

# Discussion 11

## Regular Expressions, SQL

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

# Announcements

- **Scheme Checkpoint 1** due today (8/2)
- **Lab 10** due today (8/2)
- **HW 06** due Thursday (8/4)
- **Scheme Checkpoint 2** due Friday (8/5)
- Scheme contest due Friday (8/5)

# Regular Expressions [RegEx]

# Regular Expressions

- What are Regular Expressions?
  - A way to describe sets of strings that match certain criteria
  - Useful for pattern matching
- Resources to test Regular Expressions
  - <https://regexpr.com/>
  - <https://regex101.com/>

# Character classes

- Search for any one of a set of characters
- Can specify set or use pre-defined sets

Class	Description
[abc]	Matches <code>a</code> , <code>b</code> , or <code>c</code>
[a-z]	Matches any character between <code>a</code> and <code>z</code>
[^A-Z]	Matches any character that is not between <code>A</code> and <code>Z</code>
\w	Matches any "word" character. Equivalent to <code>[A-Za-z0-9_]</code>
\d	Matches any digit. Equivalent to <code>[0-9]</code>
[0-9]	Matches a single digit in the range 0 - 9. Equivalent to <code>\d</code>
\s	Matches any whitespace character (spaces, tabs, line breaks)
.	Matches any character besides new line

# Combining Patterns

- There are multiple ways to combine patterns together in regular expressions

Combo	Description
<code>AB</code>	Matches <code>a</code> , <code>b</code> , or <code>c</code>
<code>A B</code>	Matches either A or B. Example: <code>\d+   Inf</code> matches either a sequence containing 1 <b>OR</b> more digits or <code>"Inf"</code>

# Quantifiers

- A pattern can be followed by one of these quantifiers to specify how many instances of the pattern can occur

Combo	Description
-------	-------------

*	0 or more occurrences of the preceding pattern. Example: <code>[a-z]*</code> matches any sequence of lower-case letters or the empty string
---	---

+	1 or more occurrences of the preceding pattern. Example: <code>\d+</code> matches any non-empty sequence of digits.
---	---

?	0 or 1 occurrences of the preceding pattern. Example: <code>[-+]?</code> matches an optional sign.
---	--

{1, 3}	Matches the specified quantity of the preceding pattern. <code>{1, 3}</code> will match from 1 to 3 instances. <code>{3}</code> will match exactly 3 instances. <code>{3,}</code> will match 3 or more instances. Example: <code>\d{5, 6}</code> matches either 5 or 6 digit numbers
--------	--

# Groups

- Parentheses in RegEx are similar to arithmetic
  - `5 * 4 - 3`
  - `5 * (4 - 3)`
- `(Mahna)+`
  - matches strings with 1 or more `"Mahna"`
  - `"MahnaMahna"`
- `Mahna+`
  - match strings with `"Mahn"` followed by 1 or more `"a"` characters
  - `"Mahnaaaa"`
- Can use groups to determine what to output
  - Want to match valid phone numbers but ONLY want to output the area code



# Anchors

Symbol	Description
<code>^</code>	Matches the beginning of a string. Example: <code>^(I You)</code> matches <code>I</code> or <code>You</code> at the start of a string
<code>\$</code>	Normally matches the empty string at the end of a string or just before a newline at the end of a string
<code>\b</code>	Matches a "word boundary", the beginning or end of a word. Example: <code>s\b</code> matches <code>s</code> characters at the end of words

# Special Characters

The following special characters are used above to denote types of patterns:

```
\ ( ) [ ] { } + * ? | $ ^ .
```

That means if you actually want to match one of those characters, you have to escape it using a backslash. For example, `\(1\+3\)` matches

```
"(1 + 3)".
```

**Q1 , Q2**

# SQL

# SQL

- What is SQL?
  - A declarative programming language
  - Database management
  - Do not describe computations, but rather the desired result of computations
- Resources
  - <https://sql.cs61a.org>

# SELECT

- Create a table with two rows, columns as `first` and `last`

```
sqlite> SELECT "Ben" AS first, "Bitdiddle" AS last UNION  
...> SELECT "Louis", "Reasoner";
```

first	last
Ben	Bitdiddle
Louis	Reasoner

```
SELECT [columns]
```

# FROM

- SELECT specific values from an existing table with a FROM clause

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

name	division
Alyssa P Hacker	Computer
...	...
Robert Cratchet	Accounting

```
SELECT [columns] FROM [tables]
```

# WHERE

- filter out rows using a WHERE clause

```
sqlite> SELECT * FROM records WHERE title = "Programmer";
```

name	division	title	salary	supervisor
Alyssa P Hacker	Computer	Programmer	40000	Ben Bitdiddle
Cy D Fect	Computer	Programmer	35000	Ben Bitdiddle

```
SELECT [columns] FROM [tables] WHERE [condition]
```



# ORDER BY

- Order rows with the ORDER BY clause

```
sqlite> SELECT name, salary FROM records  
...> WHERE division = "Accounting" ORDER BY salary desc;
```

name	salary
Eben Scrooge	75000
Robert Cratchet	18000

```
SELECT [columns] FROM [tables] WHERE [condition]  
ORDER BY [criteria]
```

**Q4 , Q5 , Q6**

# JOIN

- What to do when you want to combine data from multiple tables
  - JOIN them!
- What happens when you use the JOIN clause?
  - Have a new row for each combination of the input table
  - If two tables are joined and the left table has  $m$  rows and the right table has  $n$  rows, then the joined table will have  $m*n$  rows

# JOIN (Example)

- Let's say we have a `meetings` table that displays what `Day` each `Division` has a meeting
- Our goal is to determine what `Day` each employee has a meeting
- First step
  - Combine table `records` and `meetings`
- Second step
  - Only include the rows where the `records.division = meetings.division`

# JOIN (Example) (Combining)

```
SELECT name, day FROM records, meetings;
```

name	day
Alyssa P Hacker	Monday
Alyssa P Hacker	Monday
Alyssa P Hacker	Wednesday
Alyssa P Hacker	Wednesday
...	...
Robert Cratchet	Wednesday

# JOIN (Example) (Filtering)

## Ambiguous Joins

```
SELECT a.name, b.day FROM records AS a, meetings AS b
WHERE a.division = b.division;
```

name	day
Alyssa P Hacker	Wednesday
Ben Bitdiddle	Wednesday
...	...
Robert Cratchet	Monday

STEP BY STEP

**Q6 , Q7 , Q8**

# Aggregation

- What if we want to obtain data from our table (i.e. `max`, `min`)
  - `MAX`, `MIN`, `COUNT`, and `SUM`

```
SELECT name, MAX(salary) FROM records;
```

name	MAX(salary)
Oliver Warbucks	150000

```
sqlite> SELECT COUNT(*) from RECORDS;  
9
```



# GROUP BY

- group rows in a column to be aggregated with the GROUP BY clause

```
sqlite> SELECT division, MIN(salary) FROM records  
GROUP BY division;
```

division	MIN(salary)
Accounting	18000
Administration	150000
Computer	25000

STEP BY STEP

# HAVING

- Filter GROUPS with the HAVING clause

```
sqlite> SELECT title FROM records GROUP BY title  
HAVING COUNT(*) > 1;
```

**title**

---

Programmer

STEP BY STEP

**Q9 , Q10 , Q11**

# Thank you!!!

**Attendance Form -> <https://tinyurl.com/adit-disc11>**

**Anon Feedback -> <https://tinyurl.com/adit-anon>**