CS 61A SU26

Lecture 23: Aggregation and Databases

July 30, 2026

Rebecca Dang

Join pollev.com/rebeccadang831 (or scan QR code) on your phone or laptop

We'll begin at Berkeley time (10 minutes after), as per tradition!

Potpourri

10 min

- Announcements
- Computing in the News
- Quiz 4 Poll

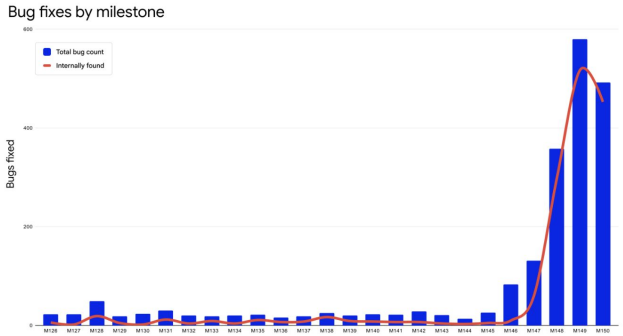
Announcements

- Final exam logistics and alterations form released
- Quiz 4 grades released
  - This quiz was more challenging on average for everyone compared to past quizzes
  - I will consider a grade adjustment policy for everyone once Quiz 5 grades are out, but nothing is guaranteed
- Reminder: Alex and Dakota's discussion sections are moving to Grimes starting next week
- Note: In the printout of Disc 10, it says that the "Slice It!" problem is tail recursive, but it is not

Computing in the News

Google says it fixed more Chrome bugs in June than over the past two years, thanks to AI (TechCrunch)

- "With help from its internal AI tools, Google says it has **patched more security flaws in its Chrome browser in the last month than in the past two years combined.**"
- "The tech giant announced on Thursday that it has fixed a whopping 1,072 security bugs in the last two versions of Chrome, both released in June. That is more than the number of bugs patched in the previous 23 versions released over the last two years, which totaled 1,036 fixes."



A CHART SHOWING THE NUMBER OF SECURITY BUGS FIXED IN RELEASED VERSIONS OF CHROME, WHICH GOOGLE CALLS "MILESTONES."

IMAGE CREDITS:GOOGLE

Quiz 4 Survey

PollEv

Review

15 min

- Dynamic Scope
- Correction: Macros
- SQL: Middle Managers

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

Consider Variant A and Variant B, with differences highlighted in yellow. What would be displayed by calling (f) as defined in Variant A vs. calling (f) as defined in Variant B?

Dynamic Scope (1 of 3)

```
(define x 1)
(define h (mu () x))
(define (g h)
    (h)
)
(define (f)
    (define x 2)
    (g h)
)

scm> (f)
1
```

Dynamic Scope (2 of 3)

```
(define x 1)
(define h (mu () x))
(define g (mu (h)
    (h)
))
(define (f)
    (define x 2)
    (g h)
)

scm> (f)
2
```

Dynamic Scope (3 of 3)

- To generalize, when you are doing variable lookup, if the variable doesn't exist in the current frame, look up the variable in the frame's parent (recursively).
- **The parent of a frame depends only on whether the frame was opened by calling a regular procedure or a mu procedure**, NOT the type of procedure that originally triggered the variable lookup
  - Regular: Use lexical scope
  - Mu: Use dynamic scope
- This can be seen in the previous example:
  - `x` evaluated to 1 in Scenario #1 because `g` was a regular procedure, so we used the `x` in the global frame where `g` was defined (even though we originally triggered lookup in `h`, which is a `mu` procedure)
  - `x` evaluated to 2 in Scenario #2 because `g` was a `mu` procedure, so we used the value of `x` in the `f` frame, since `f` called `g`

Correction: Macros (1 of 4)

Recall the check macro from Lecture 21:

```
(define-macro (check expr)
    (list 'if expr ''passed
        (list 'quote (list 'failed: expr))
    )
)
(define x -2)

scm> (check (> x 0))
(failed (> x 0))
```

Correction: Macros (2 of 4)

Recall: The macro procedure evaluation steps (from the Scheme Spec):

1. Evaluate the operator. If it is not a macro procedure, follow the normal call expression steps.
2. Apply the macro procedure from step 1 to the unevaluated operands.
3. Once the macro returns, evaluate that value in the calling environment.

I referred to Step 2 as the *"substitution step,"* and in the original slides I said we can just literally substitute, but that is inaccurate! **(I've updated Lec 21 slides 32-35.)**

It is really a ***quoted substitution*** (this is because calling a macro is equivalent to calling `eval` on the result of calling a regular procedure with quoted arguments). See Lec 21 slides.

Correction: Macros (3 of 4)

```
; original macro return expression
(list 'if expr ''passed
    (list 'quote (list 'failed: expr))
)

; substitute expr, NO quote
(list 'if (> x 0) ''passed
    (list 'quote (list 'failed: (> x 0)))
)

; evaluate list call expressions
(if #f 'passed
    (quote (failed: #f))
)
```

```
; alternative returned
(quote (failed: #f))

; interpreter displays the
; wrong thing!
(failed: #f)
```

Correction: Macros (4 of 4)

```
; original macro return expression
(list 'if expr ''passed
    (list 'quote (list 'failed: expr))
)

; substitute expr, WITH quote
(list 'if '(> x 0) ''passed
    (list 'quote (list 'failed: '(> x 0)))
)


; evaluate list call expressions
(if (> x 0) 'passed
    (quote (failed: (> x 0)))
)
```

```
; alternative returned
(quote (failed: (> x 0))


; displays correctly
(failed: (> x 0))
```

SQL: Middle Managers

- Implement Q1
- Download starter code `23.sql` from the course website under Lecture 23
- Manually test your code by copying and pasting it into a new file in code.cs61a.org, clicking on the green play button on the top right, and selecting all rows from that question's table
- 5 min: Try implementing it yourself
- 5 min: Discuss with someone next to you and compare your implementations
  - What worked?
  - What didn't?
  - Why?

Aggregation

25 min

- Group By
- Aggregation Functions
- Having
- Updated Order of Execution

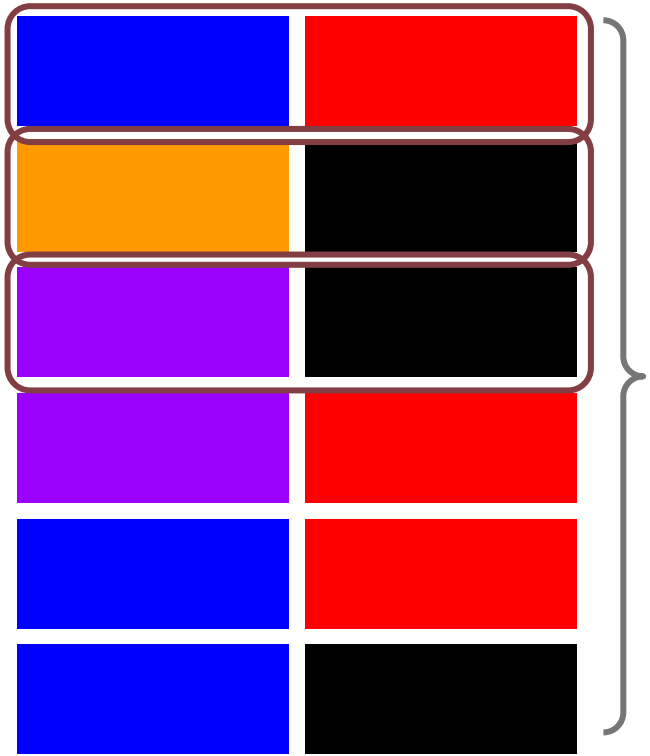
Group By (1 of 4)

`GROUP BY` allows us to **group** rows together and perform **aggregations** (combining the results of multiple rows) on them

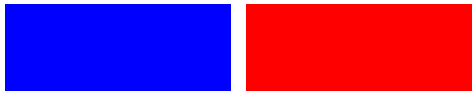
```
SELECT <columns>
FROM <table>
JOIN <table>
ON <condition>
WHERE <condition>
GROUP BY <columns>
ORDER BY <columns>
LIMIT <n>;
```

GROUP BY Visualization (1 of 2)

```
SELECT [left-column], COUNT(*)
FROM [table]
GROUP BY [left-column]
```



Blue

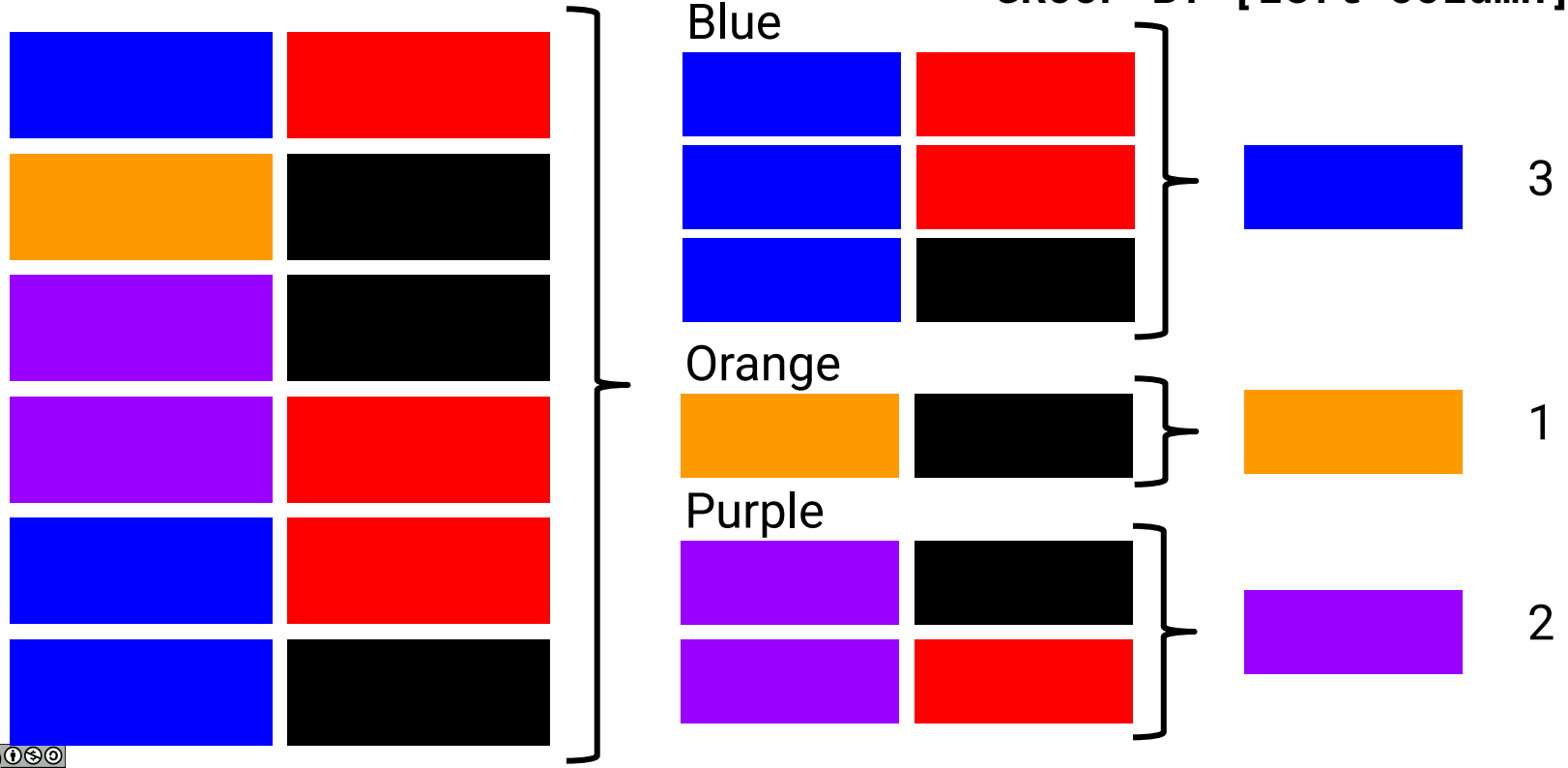


Orange

Purple

GROUP BY Visualization (2 of 2)

```
SELECT [left-column], COUNT(*)
FROM [table]
GROUP BY [left-column]
```



Group By (2 of 4)

name	division
Alyssa P Hacker	Computer
Ben Bitdiddle	Computer
Cy D Fect	Computer
Eben Scrooge	Accounting
Lana Lambda	Administration
Lem E Tweakit	Computer
Louis Reasoner	Computer
Oliver Warbucks	Administration
Robert Cratchet	Accounting

```
SELECT name, division
FROM records;
```

What happens if we run the following?

```
SELECT name, division
FROM records
GROUP BY division;
```

Group By (3 of 4)

name	division
Eben Scrooge	Accounting
Lana Lambda	Administration
Alyssa P Hacker	Computer

What happens if we run the following?

```
SELECT name, division
FROM records
GROUP BY division;
```

Group By (4 of 4)

- **In most variants of SQL,** if you have a `GROUP BY` clause, the columns in the `SELECT` clause must either be in the `GROUP BY` clause **OR** are inside of an aggregation function
- **SQLite** (which we use in 61A) **is not as strict** (they call such a column a "bare column")

Aggregation Functions

- `COUNT(*)` or `COUNT(column)` - counts number of rows
  - There is technically a difference between these, but it's not in scope for 61A
- `COUNT(DISTINCT column)` - counts the number of **unique** values in that column/group
  - You can also just use `SELECT DISTINCT <column>` to get the unique values in a column
- `MAX(column)` - computes the maximum value in that column/group
- `MIN(column)` - same as above, except the minimum value
- `AVG(column)` - same as above, except the average value
- `SUM(column)` - same as above, except the sum of all the values

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

Aggregation Functions I: What would SQL display?

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

Aggregation Functions II: What would SQL display?

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

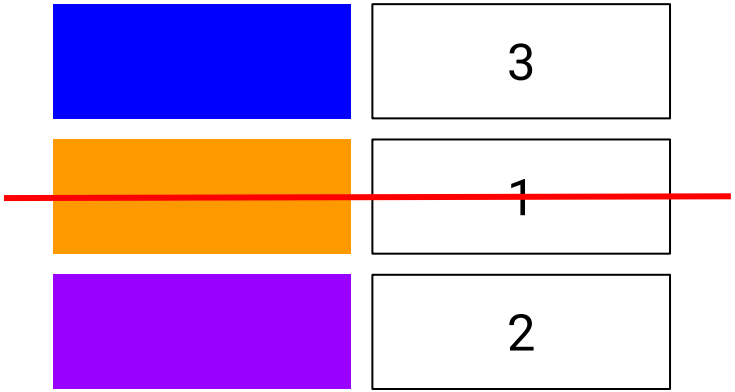
If video conferencing, you must be sharing your full screen to share the activated poll.

Aggregation Functions III: What would SQL display?

Having

- `HAVING` allows you to filter out groups based on some condition
  - Like the `WHERE` clause, but for groups instead of rows
  - You can call aggregation functions in a `HAVING` clause
- **IMPORTANT:** You can only use `HAVING` if you have `GROUP BY`

HAVING Visualization (continuation of previous GROUP BY visualization)




```
SELECT [left-column], COUNT(*)
    FROM [tbl]
    GROUP BY [left-column]
    HAVING COUNT(*) > 1
```

Updated Order of Execution

1. `FROM`, `JOIN … ON …`
2. `WHERE`, `LIKE`
3. **`GROUP BY`**
4. **`HAVING`**
5. `SELECT`
6. `AS`
7. `ORDER BY`
8. `LIMIT`

5 min break

Practice

(reminder to self to use high contrast theme)

35 min

- Setup
- Spotify
- IMDb

Setup

1. Go to https://github.com/phrdang/cs61a-su26-lec23
2. Download a `.zip` of the repository and unzip it (or alternatively, `git clone` the repo if you know how)
3. Follow all steps in the Installation section of the `README.md` file

(Demo and repo overview)

Spotify

- Implement the questions in `spotify.sql`
- Manually test your code by running it in your local SQLite shell (see `README.md` for instructions)
- 5 min: Try implementing it yourself
- 5 min: Discuss with someone next to you and compare your implementations
  - What worked?
  - What didn't?
  - Why?

IMDb

- IMDb is the Internet Movie Database
- Implement the questions in `imdb.sql`
- Manually test your code by running it in your local SQLite shell (see `README.md` for instructions)
- 5 min: Try implementing it yourself
- 5 min: Discuss with someone next to you and compare your implementations
  - What worked?
  - What didn't?
  - Why?