Convey your intent

APPLYING SQL TO REAL-WORLD PROBLEMS

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

Why is this important?

> "...if my code does what I designed it to do, who cares how its written..."

...six months from now

Always use AS

Original

```
SELECT title film_title
FROM film;
```

Improved

```
SELECT title AS film_title
FROM film;
```

What kind of JOIN?

Original

```
SELECT category, length
FROM film AS f
JOIN category AS c
ON f.film_id = c.film_id;
```

Improved

```
SELECT category, length
FROM film AS f
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Good use of aliases

Original

```
SELECT category, length
FROM film AS x1
INNER JOIN category AS x2
ON x1.film_id = x2.film_id;
```

Improved

```
SELECT category, length
FROM film AS f
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Good use of aliases

Original

```
SELECT category, length
FROM film AS x1
INNER JOIN category AS x2
ON x1.film_id = x2.film_id;
```

Improved

```
SELECT category, length
FROM film AS f
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

```
SELECT category, length
FROM film AS fil
INNER JOIN category AS cat
ON fil.film_id = cat.film_id;
```

Use comments

```
/* Use the system table, information_schema.columns to
generate a comma-separated list of columns for each table */
SELECT table_name, STRING_AGG(column_name, ', ') AS columns
FROM information_schema.columns
-- All our data is stored in the public schema.
WHERE table_schema = 'public'
GROUP BY table_name;
```

```
/* Multi-line comment */
```

```
-- Single-line comment
```

What was your intent?

APPLYING SQL TO REAL-WORLD PROBLEMS

Write readable code

APPLYING SQL TO REAL-WORLD PROBLEMS

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

datacamp

Capitalize SQL commands

Original

```
select
  title as film_title,
  description as film_description
from rental as r
join inventory as i
  on r.inventory_id = i.inventory_id
join film as f
  on i.film_id = f.film_id
where f.length > 90
  and f.replacement_cost > 20;
```

Improved

```
SELECT
  title AS film_title,
  description AS film_description
FROM rental AS r
JOIN inventory AS i
  ON r.inventory_id = i.inventory_id
JOIN film AS f
  ON i.film_id = f.film_id
WHERE f.length > 90
  AND f.replacement_cost > 20;
```

Use new lines & indentation

Original

```
SELECT title, description
FROM rental AS r
JOIN inventory AS i
ON r.inventory_id = i.inventory_id
JOIN film AS f ON i.film_id = f.film_id
WHERE f.length > 90 AND f.replacement_cost >
```

Improved

```
SELECT title,
       description
FROM rental AS r
JOIN inventory AS i
  ON r.inventory_id = i.inventory_id
JOIN film AS f
  ON i.film_id = f.film_id
WHERE f.length > 90
  AND f.replacement_cost > 20;
```

Use snake_case

Original

```
SELECT
  title expensivelongtitle,
  description expensivelongdescription
FROM rental AS r
...
```

Improved

```
SELECT
  title expensive_long_title,
  description expensive_long_description
FROM rental AS r
...
```

Use IN instead of many OR statements

Original

```
SELECT
    address_id,
    district
FROM address
WHERE district = 'Texas'
  OR district = 'Bihar'
  OR district = 'Chiba'
  OR district = 'Chiayi'
  OR district = 'Gois';
```

Improved

```
SELECT
    address_id,
    district
FROM address
WHERE district IN ('Texas', 'Bihar', 'Chiba',
                   'Chiayi', 'Gois');
```

Use BETWEEN when possible

Original

```
SELECT
  title,
  description
FROM film
WHERE replacement_cost > 15
  AND replacement_cost < 25;
```

Improved

```
SELECT
  title,
  description
FROM film
WHERE replacement_cost BETWEEN 15 AND 25;
```

Let's practice!

APPLYING SQL TO REAL-WORLD PROBLEMS

Avoid common mistakes

APPLYING SQL TO REAL-WORLD PROBLEMS

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

datacamp

Don't misuse comments

```
/* When selecting category and length
from films we need to use f, after this
I had a sandwich, it was a
good sandwich ...*/
SELECT category, length
-- FROM actor as a
FROM film AS f
/* Inner join the table category
with the film table */
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Do not

- Write an essay in your comments.

Don't misuse comments

```
SELECT category, length
-- FROM actor as a
FROM film AS f
/* Inner join the table category
with the film table */
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Do not

- **Write an essay in your comments.**
- Leave old comments in finished code.

Don't misuse comments

```
SELECT category, length

FROM film AS f
/* Inner join the table category
with the film table */
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Do not

- **Write an essay in your comments.**
- **Leave old comments in finished code.**
- Make comments redundant with code.

Don't misuse comments

```
/* When selecting category and length
from films we need to use f, after this
I had a sandwich, it was a
good sandwich ...*/
SELECT category, length
-- FROM actor as a
FROM film AS f
/* Inner join the table category
with the film table */
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

```
SELECT category, length
FROM film AS f
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

Don't SELECT everything

```
SELECT *
FROM film AS f
INNER JOIN category AS c
ON f.film_id = c.film_id;
```

```
release_year language_id rental_duration rental_rate length  .......
2009              1               4             6.99        173  .......
2006              1               7             6.99        185  .......
2004              1               5             4.99        153  .......
2007              1               7             2.99        69   .......
```

Don't use SQL for programming

```
DO $$
BEGIN
   FOR counter IN 1..5 LOOP
     IF (counter = 2) THEN
      RAISE NOTICE 'BINGO!';
     ELSE
      RAISE NOTICE 'Not BINGO :-(';
     END IF;
   END LOOP;
END; $$
```

```
NOTICE:  1 Not BINGO :-(
NOTICE:  1 BINGO!
NOTICE:  3 Not BINGO :-(
NOTICE:  4 Not BINGO :-(
NOTICE:  5 Not BINGO :-(
```

Let's practice!

APPLYING SQL TO REAL-WORLD PROBLEMS

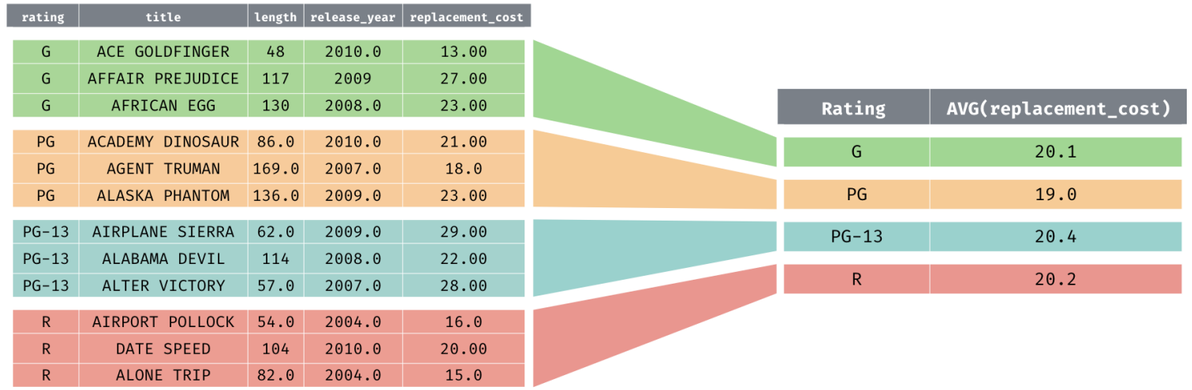
Recap

APPLYING SQL TO REAL-WORLD PROBLEMS

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

Chapter 1 - Use Real-World SQL

- Essential SQL
- Transforming Your Results
- Working With Aggregate Functions

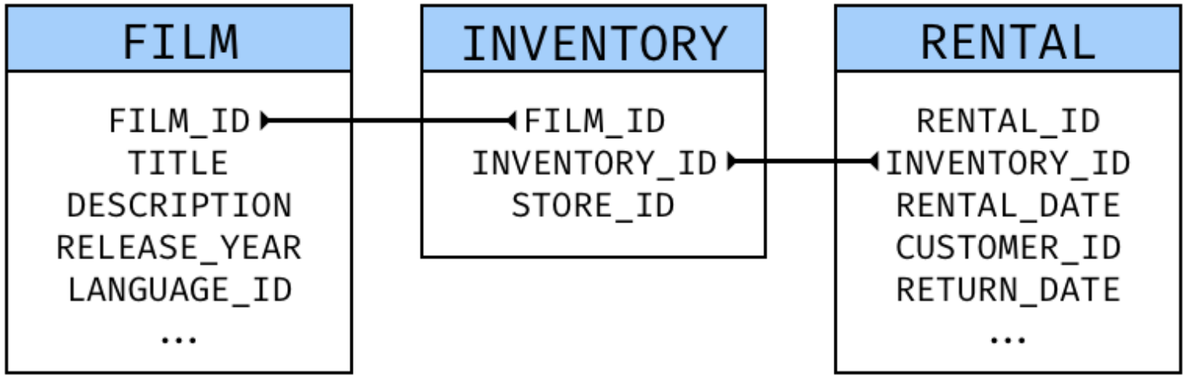


rating	title	length	release_year	replacement_cost
G	ACE GOLDFINGER	48	2010.0	13.00
G	AFFAIR PREJUDICE	117	2009	27.00
G	AFRICAN EGG	130	2008.0	23.00
PG	ACADEMY DINOSAUR	86.0	2010.0	21.00
PG	AGENT TRUMAN	169.0	2007.0	18.0
PG	ALASKA PHANTOM	136.0	2009.0	23.00
PG-13	AIRPLANE SIERRA	62.0	2009.0	29.00
PG-13	ALABAMA DEVIL	114	2008.0	22.00
PG-13	ALTER VICTORY	57.0	2007.0	28.00
R	AIRPORT POLLOCK	54.0	2004.0	16.0
R	DATE SPEED	104	2010.0	20.00
R	ALONE TRIP	82.0	2004.0	15.0

Rating	AVG(replacement_cost)
G	20.1
PG	19.0
PG-13	20.4
R	20.2

Chapter 2 - Find Your Data

- Find the Right Table
- Join the Correct Table
- Working With Aggregate Functions

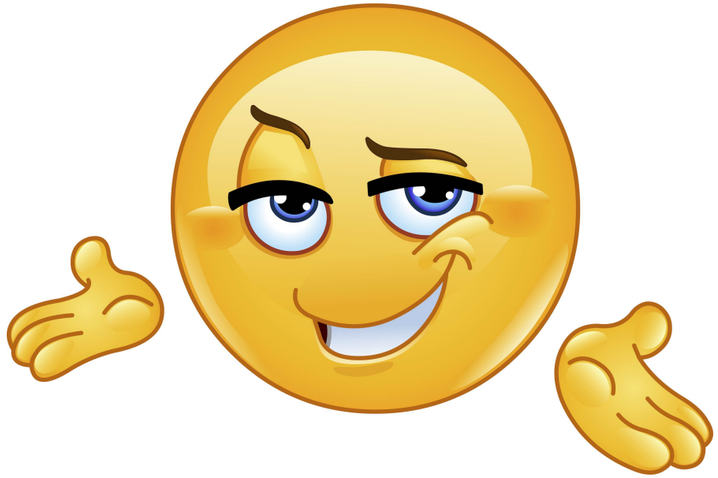


Chapter 3 - Manage Your Data

- Store Your Data
- Update Your Data
- Delete Your Data

Chapter 4 - Best Practices for Writing SQL

- Convey Your Intent
- Write Readable Code
- Avoid Common Mistakes



Congratulations!

APPLYING SQL TO REAL-WORLD PROBLEMS