PostgreSQL server Advanced 4

HR=# INSERT INTO employee(emp\_id, first\_name, last\_name, gender, birthdate)

HR-# VALUES

HR-# (5,'Sachin','Tendulkar','M', DATE '1978-01-09'),

HR-# (6,'Kapil', 'Dev', 'M', DATE '1959-03-26'),

HR-# (7,'Joe','Root','M', DATE '1988-11-29'),

HR-# (8,'Moeen', 'Ali', 'M', DATE '1999-04-16');

INSERT 0 4

HR=# INSERT INTO employee(emp\_id, first\_name, last\_name, gender, birthdate)

HR-# VALUES

HR-# (5,'Sachin','Tendulkar','M', DATE '1978-01-09'),

HR-# (6,'Kapil', 'Dev', 'M', DATE '1959-03-26'),

HR-# (7,'Joe','Root','M', DATE '1988-11-29'),

HR-# (8,'Moeen', 'Ali', 'M', DATE '1999-04-16')

HR-# RETURNING \*;

ERROR: duplicate key value violates unique constraint "employee\_pkey"

DETAIL: Key (emp\_id)=(5) already exists.

HR=# select \* from employee;

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+------------+------------------------+--------

1 | Annie | Smith | F | 1988-01-09 | ani@email.com | 5000

2 | Susan | Klassen | F | 2002-03-26 | |

3 | May | Kaasman | M | 1994-07-09 | mkaasman2@freewebs.com |

4 | Charlton | Duran | M | 2010-10-02 | |

5 | Sachin | Tendulkar | M | 1978-01-09 | |

6 | Kapil | Dev | M | 1959-03-26 | |

7 | Joe | Root | M | 1988-11-29 | |

8 | Moeen | Ali | M | 1999-04-16 | |

(8 rows)

HR=# SELECT \*

HR-# FROM employee

HR-# WHERE first\_name = 'Charlton';

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+------------+-------+--------

4 | Charlton | Duran | M | 2010-10-02 | |

(1 row)

HR=# SELECT \* FROM employee

HR-# WHERE salary > 50000;

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+-----------+-------+--------

(0 rows)

HR=# SELECT \* FROM employee

HR-# WHERE emp\_id IN (1,6);

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+------------+---------------+--------

1 | Annie | Smith | F | 1988-01-09 | ani@email.com | 5000

6 | Kapil | Dev | M | 1959-03-26 | |

(2 rows)

HR=# SELECT \* FROM employee

HR-# WHERE gender = 'F' AND salary > 50000;

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+-----------+-------+--------

(0 rows)

HR=# SELECT \* FROM employee where WHERE gender = 'M' AND salary >= 50000;

ERROR: syntax error at or near "WHERE"

LINE 1: SELECT \* FROM employee where WHERE gender = 'M' AND salary ...

^

HR=# SELECT \* FROM employee where WHERE gender = 'F' AND salary >= 50000;

ERROR: syntax error at or near "WHERE"

LINE 1: SELECT \* FROM employee where WHERE gender = 'F' AND salary ...

^

HR=# SELECT \* FROM employee

HR-# WHERE gender = 'F' OR salary > 50000;

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+------------+---------------+--------

1 | Annie | Smith | F | 1988-01-09 | ani@email.com | 5000

2 | Susan | Klassen | F | 2002-03-26 | |

(2 rows)

HR=# SELECT \* FROM employee

HR-# WHERE gender = 'F' AND salary > 50000;

emp\_id | first\_name | last\_name | gender | birthdate | email | salary

--------+------------+-----------+--------+-----------+-------+--------

(0 rows)

HR=# SELECT gender FROM employee GROUP BY gender;

gender

--------

M

F

(2 rows)

HR=# SELECT gender, COUNT(emp\_id) FROM employee GROUP BY gender;

gender | count

--------+-------

M | 6

F | 2

(2 rows)

HR=# update employee set salary=76000;

UPDATE 8

HR=# select gender,sum(salary) from employee group by gender;

gender | sum

--------+--------

M | 456000

F | 152000

(2 rows)

HR=# SELECT dept\_id, gender, SUM(salary) FROM employee

HR-# GROUP BY dept\_id, gender;

ERROR: column "dept\_id" does not exist

LINE 1: SELECT dept\_id, gender, SUM(salary) FROM employee

^

HINT: Perhaps you meant to reference the column "employee.emp\_id".

HR=# alter table employee rename emp\_id to dept\_id;

ALTER TABLE

HR=# \d employee;

Table "public.employee"

Column | Type | Collation | Nullable | Default

------------+------------------------+-----------+----------+---------

dept\_id | integer | | not null |

first\_name | character varying(50) | | not null |

last\_name | character varying(50) | | not null |

gender | character(1) | | |

birthdate | date | | |

email | character varying(100) | | |

salary | integer | | |

Indexes:

"employee\_pkey" PRIMARY KEY, btree (dept\_id)

"employee\_email\_key" UNIQUE CONSTRAINT, btree (email)

HR=# SELECT dept\_id, SUM(salary)

HR-# FROM employee

HR-# GROUP BY dept\_id;

dept\_id | sum

---------+-------

3 | 76000

5 | 76000

4 | 76000

6 | 76000

2 | 76000

7 | 76000

1 | 76000

8 | 76000

(8 rows)

HR=# SELECT gender, SUM(salary)

HR-# FROM employee

HR-# GROUP BY gender;

gender | sum

--------+--------

M | 456000

F | 152000

(2 rows)

HR=# SELECT dept\_id, gender, SUM(salary)

HR-# FROM employee

HR-# GROUP BY dept\_id, gender;

dept\_id | gender | sum

---------+--------+-------

3 | M | 76000

5 | M | 76000

4 | M | 76000

6 | M | 76000

2 | F | 76000

7 | M | 76000

1 | F | 76000

8 | M | 76000

(8 rows)

HR=# SELECT dept\_id, gender, SUM(salary) FROM employee GROUP BY dept\_id, gender

HR-# UNION ALL

HR-# SELECT dept\_id, NULL, SUM(salary) FROM employee GROUP BY dept\_id

HR-# UNION ALL

HR-# SELECT NULL, gender, SUM(salary) FROM employee GROUP BY gender

HR-# UNION ALL

HR-# SELECT NULL, NULL, SUM(salary) FROM employee;

dept\_id | gender | sum

---------+--------+--------

3 | M | 76000

5 | M | 76000

4 | M | 76000

6 | M | 76000

2 | F | 76000

7 | M | 76000

1 | F | 76000

8 | M | 76000

3 | | 76000

5 | | 76000

4 | | 76000

6 | | 76000

2 | | 76000

7 | | 76000

1 | | 76000

8 | | 76000

| M | 456000

| F | 152000

| | 608000

(19 rows)

HR=# SELECT dept\_id, gender, SUM(salary) FROM employee

HR-# GROUP BY

HR-# GROUPING SETS (

HR(# (dept\_id, gender),

HR(# (dept\_id),

HR(# (gender),

HR(# ()

HR(# );

dept\_id | gender | sum

---------+--------+--------

| | 608000

5 | M | 76000

8 | M | 76000

3 | M | 76000

1 | F | 76000

4 | M | 76000

2 | F | 76000

6 | M | 76000

7 | M | 76000

3 | | 76000

5 | | 76000

4 | | 76000

6 | | 76000

2 | | 76000

7 | | 76000

1 | | 76000

8 | | 76000

| M | 456000

| F | 152000

(19 rows)

HR=#

UPDATE department

SET last\_name = 'Blamire',

birthdate = NULL,

salary = 10000

WHERE emp\_id = 4

RETURNING first\_name, last\_name, salary;

-------------------------

DELETE FROM department

WHERE emp\_id = 7;

--------

SELECT emp\_id, gender, SUM(salary) FROM employee

GROUP BY emp\_id, gender;

------

SELECT emp\_id, SUM(salary) AS "Total Salary" FROM employee

GROUP BY emp\_id;

------------------

SELECT emp\_id, SUM(salary) AS "Total Salary" FROM employee

GROUP BY emp\_id

HAVING SUM(salary) > 200000;

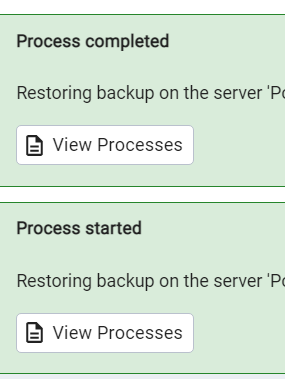
-------------------

SELECT emp\_id, COUNT(emp\_id) AS "No of Employees", MAX(salary) as "Max Salary"

FROM employee

GROUP BY emp\_id

HAVING COUNT(emp\_id) > 3 AND MAX(salary) > 5000;



We will use the DVD rental database to demonstrate the features of PostgreSQL.

The DVD rental database represents the business processes of a DVD rental store. The DVD rental database has many objects, including:

* 15 tables
* 1 trigger
* 7 views
* 8 functions
* 1 domain
* 13 sequences

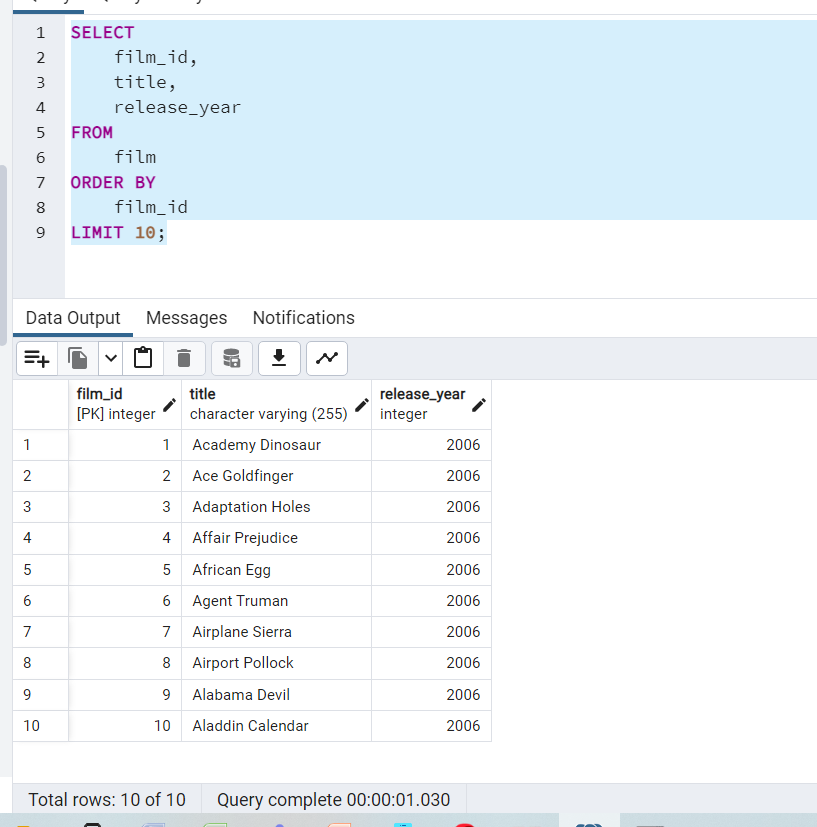
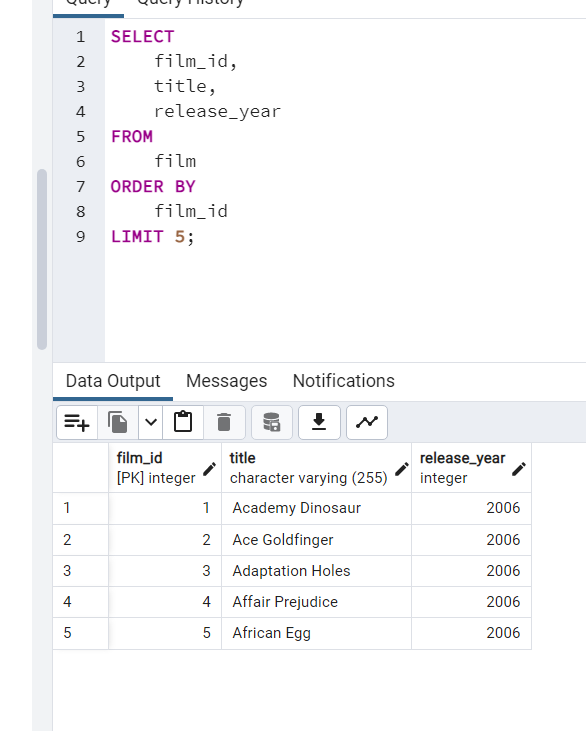
## DVD Rental ER Model

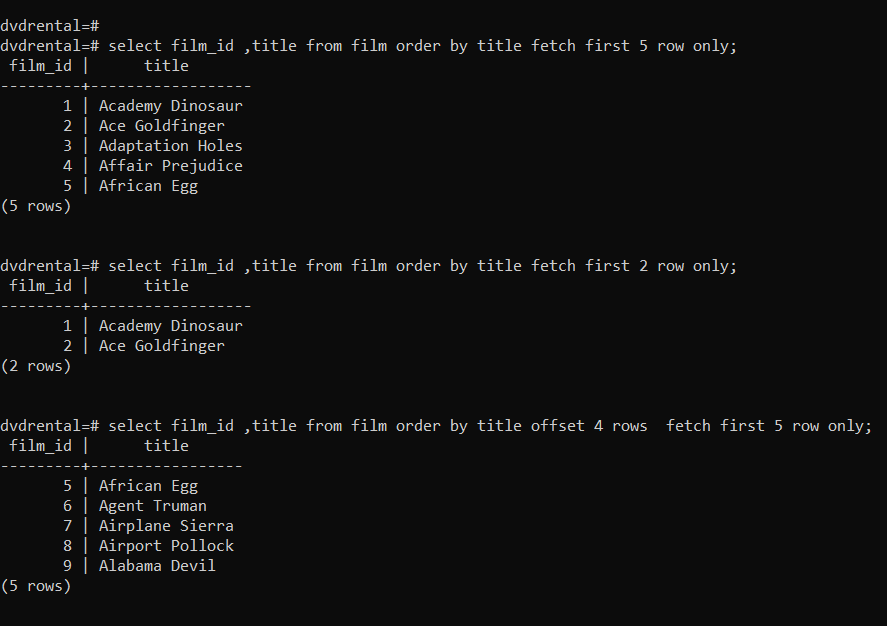
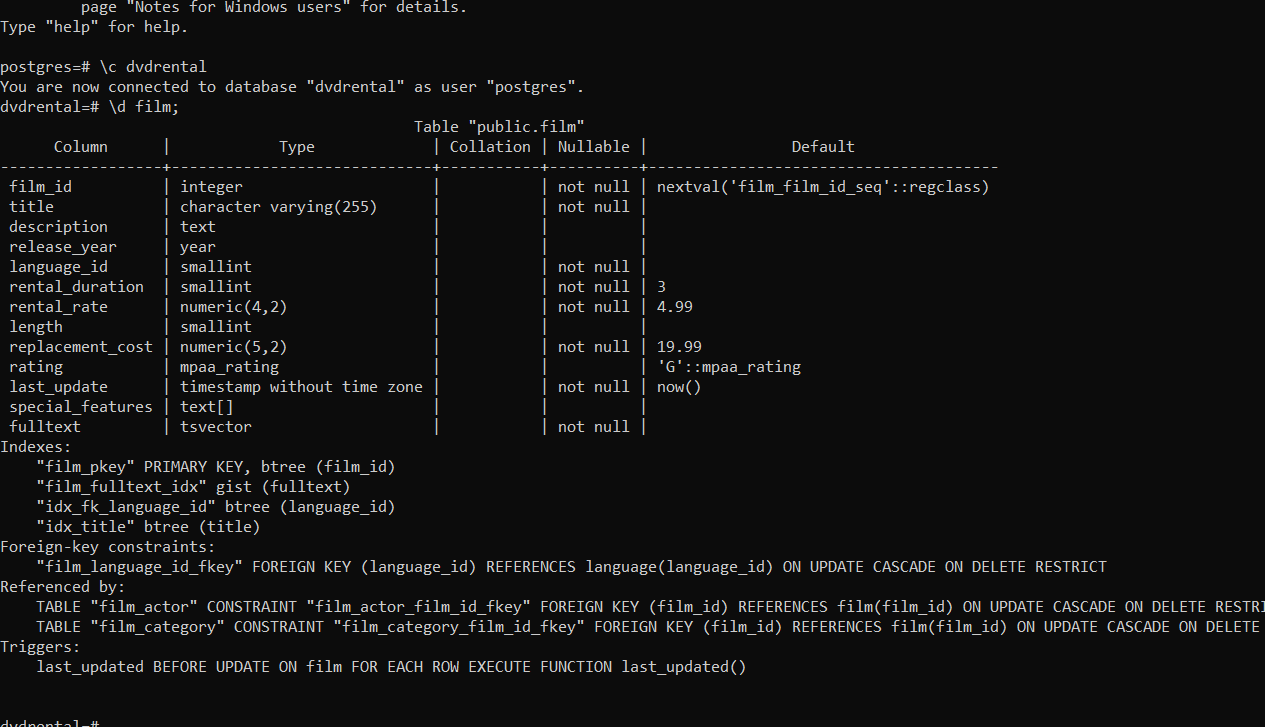
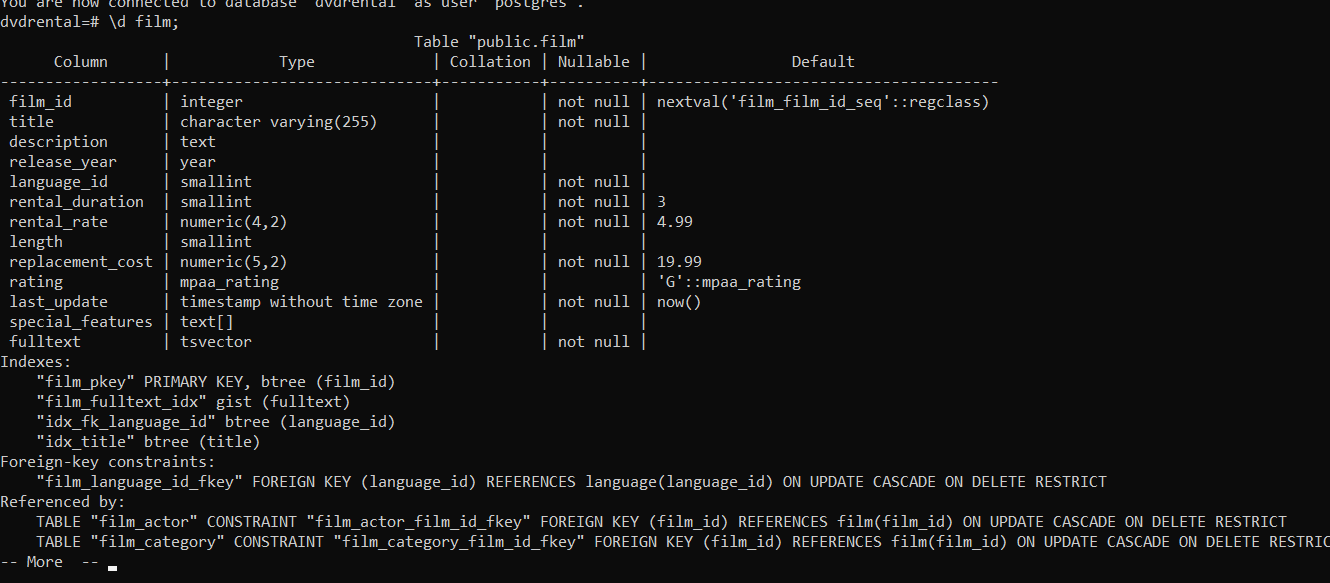
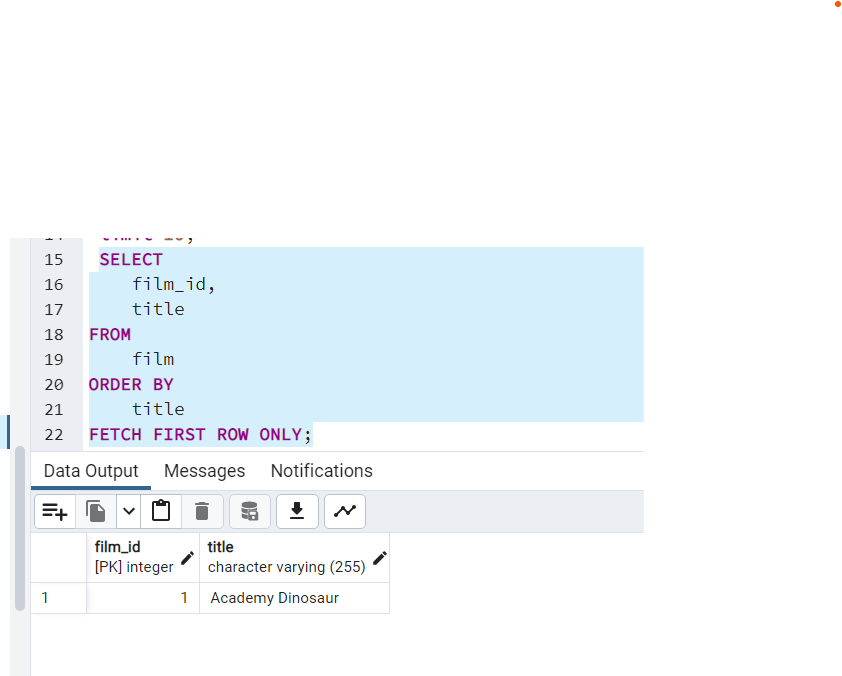
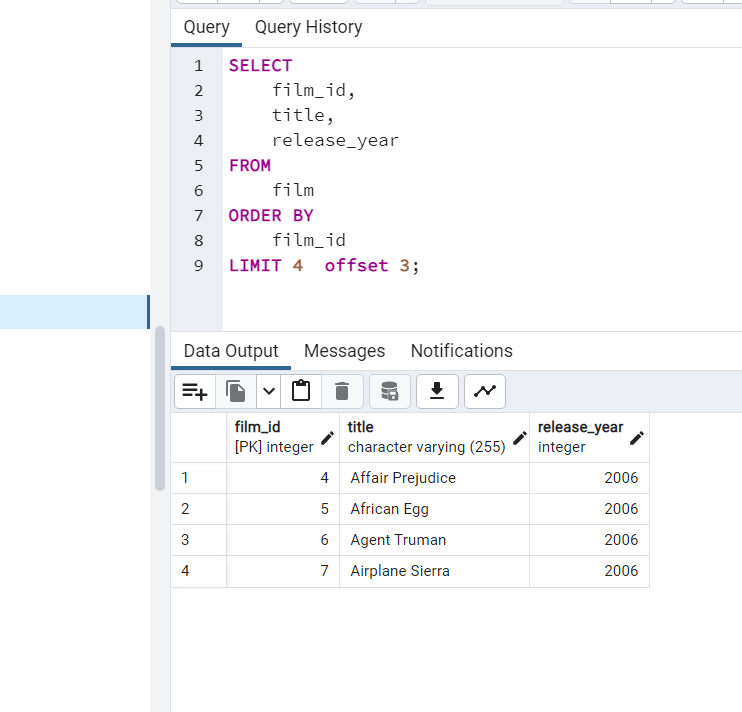
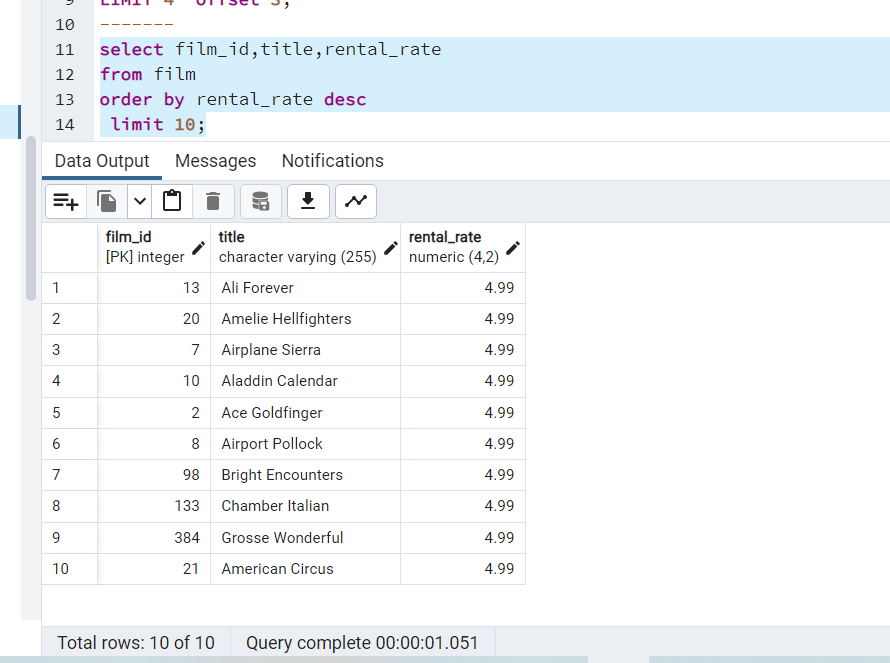
## PostgreSQL Sample Database Diagram

## PostgreSQL Sample Database Tables

There are 15 tables in the DVD Rental database:

* actor – stores actors data including first name and last name.
* film – stores film data such as title, release year, length, rating, etc.
* film\_actor – stores the relationships between films and actors.
* category – stores film’s categories data.
* film\_category- stores the relationships between films and categories.
* store – contains the store data including manager staff and address.
* inventory – stores inventory data.
* rental – stores rental data.
* payment – stores customer’s payments.
* staff – stores staff data.
* customer – stores customer data.
* address – stores address data for staff and customers
* city – stores city names.
* country – stores country names.





postgres=# \c dvdrental

You are now connected to database "dvdrental" as user "postgres".

dvdrental=# \d film;

Table "public.film"

Column | Type | Collation | Nullable | Default

------------------+-----------------------------+-----------+----------+---------------------------------------

film\_id | integer | | not null | nextval('film\_film\_id\_seq'::regclass)

title | character varying(255) | | not null |

description | text | | |

release\_year | year | | |

language\_id | smallint | | not null |

rental\_duration | smallint | | not null | 3

rental\_rate | numeric(4,2) | | not null | 4.99

length | smallint | | |

replacement\_cost | numeric(5,2) | | not null | 19.99

rating | mpaa\_rating | | | 'G'::mpaa\_rating

last\_update | timestamp without time zone | | not null | now()

special\_features | text[] | | |

fulltext | tsvector | | not null |

Indexes:

"film\_pkey" PRIMARY KEY, btree (film\_id)

"film\_fulltext\_idx" gist (fulltext)

"idx\_fk\_language\_id" btree (language\_id)

"idx\_title" btree (title)

Foreign-key constraints:

"film\_language\_id\_fkey" FOREIGN KEY (language\_id) REFERENCES language(language\_id) ON UPDATE CASCADE ON DELETE RESTRICT

Referenced by:

TABLE "film\_actor" CONSTRAINT "film\_actor\_film\_id\_fkey" FOREIGN KEY (film\_id) REFERENCES film(film\_id) ON UPDATE CASCADE ON DELETE RESTRICT

TABLE "film\_category" CONSTRAINT "film\_category\_film\_id\_fkey" FOREIGN KEY (film\_id) REFERENCES film(film\_id) ON UPDATE CASCADE ON DELETE RESTRICT

Triggers:

last\_updated BEFORE UPDATE ON film FOR EACH ROW EXECUTE FUNCTION last\_updated()

dvdrental=#

dvdrental=# select film\_id ,title from film order by title fetch first 5 row only;

film\_id | title

---------+------------------

1 | Academy Dinosaur

2 | Ace Goldfinger

3 | Adaptation Holes

4 | Affair Prejudice

5 | African Egg

(5 rows)

dvdrental=# select film\_id ,title from film order by title fetch first 2 row only;

film\_id | title

---------+------------------

1 | Academy Dinosaur

2 | Ace Goldfinger

(2 rows)

dvdrental=# select film\_id ,title from film order by title offset 4 rows fetch first 5 row only;

film\_id | title

---------+-----------------

5 | African Egg

6 | Agent Truman

7 | Airplane Sierra

8 | Airport Pollock

9 | Alabama Devil

(5 rows)

dvdrental=#

