Assignment 3 -- Accessing PostgreSQL

As given, I have restored the dvd rental database and installed python and psycopg2 library in a virtual environment dvd. For this assignment, I decided to create film_review table and insert 5 rows in dvdrental.py file. Later update the table by adding new column and deleting the last row in dvdupdate.py file.

1. Table creation and insertion:-

a. Code : This code will create a film_review table with the following columns:

review_id (Serial, Primary Key)
film_id (Integer, Not Null)
customer_id (Integer, Not Null)
rating (Integer, Not Null)
review_text (Text)
review_date (Date)

It will then insert 5 rows of data into this table, fetch and print the inserted rows, fetch and print one specific row, and fetch and print three rows with a limit.

```
1
2
3       #Drop the table if it exists to start fresh
4       cur.execute('DROP TABLE IF EXISTS film_review')
5
6       # Create the film_review table
7       create_script = '''
8       CREATE TABLE IF NOT EXISTS film_review (
9           review_id SERIAL PRIMARY KEY,
0           film_id INT NOT NULL,
1           customer_id INT NOT NULL,
2           rating INT NOT NULL,
3           review_text TEXT,
4           review_date DATE
5       )
6       '''
7       cur.execute(create_script)
8
```

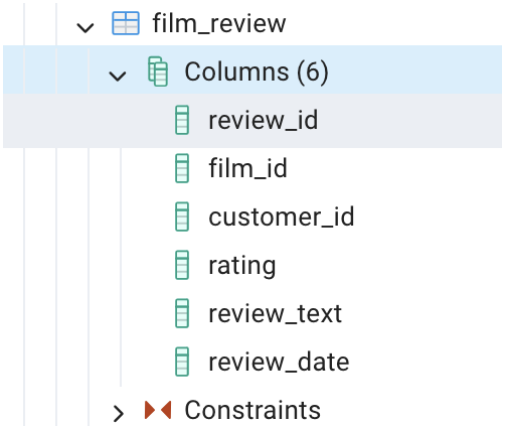
```python
    #your code goes in this
# Insert 5 rows into the film_review table
    insert_script = '''
    INSERT INTO film_review (film_id, customer_id, rating, review_text, review_date)
    VALUES (%s, %s, %s, %s, %s)
    '''
    insert_values = [
        (1, 1, 5, 'Excellent film, highly recommended!', '2024-04-20'),
        (2, 3, 4, 'Good movie, enjoyed it.', '2024-04-21'),
        (3, 2, 3, 'Average film, nothing special.', '2024-04-22'),
        (4, 4, 5, 'Amazing, must-watch!', '2024-04-23'),
        (5, 5, 2, 'Disappointing, not worth it.', '2024-04-24')
    ]

    #for multiple entries we need for loop
    for record in insert_values:
        cur.execute(insert_script, record)
```

b. Output in terminal:- by running

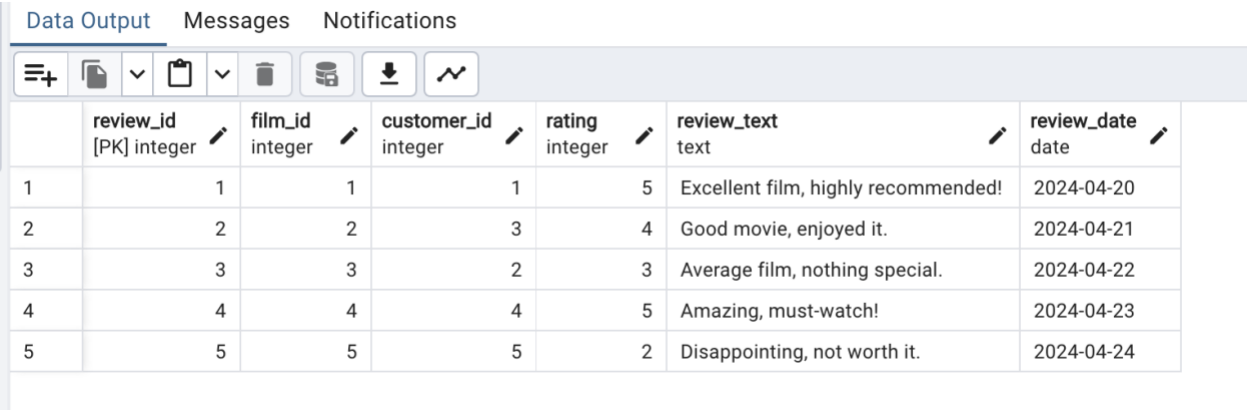
python3 dvdrental.py

```
(dvd) tejamothukuri@teja python3 dvdrental.py
All Rows:
[(1, 1, 1, 5, 'Excellent film, highly recommended!', datetime.date(2024, 4, 20)), (2, 2, 3, 4, 'Good movie, enjoyed it.',
 datetime.date(2024, 4, 21)), (3, 3, 2, 3, 'Average film, nothing special.', datetime.date(2024, 4, 22)), (4, 4, 4, 5, 'A
mazing, must-watch!', datetime.date(2024, 4, 23)), (5, 5, 5, 2, 'Disappointing, not worth it.', datetime.date(2024, 4, 24
))]


One Row (Review ID = 2):
(2, 2, 3, 4, 'Good movie, enjoyed it.', datetime.date(2024, 4, 21))


Three Rows (Limit 3):
[(1, 1, 1, 5, 'Excellent film, highly recommended!', datetime.date(2024, 4, 20)), (2, 2, 3, 4, 'Good movie, enjoyed it.',
 datetime.date(2024, 4, 21)), (3, 3, 2, 3, 'Average film, nothing special.', datetime.date(2024, 4, 22))]
(dvd) tejamothukuri@teja
```

c. Output in pgAdmin4:



Retrieving all rows using select command



Data Output | Messages | Notifications

	review_id [PK] integer	film_id integer	customer_id integer	rating integer	review_text text	review_date date
1	1	1	1	5	Excellent film, highly recommended!	2024-04-20
2	2	2	3	4	Good movie, enjoyed it.	2024-04-21
3	3	3	2	3	Average film, nothing special.	2024-04-22
4	4	4	4	5	Amazing, must-watch!	2024-04-23
5	5	5	5	2	Disappointing, not worth it.	2024-04-24

2. Update table and Drop a row

a. Code :-

- Added a new column helpful_votes to the film_review table with a default value of 0 and displayed the structure of the film_review table after adding the new column.
- Deletes a record from the film_review table where review_id is 5 and fetches and prints all rows from the film_review table after the deletion.

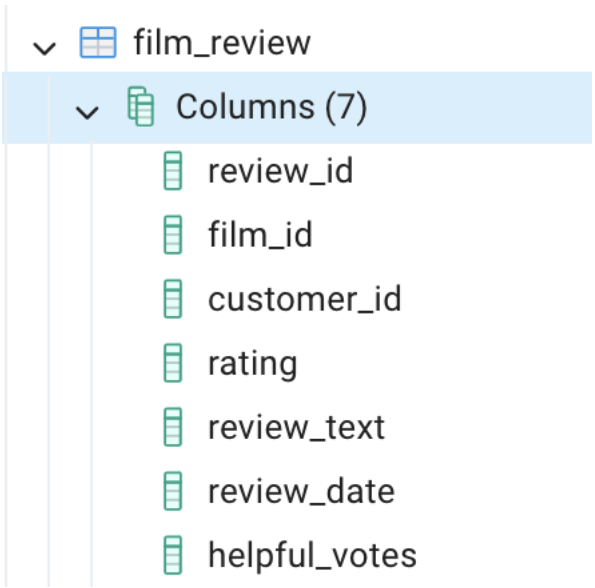
```
# Add a new column helpful_votes to film_review table
alter_script = '''
ALTER TABLE film_review
ADD COLUMN IF NOT EXISTS helpful_votes INT DEFAULT 0
'''
cur.execute(alter_script)

# Use information_schema.columns to describe the table
describe_query = '''
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'film_review'
'''
cur.execute(describe_query)
table_description = cur.fetchall()

# Print the description of the film_review table
print("Description of 'film_review' table:")
for column in table_description:
    print(column[0], "-", column[1])
```

```python
# Delete a record from the film_review table
delete_script = '''
DELETE FROM film_review
WHERE review_id = %s
'''
review_id_to_delete = 5  # Deleting the review with review_id = 5
cur.execute(delete_script, (review_id_to_delete,))

# Fetch and print all rows from the film_review table after deletion
cur.execute('SELECT * FROM film_review')
print("\nAll Rows After Deletion:")
print(cur.fetchall())
```

d. Output in terminal by running python3 dvdupdate.py

```
 datetime.date(2024, 4, 21)), (3, 3, 2, 3, 'Average film, nothing special.', datetime.date(2024, 4, 22))]
(dvd) tejamothukuri@teja python3 dvdupdate.py
Description of 'film_review' table:
review_id - integer
film_id - integer
customer_id - integer
rating - integer
review_text - text
review_date - date
helpful_votes - integer

All Rows After Deletion:
[(1, 1, 1, 5, 'Excellent film, highly recommended!', datetime.date(2024, 4, 20), 0), (2, 2, 3, 4, 'Good movie, enjoyed it
.', datetime.date(2024, 4, 21), 0), (3, 3, 2, 3, 'Average film, nothing special.', datetime.date(2024, 4, 22), 0), (4, 4,
 4, 5, 'Amazing, must-watch!', datetime.date(2024, 4, 23), 0)]
(dvd) tejamothukuri@teja
```

b. Output in pgAdmin4



Retrieving all rows using select command

Data Output | Messages | Notifications

	review_id [PK] integer	film_id integer	customer_id integer	rating integer	review_text text	review_date date	helpful_votes integer
1	1	1	1	5	Excellent film, highly recommended!	2024-04-20	0
2	2	2	3	4	Good movie, enjoyed it.	2024-04-21	0
3	3	3	2	3	Average film, nothing special.	2024-04-22	0
4	4	4	4	5	Amazing, must-watch!	2024-04-23	0