

3. SQL DML and SQL SELECT Lab

CSCI 2541 Database Systems & Team Projects

Wood presented by Parmer

Announcements

Activity 2 and **Activity 3** are to be submitted via GitClassroom for this lab.

Expect to finish at least **Activity 1** and **Activity 2** in lab

Read the webpage for homework information.

Last time...

Python Flask

SQL DDL

SQL DML

this time...

SELECT Queries

Allow you to retrieve information from your database

Can be simple:

```
SELECT * FROM Instructor;
```

id	name	department	office
1	Cat	MENG	173
2	Sam	CIVIL	185

Or complex:

```
SELECT name, class, grade,  
ROW_NUMBER()  
OVER (  
PARTITION BY class  
ORDER BY grade DESC  
)  
AS rank_in_class  
FROM grades;
```

name	class	grade	rank_in_class
lucia	1	98	1
juan	1	93	2
chen	2	90	1
raph	2	88	2

SELECT Syntax

SQL Query

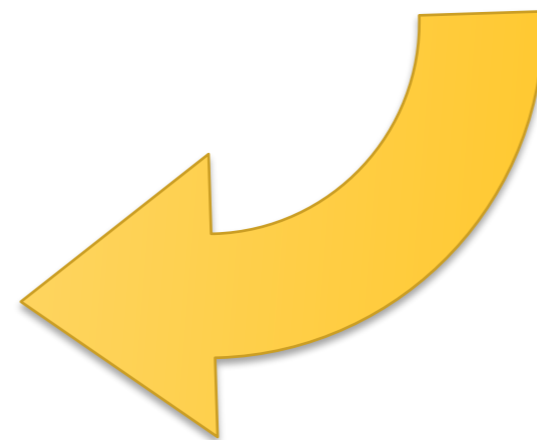
```
SELECT ID, office, name FROM  
instructor;
```

instructor Relation

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Sam	ECE	TOMP 231
E7	Sarah	LIT	Gelman 213
E8	Sarah	CS	SEH 125

Output

ID	office	name
E1	SEH 111	Sam
E2	SEH 231	Sam
E3	SEH 321	Lily
E4	SEH 451	Lily
E5	SEH 341	Nick
E6	TOMP 231	Sam
E7	Gelman 213	Sarah
E8	SEH 125	Sarah



SELECT + WHERE

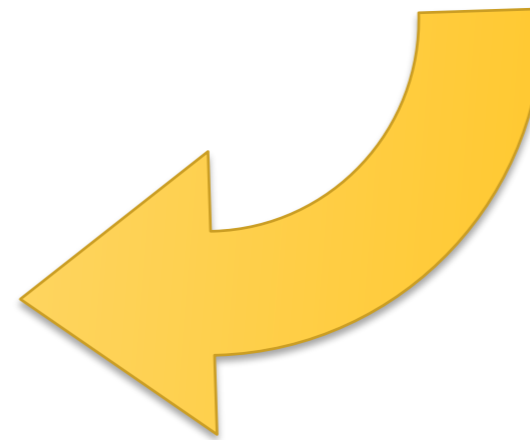
WHERE defines a predicate to match rows

```
SELECT name FROM instructor  
WHERE department = 'CS' OR  
department = 'EE';
```

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Susan	EE	TOMP 231
E7	Sarah	LIT	Gelman 213
E8	Sarah	CS	SEH 125

By default, SQL DBMS will not enforce set uniqueness in the output

name
Sam
Sam
Sarah
Susan



Logic operators:
AND, OR, NOT

SELECT + WHERE + LIKE + IN

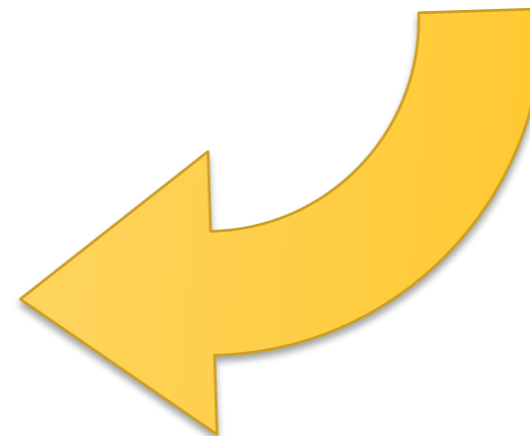
A WHERE predicate can use other keywords

- **LIKE** is used for string matching - % acts as a wildcard
- **IN** lets you list a set to test for equality

```
SELECT name FROM instructor  
WHERE office LIKE 'TOMP%' OR  
department IN ('BIO', 'ME');
```

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Susan	EE	TOMP 231
E7	Sarah	LIT	Gelman 213
E8	Sarah	CS	SEH 125

name
Lily
Nick
Susan



SELECT + ORDER + LIMIT

Tables don't have a defined sorting order, but you can control this with **ORDER BY**

LIMIT controls how many rows are returned

```
SELECT * FROM Instructor ORDER BY name LIMIT 5;
```

id	name	department	office
7	Birdie Greguoli	MATH	104
4	Brian Trewett	MATH	161
21	Candy Jeffries	CSCI	141
22	Carmon Londsdaal	STAT	145
20	Claiborne Titch	CIVIL	150

Activity 1 - Individual + Share

Course

ID	Title	InstructorID
1	Intro to EE	E1
2	Intro to CS	E2
3	Intro to ME	E3

Enrollment

CourseID	StudentID	Year
1	5	2020
2	6	2019
3	7	2020

Student

ID	name	Major	Email
5	Brooke Shimon	LIT	bshimon4@google.com
6	Dolf Fergusson	ME	dferg@gmail.com
7	Bebe Goggin	ECE	bgogg@gmail.com
8	George Harrison	BME	harrison@gmail.com
9	Bilbo Baggins	ME	myprecious@gmail.com

Instructor

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321

In a text document write out the sql syntax for the following four queries:

1. Find all the students.
2. Find only three instructors.
3. Find all students who have a major of CS or Biomedical Engineering
4. Find just the names of the courses sorted by instructorID.

Aggregation Functions

SQL can do some helpful calculations

- **AVG, MIN, MAX, SUM, COUNT**

```
SELECT COUNT(StudentID) FROM Enrollment WHERE CourseID=12;
```

```
COUNT(StudentID)
```

```
-----
```

```
4
```

```
SELECT MAX(year) AS LatestYear FROM Enrollment;
```

```
LatestYear
```

```
-----
```

```
2021
```

- Also: Use **AS** to rename/alias an output column!

Let's run some actual SQL!

We will use **sqlite**

- A simple “embedded” database -> you don't need to run a separate database server
- Comes with the python3 package

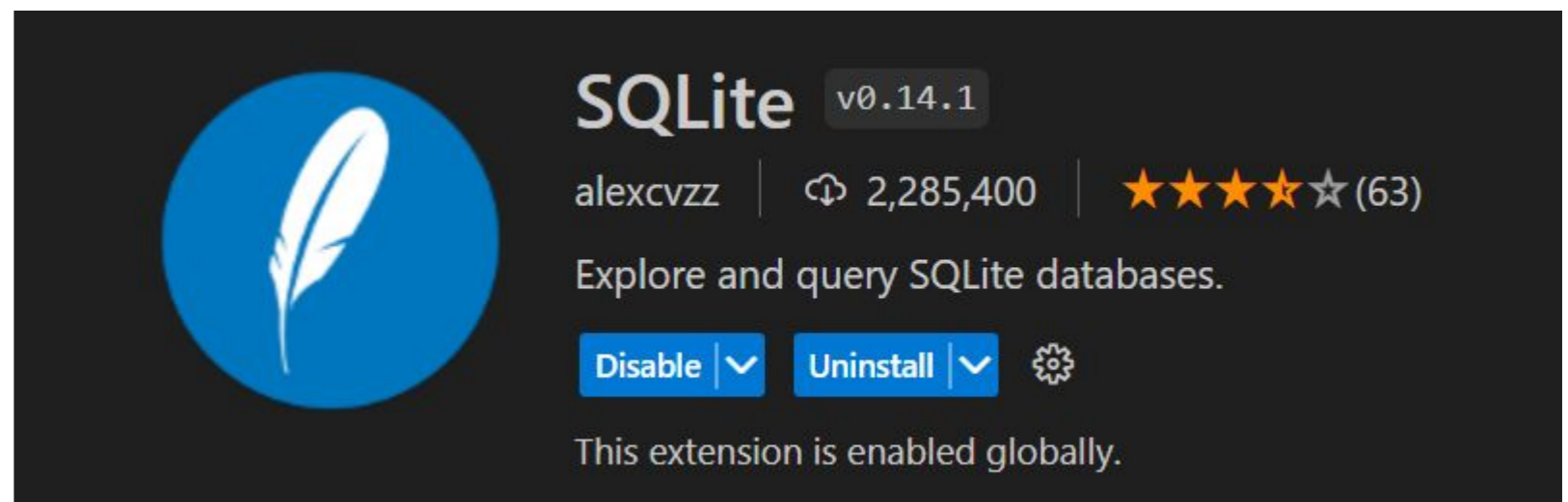
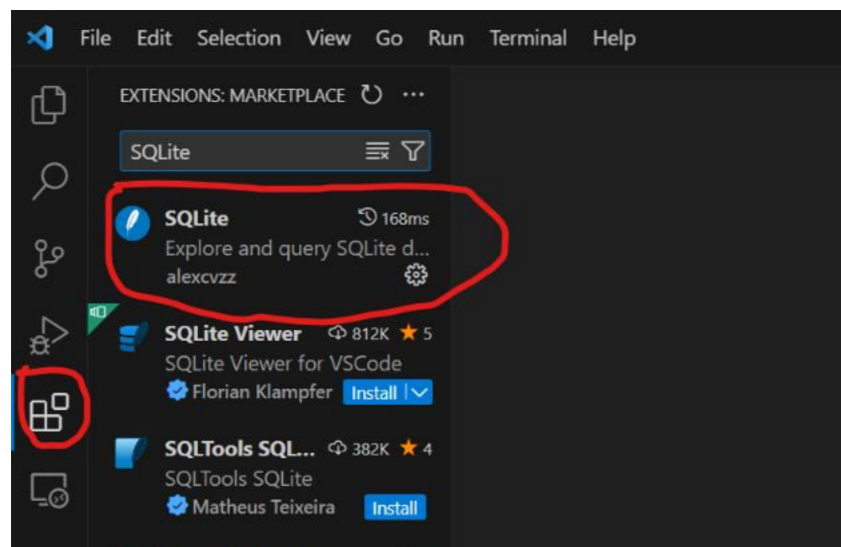
You can enter queries into the console, or you can write them in a file and “Run” the code

- We provide you a file that automatically creates a set of tables and loads data into it
- You will need to Run our script at least once to setup the data (running it multiple times is fine, the DB is reset with each run)

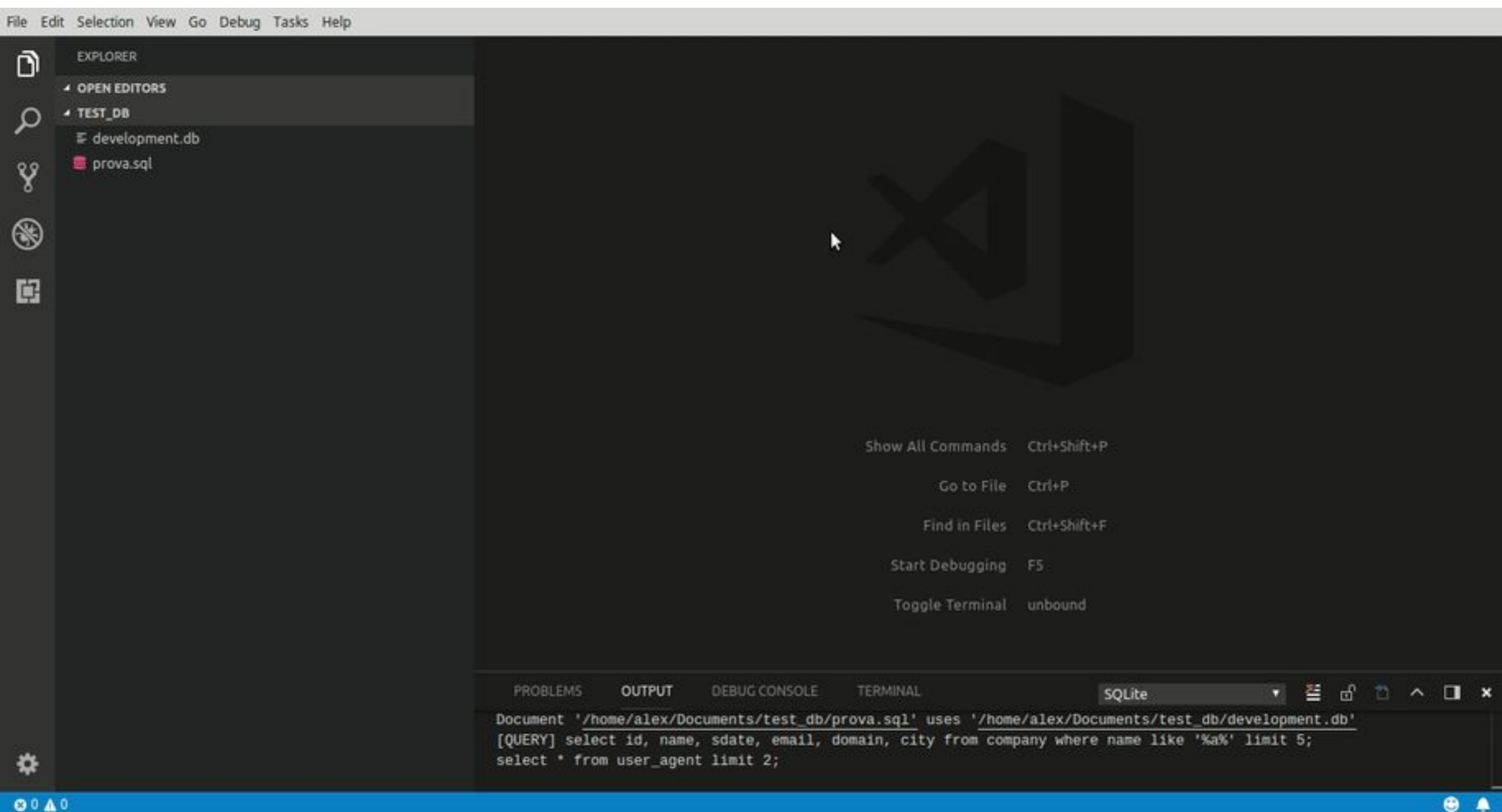
SQLite in VSCode

VSCode comes with an extension to run SQLite

- Search for the extension in VSCode, and install it
 - This allows for running .sql files and opening databases
- Instructions for installing and using it can be found in the lab github repository (and the developer page)



VSCode SQLite Usage



Activity 2 - Table Groups

Github link on the website (under /lab3/ page) in the `select-basics/` folder.

1. Enter all your queries from prior activity and make sure they work successfully.
 - Tables and column names may be slightly different!
 - View the `create.sql` file to understand schema

Complete the queries located in the `select-basics/main.sql` file

GROUP BY

What if we want to group together similar rows?

How many students are enrolled in each course?

CourseID	StudentID	Year
1	123	2019
2	123	2019
1	456	2019
1	678	2020
2	987	2019
...

We need to “group” rows based on the CourseID

Then we need to count the number of StudentIDs

GROUP BY

What if we want to group together similar rows?

How many students are enrolled in each course?

```
SELECT CourseID, COUNT(StudentID) AS NumS  
FROM Enrollment GROUP BY CourseID LIMIT 4;
```

CourseID	NumS
1	3
2	1
3	2
4	1

CourseID	StudentID	Year
1	123	2019
2	123	2019
1	456	2019
1	678	2020
2	987	2019
...

We need to “group” rows based on the CourseID

Then we need to count the number of StudentIDs

GROUP BY

What if we want to group together similar rows?

How many students are enrolled in each course?

```
SELECT CourseID, COUNT(StudentID) AS NumS  
FROM Enrollment GROUP BY CourseID;
```

VS

```
SELECT CourseID, COUNT(StudentID) AS NumS  
FROM Enrollment GROUP BY CourseID, Year;
```

CourseID	StudentID	Year
1	123	2019
2	123	2019
1	456	2019
1	678	2020
2	987	2019
...

What is the difference
between these?

GROUP BY and HAVING

How do we combine filtering and grouping?

- WHERE statements happen BEFORE grouping

```
SELECT CourseID, COUNT(StudentID) AS NumS  
FROM Enrollment WHERE Year=2019 GROUP BY CourseID;
```

To filter AFTER grouping, we use HAVING instead

```
SELECT CourseID, COUNT(StudentID) AS NumS  
FROM Enrollment GROUP BY CourseID HAVING NumS > 5;
```

- Should operate on aggregated fields

```
SELECT month  
FROM sales  
GROUP BY month  
HAVING SUM(price) > 100;
```

Month	Product	Price
Jan	tomato	2.00
Jan	grapes	3.99
Feb	water	1.99

Join Queries

Allow you to take two tables and combine them into one
instructor Relation

class Relation

ID	Instructor	Class name	Building
1	E5	BIO101	SEH
2	E3	ME201	SEH
3	E1	EE301	SEH
4	E7	LIT101	Gelman
5	E6	ECE102	TOMP
6	E4	CE101	SEH
7	Null	CS401	SEH

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Sam	ECE	TOMP 231
E7	Sarah	LIT	Gelman 213
E8	Sarah	CS	SEH 125
E9	David	CE	SEH 455

Join Queries: Inner Join

```
SELECT * FROM instructor INNER JOIN class ON
instructor.ID = class.Instructor
```

class Relation

ID	Instructor	Class name	Building
1	E5	BIO101	SEH
2	E3	ME201	SEH
3	E1	EE301	SEH
4	E7	LIT101	Gelman
5	E6	ECE102	TOMP
6	E4	CE101	SEH
7	Null	CS401	SEH

instructor Relation

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Kinga	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Sam	ECE	TOMP 231
E7	Sarah	LIT	Gelman 213
E9	David	CE	SEH 455

class.ID	Instructor	Class name	Building	instructor.ID	name	Department	Office
1	E5	BIO101	SEH	E5	Nick	BIO	SEH 341
2	E3	ME201	SEH	E3	Lily	ME	SEH 321
3	E1	EE301	SEH	E1	Sam	EE	SEH 111
4	E7	LIT101	Gelman	E7	Sarah	LIT	Gelman 213
5	E6	ECE102	TOMP	E6	Sam	ECE	Tomp 231
6	E4	CE101	SEH	E4	Lily	CE	SEH451

Join Queries: Left Join

```
SELECT * FROM class LEFT JOIN instructor ON
class.Instructor = instructor.ID
```

class Relation

ID	Instructor	Class name	Building
1	E5	BIO101	SEH
2	E3	ME201	SEH
3	E1	EE301	SEH
4	E7	LIT101	Gelman
5	E6	ECE102	TOMP
6	E4	CE101	SEH
7	Null	CS401	SEH

instructor Relation

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Kinga	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Sam	ECE	TOMP 231
E8	Sarah	CS	SEH 125
E9	David	CE	SEH 455

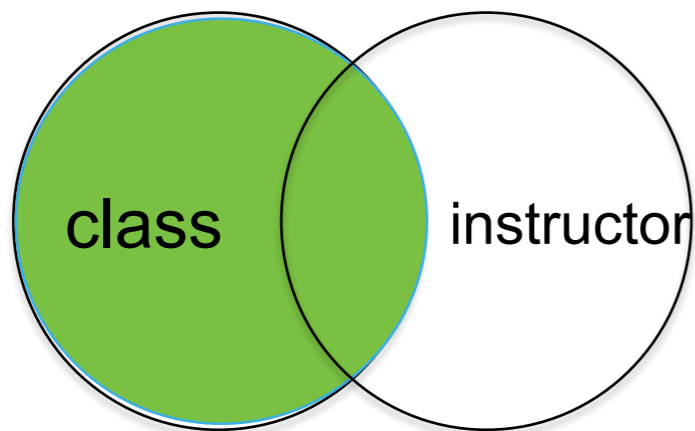
class.ID	Instructor	Class name	Building	instructor.ID	name	Department	Office
1	E4	CS101	SEH	E4	Lily	CE	SEH451
2	E5	BIO101	SEH	E5	Nick	BIO	SEH 341
3	E3	ME201	SEH	E3	Lily	ME	SEH 321
4	E1	EE301	SEH	E1	Sam	EE	SEH 111
5	E7	LIT101	Gelman	Null	Null	Null	Null
6	E6	ECE102	TOMP	E6	Sam	ECE	Tomp 231
7	Null	CS401	SEH	Null	Null	Null	Null

Remembering Join Queries

Allow you to take two tables and combine them into one

Left Join

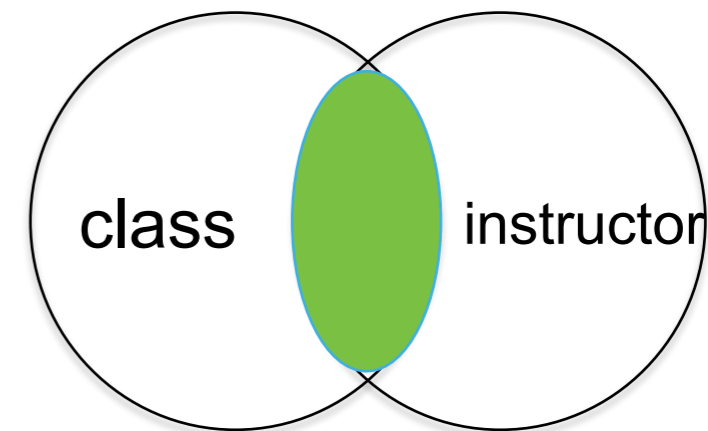
```
SELECT * FROM class  
LEFT JOIN instructor ON  
class.Instructor = instructor.ID
```



Includes every row in the left table,
rows not in the right table are set to null

Inner Join

```
SELECT * FROM instructor  
INNER JOIN class ON  
instructor.ID = class.Instructor
```



Only includes rows that
match the ON condition

Join Queries: Join + Where

```
SELECT * FROM instructor INNER JOIN class ON  
instructor.ID = class.Instructor WHERE NOT class.Building = "SEH"
```

class Relation

ID	Instructor	Class name	Building
1	E5	BIO101	SEH
2	E3	ME201	SEH
3	E1	EE301	SEH
4	E7	LIT101	Gelman
5	E6	ECE102	TOMP
6	E4	CE101	SEH
7	Null	CS401	SEH

instructor Relation

ID	name	department	office
E1	Sam	EE	SEH 111
E2	Sam	CS	SEH 231
E3	Lily	ME	SEH 321
E4	Lily	CE	SEH 451
E5	Nick	BIO	SEH 341
E6	Sam	ECE	TOMP 231
E7	Sarah	LIT	Gelman 213
E8	Sarah	CS	SEH 125
E9	David	CE	SEH 455

class.ID	Instructor	Class name	Building	instructor.ID	name	Department	Office
5	E7	LIT101	Gelman	E7	Sarah	LIT	Gelman 213
6	E6	ECE102	TOMP	E6	Sam	ECE	Tomp 231

Activity 3

Github (same as before):

Complete the queries inside `main.sql` inside `advanced-selections` located in your `github-classroom`

Complete the queries located in the `advanced-selections/main.sql` **file**

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

