Data 101/CS 187: Data Engineering Fall 2026 Discussion 01 – September 4

Review: SQL

Instructor: Aditya Parameswaran *Covers Lecture(s): Review*

I Execution Order

1. Warmup: Add a number indicating the order of execution of the below query.

SELECT	S	5
FROM	$R_1,\ R_2,\ \ldots$	1
WHERE	C_1	2
GROUP BY	$A_1,\ A_2,\ \ldots$	3
HAVING	C_2	4
ORDER BY	S	6
LIMIT	c	7

II SQL Review

Suppose a full database schema is as follows:

- `books (book_id, book_name, year, genre, author_id, times_trending)`
- `authors (author_id, author_name, publishing_company, debut_year)`
- `awards (award_id, book_id, award_name)`

Write a SQL query to accomplish each task below:

2. Select only the book names and years from the `books` table.

```
SELECT book_name, year FROM books;
```

3. Find the names of all authors whose debut year is in the 21st century (defined as after 2000).

```
SELECT author_name FROM authors WHERE debut_year >= 2001;
```

4. Find the names of all authors who released a *sci-fi* genre book in 1974 that won an award.

```
SELECT au.author_name
FROM books AS b INNER JOIN awards AS aw ON b.book_id = aw.book_id
INNER JOIN authors AS au ON b.author_id = au.author_id
WHERE b.genre = 'sci-fi' AND b.year = 1974;
```

We see that the `books` relation does not have an author name column, so we need to join the `books` relation with the `authors` relation, as well as the `awards` relation, since we also need information on award-winning books. We're interested in sci-fi books released in 1974, so we filter our records with the WHERE clause. Finally, we're only interested in author names, so we SELECT the author_name column at the end.

Note: Will the returned table have unique author names? Why or why not? Why might this query return repeated author names? Authors who write multiple books that have won an award will return the author's name for each book. If two authors who have won an award share a name, those two authors' names will be returned by the query. We could disambiguate the authors with GROUP BY author_id and author_name. Additionally, we could use the DISTINCT keyword in the SELECT statement to only return unique authors.

III SQL Aggregation Review

In some cases, we may need to accumulate results by combining multiple rows via *aggregation*. We use the same schema as in Section 2.

5. Write a query that calculates the total number of books written by author Agatha Christie.

```
SELECT COUNT(*)
FROM books AS b INNER JOIN authors AS a ON b.author_id = a.author_id
WHERE a.author_name = 'Agatha Christie';
```

If we want to count the number of rows, we can use `COUNT(*)`. So the rest of the problem is just generating one row for each book written by the author, which we can perform using a join and a filter.

6. The *group-by* construct lets us compute aggregates for different chunks of a relation. Find the total number of books released per genre. The output should contain two columns: genre and the total count in that genre. Don't include genres with a count less than 2. (Hint: `HAVING`)

```
SELECT genre, COUNT(*)
FROM books
GROUP BY genre
HAVING COUNT(*) >= 2;
```

The trick here is that when we group by a specific column, we can return that column in the result. To filter the genres by count, we use the `HAVING` keyword, which applies a filter *after* grouping. Remember that WHERE filters individual rows before aggregation, and HAVING filters groups post aggregation.

IV Optional Questions

7. Find the names of the 5 books that trended the least, ordered from least to most. Break ties by book name in alphabetical order.

```
SELECT book_name FROM books
ORDER BY times_trending ASC, book_name ASC LIMIT 5;
```

We want information on book name and times trending, and both can be found with just the `books` table. The next clause to use is the SELECT clause, so we SELECT the book_name column. The problem asks us to order the records from the least trendy to the most trendy, so we use the ORDER BY clause on the times_trending column. We break ties by book name so we also order by the book_name column after the times_trending column. Finally, we can limit our returned query to the first 5 records with the LIMIT clause.

8. **Challenge:** Write a query that computes the total times each author has had a trending book. The output should have two columns: author name and the total times. (Hint: you will need to use a join for this as well.)

```
SELECT a.author_name, SUM (b.times_trending)
FROM books AS b INNER JOIN authors AS a ON b.author_id = a.author_id
GROUP BY a.author_id, a.author_name;
```

Before we perform a group-by, we can still perform a join. The trick here is that we get all pairs of authors and their books, group by author ID, so we have a bunch of grouped rows corresponding to all the books written by each author, and then can perform an aggregation over the contents of the grouped rows.