Please work in **groups** of max 4 and min 2 to complete this Assignment. This Assignment is worth 7% of the total course grade and will be evaluated through your written submission, as well as the Assignment demo in the class worth 1%. During the Assignment demo, group members are randomly selected to present the answers to each of the lab questions. Group members not present during the Assignment demo will lose the demo mark. Individual submission will have penalty of 10%.

Please submit the following files through Blackboard. Only one person must submit for the team.

* Project SQL file: Project1\_GroupNumber.SQL
* Project document: Project1\_ GroupNumber.docx
* In the docx file use the same document as Assignment 1. Put the query and screenshot of the results below each question.

# Part I. Group work

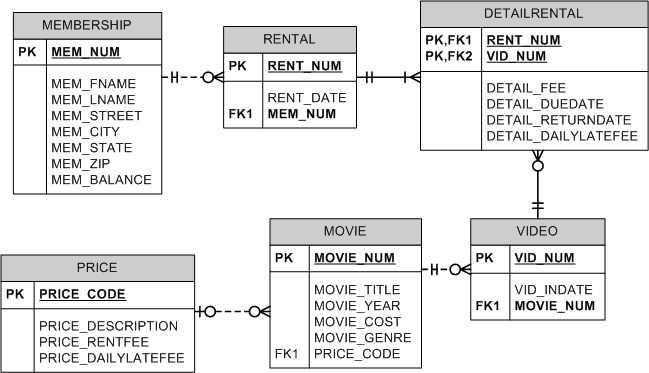
1. Add this declaration on the top of your file.

We, ------------(mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) **or distributed to other students.**

1. Specify what each member has done towards the completion of this work:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Task(s) | |
| 1 | Abiodun Oke |  | |
| 2 |  |  | |
| 3 |  | |
| 4 |  |  | |

**The ERD of movie rental database is provided below. The database is provided on Blackboard under assignment 1 link with the name of Movie\_Rental.rtf. The first 20 questions are worth 2.5 each and last 10 questions are worth 5 marks each. Run the script and then run the queries after solving in the questions below.**



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

**update Movie**

**set movie\_year = 2008**

**where movie\_num = 1245;**

1. **Write the SQL command to change the price code for all Action movies to price code 3.**

**update Movie**

**set price\_code = 3**

**where movie\_genre = 'ACTION';**

1. **Write a single SQL command to increase all price rental fee values by $0.50.**

**update Price**

**set price\_rentfee = price\_rentfee + 0.50;**

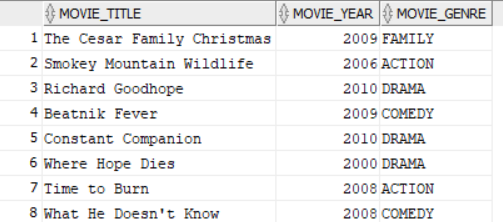
1. **Write a query to display the movie title, movie year, and movie genre for all movies (result shown in Figure P7.72).**

Figure P7.72 All Movies



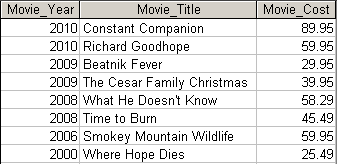
select movie\_title, movie\_year, movie\_genre

from MOVIE;



1. **Write a query to display the movie year, movie title, and movie cost sorted by movie year in descending order (result shown in Figure P7.73).**

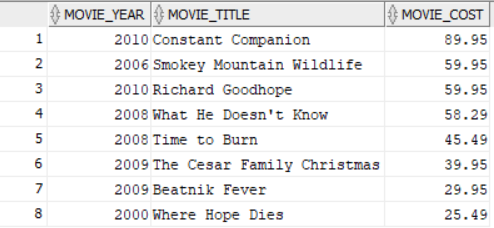
Figure P7.73 Movies by year



select movie\_year, movie\_title, movie\_cost

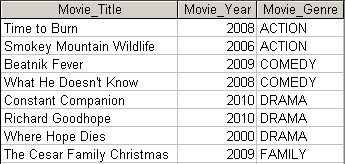
from MOVIE

order by 1 desc;



1. **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 (result shown in Figure P7.74).**

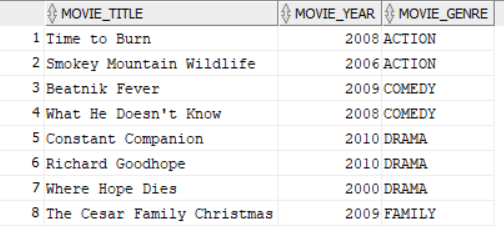
Figure P7.74 Movies with multicolumn sort



select movie\_title, movie\_year, movie\_genre

from MOVIE

order by 3 asc, 2 desc;



1. **Write a query to display the movie number, movie title, and price code for all movies with a title that starts with the letter “R” (result shown in Figure P7.75).**

Figure P7.75 Movies starting with R

FigP7-54-Movies-Starting-with-R

select movie\_num, movie\_title, price\_code

from Movie

where movie\_title like 'R%';



1. **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 (result shown in figure P7.76).**

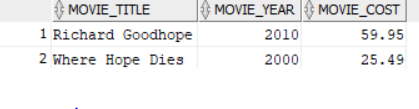
Figure P7.76 Movies with “Hope” in the title

FigP7-76-Movies-with-Hope

select movie\_title, movie\_year, movie\_cost

from Movie

where UPPER(Movie\_title) like '%HOPE%';



1. **Write a query to display the movie title, movie year, and movie genre for all action movies (result shown in Figure P7.77).**

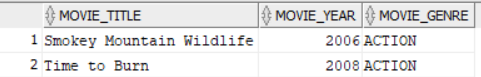
Figure P7.77 Action movies

FigP7-77-Action-Movies

select movie\_title, movie\_year, movie\_genre

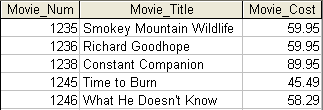
from Movie

where Movie\_genre = 'ACTION';



1. **Write a query to display the movie number, movie title, and movie cost for all movies with a cost greater than $40 (result shown in Figure P7.78).**

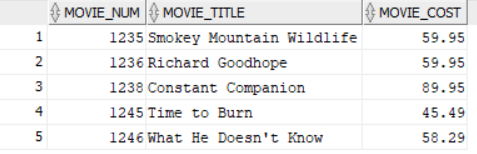
P7.78 Movies costing more than $40



select movie\_num, movie\_title, Movie\_cost

from Movie

where Movie\_cost > 40;



1. **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 and that have a cost that is less than $50. Sort the results in ascending order by genre. (Result shown in Figure P7.79.)**

Figure P7.79 Action or comedy movies costing less than $50



select movie\_num, movie\_title, Movie\_cost, movie\_genre

from Movie

where Movie\_genre in ('ACTION','COMEDY')

and movie\_cost < 50

order by 4 asc;



OR

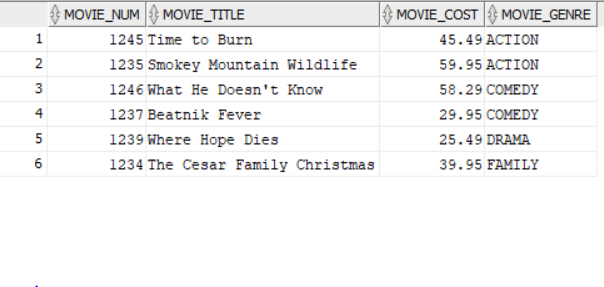
select movie\_num, movie\_title, Movie\_cost, movie\_genre

from Movie

where Movie\_genre in ('ACTION','COMEDY')

or movie\_cost < 50

order by 4 asc;



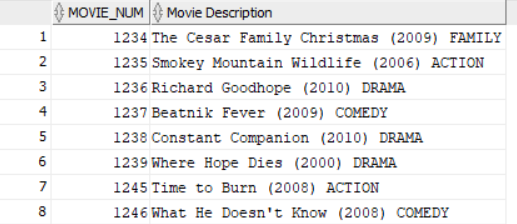
1. **Write a query to display the movie number, and movie description for all movies where the movie description is a combination of the movie title, movie year and movie genre with the movie year enclosed in parentheses (result shown in Figure P7.80).**

Figure P7.80 Movies with concatenated descriptions



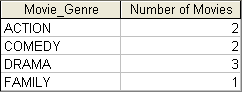
select movie\_num, movie\_title || ' (' || movie\_year || ') ' || movie\_genre "Movie Description"

from movie;



1. **Write a query to display the movie genre and the number of movies in each genre (result shown in Figure P7.81).**

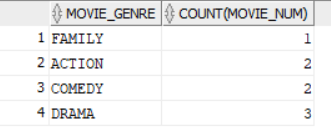
Figure P7.81 Number of movies in genre



select movie\_genre, count(movie\_num)

from movie

group by(movie\_genre);



1. **Write a query to display the average cost of all of the movies (result shown in Figure P7.82).**

Figure P7.82 Average movie cost

FigP7-61-Average-Movie-Cost

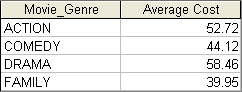
select avg(movie\_cost) "Average Movie Cost"

from Movie;



1. **Write a query to display the movie genre and average cost of movies in each genre (result shown in Figure P7.83).**

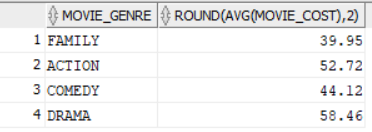
Figure P7.83 Average movie cost by genre



select movie\_genre, round(avg(movie\_cost),2)

from movie

group by(movie\_genre);



1. **Write a query to display the movie title, movie genre, price description, and price rental fee for all movies with a price code (result shown in Figure P7.84).**

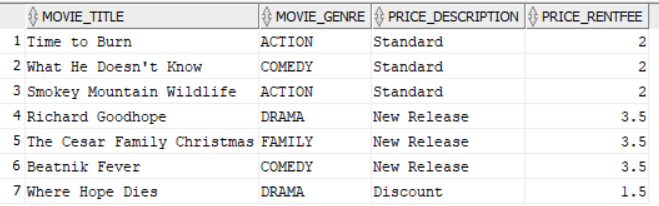
Figure P7.84 Rental fees for movies



select m.movie\_title, m.movie\_genre, p.price\_description, p.price\_rentfee

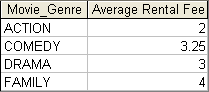
from movie m, price p

where m.price\_code = p.price\_code;



1. **Write a query to display the movie genre and average price rental fee for movies in each genre that have a price (result shown in Figure P7.85).**

Figure P7.85 Average rental fee by genre

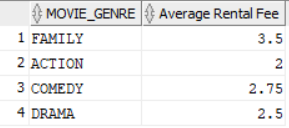


select m.movie\_genre, avg(p.price\_rentfee) "Average Rental Fee"

from Movie m join price p

on m.price\_code = p.price\_code

group by movie\_genre;



1. **Write a query to display the movie title, movie year, and the movie cost divided by the price rental fee for each movie that has a price to determine the number of rentals it will take to break even on the purchase of the movie (result shown in Figure P7.86).**

Figure P7.86 Breakeven rentals



select movie\_title, movie\_year, round(movie\_cost / price\_rentfee , 2)"Breakeven Rentals"

from movie join price

on movie.price\_code = price.price\_code;



1. **Write a query to display the movie title and movie year for all movies that have a price code (result shown in Figure P7.87).**

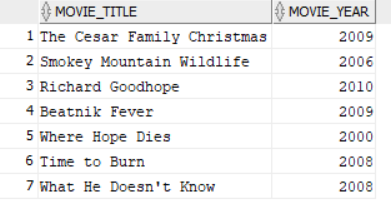
P7.87 Movies with a price



select movie\_title, movie\_year

from movie

where price\_code is not null;



1. **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 (result shown in Figure P7.88).**

Figure P7.88 Movies costs within a range

FigP7-88-Movie-Costs-within-a-Range

select movie\_title, movie\_year, movie\_cost

from movie

where movie\_cost between 44.99 and 49.99;



1. **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 (result shown in Figure P7.89).**

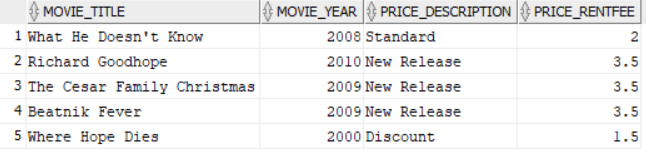
Figure P7.89 Movies with specific genres



select movie\_title, movie\_year, price\_description, price\_rentfee

from movie join price using(price\_code)

where movie.movie\_genre in ('FAMILY','COMEDY','DRAMA');



1. **Write a query to display the movie number, movie title, and movie year for all movies that do not have a video (result shown in Figure P7.90).**

Figure P7.90 Movies without videos

FigP7-90-Movies-without-Videos

select distinct movie.movie\_num, movie\_title, movie\_year

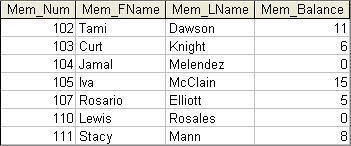
FROM movie

where price\_code is null;



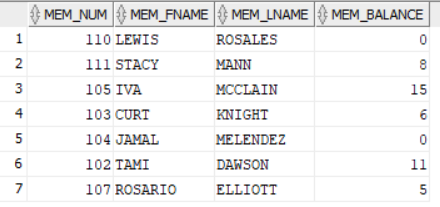
1. **Write a query to display the membership number, first name, last name, and balance of the memberships that have a rental (result shown in Figure P7.91).**

Figure P7.91 Balances of memberships with rentals



select distinct mem\_num, mem\_fname, mem\_lname, mem\_balance

from membership join rental using(mem\_num);



1. **Write a query to display the minimum balance, maximum balance, and average balance for memberships that have a rental (result shown in Figure P7.92).**

Figure P7.92 Minimum, maximum, and average balances

FigP7-71-Min-Max-Avg-of-Balances

select min(m.mem\_balance) "Minimum Balance", max(m.mem\_balance) "Maximum Balance", round(sum(distinct m.mem\_balance)/count(distinct r.mem\_num),2) "Average Balance"

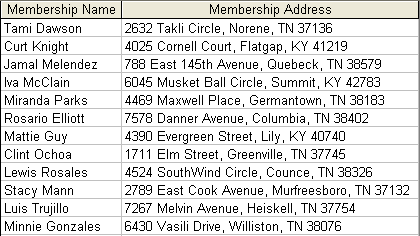
from membership m join rental r

on m.mem\_num = r.mem\_num;



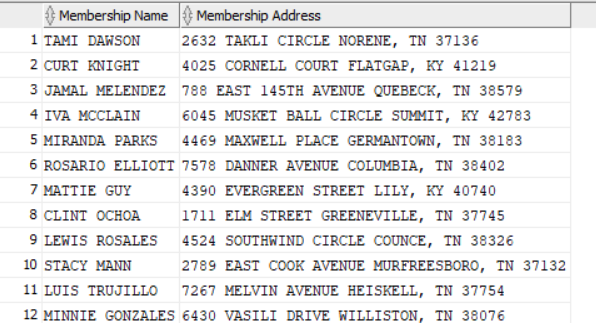
1. **Write a query to display the membership name (concatenate the first name and last name with a space between them into a single column), membership address (concatenate the street, city, state, and zip codes into a single column with spaces (result shown in Figure P7.93).**

Figure P7.93 Concatenated membership data



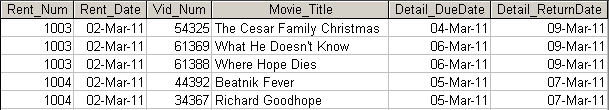
select mem\_fname || ' ' || mem\_lname "Membership Name", mem\_street || ' ' || mem\_city || ', ' || mem\_state || ' ' || mem\_ZIP "Membership Address"

from membership;



1. **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 th** **All Reports/User Defined Reportse results by rental number and movie title (result shown in Figure P7.94).**

Figure P7.94 Late video returns



select r.rent\_num, r.rent\_date, v.vid\_num, m.movie\_title, d.detail\_duedate, d.detail\_returnDate

from video v, rental r , detailrental d, movie m

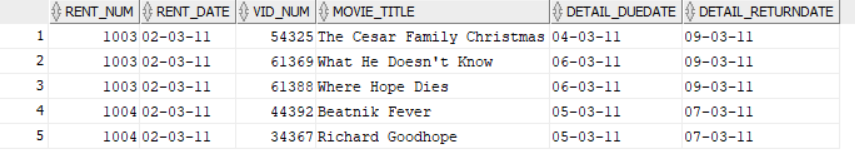
where m.movie\_num = v.movie\_num

and v.vid\_num = d.vid\_num

and r.rent\_num = d.rent\_num

and d.detail\_returndate > d.detail\_duedate

order by 1,4;



1. **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. (Result shown in Figure P7.95.)**

Figure P7.95 Number of days late



select rent\_num, rent\_date, vid\_num, movie\_title, detail\_duedate, detail\_returnDate, detail\_fee, EXTRACT( year from detail\_returndate) - EXTRACT( year from detail\_duedate) "Days Past due"

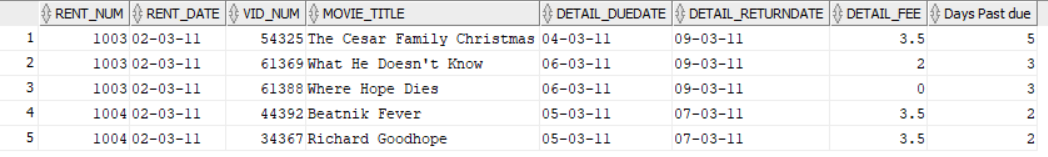
from movie join video using(movie\_num)

join detailrental using(vid\_num)

join rental using(rent\_num)

where detail\_returndate > detail\_duedate

order by 1,4;



1. **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 (result shown in Figure P7.96).**

Figure P7.96 Actual rental fees charged



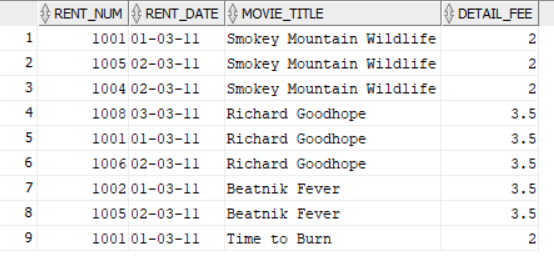
select rent\_num, rent\_date, movie\_title, detail\_fee

from rental join detailrental using(rent\_num)

join video using(vid\_num)

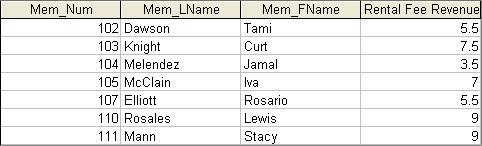
join movie using(movie\_num)

where detailrental.detail\_returndate <= detailrental.detail\_duedate;



1. **Write a query to display the membership number, last name, and total rental fees earned from that membership (result shown in Figure P7.97). 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.**

Figure P7.97 Total rental fees paid by membership



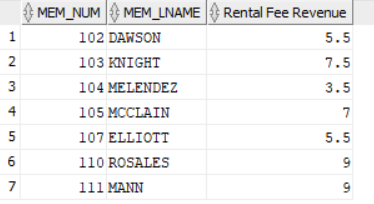
select mem\_num,mem\_lname, sum(detail\_fee) "Rental Fee Revenue"

from membership join rental using(mem\_num)

join detailrental using(rent\_num)

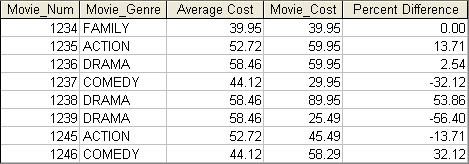
group by(mem\_num, mem\_lname)

order by 1;



1. **Write a query to display the movie number, movie genre, average movie cost of movies in that genre, movie cost of that individual movie, and the percentage difference between the average movie cost and the individual movie cost (result shown in Figure P7.98). Note: the percentage difference is calculated as the cost of the individual movie minus the average cost of movies in that genre, divided by the average cost of movies in that genre multiplied by 100. For example, if the average cost of movies in the “Family” genre is $25, if a given Family movie cost $26, then the calculation would be ((26 – 25) / 25 \* 100), which would work out to be 4.00%. This indicates that this movie costs 4% more than the average Family movie.**

Figure P7.98 Movie difference from genre average



select l.movie\_num, m.movie\_genre, round(avg(m.movie\_cost),2) "Average Cost", l.movie\_cost, round((l.movie\_cost - avg(m.movie\_cost))/avg(m.movie\_cost) \* 100 ,2) "Percent difference"

from movie m full join movie l

on l.movie\_genre = m.movie\_genre

group by(l.Movie\_num, m.movie\_genre, l.movie\_cost)

order by 1;

