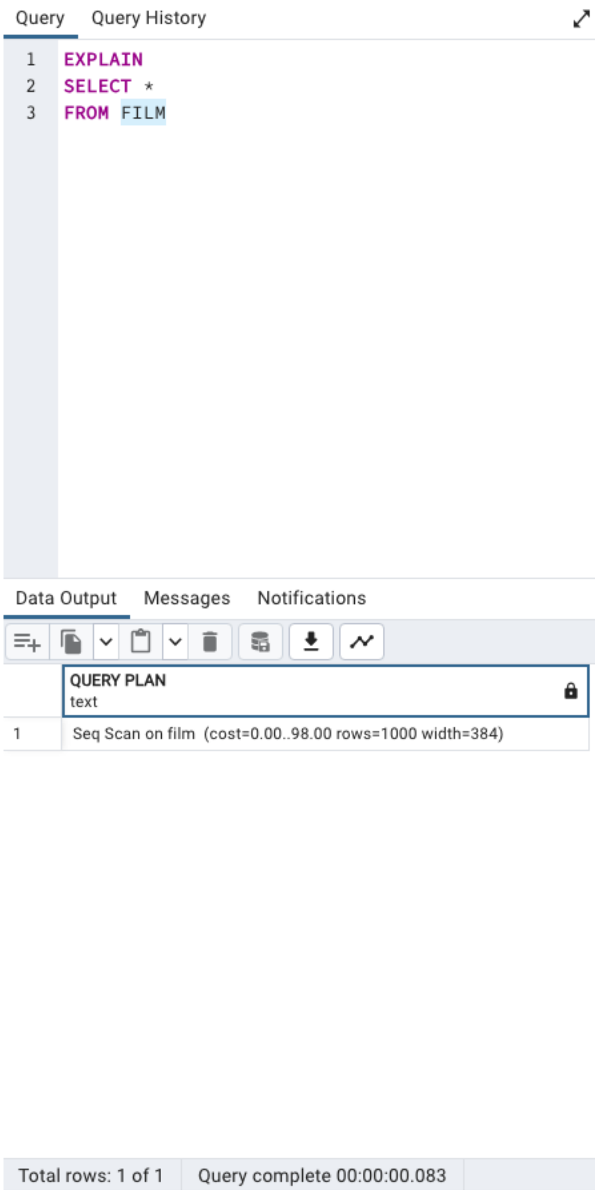
Jeremy Obach
CareerFoundry DA Immersion
Task 3.4

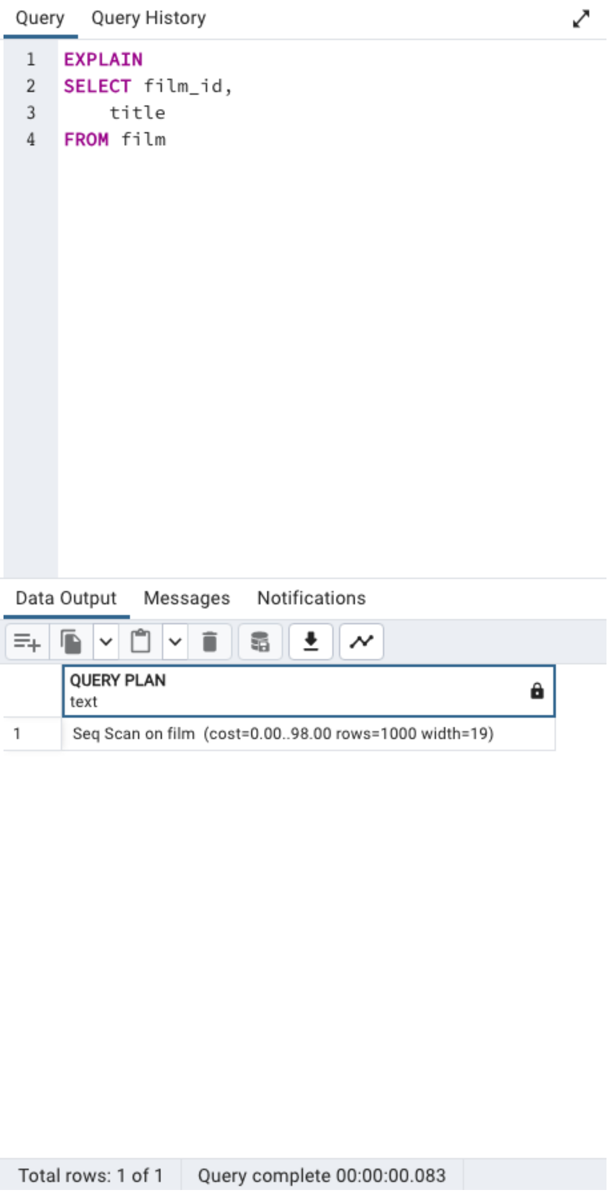
1.



```
EXPLAIN
SELECT *
FROM FILM
```

```
QUERY PLAN
text
1   Seq Scan on film  (cost=0.00..98.00 rows=1000 width=384)
```

Total rows: 1 of 1   Query complete 00:00:00.083

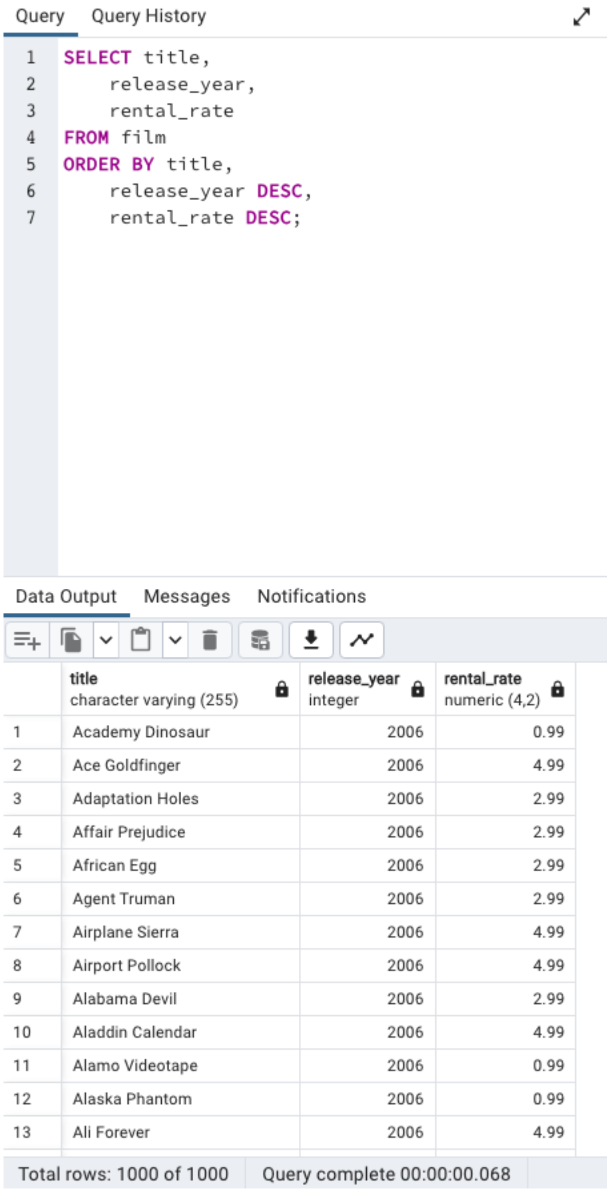


```
EXPLAIN
SELECT film_id,
    title
FROM film
```

```
QUERY PLAN
text
1   Seq Scan on film  (cost=0.00..98.00 rows=1000 width=19)
```

Total rows: 1 of 1   Query complete 00:00:00.083

Based on lesser width from the latter query specifying columns, the latter query will be faster. Running each, the latter took 76ms and the former took 75 msec tho. Running the latter repeatedly, the msec count varies slightly from each run, as low as 50msec and as high as 93msec. To optimize query further, maybe use WHERE conditions or LIMIT X amount depending on what data you need.



```
SELECT title,
    release_year,
    rental_rate
FROM film
ORDER BY title,
    release_year DESC,
    rental_rate DESC;
```

| | title<br>character varying (255) | release_year<br>integer | rental_rate<br>numeric (4,2) |
|---|---|---|---|
| 1 | Academy Dinosaur | 2006 | 0.99 |
| 2 | Ace Goldfinger | 2006 | 4.99 |
| 3 | Adaptation Holes | 2006 | 2.99 |
| 4 | Affair Prejudice | 2006 | 2.99 |
| 5 | African Egg | 2006 | 2.99 |
| 6 | Agent Truman | 2006 | 2.99 |
| 7 | Airplane Sierra | 2006 | 4.99 |
| 8 | Airport Pollock | 2006 | 4.99 |
| 9 | Alabama Devil | 2006 | 2.99 |
| 10 | Aladdin Calendar | 2006 | 4.99 |
| 11 | Alamo Videotape | 2006 | 0.99 |
| 12 | Alaska Phantom | 2006 | 0.99 |
| 13 | Ali Forever | 2006 | 4.99 |

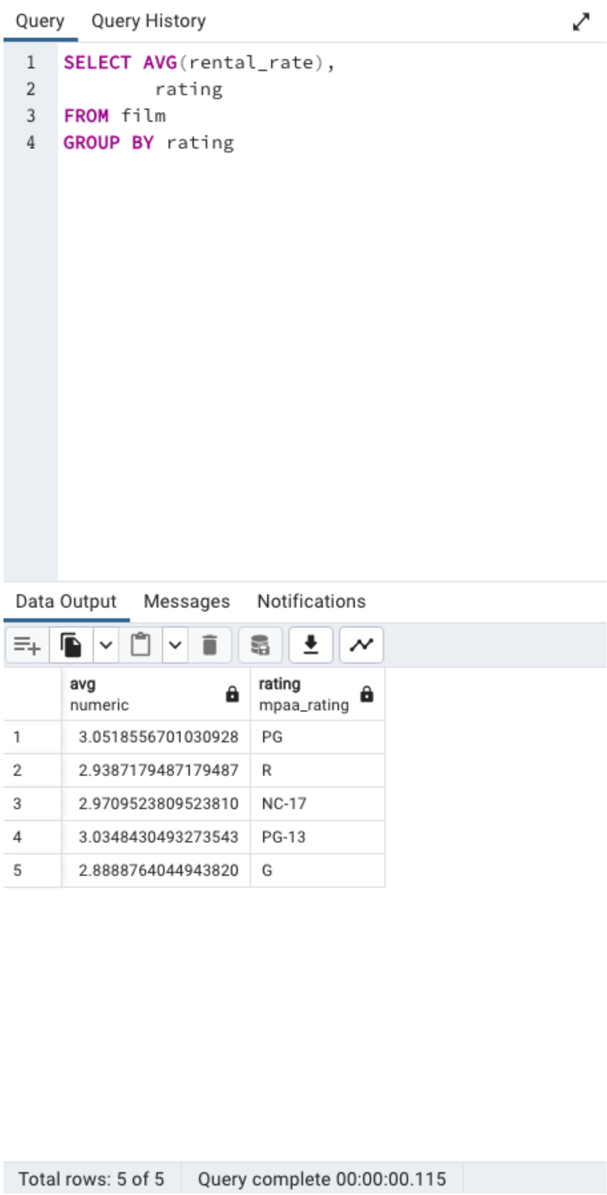
Total rows: 1000 of 1000    Query complete 00:00:00.068

2.

Couldn't get it to run with DESC for 2nd and 3rd conditions at first, realized I was using GROUP BY instead of ORDER BY. Not sure that the other two conditions are doing anything though, unless there was a duplicate title the release year and rental rate would be superseded in order by the titles in alphabetical order.

3. Grouping Data

Average rental rate for each rating category:

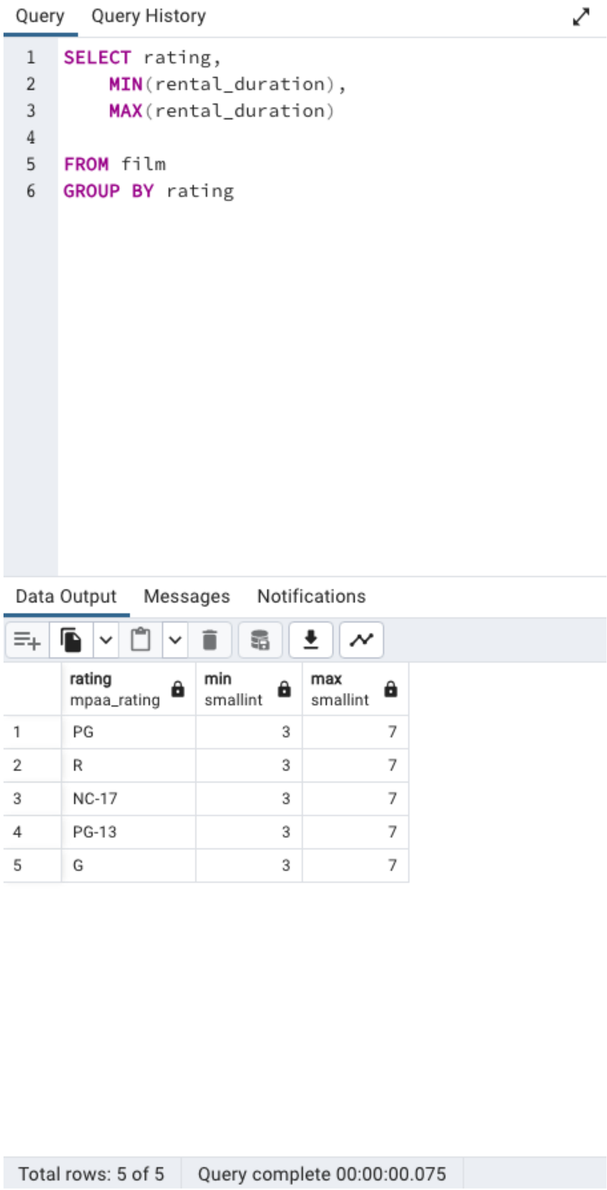


```sql
SELECT AVG(rental_rate),
        rating
FROM film
GROUP BY rating
```

| | avg<br>numeric | rating<br>mpaa_rating |
|---|---|---|
| 1 | 3.0518556701030928 | PG |
| 2 | 2.9387179487179487 | R |
| 3 | 2.9709523809523810 | NC-17 |
| 4 | 3.0348430493273543 | PG-13 |
| 5 | 2.8888764044943820 | G |

Total rows: 5 of 5 Query complete 00:00:00.115

Min and max rental durations for each rating category:



```sql
SELECT rating,
    MIN(rental_duration),
    MAX(rental_duration)

FROM film
GROUP BY rating
```

| | rating<br>mpaa_rating | min<br>smallint | max<br>smallint |
|---|---|---|---|
| 1 | PG | 3 | 7 |
| 2 | R | 3 | 7 |
| 3 | NC-17 | 3 | 7 |
| 4 | PG-13 | 3 | 7 |
| 5 | G | 3 | 7 |

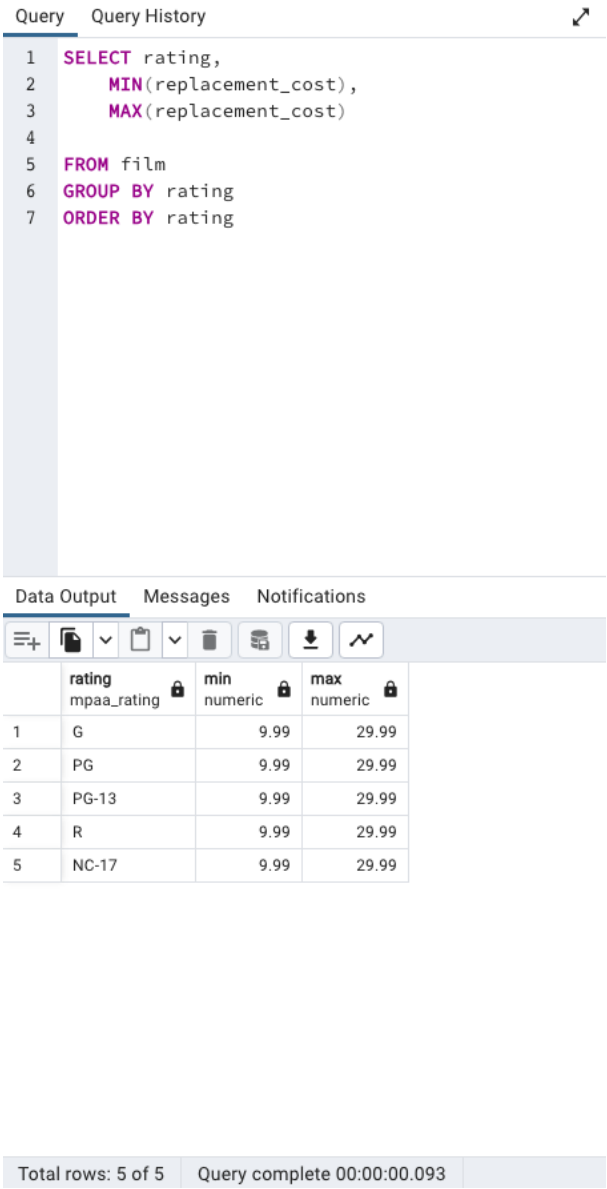
Total rows: 5 of 5 Query complete 00:00:00.075

4. Database Migration

a. The procedure to move data from this new source to the data warehouse can be broken into three main steps: Extract, Transform, and Load (ETL). Extraction involves collecting the data from the source systems, in this case the external data collection tool and the Rockbuster Android app. Next is transformation, where the extracted data is converted into another format. Finally, the transformed data is loaded into the data warehouse. This is generally the responsibility of a data engineer, but it's important that a data analyst be at least familiar with the steps of the process. Sidebar: a girl that works as a DA in NYC that I went to college with is pretty involved in ETL processes – I picked her brain on LinkedIn.

b. Should you analyze the data prior to being loaded into the data warehouse, you could run into issues with misidentifying scope or scale, as you're only working with the limited data that exists at the source at the time. Additionally, you may not be able to interact with the data in the same depth (or even at all) at the source level, versus the data warehouse level where you’re likely to be proficient in this scenario.

BONUS:



```
SELECT rating,
    MIN(replacement_cost),
    MAX(replacement_cost)

FROM film
GROUP BY rating
ORDER BY rating
```

| | rating<br>mpaa_rating | min<br>numeric | max<br>numeric |
|---|---|---|---|
| 1 | G | 9.99 | 29.99 |
| 2 | PG | 9.99 | 29.99 |
| 3 | PG-13 | 9.99 | 29.99 |
| 4 | R | 9.99 | 29.99 |
| 5 | NC-17 | 9.99 | 29.99 |

Total rows: 5 of 5    Query complete 00:00:00.093

Didn’t actually have to break out the  custom sorting technique from the links for this one; Ordering by ascending already put it in G, PG, PG-13, R and NC-17 sequence.