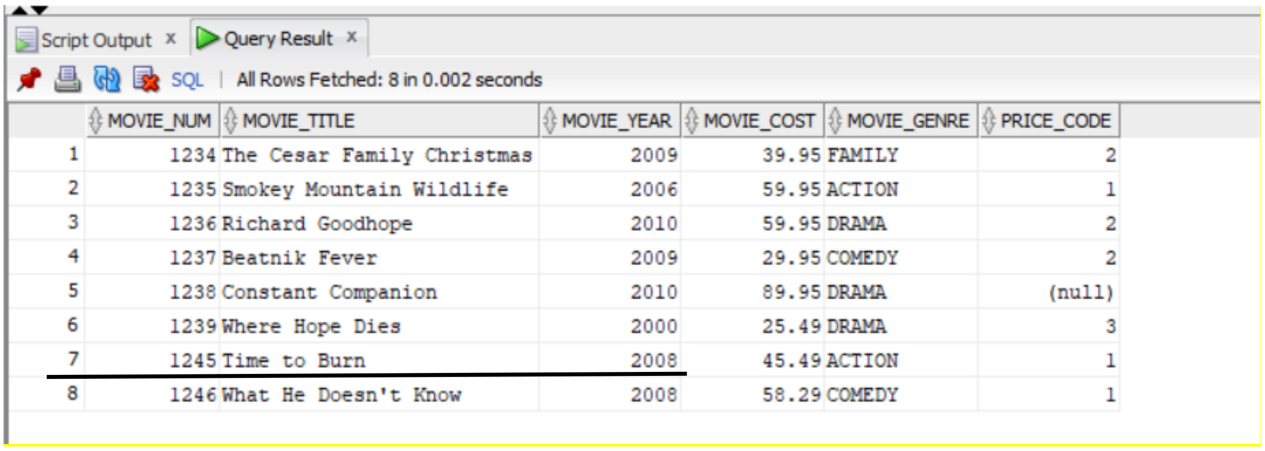
DBMS & SQL – Graded Microproject

1. **Write the SQL command to change the movie year for movie number 1245 to 2008.**

SQL Query: **update movie set movie_year = 2008 where movie_num = 1245; (select * from movie)**

Output:



Script Output × | Query Result ×

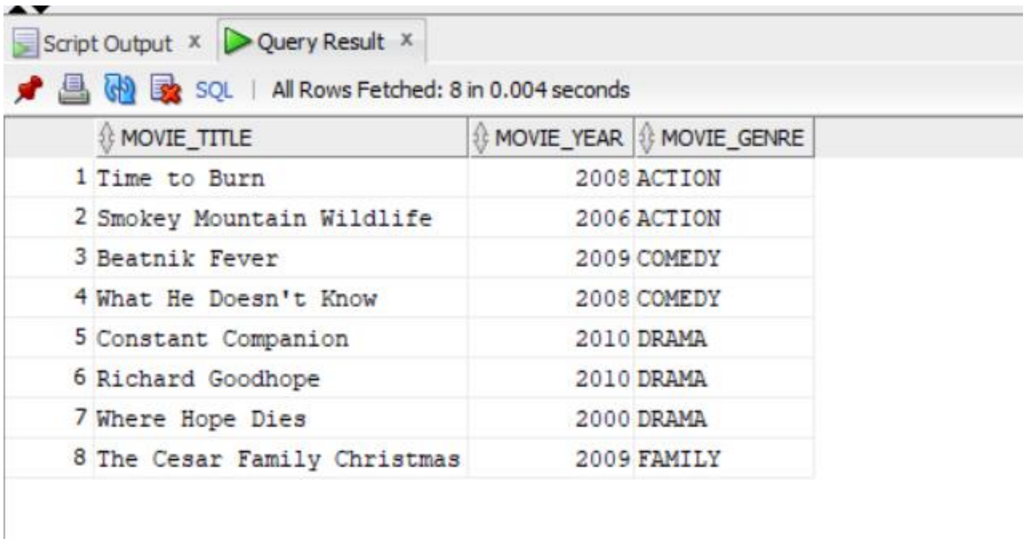
SQL | All Rows Fetched: 8 in 0.002 seconds

	MOVIE_NUM	MOVIE_TITLE	MOVIE_YEAR	MOVIE_COST	MOVIE_GENRE	PRICE_CODE
1	1234	The Cesar Family Christmas	2009	39.95	FAMILY	2
2	1235	Smokey Mountain Wildlife	2006	59.95	ACTION	1
3	1236	Richard Goodhope	2010	59.95	DRAMA	2
4	1237	Beatnik Fever	2009	29.95	COMEDY	2
5	1238	Constant Companion	2010	89.95	DRAMA	(null)
6	1239	Where Hope Dies	2000	25.49	DRAMA	3
7	1245	Time to Burn	2008	45.49	ACTION	1
8	1246	What He Doesn't Know	2008	58.29	COMEDY	1

2. **Write a query to display the movie title, movie year, and movie genre for all movies sorted by movie genre in ascending order, then sorted by movie year in descending order within genre**

SQL Query: **select movie_title, movie_year, movie_genre from movie order by movie_genre asc, movie_year desc;**

Output:



Script Output × | Query Result ×

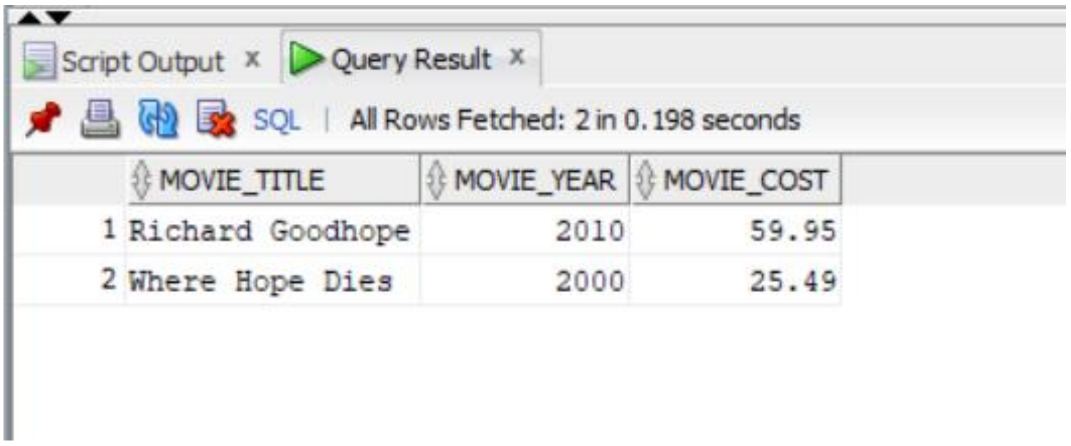
SQL | All Rows Fetched: 8 in 0.004 seconds

	MOVIE_TITLE	MOVIE_YEAR	MOVIE_GENRE
1	Time to Burn	2008	ACTION
2	Smokey Mountain Wildlife	2006	ACTION
3	Beatnik Fever	2009	COMEDY
4	What He Doesn't Know	2008	COMEDY
5	Constant Companion	2010	DRAMA
6	Richard Goodhope	2010	DRAMA
7	Where Hope Dies	2000	DRAMA
8	The Cesar Family Christmas	2009	FAMILY

3. **Write a query to display the movie title, movie year, and movie cost for all movies that contain the word “hope” anywhere in the title. Sort the results in ascending order by title.**

   SQL Query: select movie_title, movie_year, movie_cost from movie where movie_title like '%hope%' or movie_title like '%Hope%' order by movie_title asc;

   Output:



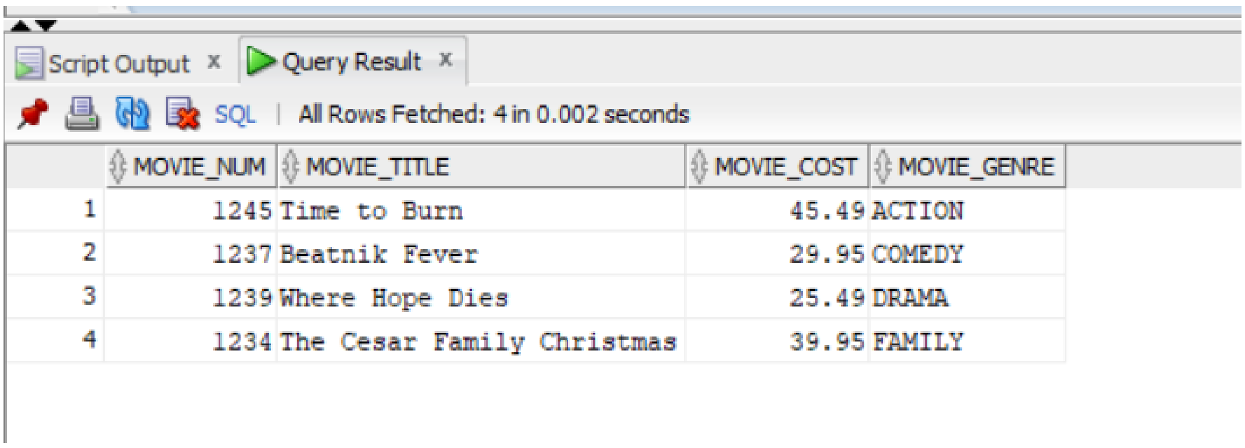
All Rows Fetched: 2 in 0.198 seconds

```
  MOVIE_TITLE       MOVIE_YEAR  MOVIE_COST
1 Richard Goodhope        2010       59.95
2 Where Hope Dies         2000       25.49
```

4. **Write a query to display the movie number, movie title, movie cost, and movie genre for all movies that are either action or comedy movies or that have a cost that is less than $50. Sort the results in ascending order by genre. (output is different from pdf)**

   SQL Query: select movie_num, movie_title, movie_cost, movie_genre from movie where movie_genre in ('Action', 'Comedy') or movie_cost < 50 order by movie_genre asc;

   Output:



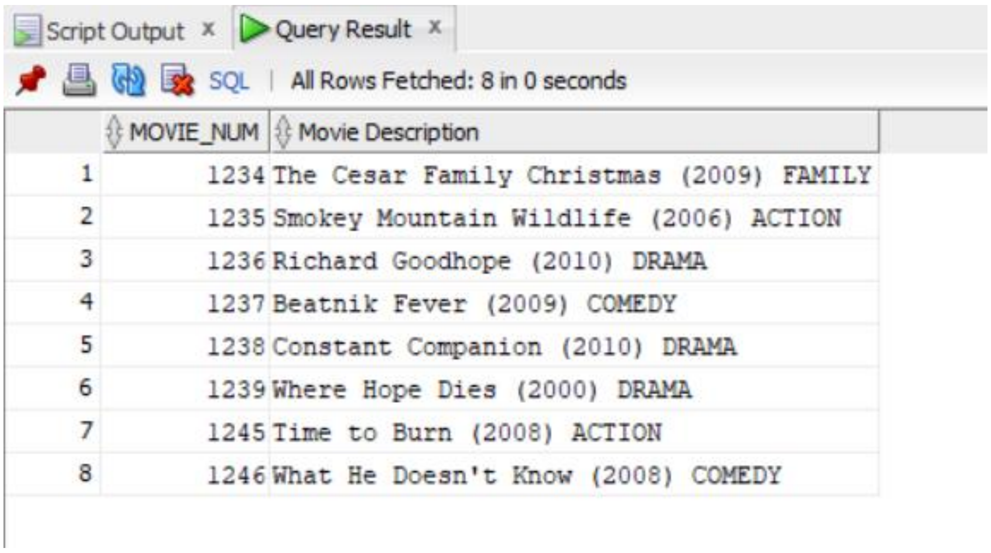
All Rows Fetched: 4 in 0.002 seconds

```
  MOVIE_NUM MOVIE_TITLE                 MOVIE_COST MOVIE_GENRE
1      1245 Time to Burn                     45.49 ACTION
2      1237 Beatnik Fever                    29.95 COMEDY
3      1239 Where Hope Dies                  25.49 DRAMA
4      1234 The Cesar Family Christmas       39.95 FAMILY
```

5. **Write a query to display the movie number, and movie description for all movies where the movie description is a combination of the movie title, movie year and movie genre with the movie year enclosed in parentheses**

   SQL Query: select movie_num, movie_title || ' (' || movie_year || ') ' || movie_genre as "Movie Description" from movie;

   Output:



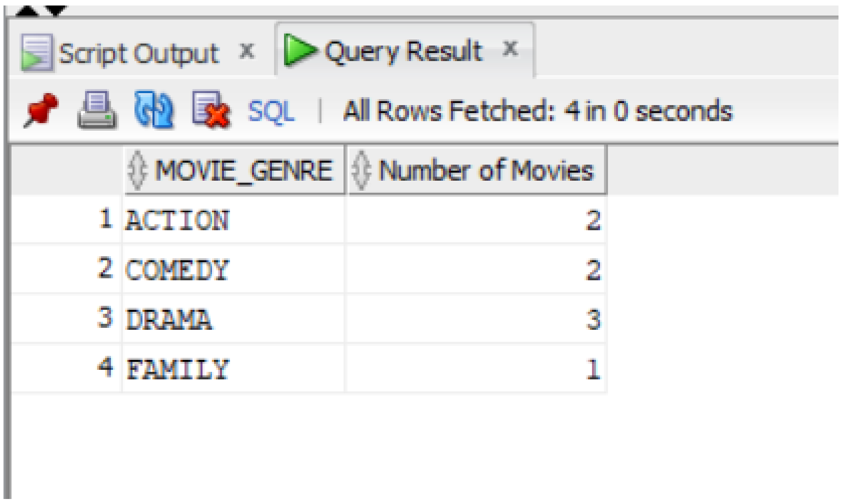
```
All Rows Fetched: 8 in 0 seconds

     MOVIE_NUM  Movie Description
1         1234  The Cesar Family Christmas (2009) FAMILY
2         1235  Smokey Mountain Wildlife (2006) ACTION
3         1236  Richard Goodhope (2010) DRAMA
4         1237  Beatnik Fever (2009) COMEDY
5         1238  Constant Companion (2010) DRAMA
6         1239  Where Hope Dies (2000) DRAMA
7         1245  Time to Burn (2008) ACTION
8         1246  What He Doesn't Know (2008) COMEDY
```

6. **Write a query to display the movie genre and the number of movies in each genre**

   SQL Query: select movie_genre, count(*) as "Number of Movies" from movie group by movie_genre order by movie_genre asc;

   Output:



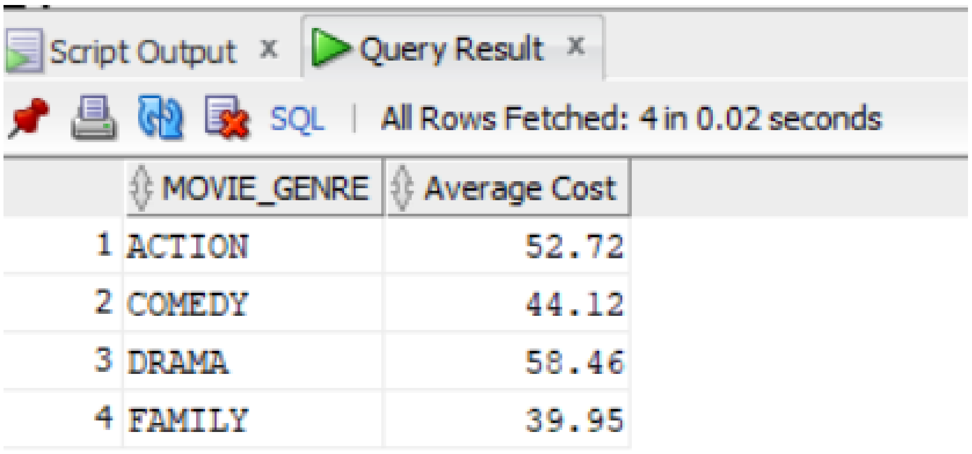
```
All Rows Fetched: 4 in 0 seconds

  MOVIE_GENRE  Number of Movies
1 ACTION                      2
2 COMEDY                      2
3 DRAMA                       3
4 FAMILY                      1
```

7. **Write a query to display the movie genre and average cost of movies in each genre**

   SQL Query: select movie_genre, ROUND(avg(movie_cost),2)as "Average Cost" from movie group by movie_genre order by movie_genre asc;

   Output:



Script Output × | Query Result ×

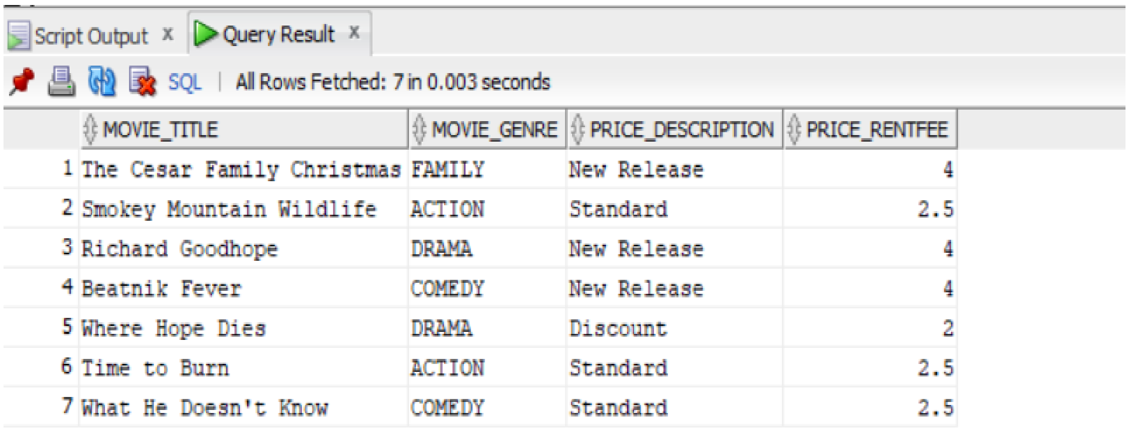
SQL | All Rows Fetched: 4 in 0.02 seconds

```
  MOVIE_GENRE  Average Cost
1 ACTION              52.72
2 COMEDY              44.12
3 DRAMA               58.46
4 FAMILY              39.95
```

8. **Write a query to display the movie title, movie genre, price description, and price rental fee for all movies with a price code (order is different from pdf)**

   SQL Query: select movie_title, movie_genre, price_description, price_rentfee from movie join price on movie.price_code = price.price_code;

   Output:



Script Output × | Query Result ×

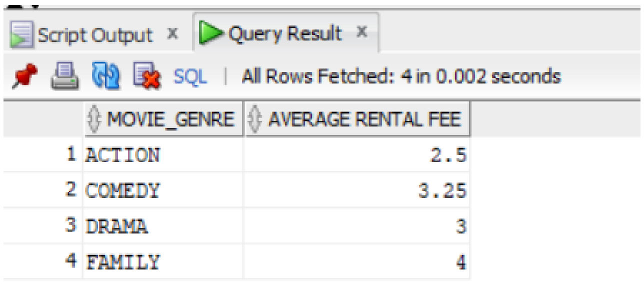
SQL | All Rows Fetched: 7 in 0.003 seconds

```
  MOVIE_TITLE                 MOVIE_GENRE PRICE_DESCRIPTION PRICE_RENTFEE
1 The Cesar Family Christmas FAMILY      New Release                   4
2 Smokey Mountain Wildlife   ACTION      Standard                    2.5
3 Richard Goodhope           DRAMA       New Release                   4
4 Beatnik Fever              COMEDY      New Release                   4
5 Where Hope Dies            DRAMA       Discount                      2
6 Time to Burn               ACTION      Standard                    2.5
7 What He Doesn't Know       COMEDY      Standard                    2.5
```

9. **Write a query to display the movie genre and average price rental fee for movies in each genre that have a price**

   SQL Query: SELECT MOVIE_GENRE, AVG(PRICE_RENTFEE) AS "AVERAGE RENTAL FEE" FROM MOVIE INNER JOIN PRICE ON MOVIE.PRICE_CODE = PRICE.PRICE_CODE GROUP BY MOVIE_GENRE ORDER BY MOVIE_GENRE ASC;

   Output:



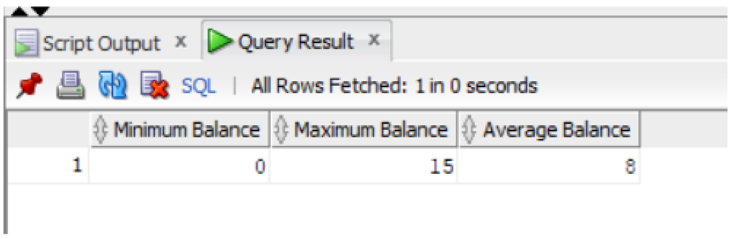
All Rows Fetched: 4 in 0.002 seconds

	MOVIE_GENRE	AVERAGE RENTAL FEE
1	ACTION	2.5
2	COMEDY	3.25
3	DRAMA	3
4	FAMILY	4

10. **Write a query to display the minimum balance, maximum balance, and average balance for memberships that have a rental**

    SQL Query: select min(mem_balance) as "Minimum Balance", max(mem_balance) as "Maximum Balance", Round(avg(mem_balance)) as "Average Balance" from membership m full join rental r on m.mem_num = r.mem_num where r.rent_num is not null;

    Output:



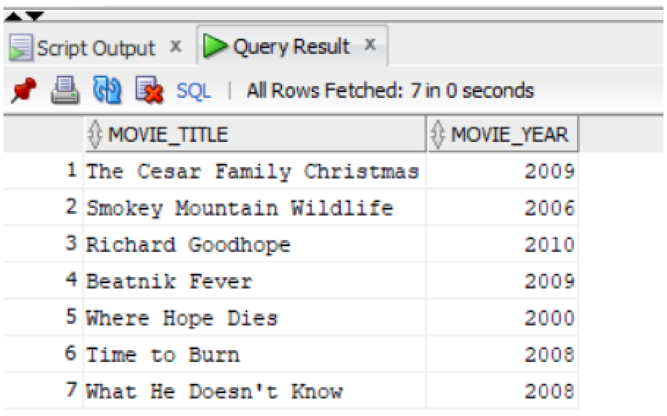
All Rows Fetched: 1 in 0 seconds

	Minimum Balance	Maximum Balance	Average Balance
1	0	15	8

11. **Write a query to display the movie title and movie year for all movies that have a price code**

    SQL Query: select movie_title, movie_year from movie inner join price on movie.price_code = price.price_code where price.price_code is not null;

    Output:



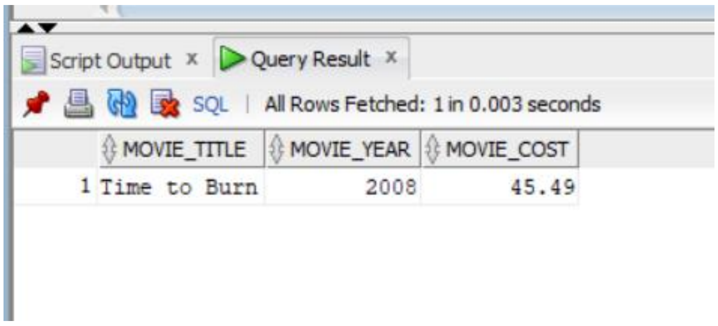
All Rows Fetched: 7 in 0 seconds

	MOVIE_TITLE	MOVIE_YEAR
1	The Cesar Family Christmas	2009
2	Smokey Mountain Wildlife	2006
3	Richard Goodhope	2010
4	Beatnik Fever	2009
5	Where Hope Dies	2000
6	Time to Burn	2008
7	What He Doesn't Know	2008

12. **Write a query to display the movie title, movie year, and movie cost for all movies that have a cost between $44.99 and $49.99**

    SQL Query: select movie_title, movie_year, movie_cost from movie where movie_cost between 44.99 and 49.99;

    Output:



Script Output × Query Result ×

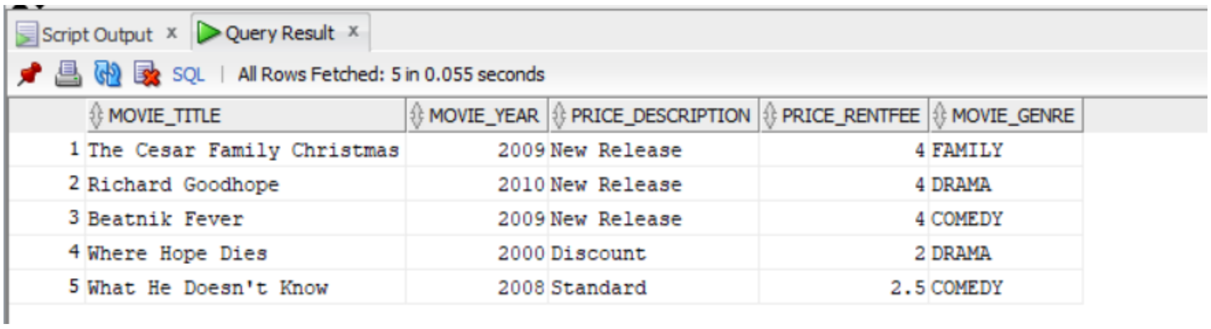
SQL | All Rows Fetched: 1 in 0.003 seconds

	MOVIE_TITLE	MOVIE_YEAR	MOVIE_COST
1	Time to Burn	2008	45.49

13. **Write a query to display the movie title, movie year, price description, and price rental fee for all movies that are in the genres Family, Comedy, or Drama**

    SQL Query: select movie_title, movie_year, price_description, price_rentfee, movie_genre from movie inner join price on movie.price_code = price.price_code where movie_genre = 'FAMILY' OR movie_genre = 'COMEDY' OR movie_genre = 'DRAMA';

    Output:



Script Output × Query Result ×

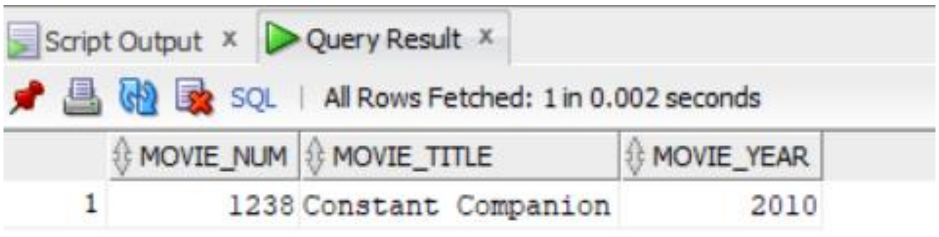
SQL | All Rows Fetched: 5 in 0.055 seconds

	MOVIE_TITLE	MOVIE_YEAR	PRICE_DESCRIPTION	PRICE_RENTFEE	MOVIE_GENRE
1	The Cesar Family Christmas	2009	New Release	4	FAMILY
2	Richard Goodhope	2010	New Release	4	DRAMA
3	Beatnik Fever	2009	New Release	4	COMEDY
4	Where Hope Dies	2000	Discount	2	DRAMA
5	What He Doesn't Know	2008	Standard	2.5	COMEDY

14. **Write a query to display the movie number, movie title, and movie year for all movies that do not have a video**

    SQL Query: select movie_num, movie_title, movie_year from movie where movie_num not in (select movie_num from video);

    Output:



Script Output × Query Result ×

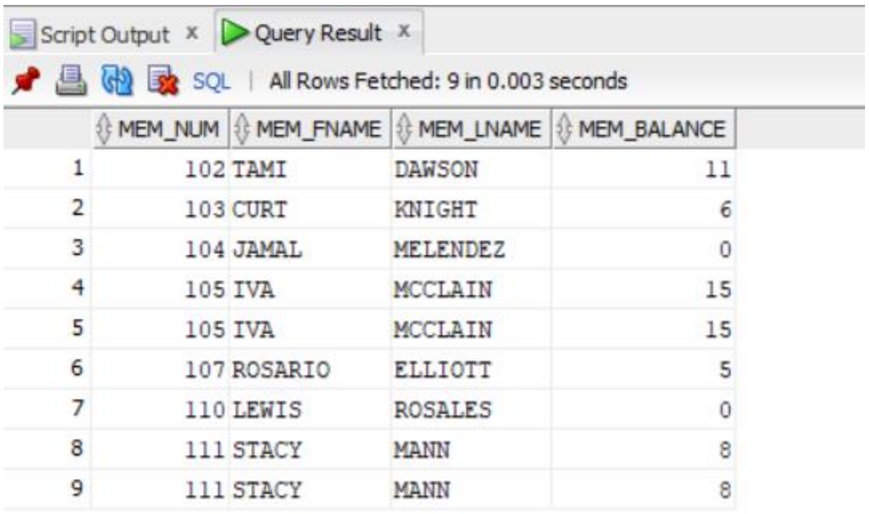
SQL | All Rows Fetched: 1 in 0.002 seconds

	MOVIE_NUM	MOVIE_TITLE	MOVIE_YEAR
1	1238	Constant Companion	2010

15. **Write a query to display the membership number, first name, last name, and balance of the memberships that have a rental**

    SQL Query: select membership.mem_num, mem_fname, mem_lname, mem_balance from membership inner join rental ON membership.mem_num = rental.mem_num;

    Output:



All Rows Fetched: 9 in 0.003 seconds

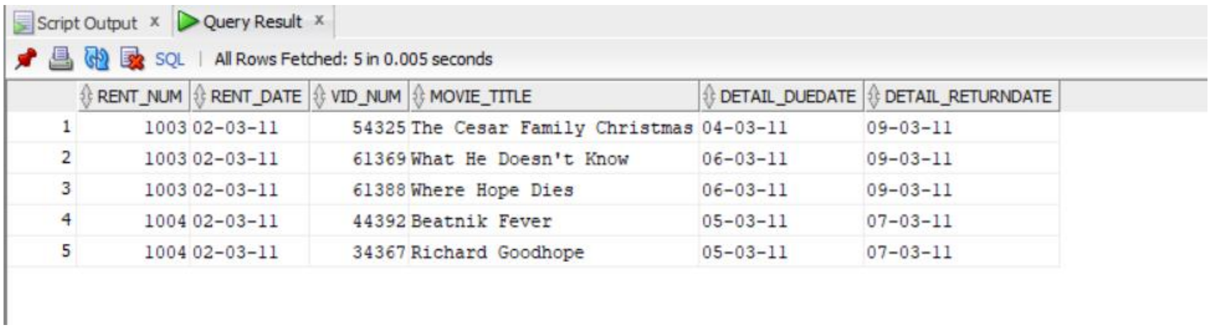
	MEM_NUM	MEM_FNAME	MEM_LNAME	MEM_BALANCE
1	102	TAMI	DAWSON	11
2	103	CURT	KNIGHT	6
3	104	JAMAL	MELENDEZ	0
4	105	IVA	MCCLAIN	15
5	105	IVA	MCCLAIN	15
6	107	ROSARIO	ELLIOTT	5
7	110	LEWIS	ROSALES	0
8	111	STACY	MANN	8
9	111	STACY	MANN	8

16. **Write a query to display the rental number, rental date, video number, movie title, due date, and return date for all videos that were returned after the due date. Sort the results by rental number and movie title**

    SQL Query:

```
select rental.rent_num, rental.rent_date, detailrental.vid_num, movie_title,
detailrental.detail_duedate,detailrental.detail_returndate from rental
join detailrental ON  rental.rent_num =detailrental.rent_num
join video ON detailrental.vid_num = video.vid_num
join movie on video.movie_num = movie.movie_num
where detailrental.detail_returndate > detailrental.detail_duedate
order by rental.rent_num, movie_title;
```

    Output:



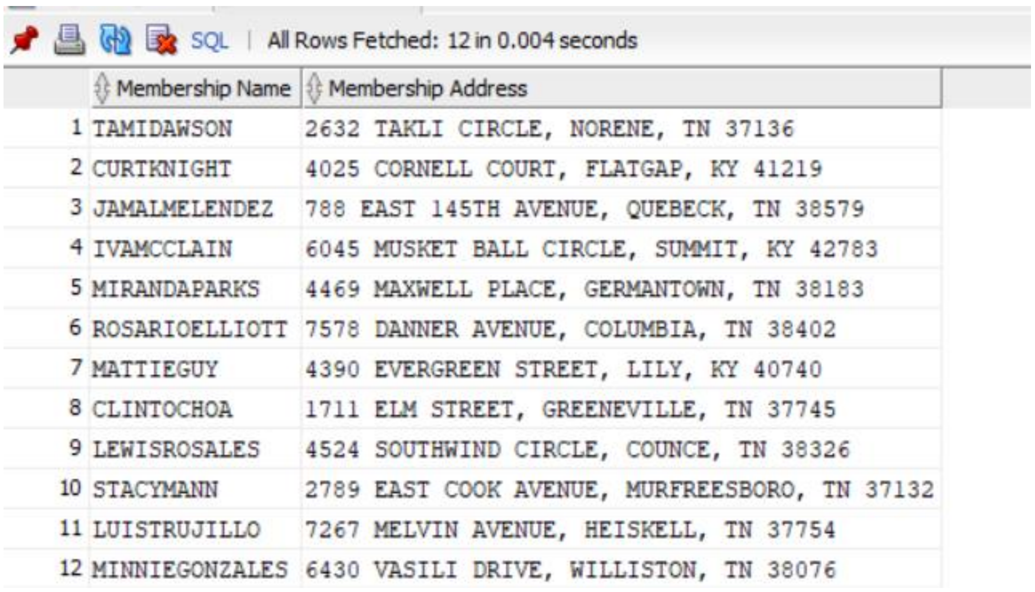
All Rows Fetched: 5 in 0.005 seconds

	RENT_NUM	RENT_DATE	VID_NUM	MOVIE_TITLE	DETAIL_DUEDATE	DETAIL_RETURNDATE
1	1003	02-03-11	54325	The Cesar Family Christmas	04-03-11	09-03-11
2	1003	02-03-11	61369	What He Doesn't Know	06-03-11	09-03-11
3	1003	02-03-11	61388	Where Hope Dies	06-03-11	09-03-11
4	1004	02-03-11	44392	Beatnik Fever	05-03-11	07-03-11
5	1004	02-03-11	34367	Richard Goodhope	05-03-11	07-03-11

17. **Write a query to display the membership name (concatenate the first name and last name with a space between them into a single column), membership address (concatenate the street, city, state, and zip codes into a single column with spaces**

SQL Query: select mem_fname || '' || mem_lname as "Membership Name", mem_street || ', ' || mem_city || ', ' || mem_state || ' ' || mem_zip as "Membership Address" from membership;

Output:



SQL | All Rows Fetched: 12 in 0.004 seconds

	Membership Name	Membership Address
1	TAMIDAWSON	2632 TAKLI CIRCLE, NORENE, TN 37136
2	CURTKNIGHT	4025 CORNELL COURT, FLATGAP, KY 41219
3	JAMALMELENDEZ	788 EAST 145TH AVENUE, QUEBECK, TN 38579
4	IVAMCCLAIN	6045 MUSKET BALL CIRCLE, SUMMIT, KY 42783
5	MIRANDAPARKS	4469 MAXWELL PLACE, GERMANTOWN, TN 38183
6	ROSARIOELLIOTT	7578 DANNER AVENUE, COLUMBIA, TN 38402
7	MATTIEGUY	4390 EVERGREEN STREET, LILY, KY 40740
8	CLINTOCHOA	1711 ELM STREET, GREENEVILLE, TN 37745
9	LEWISROSALES	4524 SOUTHWIND CIRCLE, COUNCE, TN 38326
10	STACYMANN	2789 EAST COOK AVENUE, MURFREESBORO, TN 37132
11	LUISTRUJILLO	7267 MELVIN AVENUE, HEISKELL, TN 37754
12	MINNIEGONZALES	6430 VASILI DRIVE, WILLISTON, TN 38076

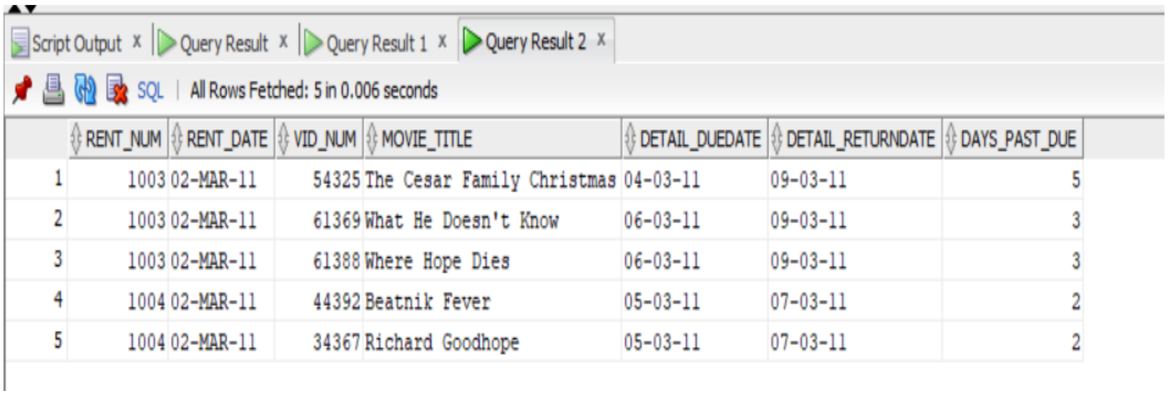
18. **Write a query to display the rental number, rental date, video number, movie title, due date, return date, detail fee, and number of days past the due date that the video was returned for each video that was returned after the due date. Sort the results by rental number and movie title.**

SQL Query:

```
SELECT
  r.rent_num,
  TO_CHAR(r.rent_date, 'DD-MON-YY')as RENT_DATE,
  v.vid_num,
  m.movie_title,
  dr.detail_duedate,
  dr.detail_returndate,

  CASE
    WHEN dr.detail_returndate > dr.detail_duedate THEN TRUNC(dr.detail_returndate)-
TRUNC (dr.detail_duedate)
    ELSE 0
  END AS days_past_due
FROM
  rental r
JOIN detailrental dr ON r.rent_num = dr.rent_num
JOIN video v ON dr.vid_num = v.vid_num
JOIN movie m ON v.movie_num = m.movie_num
WHERE
  dr.detail_returndate > dr.detail_duedate
ORDER BY
  r.rent_num,
  m.movie_title;
```

Output:



Script Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | All Rows Fetched: 5 in 0.006 seconds

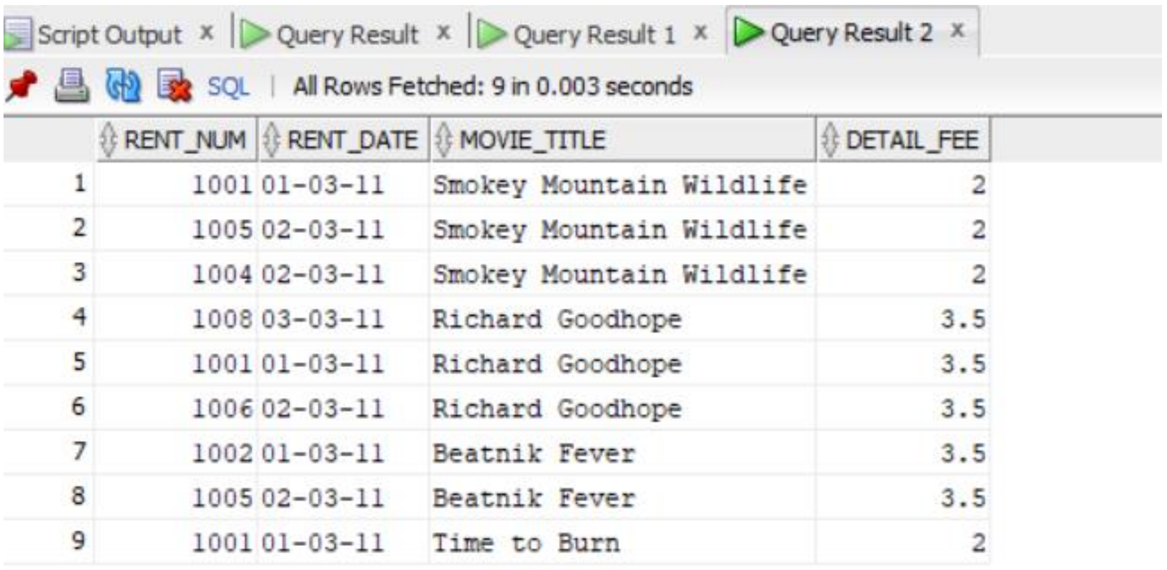
	RENT_NUM	RENT_DATE	VID_NUM	MOVIE_TITLE	DETAIL_DUEDATE	DETAIL_RETURNDATE	DAYS_PAST_DUE
1	1003	02-MAR-11	54325	The Cesar Family Christmas	04-03-11	09-03-11	5
2	1003	02-MAR-11	61369	What He Doesn't Know	06-03-11	09-03-11	3
3	1003	02-MAR-11	61388	Where Hope Dies	06-03-11	09-03-11	3
4	1004	02-MAR-11	44392	Beatnik Fever	05-03-11	07-03-11	2
5	1004	02-MAR-11	34367	Richard Goodhope	05-03-11	07-03-11	2

19. **Write a query to display the rental number, rental date, movie title, and detail fee for each movie that was returned on or before the due date (order is different from pdf)**

SQL Query:

```
select r.rent_num, r.rent_date, m.movie_title, dr.detail_fee from rental r
join detailrental dr on r.rent_num = dr.rent_num
join video v on dr.vid_num = v.vid_num
join movie m on v.movie_num = m.movie_num
where dr.detail_returndate < = dr.detail_duedate;
```

Output:



Script Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | All Rows Fetched: 9 in 0.003 seconds

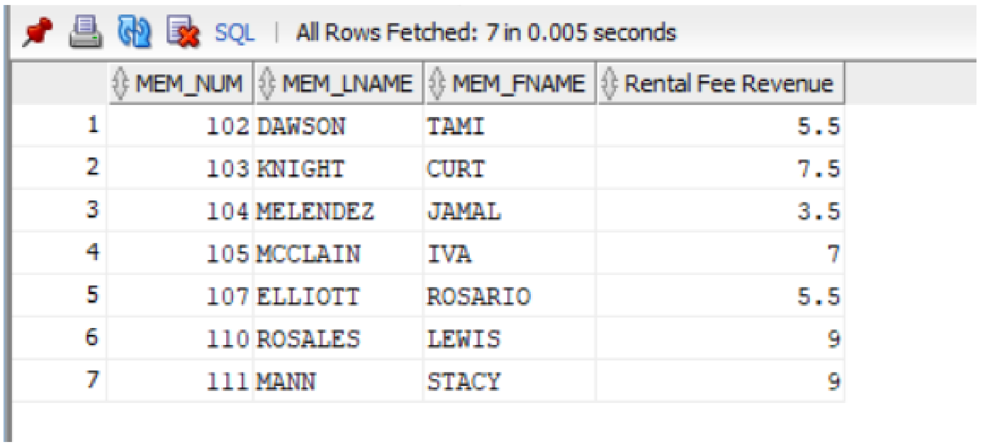
	RENT_NUM	RENT_DATE	MOVIE_TITLE	DETAIL_FEE
1	1001	01-03-11	Smokey Mountain Wildlife	2
2	1005	02-03-11	Smokey Mountain Wildlife	2
3	1004	02-03-11	Smokey Mountain Wildlife	2
4	1008	03-03-11	Richard Goodhope	3.5
5	1001	01-03-11	Richard Goodhope	3.5
6	1006	02-03-11	Richard Goodhope	3.5
7	1002	01-03-11	Beatnik Fever	3.5
8	1005	02-03-11	Beatnik Fever	3.5
9	1001	01-03-11	Time to Burn	2

20. **Write a query to display the membership number, last name, and total rental fees earned from that membership. The total rental fee is the sum of all of the detail fees (without the late fees) from all movies that the membership has rented.**

SQL Query:

```
 SELECT
  m.mem_num,
  m.mem_lname,
  m.mem_fname,
  SUM(dr.detail_fee) AS "Rental Fee Revenue"
FROM
  membership m
LEFT JOIN rental r ON m.mem_num = r.mem_num
LEFT JOIN detailrental dr ON r.rent_num = dr.rent_num
GROUP BY
  m.mem_num,
  m.mem_lname,
  m.mem_fname
HAVING
  SUM(dr.detail_fee) > 0
ORDER BY
  m.mem_num ASC;
```

Output:



SQL | All Rows Fetched: 7 in 0.005 seconds

	MEM_NUM	MEM_LNAME	MEM_FNAME	Rental Fee Revenue
1	102	DAWSON	TAMI	5.5
2	103	KNIGHT	CURT	7.5
3	104	MELENDEZ	JAMAL	3.5
4	105	MCCLAIN	IVA	7
5	107	ELLIOTT	ROSARIO	5.5
6	110	ROSALES	LEWIS	9
7	111	MANN	STACY	9