CREATE TABLE PRICE (

PRICE\_CODE int PRIMARY KEY,

PRICE\_DESCRIPTION VARCHAR(20) NOT NULL ,

PRICE\_RENTFEE NUMBER(8,2) CHECK (PRICE\_RENTFEE >= 0),

PRICE\_DAILYLATEFEE NUMBER(8,2) CHECK (PRICE\_DAILYLATEFEE >= 0)

);

CREATE TABLE MOVIE (

MOVIE\_NUM int PRIMARY KEY,

MOVIE\_TITLE VARCHAR(75) NOT NULL,

MOVIE\_YEAR int CHECK (MOVIE\_YEAR > 1900),

MOVIE\_COST NUMBER(8,2),

MOVIE\_GENRE VARCHAR(50),

PRICE\_CODE int CONSTRAINT MOVIE\_PRICE\_CODE\_FK REFERENCES PRICE

);

CREATE TABLE VIDEO (

VID\_NUM int PRIMARY KEY,

VID\_INDATE DATE,

MOVIE\_NUM int CONSTRAINT VIDEO\_MOVIE\_NUM\_FK REFERENCES MOVIE

);

CREATE TABLE MEMBERSHIP (

MEM\_NUM int PRIMARY KEY,

MEM\_FNAME VARCHAR(30) NOT NULL,

MEM\_LNAME VARCHAR(30) NOT NULL,

MEM\_STREET VARCHAR(120),

MEM\_CITY VARCHAR(50),

MEM\_STATE CHAR(2),

MEM\_ZIP CHAR(5),

MEM\_BALANCE int

);

CREATE TABLE RENTAL (

RENT\_NUM int PRIMARY KEY,

RENT\_DATE DATE DEFAULT SYSDATE, --CURRENT\_DATE,

MEM\_NUM int CONSTRAINT RENTAL\_MEM\_NUM\_FK REFERENCES MEMBERSHIP

);

CREATE TABLE DETAILRENTAL (

RENT\_NUM int CONSTRAINT DETAIL\_RENT\_NUM\_FK REFERENCES RENTAL,

VID\_NUM int CONSTRAINT DETAIL\_VID\_NUM\_FK REFERENCES VIDEO,

DETAIL\_FEE NUMBER(8,2),

DETAIL\_DUEDATE DATE,

DETAIL\_RETURNDATE DATE,

DETAIL\_DAILYLATEFEE INT,

CONSTRAINT DETAIL\_RENT\_VID\_PK PRIMARY KEY (RENT\_NUM, VID\_NUM)

);

-- PRICE

INSERT INTO PRICE VALUES (1, 'Standard', 2, 1);

INSERT INTO PRICE VALUES (2, 'New Release', 3.5, 3);

INSERT INTO PRICE VALUES (3, 'Discount', 1.5, 1);

INSERT INTO PRICE VALUES (4, 'Weekly Special', 1, .5);

--MOVIE

INSERT INTO MOVIE VALUES (1234, 'The Cesar Family Christmas', 2009, 39.95, 'FAMILY', 2);

INSERT INTO MOVIE VALUES (1235, 'Smokey Mountain Wildlife', 2006, 59.95, 'ACTION', 1);

INSERT INTO MOVIE VALUES (1236, 'Richard Goodhope', 2010, 59.95, 'DRAMA', 2);

INSERT INTO MOVIE VALUES (1237, 'Beatnik Fever', 2009, 29.95, 'COMEDY', 2);

INSERT INTO MOVIE VALUES (1238, 'Constant Companion', 2010, 89.95, 'DRAMA', NULL);

INSERT INTO MOVIE VALUES (1239, 'Where Hope Dies', 2000, 25.49, 'DRAMA', 3);

INSERT INTO MOVIE VALUES (1245, 'Time to Burn', 2007, 45.49, 'ACTION', 1);

INSERT INTO MOVIE VALUES (1246, 'What He Doesn''t Know', 2008, 58.29, 'COMEDY', 1);

-- VIDEO:

INSERT INTO VIDEO VALUES (34341, '22-JAN-09', 1235);

INSERT INTO VIDEO VALUES (34342, '22-JAN-09', 1235);

INSERT INTO VIDEO VALUES (34366, '02-MAR-11', 1236);

INSERT INTO VIDEO VALUES (34367, '02-MAR-11', 1236);

INSERT INTO VIDEO VALUES (34368, '02-MAR-11', 1236);

INSERT INTO VIDEO VALUES (34369, '02-MAR-11', 1236);

INSERT INTO VIDEO VALUES (44392, '21-OCT-10', 1237);

INSERT INTO VIDEO VALUES (44397, '21-OCT-10', 1237);

INSERT INTO VIDEO VALUES (54321, '18-JUN-10', 1234);

INSERT INTO VIDEO VALUES (54324, '18-JUN-10', 1234);

INSERT INTO VIDEO VALUES (54325, '18-JUN-10', 1234);

INSERT INTO VIDEO VALUES (59237, '14-FEB-11', 1237);

INSERT INTO VIDEO VALUES (61353, '28-JAN-08', 1245);

INSERT INTO VIDEO VALUES (61354, '28-JAN-08', 1245);

INSERT INTO VIDEO VALUES (61367, '30-JUL-10', 1246);

INSERT INTO VIDEO VALUES (61369, '30-JUL-10', 1246);

INSERT INTO VIDEO VALUES (61388, '25-JAN-09', 1239);

--MEMBERSHIP:

INSERT INTO MEMBERSHIP VALUES (102, 'TAMI', 'DAWSON', '2632 TAKLI CIRCLE', 'NORENE', 'TN', '37136', 11);

INSERT INTO MEMBERSHIP VALUES (103, 'CURT', 'KNIGHT', '4025 CORNELL COURT', 'FLATGAP', 'KY', '41219', 6);

INSERT INTO MEMBERSHIP VALUES (104, 'JAMAL', 'MELENDEZ', '788 EAST 145TH AVENUE', 'QUEBECK', 'TN', '38579', 0);

INSERT INTO MEMBERSHIP VALUES (105, 'IVA', 'MCCLAIN', '6045 MUSKET BALL CIRCLE', 'SUMMIT', 'KY', '42783', 15);

INSERT INTO MEMBERSHIP VALUES (106, 'MIRANDA', 'PARKS', '4469 MAXWELL PLACE', 'GERMANTOWN', 'TN', '38183', 0);

INSERT INTO MEMBERSHIP VALUES (107, 'ROSARIO', 'ELLIOTT', '7578 DANNER AVENUE', 'COLUMBIA', 'TN', '38402', 5);

INSERT INTO MEMBERSHIP VALUES (108, 'MATTIE', 'GUY', '4390 EVERGREEN STREET', 'LILY', 'KY', '40740', 0);

INSERT INTO MEMBERSHIP VALUES (109, 'CLINT', 'OCHOA', '1711 ELM STREET', 'GREENEVILLE', 'TN', '37745', 10);

INSERT INTO MEMBERSHIP VALUES (110, 'LEWIS', 'ROSALES', '4524 SOUTHWIND CIRCLE', 'COUNCE', 'TN', '38326', 0);

INSERT INTO MEMBERSHIP VALUES (111, 'STACY', 'MANN', '2789 EAST COOK AVENUE', 'MURFREESBORO', 'TN', '37132', 8);

INSERT INTO MEMBERSHIP VALUES (112, 'LUIS', 'TRUJILLO', '7267 MELVIN AVENUE', 'HEISKELL', 'TN', '37754', 3);

INSERT INTO MEMBERSHIP VALUES (113, 'MINNIE', 'GONZALES', '6430 VASILI DRIVE', 'WILLISTON', 'TN', '38076', 0);

--RENTAL:

INSERT INTO RENTAL VALUES (1001, '01-MAR-11', 103);

INSERT INTO RENTAL VALUES (1002, '01-MAR-11', 105);

INSERT INTO RENTAL VALUES (1003, '02-MAR-11', 102);

INSERT INTO RENTAL VALUES (1004, '02-MAR-11', 110);

INSERT INTO RENTAL VALUES (1005, '02-MAR-11', 111);

INSERT INTO RENTAL VALUES (1006, '02-MAR-11', 107);

INSERT INTO RENTAL VALUES (1007, '02-MAR-11', 104);

INSERT INTO RENTAL VALUES (1008, '03-MAR-11', 105);

INSERT INTO RENTAL VALUES (1009, '03-MAR-11', 111);

--DETAILRENTAL:

INSERT INTO DETAILRENTAL VALUES (1001, 34342, 2, '04-MAR-11', '02-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1001, 34366, 3.5, '04-MAR-11', '02-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1001, 61353, 2, '04-MAR-11', '03-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1002, 59237, 3.5, '04-MAR-11', '04-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1003, 54325, 3.5, '04-MAR-11', '09-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1003, 61369, 2, '06-MAR-11', '09-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1003, 61388, 0, '06-MAR-11', '09-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1004, 34341, 2, '07-MAR-11', '07-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1004, 34367, 3.5, '05-MAR-11', '07-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1004, 44392, 3.5, '05-MAR-11', '07-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1005, 34342, 2, '07-MAR-11', '05-MAR-11', 1);

INSERT INTO DETAILRENTAL VALUES (1005, 44397, 3.5, '05-MAR-11', '05-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1006, 34366, 3.5, '05-MAR-11', '04-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1006, 61367, 2, '07-MAR-11', NULL, 1);

INSERT INTO DETAILRENTAL VALUES (1007, 34368, 3.5, '05-MAR-11', NULL, 3);

INSERT INTO DETAILRENTAL VALUES (1008, 34369, 3.5, '05-MAR-11', '05-MAR-11', 3);

INSERT INTO DETAILRENTAL VALUES (1009, 54324, 3.5, '05-MAR-11', NULL, 3);

-- 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;

-- 2. 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';

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

UPDATE Price

SET price\_rentfee = price\_rentfee + 0.5;

-- 4. WRITE A QUERY TO display the movie title, movie YEAR, AND movie genre FOR ALL movies (RESULT shown IN Figure P7.72).

SELECT movie\_title,

movie\_year,

movie\_genre

FROM Movie;

-- 5. 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).

SELECT movie\_year,

movie\_title,

movie\_cost

FROM Movie

ORDER BY movie\_year DESC;

-- 6. 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).

SELECT movie\_title,

movie\_year,

movie\_genre

FROM Movie

ORDER BY movie\_genre, movie\_year DESC;

-- 7. 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).

SELECT movie\_num,

movie\_title,

movie\_cost

FROM Movie

WHERE movie\_title LIKE 'R%';

-- 8. 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).

SELECT movie\_title,

movie\_year,

movie\_cost

FROM Movie

WHERE movie\_title LIKE '%hope%'

OR movie\_title LIKE '%Hope%'

ORDER BY movie\_title;

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

SELECT movie\_title,

movie\_year,

movie\_genre

FROM Movie

WHERE movie\_genre = 'ACTION';

-- 10. 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).

SELECT movie\_num,

movie\_title,

movie\_cost

FROM Movie

WHERE movie\_cost > 40;

-- 11. 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.)

SELECT movie\_num,

movie\_title,

movie\_cost,

movie\_genre

FROM Movie

WHERE movie\_cost < 50

AND movie\_genre IN ('ACTION','COMEDY')

ORDER BY movie\_genre;

-- 12. 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).

SELECT movie\_num,

movie\_title || ' (' || movie\_year || ') ' || movie\_genre AS "DESCRIPTION"

FROM Movie;

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

SELECT movie\_genre,

COUNT(movie\_num)

FROM Movie

GROUP BY movie\_genre

ORDER BY movie\_genre;

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

SELECT AVG(movie\_cost)

FROM Movie;

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

SELECT movie\_genre,

ROUND(AVG(movie\_cost))

FROM Movie

GROUP BY movie\_genre

ORDER BY movie\_genre;

-- 16. 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).

SELECT movie\_title,

movie\_genre,

price\_description,

price\_rentfee

FROM Movie M

INNER JOIN Price P

ON M.price\_code = P.price\_code

WHERE m.price\_code IS NOT NULL

ORDER BY price\_description DESC;

-- 17. 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).

SELECT movie\_genre,

AVG(price\_rentfee) AS "Average Rental FEE"

FROM Movie, Price

WHERE movie\_cost IS NOT NULL

GROUP BY movie\_genre;

-- 18. 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).

SELECT m.MOVIE\_TITLE "Movie\_title" ,

m.MOVIE\_YEAR "Movie\_Year",

Round( m.MOVIE\_COST / p.PRICE\_RENTFEE,2 ) "Breakeven Rentals"

FROM Movie m, Price p

where m.PRICE\_CODE = p.PRICE\_CODE;

-- 19. 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).

SELECT MOVIE\_TITLE "Movie\_title",

MOVIE\_YEAR "Movie\_Year"

from Movie

where PRICE\_CODE is not null;

-- 20. 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).

SELECT MOVIE\_TITLE "Movie\_title",

MOVIE\_YEAR "Movie\_Year" ,

MOVIE\_COST "Movie\_Cost"

from Movie

where MOVIE\_COST between 44.99 and 49.99;

-- 21. 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).

SELECT m.MOVIE\_TITLE "Movie\_title" ,

m.MOVIE\_YEAR "Movie\_Year",

p.PRICE\_DESCRIPTION "Price\_Description",

p.PRICE\_RENTFEE "Price\_RentFee" ,

m.MOVIE\_GENRE "Movie\_Genre"

from Movie m , Price p

where m.MOVIE\_GENRE in ('FAMILY', 'COMEDY' , 'DRAMA');

-- 22. 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).

SELECT m.MOVIE\_NUM "Movie\_Num",

m.MOVIE\_TITLE "Movie\_Title",

m.MOVIE\_YEAR "Movie\_Year"

from Movie m full outer join Video v

on m.MOVIE\_NUM = v.MOVIE\_NUM

where v.VID\_NUM is null;

-- 23. 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).

SELECT m.MEM\_NUM "Mem\_Num",

m.MEM\_FNAME "Mem\_FName",

m.MEM\_LNAME "Mem\_LName",

m.MEM\_Balance "Mem\_Balance", r.RENT\_NUM

FROM Membership m join Rental r

ON m.MEM\_NUM = r.MEM\_NUM

WHERE r.RENT\_NUM IS NOT NULL;

-- 24. 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).

SELECT MIN(MEM\_BALANCE) AS "Minimum Balance",

MAX(MEM\_BALANCE) AS "Maximum Balance",

ROUND(AVG(MEM\_BALANCE),2) AS "Average Balance"

FROM MEMBERSHIP

WHERE MEM\_NUM IN (SELECT MEM\_NUM FROM RENTAL);

-- 25. 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).

SELECT MEM\_FNAME || ' ' || MEM\_LNAME AS "Membership Name",

MEM\_STREET || ' ' || MEM\_CITY || ', ' || MEM\_STATE || ' ' || MEM\_ZIP

AS "Membership Address"

FROM MEMBERSHIP;

-- 26. 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 (result shown in Figure P7.94).

SELECT RENTAL.RENT\_NUM, RENT\_DATE, VIDEO.VID\_NUM, MOVIE\_TITLE,

DETAIL\_DUEDATE, DETAIL\_RETURNDATE

FROM RENTAL, DETAILRENTAL, VIDEO, MOVIE

WHERE RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

AND DETAILRENTAL.VID\_NUM = VIDEO.VID\_NUM

AND VIDEO.MOVIE\_NUM = MOVIE.MOVIE\_NUM

ORDER BY RENTAL.RENT\_NUM, MOVIE\_TITLE;

-- 27. 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.)

SELECT RENTAL.RENT\_NUM, RENT\_DATE, VIDEO.VID\_NUM, MOVIE\_TITLE,

DETAIL\_DUEDATE, DETAIL\_RETURNDATE,

DETAIL\_RETURNDATE - DETAIL\_DUEDATE AS "Days Past Due"

FROM RENTAL, DETAILRENTAL, VIDEO, MOVIE

WHERE RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

AND DETAILRENTAL.VID\_NUM = VIDEO.VID\_NUM

AND VIDEO.MOVIE\_NUM = MOVIE.MOVIE\_NUM

AND DETAIL\_RETURNDATE > DETAIL\_DUEDATE

ORDER BY RENTAL.RENT\_NUM, MOVIE\_TITLE;

-- 28. 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).

SELECT RENTAL.RENT\_NUM, RENT\_DATE, MOVIE\_TITLE, DETAIL\_FEE

FROM RENTAL, DETAILRENTAL, VIDEO, MOVIE

WHERE RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

AND DETAILRENTAL.VID\_NUM = VIDEO.VID\_NUM

AND VIDEO.MOVIE\_NUM = MOVIE.MOVIE\_NUM

AND DETAIL\_RETURNDATE <= DETAIL\_DUEDATE;

-- 29. 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.

SELECT MEMBERSHIP.MEM\_NUM, MEM\_LNAME, MEM\_FNAME,

SUM(DETAILRENTAL.DETAIL\_FEE) AS "Rental Fee Revenue"

FROM MEMBERSHIP, RENTAL, DETAILRENTAL

WHERE MEMBERSHIP.MEM\_NUM = RENTAL.MEM\_NUM

AND RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

GROUP BY MEMBERSHIP.MEM\_NUM, MEM\_LNAME, MEM\_FNAME;

-- 30. 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.

SELECT MOVIE\_NUM, M.MOVIE\_GENRE, AVGCOST AS "Average Cost",

MOVIE\_COST,

(MOVIE\_COST - AVGCOST)/AVGCOST \* 100 AS "Percent Difference"

FROM MOVIE M, (SELECT MOVIE\_GENRE, AVG(MOVIE\_COST) AS AVGCOST

FROM MOVIE

GROUP BY MOVIE\_GENRE) S

WHERE M.MOVIE\_GENRE = S.MOVIE\_GENRE;