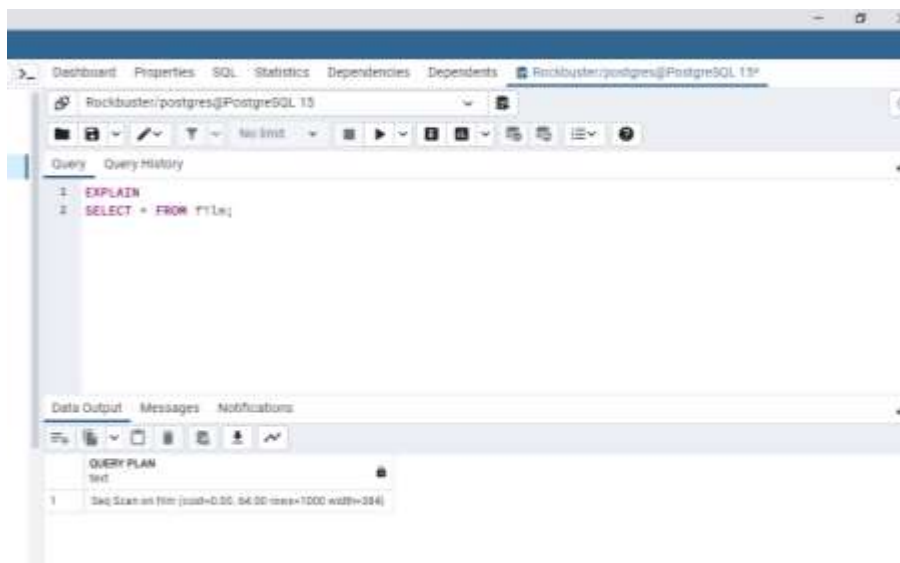


Refining Your Query: You need to get some data from the “film” table and decide to use the query `SELECT * FROM film`.

- You realize that only the “film_id” and “title” columns are needed. Write a new query that selects only those 2 columns.

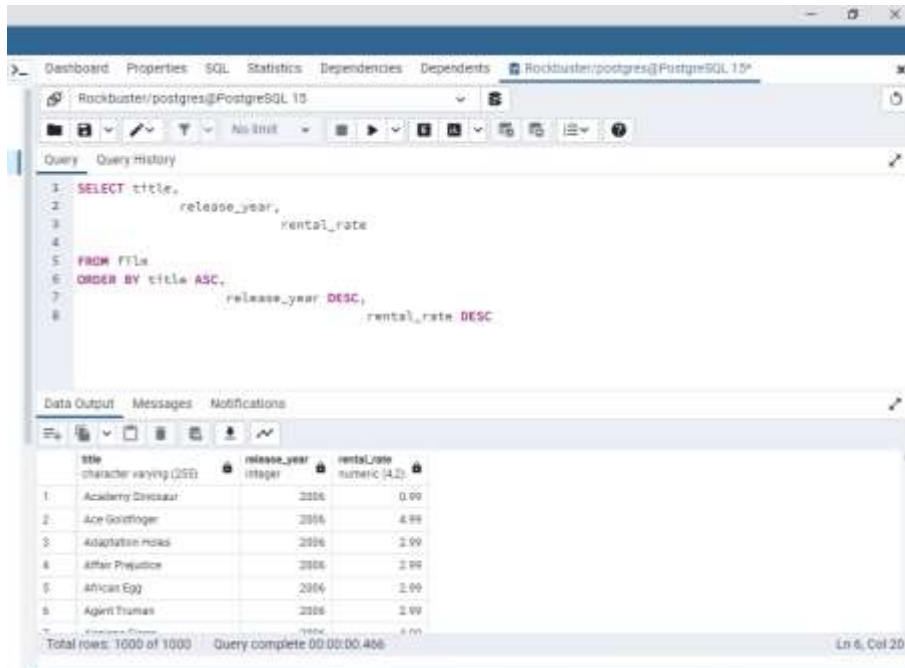
Compare the cost of the original query and the revised query, and write a few sentences explaining the comparison. Can you suggest any ways to optimize this query?



- The cost first query is more expensive than the second one. Because the first one takes longer time, checking all the 64 rows and having a width value 384. The second one only targets the intended columns, hence cheaper and faster in processing the result. Such kind of differences is more visible especially when we have a big data, overall, no need to go into all the records in a table – specific queries are efficient and cheaper.

Ordering the Data:

- In the pgAdmin Query Tool, run a query that selects every film from the “film” table, with the movies sorted by title from A to Z, then by most recent release year, and then by highest to lowest rental rate.
- Extract the data output of your query into a CSV file for the film collection department to analyze in Excel. To do this, click the button “Save results to file”:



The screenshot shows the pgAdmin Query Tool interface. The query editor contains the following SQL query:

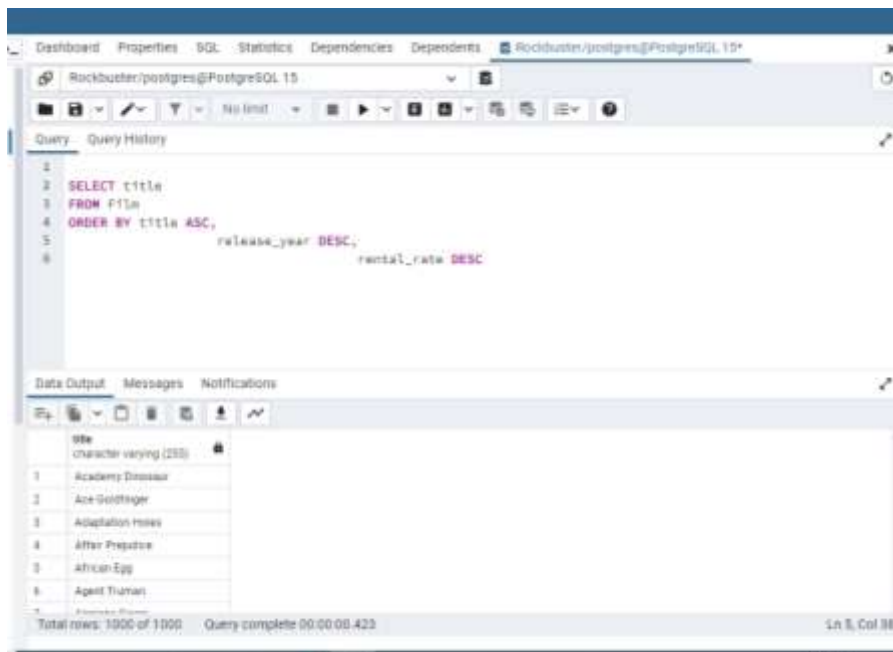
```
1. SELECT title,
2.        release_year,
3.        rental_rate
4.
5. FROM film
6. ORDER BY title ASC,
7.        release_year DESC,
8.        rental_rate DESC
```

The Data Output tab shows the results of the query. The table has three columns: title (character varying (255)), release_year (integer), and rental_rate (numeric (4,2)). The results are sorted by title, then by release_year, and then by rental_rate.

	title	release_year	rental_rate
1.	Academy Dinosaur	2006	0.99
2.	Ace Goldfinger	2006	4.99
3.	Adaptation notes	2006	2.99
4.	After Prejudice	2006	2.99
5.	African Egg	2006	2.99
6.	Agent Thomas	2006	2.99

Total rows: 1000 of 1000 Query complete 00:00:00.456 Ln 8, Col 20

- Without release year and rental rate



The screenshot shows the pgAdmin Query Tool interface. The query editor contains the following SQL query:

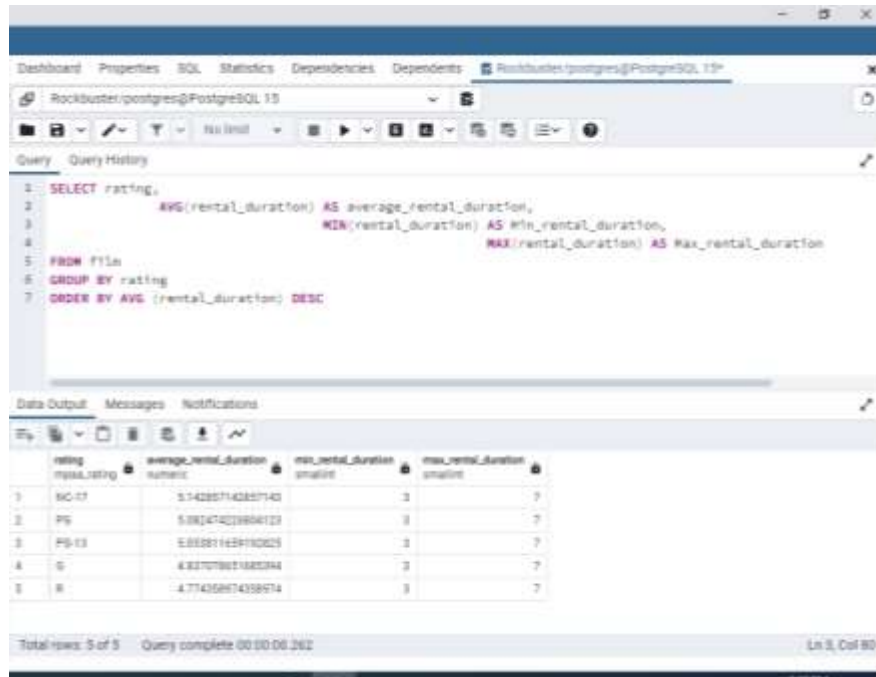
```
1
2 SELECT title
3 FROM film
4 ORDER BY title ASC,
5        release_year DESC,
6        rental_rate DESC
```

The Data Output tab shows the results of the query. The table has one column: title (character varying (255)). The results are sorted by title, then by release_year, and then by rental_rate.

	title
1.	Academy Dinosaur
2.	Ace Goldfinger
3.	Adaptation notes
4.	After Prejudice
5.	African Egg
6.	Agent Thomas

Total rows: 1000 of 1000 Query complete 00:00:00.422 Ln 6, Col 38

1. **Grouping Data:** The strategy department has asked you the questions below. Write a SQL query to retrieve the correct answers, then extract your results as a CSV file.
 - What is the average rental rate for each rating category?
 - What are the minimum and maximum rental durations for each rating category?



The screenshot shows a PostgreSQL query editor with a query window and a data output window. The query window contains the following SQL query:

```
1. SELECT rating,
2.        AVG(rental_duration) AS average_rental_duration,
3.        MIN(rental_duration) AS Min_rental_duration,
4.        MAX(rental_duration) AS Max_rental_duration
5. FROM Film
6. GROUP BY rating
7. ORDER BY AVG (rental_duration) DESC
```

The data output window shows the results of the query, which are 5 rows of data. The columns are: rating, average_rental_duration, min_rental_duration, and max_rental_duration. The data is as follows:

rating	average_rental_duration	min_rental_duration	max_rental_duration
PG-13	5.142857142857143	3	7
PG	5.081474239861229	3	7
PG-13	5.058811429130625	3	7
G	4.827079621885394	3	7
R	4.774358674358674	3	7

1. **Database Migration:** Your team has decided to use an external tool to collect data on user behavior in the new Rockbuster Android app. Data collected from this new source will need to be loaded into the data warehouse before you can analyze it.
 - Can you outline the procedure for migrating the data and who will be responsible for it?
 - What problems do you foresee if you start analyzing the data before it's been loaded into the data warehouse?

The data must be collected from Android APP data base, transformed (to a suitable format which will be needed in a data warehouse) and finally loaded to data warehouse. Process is owned by data engineers with additional support of data analysts for finding errors in migration process (pipelines).

There are risks of different format in Android APP Data base in compare to the data warehouse, which itself creates more workload like adding different variables in the script etc. In order to receive values that needed to be analyzed. This itself creates extra costs due to longer code and more functions in it which could be avoided if data would be already loaded into data warehouse