

Assignment 1 SQL Fundamentals

Total points: 50

This assignment should be completed individually. For each problem, submit your SQL statement and a screen shot of the SQL results in a single Word document or pdf file. Submit the file via eLearning.

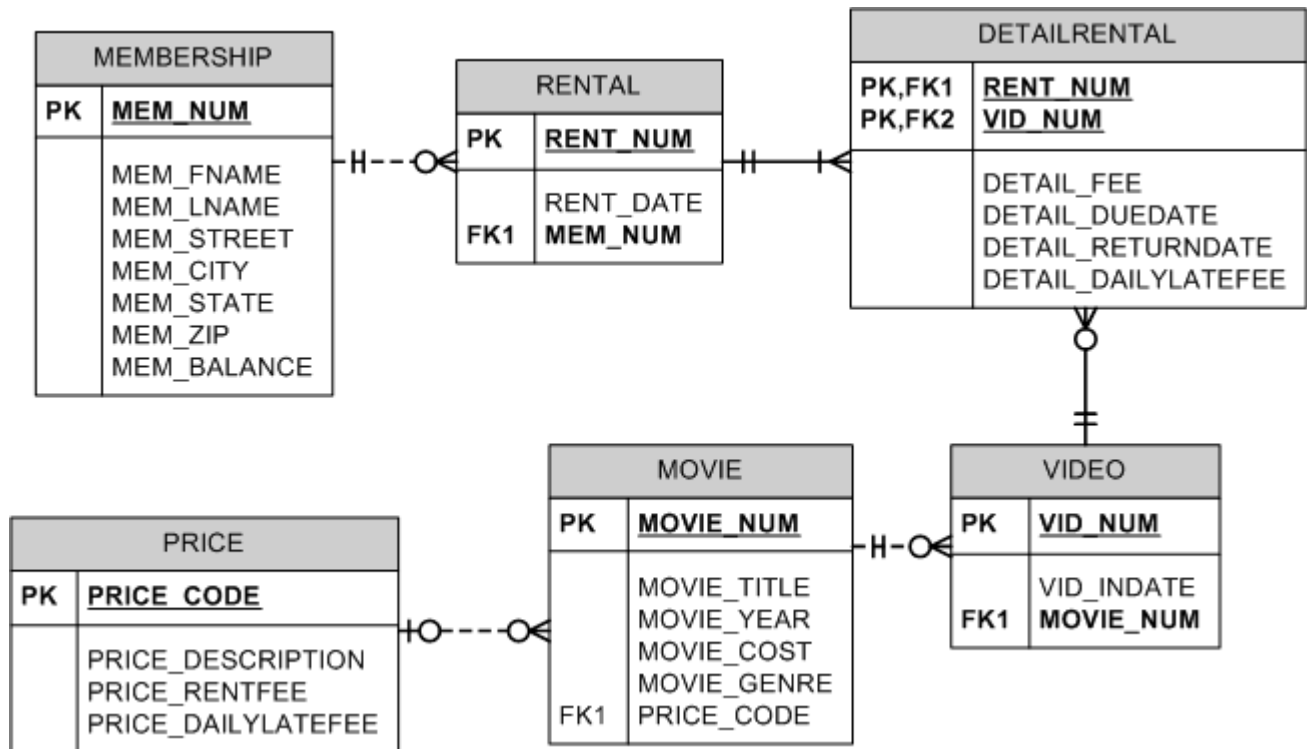
I recommend creating a new user and workspace, log in as that user and load the database script ourvideo.sql (provided in this week's assignment folder).

Before you attempt to write any SQL queries, familiarize yourself with the database structure and data. I have provided a relational diagram and sample data for this database.

Write queries to address each of the problems below. Submit both the SQL statements and the screen prints of the outputs from Oracle.

OurVideo is a small movie rental company with a single store. OurVideo needs a database system to track the rental of movies to its members. OurVideo can own several copies (VIDEO) of each movie (MOVIE). For example, the store may have 10 copies of the movie "Twist in the Wind". "Twist in the Wind" would be one 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 ERD is provided in the Figure below.

OurVideo ERD



1.A. Explain the correct sequence in which the tables should be created and why it should be in that order. (NOTE: this is based on referential integrity constraints). (2pts)

PRICE → MOVIE → VIDEO →
MEMBERSHIP → RENTAL → DETAILRENTAL

“Because the PRODUCT table contains a foreign key that references the VENDOR table, create the VENDOR table first. (In fact, the M side of a relationship always references the 1 side. Therefore, in a 1:M relationship, you must always create the table for the 1 side first.)”—from textbook page 230

According to the above instruction, we can find that the DETAILRENTAL has 2 foreign keys (FK). One FK references the VIDEO table, the VIDEO has a FK references the MOVIE table, and the MOVIE has a FK references the PRICE table. The other FK references the RENTAL table, and the RENTAL table has FK references the MEMBERSHIP table. So we create the tables according to the sequence displayed in the very beginning.

1.B. Write the SQL code to create the table structures for the RENTAL AND DETAILRENTAL entities shown in the Figure. The structures should contain the attributes specified in the ERD. 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. (4 pts)

```
CREATE TABLE RENTAL (  
  RENT_NUM    NUMBER(8,0),  
  RENT_DATE   DATE DEFAULT SYSDATE,  
  MEM_NUM     NUMBER(8,0),  
  PRIMARY KEY (RENT_NUM),  
  FOREIGN KEY (MEM_NUM) REFERENCES MEMBERSHIP  
);
```

```
CREATE TABLE RENTAL (
  RENT_NUM          NUMBER(8, 0),
  RENT_DATE         DATE DEFAULT SYSDATE,
  MEM_NUM          NUMBER(8, 0),
  PRIMARY KEY (RENT_NUM),
  FOREIGN KEY (MEM_NUM) REFERENCES MEMBERSHIP
);
```

[Results](#)
[Explain](#)
[Describe](#)
[Saved SQL](#)
[History](#)

Table created

0.01 seconds

```
CREATE TABLE DETAILRENTAL (
  RENT_NUM          NUMBER(8,0),
  VID_NUM           NUMBER(8,0),
  DETAIL_FEE        NUMBER(9,2) DEFAULT 0.00 NOT NULL,
  DETAIL_DUEDATE    DATE,
  DETAIL_RETURNDATE DATE,
  DETAIL_DAILYLATEFEE NUMBER(9,2) DEFAULT 0.00 NOT NULL,
  PRIMARY KEY (RENT_NUM, VID_NUM),
  FOREIGN KEY (RENT_NUM) REFERENCES RENTAL,
  FOREIGN KEY (VID_NUM) REFERENCES VIDEO
);
```



```
CREATE TABLE DETAILRENTAL (
  RENT_NUM          NUMBER(8, 0),
  VID_NUM           NUMBER(8, 0),
  DETAIL_FEE        NUMBER(9, 2) DEFAULT 0.00 NOT NULL,
  DETAIL_DUEDATE    DATE,
  DETAIL_RETURNDATE DATE,
  DETAIL_DAILYLATEFEE NUMBER(9, 2) DEFAULT 0.00 NOT NULL,
  PRIMARY KEY (RENT_NUM, VID_NUM),
  FOREIGN KEY (RENT_NUM) REFERENCES RENTAL,
  FOREIGN KEY (VID_NUM) REFERENCES VIDEO
);
```

Table created.

0.11 seconds

2.A. Based on the referential integrity constraints, identify the correct sequence in which to insert data into the tables (list the order). (2 pts)

Because the MOVIE table uses its PRICE_CODE to reference the PRICE table's PRICE_CODE, an integrity violation will occur if those PRICE table PRICE_CODE values don't yet exist. Therefore, we need to enter the PRICE rows before the MOVIE rows.

And because the same reason, the correct sequence to insert data into the tables is:

PRICE → MOVIE → VIDEO →

MEMBERSHIP → RENTAL → DETAILRENTAL

2.B. Write the INSERT commands necessary to place the following data in the tables that were created in problem 1.B. (5 pts)

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

```

INSERT INTO RENTAL VALUES (1001, TO_DATE('01-MAR-09','DD-MM-YY'), 103);
INSERT INTO RENTAL VALUES (1002, TO_DATE('01-MAR-09','DD-MM-YY'), 105);
INSERT INTO RENTAL VALUES (1003, TO_DATE('02-MAR-09','DD-MM-YY'), 102);
INSERT INTO RENTAL VALUES (1004, TO_DATE('02-MAR-09','DD-MM-YY'), 110);
INSERT INTO RENTAL VALUES (1005, TO_DATE('02-MAR-09','DD-MM-YY'), 111);
INSERT INTO RENTAL VALUES (1006, TO_DATE('02-MAR-09','DD-MM-YY'), 107);
INSERT INTO RENTAL VALUES (1007, TO_DATE('02-MAR-09','DD-MM-YY'), 104);
INSERT INTO RENTAL VALUES (1008, TO_DATE('03-MAR-09','DD-MM-YY'), 105);
INSERT INTO RENTAL VALUES (1009, TO_DATE('03-MAR-09','DD-MM-YY'), 111);

```

ORACLE® Application Express

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)
[Home](#) > [SQL Workshop](#) > [SQL Commands](#)
☒ Autocommit

Rows

10 ▼



Save

Run

```
INSERT INTO RENTAL VALUES (1001, TO_DATE('01-MAR-09','DD-MM-YY'), 103);
INSERT INTO RENTAL VALUES (1002, TO_DATE('01-MAR-09','DD-MM-YY'), 105);
INSERT INTO RENTAL VALUES (1003, TO_DATE('02-MAR-09','DD-MM-YY'), 102);
INSERT INTO RENTAL VALUES (1004, TO_DATE('02-MAR-09','DD-MM-YY'), 110);
INSERT INTO RENTAL VALUES (1005, TO_DATE('02-MAR-09','DD-MM-YY'), 111);
INSERT INTO RENTAL VALUES (1006, TO_DATE('02-MAR-09','DD-MM-YY'), 107);
INSERT INTO RENTAL VALUES (1007, TO_DATE('02-MAR-09','DD-MM-YY'), 104);
INSERT INTO RENTAL VALUES (1008, TO_DATE('03-MAR-09','DD-MM-YY'), 105);
INSERT INTO RENTAL VALUES (1009, TO_DATE('03-MAR-09','DD-MM-YY'), 111);
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

1 row(s) inserted

0.00 seconds



DETAILRENTAL					
Rent_Num	Vid_Num	Detail_Fee	Detail_Duedate	Detail_Returndate	Detail_Dailyratefee
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

```

INSERT INTO DETAILRENTAL VALUES (1001, 34342, 2,
TO_DATE('02-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1001, 61353, 2,
TO_DATE('03-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1002, 59237, 3.5,
TO_DATE('04-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1003, 54325, 3.5,
TO_DATE('09-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1003, 61369, 2,
TO_DATE('09-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1003, 61388, 0,
TO_DATE('09-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1004, 44392, 3.5,
TO_DATE('07-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1004, 34367, 3.5,
TO_DATE('07-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1004, 34341, 2,
TO_DATE('07-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1005, 34342, 2,
TO_DATE('05-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1005, 44397, 3.5,
TO_DATE('05-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1006, 34366, 3.5,
TO_DATE('04-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1006, 61367, 2,
INSERT INTO DETAILRENTAL VALUES (1007, 34368, 3.5,
INSERT INTO DETAILRENTAL VALUES (1008, 34369, 3.5,
TO_DATE('05-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1009, 54324, 3.5,
INSERT INTO DETAILRENTAL VALUES (1001, 34366, 3.5,
TO_DATE('02-MAR-09','DD-MM-YY'), 3);
TO_DATE('04-MAR-09','DD-MM-YY'),
TO_DATE('04-MAR-09','DD-MM-YY'),
TO_DATE('04-MAR-09','DD-MM-YY'),
TO_DATE('04-MAR-09','DD-MM-YY'),
TO_DATE('06-MAR-09','DD-MM-YY'),
TO_DATE('06-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('07-MAR-09','DD-MM-YY'),
TO_DATE('07-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('07-MAR-09','DD-MM-YY'), NULL, 1);
TO_DATE('05-MAR-09','DD-MM-YY'), NULL, 3);
TO_DATE('05-MAR-09','DD-MM-YY'),
TO_DATE('05-MAR-09','DD-MM-YY'), NULL, 3);
TO_DATE('04-MAR-09','DD-MM-YY'),

```


☒ Autocommit Rows 10   Save Run

```

INSERT INTO DETAILRENTAL VALUES (1001, 34342, 2, TO_DATE('04-MAR-09','DD-MM-YY'), TO_DATE('02-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1001, 61353, 2, TO_DATE('04-MAR-09','DD-MM-YY'), TO_DATE('03-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1002, 59237, 3.5, TO_DATE('04-MAR-09','DD-MM-YY'), TO_DATE('04-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1003, 54325, 3.5, TO_DATE('04-MAR-09','DD-MM-YY'), TO_DATE('09-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1003, 61369, 2, TO_DATE('06-MAR-09','DD-MM-YY'), TO_DATE('09-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1003, 61388, 0, TO_DATE('06-MAR-09','DD-MM-YY'), TO_DATE('09-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1004, 44392, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), TO_DATE('07-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1004, 34367, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), TO_DATE('07-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1004, 34341, 2, TO_DATE('07-MAR-09','DD-MM-YY'), TO_DATE('07-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1005, 34342, 2, TO_DATE('07-MAR-09','DD-MM-YY'), TO_DATE('05-MAR-09','DD-MM-YY'), 1);
INSERT INTO DETAILRENTAL VALUES (1005, 44397, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), TO_DATE('05-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1006, 34366, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), TO_DATE('04-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1006, 61367, 2, TO_DATE('07-MAR-09','DD-MM-YY'), NULL, 1);
INSERT INTO DETAILRENTAL VALUES (1007, 34368, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), NULL, 3);
INSERT INTO DETAILRENTAL VALUES (1008, 34369, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), TO_DATE('05-MAR-09','DD-MM-YY'), 3);
INSERT INTO DETAILRENTAL VALUES (1009, 54324, 3.5, TO_DATE('05-MAR-09','DD-MM-YY'), NULL, 3);
INSERT INTO DETAILRENTAL VALUES (1001, 34366, 3.5, TO_DATE('04-MAR-09','DD-MM-YY'), TO_DATE('02-MAR-09','DD-MM-YY'), 3);
    
```

Results Explain Describe Saved SQL History



1 row(s) inserted.

0.00 seconds

2.C Write the SQL command to save the rows inserted in Problem 2.B. (1 pts)
COMMIT;

Home	Application Builder ▼	SQL Workshop ▼	Team Development ▼
------	-----------------------	----------------	--------------------

Home > SQL Workshop > SQL Commands

☐ Autocommit Rows   Save Run

COMMIT;
|

Results Explain Describe Saved SQL History

Statement processed.

0.04 seconds

3. How will you make sure that the value used in MEMBERSHIP.STATE is only TN, KY or TX? Show the syntax to add this check to the table and execute the statement. What happens when you try to insert a new row where the state is FL? Display the error message. (3 pts)

```
ALTER TABLE MEMBERSHIP MODIFY(MEM_STATE CHAR(2) CHECK(MEM_STATE IN ('TN','KY','TX')));
```

[Home](#)[Application Builder ▼](#)[SQL Workshop ▼](#)[Team Development ▼](#)[Home](#) > [SQL Workshop](#) > **SQL Commands**☒ Autocommit

Rows

[Save](#)[Run](#)

```
ALTER TABLE MEMBERSHIP MODIFY (MEM_STATE CHAR(2) CHECK (MEM_STATE IN ('TN','KY','TX')));|
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Table altered.

0.36 seconds

```
INSERT INTO MEMBERSHIP VALUES (114, 'MINNIE', 'GONZALES', '6430 VASILI DRIVE', 'WILLISTON', 'FL', '38076', 0);|
```

ORA-02290: check constraint (TEST.SYS_C007123) violated



0.01 seconds

4. Write a single SQL command to increase all price rental fee values by \$0.50. (2 pts)

```
UPDATE PRICE SET PRICE_RENTFEE = PRICE_RENTFEE + 0.5;
```

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)

[Home](#) > [SQL Workshop](#) > [SQL Commands](#)

☒ Autocommit
 Rows



```
UPDATE PRICE SET PRICE_RENTFEE = PRICE_RENTFEE + 0.5;
```

[Results](#)
[Explain](#)
[Describe](#)
[Saved SQL](#)
[History](#)

4 row(s) updated.

0.20 seconds

- 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 5). (2 pts)

Figure 5 Movies with "Hope" in the title

Movie_Title	Movie_Year	Movie_Cost
Richard Goodhope	2010	59.95
Where Hope Dies	2000	25.49

```

SELECT  MOVIE_TITLE, MOVIE_YEAR, MOVIE_COST
FROM    MOVIE
WHERE   UPPER(MOVIE_TITLE) LIKE '%HOPE%'
ORDER BY MOVIE_TITLE;

```

ORACLE® Application Express

Home
Application Builder ▼
SQL Workshop ▼
Team Development ▼

Home > SQL Workshop > SQL Commands

☒ Autocommit
Rows 10
Save
Run

```

SELECT      MOVIE_TITLE, MOVIE_YEAR, MOVIE_COST
FROM        MOVIE
WHERE       UPPER(MOVIE_TITLE) LIKE '%HOPE%'
ORDER BY   MOVIE_TITLE;

```

Results Explain Describe Saved SQL History

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

2 rows returned in 0.05 seconds [Download](#)

- 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. (sample result shown in Figure 6) (3pts)

Figure 6 Action or comedy movies costing less than \$50

Movie_Num	Movie_Title	Movie_Cost	Movie_Genre
1245	Time to Burn	45.49	ACTION
1235	Smokey Mountain Wildlife	59.95	ACTION
1246	What He Doesn't Know	58.29	COMEDY
1237	Beatnik Fever	29.95	COMEDY
1239	Where Hope Dies	25.49	DRAMA
1234	The Cesar Family Christmas	39.95	FAMILY

```
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;
```

ORACLE® Application Express

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)
[Home](#) > [SQL Workshop](#) > [SQL Commands](#)


Autocommit

Rows

10



Save

Run

```
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;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

MOVIE_NUM	MOVIE_TITLE	MOVIE_COST	MOVIE_GENRE
1245	Time to Burn	45.49	ACTION
1237	Beatnik Fever	29.95	COMEDY

2 rows returned in 0.02 seconds

[Download](#)

7. Write a query to display the movie genre and the number of movies in each genre (sample result shown in Figure 7). (3 pts)

Figure 7 Number of movies in genre

Movie_Genre	Number of Movies
ACTION	2
COMEDY	2
DRAMA	3
FAMILY	1

```
SELECT MOVIE_GENRE, COUNT(MOVIE_GENRE) AS "Number of Movies"
FROM MOVIE
GROUP BY MOVIE_GENRE
ORDER BY MOVIE_GENRE;
```



```

SELECT MOVIE_GENRE, COUNT(MOVIE_GENRE) AS "Number of Movies"
FROM MOVIE
GROUP BY MOVIE_GENRE
ORDER BY MOVIE_GENRE;

```

[Results](#)
[Explain](#)
[Describe](#)
[Saved SQL](#)
[History](#)

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

4 rows returned in 0.00 seconds [Download](#)

8. Write a query to display the average cost of all of the movies – the field should be titled “Average Movie Cost” (sample result shown in Figure 8). (2 pts)

Figure 8 Average movie cost



Average Movie Cost
51.1275

SELECT AVG(MOVIE_COST) AS "Average Movie Cost" FROM MOVIE;

ORACLE® Application Express

[Home](#) [Application Builder ▼](#) [SQL Workshop ▼](#) [Team Development ▼](#)

[Home](#) > [SQL Workshop](#) > [SQL Commands](#)

☒ Autocommit Rows   [Save](#) [Run](#)

SELECT AVG(MOVIE_COST) AS "Average Movie Cost" FROM MOVIE;|

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Average Movie Cost
51.1275

1 rows returned in 0.00 seconds [Download](#)

9. Write a query to display the movie title, movie year, and the movie cost divided by the price rental fee "Breakeven Rentals" 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 (sample result shown in Figure 9). (3 pts)

Figure 9 Breakeven rentals

Movie_Title	Movie_Year	Breakeven Rentals
What He Doesn't Know	2008	23.32
The Cesar Family Christmas	2009	9.99
Richard Goodhope	2010	14.99
Beatnik Fever	2009	7.49
Smokey Mountain Wildlife	2006	29.98
Where Hope Dies	2000	12.75
Time to Burn	2008	22.75

```

SELECT MOVIE_TITLE, MOVIE_YEAR, ROUND(MOVIE_COST/PRICE_RENTFEE,2) AS "Breakeven Rentals"
FROM MOVIE, PRICE
WHERE MOVIE.PRICE_CODE = PRICE.PRICE_CODE;

```

ORACLE® Application Express

[Home](#)[Application Builder ▼](#)[SQL Workshop ▼](#)[Team Development ▼](#)[Home](#) > [SQL Workshop](#) > **SQL Commands**☒ Autocommit

Rows

[Save](#)[Run](#)

```
SELECT MOVIE_TITLE, MOVIE_YEAR, ROUND(MOVIE_COST/PRICE_RENTFEE, 2) AS "Breakeven Rentals"
FROM MOVIE, PRICE
WHERE MOVIE.PRICE_CODE = PRICE.PRICE_CODE;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

MOVIE_TITLE	MOVIE_YEAR	Breakeven Rentals
The Cesar Family Christmas	2009	9.99
Smokey Mountain Wildlife	2006	23.98
Richard Goodhope	2010	14.99
Beatnik Fever	2009	7.49
Where Hope Dies	2000	12.75
Time to Burn	2007	18.2
What He Doesn't Know	2008	23.32

7 rows returned in 0.05 seconds

[Download](#)

10. 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 10). (2 pts)

Figure 10 Movies costs within a range


Movie_Title	Movie_Year	Movie_Cost
Time to Burn	2008	45.49

```
SELECT MOVIE_TITLE, MOVIE_YEAR, MOVIE_COST
FROM MOVIE
WHERE MOVIE_COST BETWEEN 44.99 AND 49.99;
```

ORACLE Application Express

[Home](#) [Application Builder ▼](#) [SQL Workshop ▼](#) [Team Development ▼](#)

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows   [Save](#) [Run](#)

```
SELECT MOVIE_TITLE, MOVIE_YEAR, MOVIE_COST
FROM MOVIE
WHERE MOVIE_COST BETWEEN 44.99 AND 49.99;
|
```

Results [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

MOVIE_TITLE	MOVIE_YEAR	MOVIE_COST
Time to Burn	2007	45.49

1 rows returned in 0.00 seconds [Download](#)

11. Write a query to display the movie title, movie year, price description, price rental fee for all movies that are in the genres Family, Comedy, or Drama (sample result shown in Figure 11). (3 pts)

Figure 11 Movies with specific genres

Movie_Title	Movie_Year	Price_Description	Price_RentFee	Movie_Genre
The Cesar Family Christmas	2009	New Release	4	FAMILY
Richard Goodhope	2010	New Release	4	DRAMA
Beatnik Fever	2009	New Release	4	COMEDY
Where Hope Dies	2000	Discount	2	DRAMA
What He Doesn't Know	2008	Standard	2.5	COMEDY

```
SELECT MOVIE_TITLE, 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');
```

```
SELECT MOVIE_TITLE, 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');
```

Results Explain Describe Saved SQL History

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

5 rows returned in 0.00 seconds [Download](#)

12. Write a query to display the movie number, movie title, and movie year for all movies that do not have a video (sample result shown in Figure 12). (3 pts)

Figure 12 Movies without videos

Movie_Num	Movie_Title	Movie_Year
1238	Constant Companion	2010

```
SELECT MOVIE_NUM, MOVIE_TITLE, MOVIE_YEAR
FROM MOVIE
WHERE MOVIE_NUM NOT IN
(SELECT DISTINCT MOVIE_NUM FROM VIDEO);
```

```
SELECT MOVIE_NUM, MOVIE_TITLE, MOVIE_YEAR
FROM MOVIE
WHERE MOVIE_NUM NOT IN
(SELECT DISTINCT MOVIE_NUM FROM VIDEO);
```

Results [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

MOVIE_NUM	MOVIE_TITLE	MOVIE_YEAR
1238	Constant Companion	2010

1 rows returned in 0.06 seconds [Download](#)

- 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 (sample result shown in Figure 13). (3 pts)

Figure 13 Concatenated membership data

Membership Name	Membership Address
Tami Dawson	2632 Takli Circle, Norene, TN 37136
Curt Knight	4025 Cornell Court, Flatgap, KY 41219
Jamal Melendez	788 East 145th Avenue, Quebeck, TN 38579
Iva McClain	6045 Musket Ball Circle, Summit, KY 42783
Miranda Parks	4469 Maxwell Place, Germantown, TN 38183
Rosario Elliott	7578 Danner Avenue, Columbia, TN 38402
Mattie Guy	4390 Evergreen Street, Lily, KY 40740
Clint Ochoa	1711 Elm Street, Greenville, TN 37745
Lewis Rosales	4524 SouthWind Circle, Counce, TN 38326
Stacy Mann	2789 East Cook Avenue, Murfreesboro, TN 37132
Luis Trujillo	7267 Melvin Avenue, Heiskell, TN 37754
Minnie Gonzales	6430 Vasili Drive, Williston, TN 38076

```

SELECT
MEM_FNAME || ' ' || MEM_LNAME AS "Membership Name",
MEM_STREET || ', ' || MEM_CITY || ', ' || MEM_STATE || ' ' || MEM_ZIP AS "Membership Address"
FROM MEMBERSHIP;

```

ORACLE® Application Express

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)
[Home](#) > [SQL Workshop](#) > **SQL Commands**
☒ Autocommit

Rows

15



Save

Run

```
SELECT
MEM_FNAME || ' ' || MEM_LNAME AS "Membership Name",
MEM_STREET || ' ' || MEM_CITY || ' ' || MEM_STATE || ' ' || MEM_ZIP AS "Membership Address"
FROM MEMBERSHIP;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Membership Name	Membership Address
TAMI DAWSON	2632 TAKLI CIRCLE, NORENE, TN 37136
CURT KNIGHT	4025 CORNELL COURT, FLATGAP, KY 41219
JAMAL MELENDEZ	788 EAST 145TH AVENUE, QUEBECK, TN 38579
IVA MCCLAIN	6045 MUSKET BALL CIRCLE, SUMMIT, KY 42783
MIRANDA PARKS	4469 MAXWELL PLACE, GERMANTOWN, TN 38183
ROSARIO ELLIOTT	7578 DANNER AVENUE, COLUMBIA, TN 38402
MATTIE GUY	4390 EVERGREEN STREET, LILY, KY 40740
CLINT OCHOA	1711 ELM STREET, GREENEVILLE, TN 37745
LEWIS ROSALES	4524 SOUTHWIND CIRCLE, COUNCE, TN 38326
STACY MANN	2789 EAST COOK AVENUE, MURFREESBORO, TN 37132
LUIS TRUJILLO	7267 MELVIN AVENUE, HEISKELL, TN 37754
MINNIE GONZALES	6430 VASILI DRIVE, WILLISTON, TN 38076

12 rows returned in 0.00 seconds

[Download](#)

14. 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. (sample result shown in Figure 14.) (3 pts)

Figure 14 Number of days late

Rent_Num	Rent_Date	Vid_Num	Movie_Title	Detail_DueDate	Detail_ReturnDate	Detail_Fee	Days Past Due
1003	02-Mar-11	54325	The Cesar Family Christmas	04-Mar-11	09-Mar-11	3.5	5
1003	02-Mar-11	61369	What He Doesn't Know	06-Mar-11	09-Mar-11	2	3
1003	02-Mar-11	61388	Where Hope Dies	06-Mar-11	09-Mar-11	0	3
1004	02-Mar-11	44392	Beatnik Fever	05-Mar-11	07-Mar-11	3.5	2
1004	02-Mar-11	34367	Richard Goodhope	05-Mar-11	07-Mar-11	3.5	2

```
alter session set nls_date_format = 'DD-MON-YY';
```

```
SELECT DETAILRENTAL.RENT_NUM, RENT_DATE, DETAILRENTAL.VID_NUM, MOVIE_TITLE,
DETAIL_DUEDATE, DETAIL_RETURNDATE, DETAIL_FEE,
DETAIL_RETURNDATE - DETAIL_DUEDATE AS "Days Past Due"
FROM DETAILRENTAL, RENTAL, VIDEO, MOVIE, PRICE
WHERE
DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
DETAILRENTAL.VID_NUM = VIDEO.VID_NUM AND
VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM AND
MOVIE.PRICE_CODE = PRICE.PRICE_CODE AND
DETAIL_RETURNDATE > DETAIL_DUEDATE
ORDER BY RENT_NUM, MOVIE_TITLE;
```

☐ Autocommit Rows 15   Save Run

```
alter session set nls_date_format = 'DD-MON-YY' ;

SELECT DETAILRENTAL.RENT_NUM, RENT_DATE, DETAILRENTAL.VID_NUM, MOVIE_TITLE, DETAIL_DUEDATE, DETAIL_RETURNDATE, DETAIL_FEE,
DETAIL_RETURNDATE - DETAIL_DUEDATE AS "Days Past Due"
FROM DETAILRENTAL, RENTAL, VIDEO, MOVIE, PRICE
WHERE
DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
DETAILRENTAL.VID_NUM = VIDEO.VID_NUM AND
VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM AND
MOVIE.PRICE_CODE = PRICE.PRICE_CODE AND
DETAIL_RETURNDATE > DETAIL_DUEDATE
ORDER BY RENT_NUM, MOVIE_TITLE;
```

Results Explain Describe Saved SQL History

RENT_NUM	RENT_DATE	VID_NUM	MOVIE_TITLE	DETAIL_DUEDATE	DETAIL_RETURNDATE	DETAIL_FEE	Days Past Due
1003	02-MAR-09	54325	The Cesar Family Christmas	04-MAR-09	09-MAR-09	3.5	5
1003	02-MAR-09	61369	What He Doesn't Know	06-MAR-09	09-MAR-09	2	3
1003	02-MAR-09	61388	Where Hope Dies	06-MAR-09	09-MAR-09	0	3
1004	02-MAR-09	44392	Beatnik Fever	05-MAR-09	07-MAR-09	3.5	2
1004	02-MAR-09	34367	Richard Goodhope	05-MAR-09	07-MAR-09	3.5	2

5 rows returned in 0.00 seconds

Method 2:

```
SELECT
DETAILRENTAL.RENT_NUM,
TO_CHAR(RENT_DATE,'DD-MON-YY') AS RENT_DATE,
DETAILRENTAL.VID_NUM,
MOVIE_TITLE,
TO_CHAR(DETAIL_DUEDATE,'DD-MON-YY') AS DETAIL_DUEDATE,
TO_CHAR(DETAIL_RETURNDATE,'DD-MON-YY') AS DETAIL_RETURNDATE,
DETAIL_FEE,
DETAIL_RETURNDATE - DETAIL_DUEDATE AS "Days Past Due"
FROM DETAILRENTAL, RENTAL, VIDEO, MOVIE, PRICE
WHERE
DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
DETAILRENTAL.VID_NUM = VIDEO.VID_NUM AND
VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM AND
MOVIE.PRICE_CODE = PRICE.PRICE_CODE AND
DETAIL_RETURNDATE > DETAIL_DUEDATE
ORDER BY RENT_NUM, MOVIE_TITLE;
```

Home
Application Builder ▼
SQL Workshop ▼
Team Development ▼
Administration ▼

Home > SQL Workshop > SQL Commands

☒ Autocommit
Rows: 15
Save Run

```

SELECT
DETAILRENTAL.RENT_NUM,
TO_CHAR(RENT_DATE,'DD-MON-YY') AS RENT_DATE,
DETAILRENTAL.VID_NUM,
MOVIE_TITLE,
TO_CHAR(DETAIL_DUEDATE,'DD-MON-YY') AS DETAIL_DUEDATE,
TO_CHAR(DETAIL_RETURNDATE,'DD-MON-YY') AS DETAIL_RETURNDATE,
DETAIL_FEE,
DETAIL_RETURNDATE - DETAIL_DUEDATE AS "Days Past Due"
FROM DETAILRENTAL, RENTAL, VIDEO, MOVIE, PRICE
WHERE
DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
DETAILRENTAL.VID_NUM = VIDEO.VID_NUM AND
VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM AND
MOVIE.PRICE_CODE = PRICE.PRICE_CODE AND
DETAIL_RETURNDATE > DETAIL_DUEDATE
ORDER BY RENT_NUM, MOVIE_TITLE;

```

Results Explain Describe Saved SQL History

RENT_NUM	RENT_DATE	VID_NUM	MOVIE_TITLE	DETAIL_DUEDATE	DETAIL_RETURNDATE	DETAIL_FEE	Days Past Due
1003	02-MAR-09	54325	The Cesar Family Christmas	04-MAR-09	09-MAR-09	3.5	5
1003	02-MAR-09	61369	What He Doesn't Know	06-MAR-09	09-MAR-09	2	3
1003	02-MAR-09	61388	Where Hope Dies	06-MAR-09	09-MAR-09	0	3
1004	02-MAR-09	44392	Beatnik Fever	05-MAR-09	07-MAR-09	3.5	2
1004	02-MAR-09	34367	Richard Goodhope	05-MAR-09	07-MAR-09	3.5	2

5 rows returned in 0.00 seconds [Download](#)

15. Write a query to display the membership number, last name, and total rental fees earned from that membership (result shown in Figure 15). 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. (4 pts)

Figure 15 Total rental fees paid by membership

Mem_Num	Mem_LName	Mem_FName	Rental Fee Revenue
102	Dawson	Tami	5.5
103	Knight	Curt	7.5
104	Melendez	Jamal	3.5
105	McClain	Iva	7
107	Elliott	Rosario	5.5
110	Rosales	Lewis	9
111	Mann	Stacy	9

```

SELECT MEMBERSHIP.MEM_NUM, MEM_LNAME, MEM_FNAME, SUM(DETAIL_FEE) AS "Rental Fee Revenue"
FROM MEMBERSHIP, DETAILRENTAL, RENTAL
WHERE DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
RENTAL.MEM_NUM = MEMBERSHIP.MEM_NUM
GROUP BY MEMBERSHIP.MEM_NUM, MEM_LNAME, MEM_FNAME
ORDER BY MEMBERSHIP.MEM_NUM;

```

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)
[Home](#) > [SQL Workshop](#) > **SQL Commands**
☒ Autocommit

Rows



Save

Run

```
SELECT MEMBERSHIP.MEM_NUM, MEM_LNAME, MEM_FNAME, SUM(DETAIL_FEE) AS "Rental Fee Revenue"
FROM MEMBERSHIP, DETAILRENTAL, RENTAL
WHERE DETAILRENTAL.RENT_NUM = RENTAL.RENT_NUM AND
      RENTAL.MEM_NUM = MEMBERSHIP.MEM_NUM
GROUP BY MEMBERSHIP.MEM_NUM, MEM_LNAME, MEM_FNAME
ORDER BY MEMBERSHIP.MEM_NUM;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

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

7 rows returned in 0.05 seconds

[Download](#)

Extra Credit:

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 EC). 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. (5 pts)

Figure EC Movie difference from genre average

Movie_Num	Movie_Genre	Average Cost	Movie_Cost	Percent Difference
1234	FAMILY	39.95	39.95	0.00
1235	ACTION	52.72	59.95	13.71
1236	DRAMA	58.46	59.95	2.54
1237	COMEDY	44.12	29.95	-32.12
1238	DRAMA	58.46	89.95	53.86
1239	DRAMA	58.46	25.49	-56.40
1245	ACTION	52.72	45.49	-13.71
1246	COMEDY	44.12	58.29	32.12

```

SELECT MOVIE_NUM, MOVIE.MOVIE_GENRE, B.M_AVG AS "Average Cost", MOVIE_COST,
ROUND(((MOVIE_COST - B.M_AVG)/B.M_AVG) * 100,2) AS "Percent Difference"
FROM MOVIE,
(
SELECT MOVIE_GENRE, ROUND(AVG(MOVIE_COST),2) AS M_AVG
FROM MOVIE
GROUP BY MOVIE_GENRE
)B
WHERE MOVIE.MOVIE_GENRE = B.MOVIE_GENRE
ORDER BY MOVIE.MOVIE_NUM;

```

ORACLE® Application Express

[Home](#)
[Application Builder ▼](#)
[SQL Workshop ▼](#)
[Team Development ▼](#)
[Home](#) > [SQL Workshop](#) > [SQL Commands](#)
☒ Autocommit

Rows

10



Save

Run

```
SELECT MOVIE_NUM, MOVIE.MOVIE_GENRE, B.M_AVG AS "Average Cost", MOVIE_COST,
ROUND(((MOVIE_COST - B.M_AVG)/B.M_AVG) * 100, 2) AS "Percent Difference"
FROM MOVIE,
(
SELECT MOVIE_GENRE, ROUND(AVG(MOVIE_COST), 2) AS M_AVG
FROM MOVIE
GROUP BY MOVIE_GENRE
)B
WHERE MOVIE.MOVIE_GENRE = B.MOVIE_GENRE
ORDER BY MOVIE.MOVIE_NUM;
|
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

MOVIE_NUM	MOVIE_GENRE	Average Cost	MOVIE_COST	Percent Difference
1234	FAMILY	39.95	39.95	0
1235	ACTION	52.72	59.95	13.71
1236	DRAMA	58.46	59.95	2.55
1237	COMEDY	44.12	29.95	-32.12
1238	DRAMA	58.46	89.95	53.87
1239	DRAMA	58.46	25.49	-56.4
1245	ACTION	52.72	45.49	-13.71
1246	COMEDY	44.12	58.29	32.12

8 rows returned in 0.00 seconds

[Download](#)