.email, u.city, u.state
FROM Users u
LEFT OUTER JOIN Sales s ON u.userid = s.buyerid
WHERE s.buyerid IS NULL
INTERSECT
SELECT u.userid, u.firstname, u.lastname, u.email, u.city, u.state
FROM Users u
LEFT OUTER JOIN Sales s ON u.userid = s.sellerid
WHERE s.sellerid IS NULL;
```

## Practice Problem 3

Use a `FULL OUTER JOIN` between the Category and Event tables. Then look for entries which have empty values in the Category or the Event fields.

```sql
SELECT c.catid, c.catname, e.eventid, e.eventname
FROM Category c
FULL OUTER JOIN Event e ON c.catid = e.catid
WHERE c.catid IS NULL
OR e.catid IS NULL
ORDER BY c.catid, e.eventid;
```