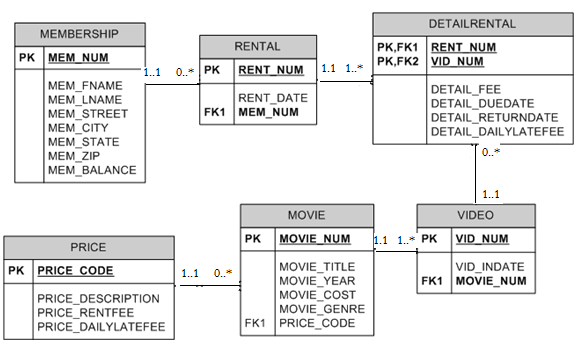
Assignment 3 SQL Programming

Problem 1 through 30

RentalA Video is a movie rental store. It needs a database system to track the rental of movies to its members. RentalAVideo can own several copies (VIDEO) of each movie (MOVIE). For example, the store may have 10 copies of the movie “Gone with the Wind”. “Gone with the Wind” would be o ne MOVIE and each copy would be a VIDEO. A rental transaction (RENTAL) involves one or more videos being rented to a member (MEMBERSHIP). A video can be rented many times over its lifetime, therefore, there is a M:N relationship between RENTAL and VIDEO. DETAILRENTAL is the bridge table to resolve this relationship. The complete RDM is provided below.

RentalAVideo ERD (Figure 1)



1. Write the SQL code to create the table structures for the entities shown in Figure 1. The structures should contain the specified attributes. Use data types that would be appropriate for the data that will need to be stored in each attribute. Enforce primary key and foreign key constraints as indicated by the ERD.

CREATE TABLE membership (

mem\_num VARCHAR(30) NOT NULL,

mem\_fname VARCHAR(30),

mem\_lname VARCHAR(30),

mem\_street VARCHAR(30),

mem\_city VARCHAR(30),

mem\_state VARCHAR(30),

mem\_zip VARCHAR(30),

mem\_balance NUMBER,

CONSTRAINT pk\_membership PRIMARY KEY (mem\_num)

);

CREATE TABLE rental (

rent\_num VARCHAR(20) NOT NULL,

rant\_date DATE,

mem\_num VARCHAR(30),

CONSTRAINT pk\_rental PRIMARY KEY (rent\_num),

CONSTRAINT fk\_rental FOREIGN KEY (mem\_num) REFERENCES membership (mem\_num)

);

CREATE TABLE detailrental (

rent\_num VARCHAR(20) NOT NULL,

vid\_num VARCHAR(20) NOT NULL,

detail\_fee NUMBER,

detail\_duedate DATE,

detail\_returndate DATE,

detail\_dailylatefee NUMBER,

CONSTRAINT pk\_detailrental PRIMARY KEY (rent\_num, vid\_num),

CONSTRAINT fk\_detailrental FOREIGN KEY (rent\_num) REFERENCES rental (rent\_num)

);

CREATE TABLE price (

price\_code VARCHAR(30) NOT NULL,

price\_description VARCHAR(30),

price\_rentfee NUMBER,

price\_dailylatefee NUMBER,

CONSTRAINT pk\_price PRIMARY KEY (price\_code)

);

CREATE TABLE movie (

movie\_num VARCHAR(20) NOT NULL,

movie\_title VARCHAR(20),

movie\_year DATE,

movie\_cost NUMBER,

movie\_genre VARCHAR(20),

price\_code VARCHAR(30),

CONSTRAINT pk\_movie PRIMARY KEY (movie\_num),

CONSTRAINT fk\_movie FOREIGN KEY (price\_code) REFERENCES price (price\_code)

);

CREATE TABLE video (

vid\_num VARCHAR(20) NOT NULL,

vid\_indate DATE,

movie\_num VARCHAR(20),

CONSTRAINT pk\_video PRIMARY KEY (vid\_num),

CONSTRAINT fk\_video FOREIGN KEY (movie\_num) REFERENCES movie (movie\_num)

);

ALTER TABLE detailrental

ADD CONSTRAINT fk2\_detailrental

FOREIGN KEY (vid\_num) REFERENCES video (vid\_num)

1. The following tables provide a very small portion of the data that will be kept in the database. This data needs to be inserted into the database for testing purposes. Write the INSERT commands necessary to place the following data in the tables that were created in problem 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **MEMBERSHIP** | | | | | | | |
| **Mem\_ Num** | **Mem\_ Fname** | **Mem\_ Lname** | **Mem\_Street** | **Mem\_City** | **Mem\_ State** | **Mem\_Zip** | **Mem\_ Balance** |
| 102 | Tami | Dawson | 2632 Takli Circle | Norene | TN | 37136 | 11 |
| 103 | Curt | Knight | 4025 Cornell Court | Flatgap | KY | 41219 | 6 |
| 104 | Jamal | Melendez | 788 East 145th Avenue | Quebeck | TN | 38579 | 0 |
| 105 | Iva | Mcclain | 6045 Musket Ball Circle | Summit | KY | 42783 | 15 |
| 106 | Miranda | Parks | 4469 Maxwell Place | Germantown | TN | 38183 | 0 |
| 107 | Rosario | Elliott | 7578 Danner Avenue | Columbia | TN | 38402 | 5 |
| 108 | Mattie | Guy | 4390 Evergreen Street | Lily | KY | 40740 | 0 |
| 109 | Clint | Ochoa | 1711 Elm Street | Greeneville | TN | 37745 | 10 |
| 110 | Lewis | Rosales | 4524 Southwind Circle | Counce | TN | 38326 | 0 |
| 111 | Stacy | Mann | 2789 East Cook Avenue | Murfreesboro | TN | 37132 | 8 |
| 112 | Luis | Trujillo | 7267 Melvin Avenue | Heiskell | TN | 37754 | 3 |
| 113 | Minnie | Gonzales | 6430 Vasili Drive | Williston | TN | 38076 | 0 |

|  |  |  |
| --- | --- | --- |
| **RENTAL** | | |
| **Rent\_Num** | **Rent\_Date** | **Mem\_Num** |
| 1001 | 01-MAR-09 | 103 |
| 1002 | 01-MAR-09 | 105 |
| 1003 | 02-MAR-09 | 102 |
| 1004 | 02-MAR-09 | 110 |
| 1005 | 02-MAR-09 | 111 |
| 1006 | 02-MAR-09 | 107 |
| 1007 | 02-MAR-09 | 104 |
| 1008 | 03-MAR-09 | 105 |
| 1009 | 03-MAR-09 | 111 |

|  |  |  |
| --- | --- | --- |
| **VIDEO** | | |
| **Vid\_Num** | **Vid\_Indate** | **Movie\_Num** |
| 54321 | 18-JUN-08 | 1234 |
| 54324 | 18-JUN-08 | 1234 |
| 54325 | 18-JUN-08 | 1234 |
| 34341 | 22-JAN-07 | 1235 |
| 34342 | 22-JAN-07 | 1235 |
| 34366 | 02-MAR-09 | 1236 |
| 34367 | 02-MAR-09 | 1236 |
| 34368 | 02-MAR-09 | 1236 |
| 34369 | 02-MAR-09 | 1236 |
| 44392 | 21-OCT-08 | 1237 |
| 44397 | 21-OCT-08 | 1237 |
| 59237 | 14-FEB-09 | 1237 |
| 61388 | 25-JAN-07 | 1239 |
| 61353 | 28-JAN-06 | 1245 |
| 61354 | 28-JAN-06 | 1245 |
| 61367 | 30-JUL-08 | 1246 |
| 61369 | 30-JUL-08 | 1246 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DETAILRENTAL** | | | | | |
| **Rent\_Num** | **Vid\_Num** | **Detail\_Fee** | **Detail\_Duedate** | **Detail\_Returndate** | **Detail\_Dailylatefee** |
| 1001 | 34342 | 2 | 04-MAR-09 | 02-MAR-09 | 1 |
| 1001 | 61353 | 2 | 04-MAR-09 | 03-MAR-09 | 1 |
| 1002 | 59237 | 3.5 | 04-MAR-09 | 04-MAR-09 | 3 |
| 1003 | 54325 | 3.5 | 04-MAR-09 | 09-MAR-09 | 3 |
| 1003 | 61369 | 2 | 06-MAR-09 | 09-MAR-09 | 1 |
| 1003 | 61388 | 0 | 06-MAR-09 | 09-MAR-09 | 1 |
| 1004 | 44392 | 3.5 | 05-MAR-09 | 07-MAR-09 | 3 |
| 1004 | 34367 | 3.5 | 05-MAR-09 | 07-MAR-09 | 3 |
| 1004 | 34341 | 2 | 07-MAR-09 | 07-MAR-09 | 1 |
| 1005 | 34342 | 2 | 07-MAR-09 | 05-MAR-09 | 1 |
| 1005 | 44397 | 3.5 | 05-MAR-09 | 05-MAR-09 | 3 |
| 1006 | 34366 | 3.5 | 05-MAR-09 | 04-MAR-09 | 3 |
| 1006 | 61367 | 2 | 07-MAR-09 |  | 1 |
| 1007 | 34368 | 3.5 | 05-MAR-09 |  | 3 |
| 1008 | 34369 | 3.5 | 05-MAR-09 | 05-MAR-09 | 3 |
| 1009 | 54324 | 3.5 | 05-MAR-09 |  | 3 |
| 1001 | 34366 | 3.5 | 04-MAR-09 | 02-MAR-09 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MOVIE** | | | | | |
| **Movie\_Num** | **Movie\_Name** | **Movie\_Year** | **Movie\_Cost** | **Movie\_Genre** | **Price\_Code** |
| 1234 | The Cesar Family Christmas | 2007 | 39.95 | FAMILY | 2 |
| 1235 | Smokey Mountain Wildlife | 2004 | 59.95 | ACTION | 1 |
| 1236 | Richard Goodhope | 2008 | 59.95 | DRAMA | 2 |
| 1237 | Beatnik Fever | 2007 | 29.95 | COMEDY | 2 |
| 1238 | Constant Companion | 2008 | 89.95 | DRAMA | 2 |
| 1239 | Where Hope Dies | 1998 | 25.49 | DRAMA | 3 |
| 1245 | Time to Burn | 2005 | 45.49 | ACTION | 1 |
| 1246 | What He Doesn't Know | 2006 | 58.29 | COMEDY | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRICE** | | | |
| **Price\_Code** | **Price\_Description** | **Price\_Rentfee** | **Price\_Dailylatefee** |
| 1 | Standard | 2 | 1 |
| 2 | New Release | 3.5 | 3 |
| 3 | Discount | 1.5 | 1 |
| 4 | Weekly Special | 1 | .5 |

INSERT INTO membership VALUES ( "102", "Tami", "Dawson", "2632 Takli Circle", "Norene", "TN", "37136", 11)

INSERT INTO rental VALUES ( "1003", #02-MAR-09#, "102")

INSERT INTO detailrental VALUES ( "1003", "61369", 2, #09-MAR-09#, #02-MAR-09#, 1)

INSERT INTO price VALUES ( "1", "Standard", 2, 1)

INSERT INTO movie VALUES ( "1246", "What he doesn't know", ‘2006’, 58.29, "COMEDY", "1")

INSERT INTO video VALUES ( "61369", #30-JUL-08#, "1246")

For questions 3– 31, use the tables that were created in Problem 1 and the data that was loaded into those tables in Problem 2.

1. Write the SQL command to change the movie year for movie number 1245 to 2006.

UPDATE movie

SET movie\_year = "2006"

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.

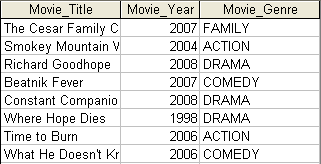
UPDATE price

SET Price\_Rentfee = Price\_Rentfee + 0.50

1. Write a query to display the movie title, movie year, and movie genre for all movies.

SELECT Movie\_Name, Movie\_Year, Movie\_Genre

FROM movie

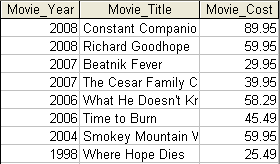


1. Write a query to display the movie year, movie title, and movie cost sorted by movie year in descending order.

SELECT Movie\_Year, Movie\_Name, Movie\_Cost

FROM movie

ORDER BY Movie\_Year DESC

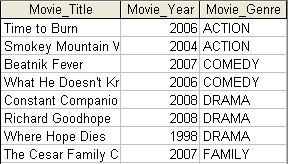


1. Write a query to display the movie title, movie year, and movie genre for all movies sorted by movie genre in ascending order, then sorted by movie year in descending order within genre.

SELECT Movie\_Name, Movie\_Year, Movie\_Genre

FROM movie

ORDER BY Movie\_Genre, Movie\_Year 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”.

SELECT Movie\_Num, Movie\_Name, Price\_Code

FROM movie

WHERE Movie\_Name 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.

SELECT Movie\_Name, Movie\_Year, Movie\_Cost

FROM movie

WHERE Movie\_Name LIKE "\*hope\*"

ORDER BY Movie\_Name



1. Write a query to display the movie title, movie year, and movie genre for all action movies.

SELECT Movie\_Name, Movie\_Year, Movie\_Genre

FROM movie

WHERE Movie\_Genre = "ACTION"

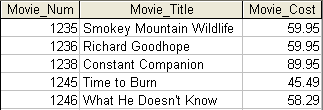


1. Write a query to display the movie number, movie title, and movie cost for all movies with a cost greater than $40.

SELECT Movie\_Num, Movie\_Name, Movie\_Cost

FROM movie

WHERE Movie\_Cost > "40"



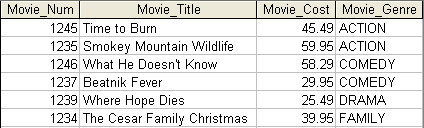
1. Write a query to display the movie number, movie title, movie cost, and movie genre for movies that are either action or comedy movies or movies that have a cost that is less than $50. Sort the results in ascending order by genre.

SELECT Movie\_Num, Movie\_Name, Movie\_Cost, Movie\_Genre

FROM movie

WHERE Movie\_Genre = "ACTION" OR Movie\_Genre = "COMEDY" OR Movie\_Cost < "50"

ORDER BY Movie\_Genre

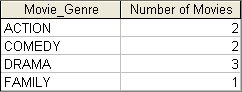


1. Write a query to display the movie genre and the number of movies in each genre.

SELECT Movie\_Genre, COUNT(Movie\_Genre) AS Number\_of\_Movies

FROM movie

GROUP BY Movie\_Genre



1. Write a query to display the average cost of all of the movies.

SELECT AVG(Movie\_Cost) AS Average\_Movie\_Cost

FROM movie



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

SELECT Movie\_Genre, Round(AVG(Movie\_Cost), 2) AS Average\_Cost

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.

SELECT m.Movie\_Name, m.Movie\_Genre, p.Price\_Description, p.Price\_Rentfee

FROM movie m, price p

WHERE m.Price\_Code = p.Price\_Code

|  |  |  |  |
| --- | --- | --- | --- |
| Movie\_Title | Movie\_Genre | Price\_Descriptioin | Price\_Rentfee |
| What He Doesn't Know | COMEDY | Standard | 2.5 |
| The Cesar Family Christmas | FAMILY | New Release | 4 |
| Richard Goodhope | DRAMA | New Release | 4 |
| Beatnik Fever | COMEDY | New Release | 4 |
| Constant Companion | DRAMA | New Release | 4 |
| Smokey Mountain Wildlife | ACTION | Discount | 2 |
| Where Hope Dies | DRAMA | Discount | 2 |
| Time to Burn | ACTION | Discount | 2 |

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

SELECT Movie\_Genre, ROUND(AVG(Price\_Rentfee), 2) AS Average\_Rental\_Fee

FROM movie, price

WHERE movie.Price\_Code = price.Price\_Code

GROUP BY Movie\_Genre

|  |  |
| --- | --- |
| Movie\_Genre | AVERAGE\_RENTAL\_FEE |
| FAMILY | 4 |
| COMEDY | 3.25 |
| ACTION | 2 |
| DRAMA | 3.33 |

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.

SELECT Movie\_Name, Movie\_Year, ROUND(Movie\_Cost/Price\_Rentfee, 2) AS Breakeven\_Rentals

FROM movie, price

WHERE movie.Price\_Code = price.Price\_Code

|  |  |  |
| --- | --- | --- |
| MOVIE\_TITLE | MOVIE\_YEAR | BREAKEVEN\_RENTALS |
| What He Doesn't Know | 2006 | 23.32 |
| Richard Goodhope | 2008 | 14.99 |
| The Cesar Family | 2007 | 9.99 |
| Beatnik Fever | 2007 | 7.49 |
| Constant Companion | 2008 | 22.49 |
| Where Hope Dies | 1998 | 12.75 |
| Time to Burn | 2006 | 22.75 |
| Smokey Mountain Wildlife | 2004 | 29.98 |

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

SELECT Movie\_Name, Movie\_Year

FROM movie

ORDER BY Movie\_Name

| **MOVIE\_TITLE** | **MOVIE\_YEAR** |
| --- | --- |
| Beatnik Fever | 2007 |
| Constant Companion | 2008 |
| Richard Goodhope | 2008 |
| Smokey Mountain Wildlife | 2004 |
| Time to Burn | 2005 |
| What He Doesn't Know | 2006 |
| Where Hope Dies | 1998 |
| The Cesar Family Christmas | 2007 |

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.

SELECT Movie\_Name, Movie\_Year, Movie\_Cost

FROM movie

WHERE Movie\_Cost > "44.99" AND Movie\_Cost < "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 .

SELECT Movie\_Name, Movie\_Year, Price\_Description, Price\_Rentfee, Movie\_Genre

FROM movie, price

WHERE movie.Price\_Code = price.Price\_Code AND Movie\_Genre IN ("FAMILY", "COMEDY", "DRAMA")

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Movie\_Title | Movie\_Year | Price\_Desciption | Price\_Rentfee | Movie\_Genre |
| The Cesar Family | 2007 | New Release | 4 | FAMILY |
| Richard Goodhope | 2008 | New Release | 4 | DRAMA |
| Beatnik Fever | 2007 | New Release | 4 | COMEDY |
| Constant Companion | 2008 | New Release | 4 | DRAMA |
| Where Hope Dies | 1998 | Discount | 2 | DRAMA |
| What He Doesn't Know | 2006 | Standard | 2.5 | COMEDY |

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

SELECT movie.Movie\_Num, Movie\_Name, Movie\_Year

FROM movie

WHERE movie.Movie\_Num NOT IN (SELECT video.Movie\_Num FROM video)

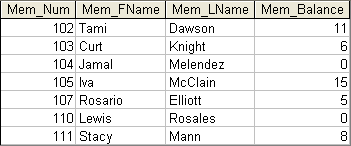


1. Write a query to display the membership number, first name, last name, and balance of the memberships that have a rental.

SELECT m.Mem\_Num, Mem\_Fname, Mem\_Lname, Mem\_Balance

FROM (SELECT DISTINCT Mem\_Num FROM rental) r, membership m

WHERE r.Mem\_Num=m.Mem\_Num



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

SELECT MIN(Mem\_Balance) AS minimum\_balance, MAX(Mem\_Balance) AS maximum\_balance, ROUND(AVG(Mem\_Balance),2) AS average\_balance

FROM (SELECT DISTINCT Mem\_Num FROM rental) r, membership m

WHERE r.Mem\_Num=m.Mem\_Num

|  |  |  |
| --- | --- | --- |
| MINIMUM\_BALANCE | MAXIMUM\_BALANCE | AVERAGE\_BALANCE |
| 0 | 15 | 6.43 |

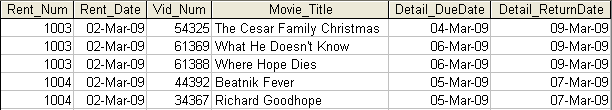
1. Write a query to display the rental number, rental date, video number, movie title, due date, and return date for all videos that were returned after the due date. Sort the results by rental number and movie title.

SELECT r.Rent\_Num, Rent\_Date, v.Vid\_Num, Movie\_Name, Detail\_Duedate, Detail\_Returndate

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

WHERE r.Rent\_Num=d.Rent\_Num AND d.Vid\_Num=v.Vid\_Num AND v.Movie\_Num=m.Movie\_Num AND Detail\_Returndate > Detail\_Duedate

ORDER BY r.Rent\_Num, Movie\_Name



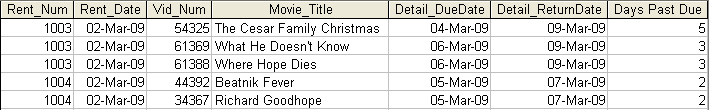
1. Write a query to display the rental number, rental date, video number, movie title, due date, return date, detail fee, and number of days past the due date that the video was returned for each video that was returned after the due date. Sort the results by rental number and movie title.

SELECT r.Rent\_Num, FORMAT(Rent\_Date, "DD-MMM-YY"), v.Vid\_Num, Movie\_Name, FORMAT(Detail\_Duedate, "DD-MMM-YY"), FORMAT(Detail\_Returndate, "DD-MMM-YY"), (Detail\_Returndate - Detail\_Duedate) AS Days\_Past\_Due

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

WHERE r.Rent\_Num=d.Rent\_Num AND d.Vid\_Num=v.Vid\_Num AND v.Movie\_Num=m.Movie\_Num AND Detail\_Returndate > Detail\_Duedate

ORDER BY r.Rent\_Num, Movie\_Name

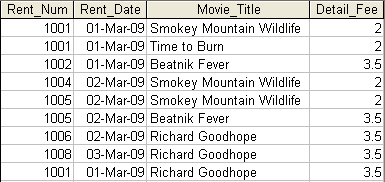


1. Write a query to display the rental number, rental date, movie title, and detail fee for each movie that was returned on or before the due date.

SELECT r.Rent\_Num, FORMAT(Rent\_Date, "DD-MMM-YY"), Movie\_Name, Detail\_Fee

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

WHERE r.Rent\_Num=d.Rent\_Num AND d.Vid\_Num=v.Vid\_Num AND v.Movie\_Num=m.Movie\_Num AND (Detail\_Returndate < Detail\_Duedate OR Detail\_Returndate = Detail\_Duedate)



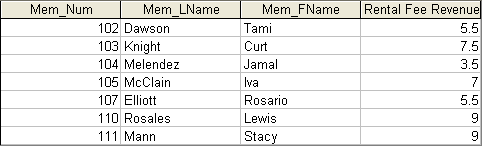
1. Write a query to display the membership number, last name, and total rental fees earned from that membership. The total rental fee is the sum of all of the detail fees (without the late fees) from all movies that the membership has rented.

SELECT m.Mem\_Num, Mem\_Lname, Mem\_Fname, SUM(Detail\_Fee) AS Rental\_Fee\_Revenue

FROM membership m, rental r, detailrental d

WHERE m.Mem\_Num=r.Mem\_Num AND r.Rent\_Num=d.Rent\_Num

GROUP BY m.Mem\_Num, Mem\_Lname, Mem\_Fname



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 26). 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.

SELECT m.Movie\_Num, m.Movie\_Genre, x AS Average\_Cost, Movie\_Cost, ROUND(((Movie\_Cost - x) / x)\* 100 ,2) AS Percentage\_Difference

FROM movie m, (SELECT Movie\_Genre, ROUND(AVG(Movie\_Cost),2) AS x FROM movie GROUP BY Movie\_Genre) s

WHERE s.Movie\_Genre = m.Movie\_Genre

