

Task 3.4: Database Querying in SQL

1a) New query: **SELECT** film_id, title **FROM** film;

1b) Original:

The screenshot shows a database query interface with a 'Query' tab. The query history contains two entries:

- SELECT** film_id, title **FROM** film;
- EXPLAIN SELECT** * **FROM** film;

The 'Data output' tab is active, showing the 'QUERY PLAN' for the second query. The plan indicates a 'Seq Scan on film' with a cost of 0.00..64.00, 1000 rows, and a width of 384. The status bar at the bottom shows 'Total rows: 1 of 1', 'Query complete 00:00:00.049', and 'Ln 2, Col 28'.

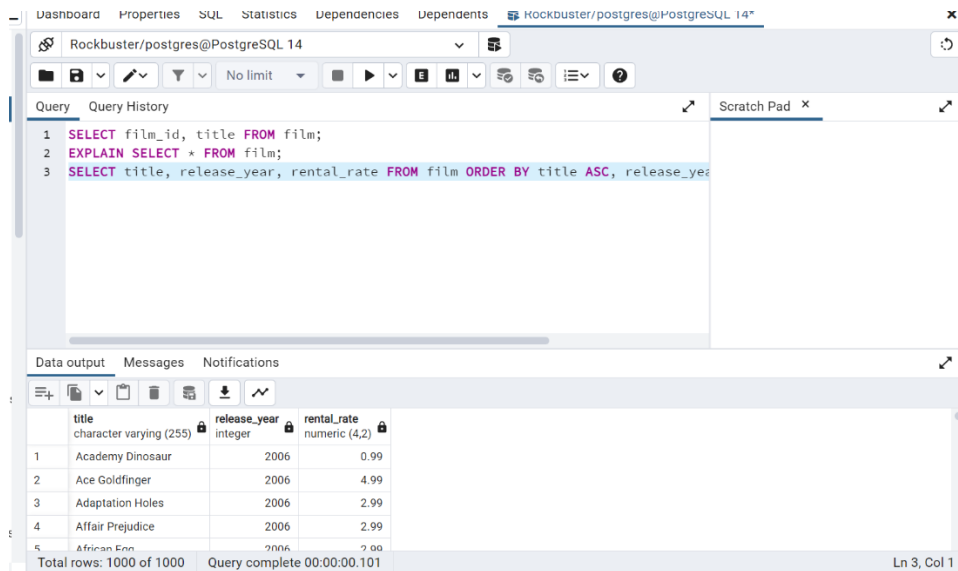
Revised:

The screenshot shows the same database query interface, but with a revised query. The query history now contains three entries:

- SELECT** film_id, title **FROM** film;
- EXPLAIN SELECT** * **FROM** film;
- EXPLAIN SELECT** film_id, title **FROM** film;

The 'Data output' tab is active, showing the 'QUERY PLAN' for the third query. The plan indicates a 'Seq Scan on film' with a cost of 0.00..64.00, 1000 rows, and a width of 19. The status bar at the bottom shows 'Total rows: 1 of 1', 'Query complete 00:00:00.082', and 'Ln 3, Col 41'. A tooltip 'Execute/Refresh F5' is visible over the query history.

The cost of both the original and revised query is the same (cost=0.00...64.00). The width value of the original is 384 and the width value of the revised is 19. The revised query targets the specific columns, processing the result faster. Therefore, the revised query would be more cost efficient than the original query.



The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 SELECT film_id, title FROM film;
2 EXPLAIN SELECT * FROM film;
3 SELECT title, release_year, rental_rate FROM film ORDER BY title ASC, release_year
```

The results window displays the output of the third query. The data is as follows:

	title	release_year	rental_rate
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 Ego	2006	2.00

At the bottom of the results window, it states: "Total rows: 1000 of 1000" and "Query complete 00:00:00.101".

2b) 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”: 

3a) Average rental rate SQL query: **SELECT** rating, **AVG**(rental_rate) **AS** average_rental_rate **FROM** film **GROUP BY** rating;

3b) Min and Max SQL query: **SELECT** rating, **MAX**(rental_rate) **AS** maximum_rental_rate, **MIN**(rental_rate) **AS** minimum_rental_rate **FROM** film **GROUP BY** rating;

4a) Outline:

1. Extract: Collect all the necessary data from the Android app (data engineers or the app team)
2. Transform: Convert the extracted data to standardized data types to determine table and column structures and to find the keys to build relationships with tables. (data engineers/data analysts)
3. Load: Load transformed data into new database. (data engineers)

4b) Problems: If you start analyzing data before it’s been loaded into the data warehouse, you are looking at incomplete data. Incomplete data hinders the ability to see relationships because it has not been filtered out. Therefore, the information is not yet meaningful. Also, analyzing data before it’s been filtered out would be inefficient and difficult.