Assignment 5 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.

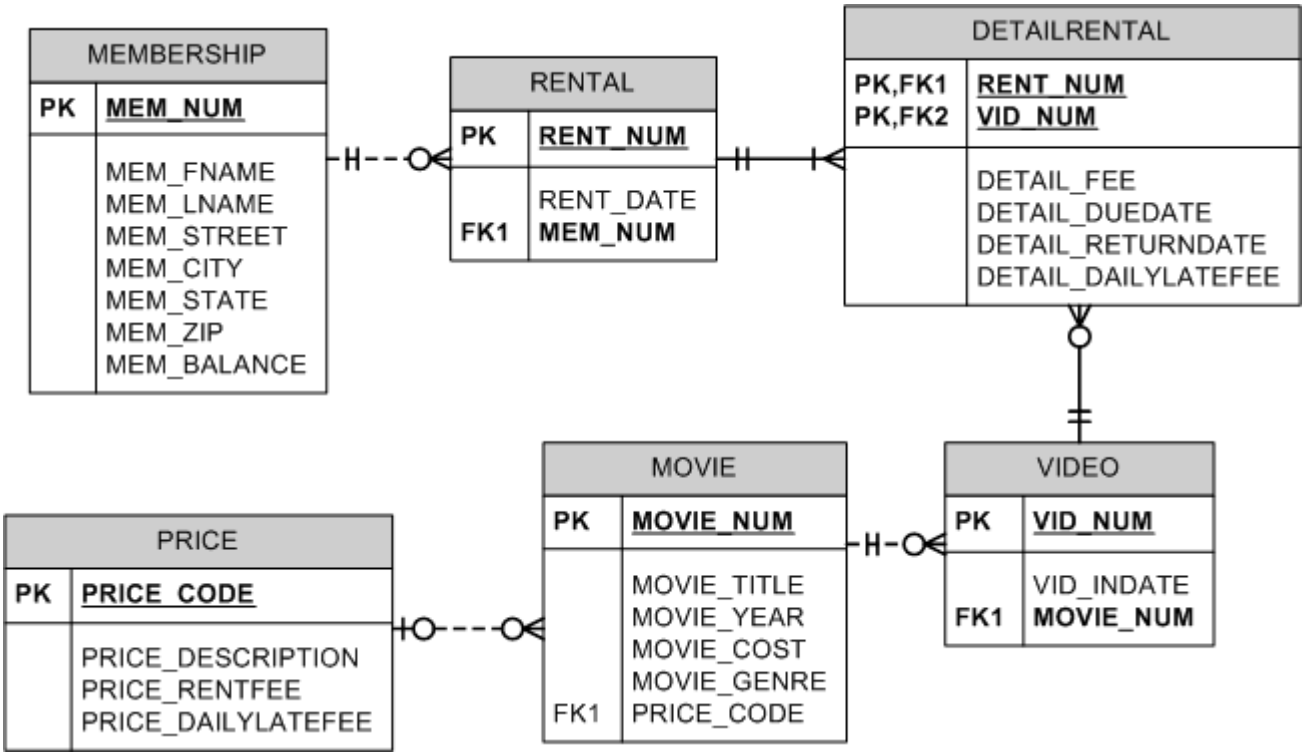
I recommend creating a new user and workspace, log in as that user and load the database script ourvideo_A2.sql (provided in this week's assignment folder).

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.

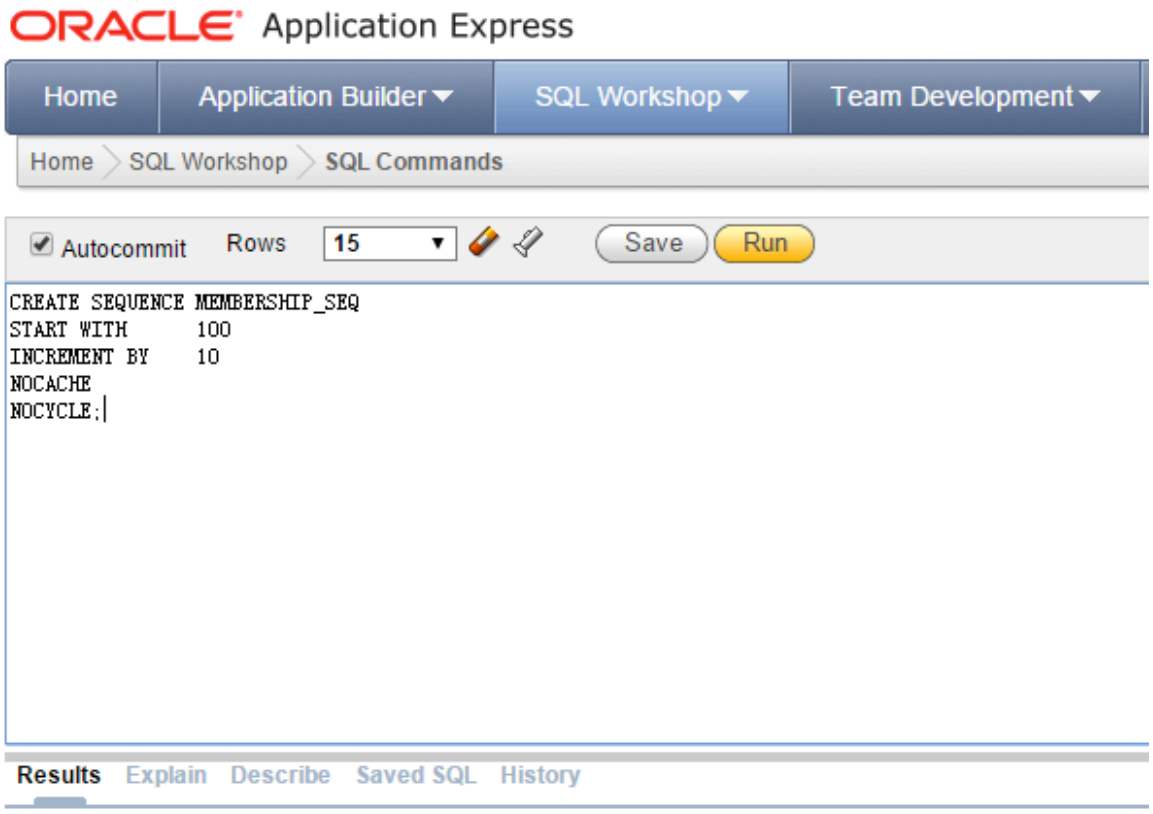
OurVideo ERD



Each question is worth 5 points.

1. Create sequences for the RENTAL and MEMBERSHIP tables. The MEMBERSHIP sequence should start with 100 and increment by 10 and the RENTAL sequence should start with 1 and increment by 1.

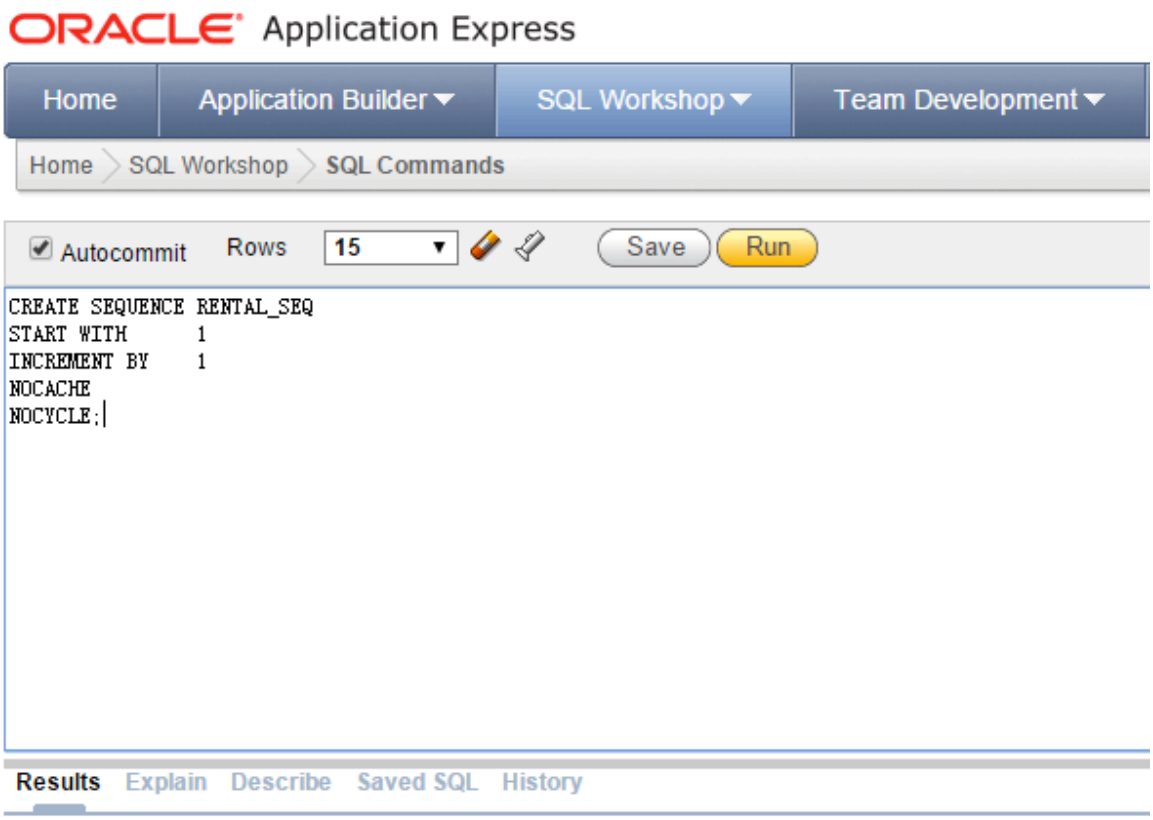
```
CREATE SEQUENCE MEMBERSHIP_SEQ
START WITH        100
INCREMENT BY    10
NOCACHE
NOCYCLE;
```




```
Sequence created.

0.07 seconds
```

```
CREATE SEQUENCE RENTAL_SEQ
START WITH        1
INCREMENT BY    1
NOCACHE
NOCYCLE;
```



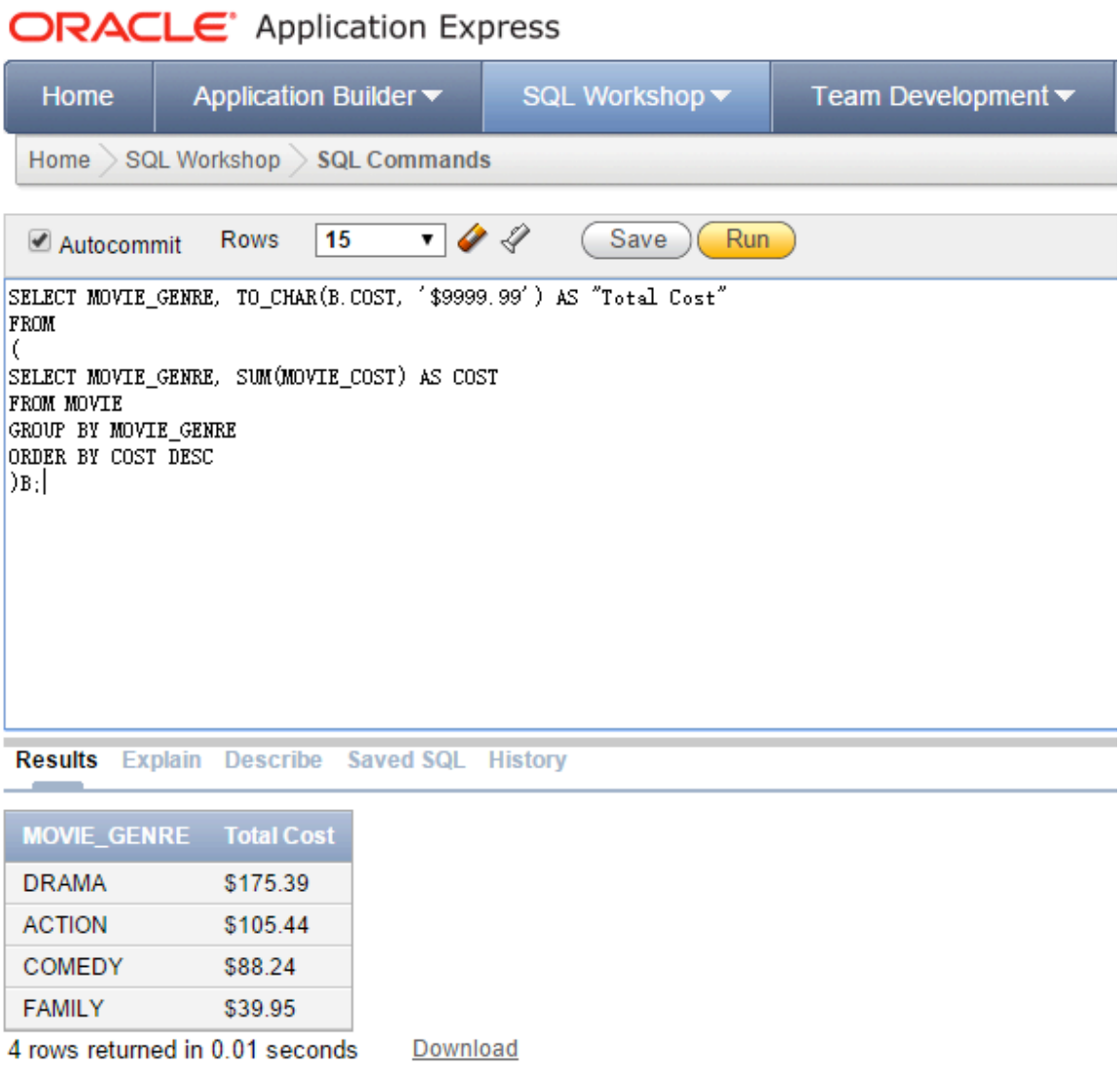

```
Sequence created.
```

0.01 seconds

2. What is the total movie cost for each movie genre? (order the results by total cost in descending order).

Movie_Genre	Total Cost
DRAMA	$175.39
ACTION	$105.44
COMEDY	$88.24
FAMILY	$39.95

```
SELECT MOVIE_GENRE, TO_CHAR(B.COST, '$9999.99') AS "Total Cost"
FROM
(
SELECT MOVIE_GENRE, SUM(MOVIE_COST) AS COST
FROM MOVIE
GROUP BY MOVIE_GENRE
ORDER BY COST DESC
)B;
```

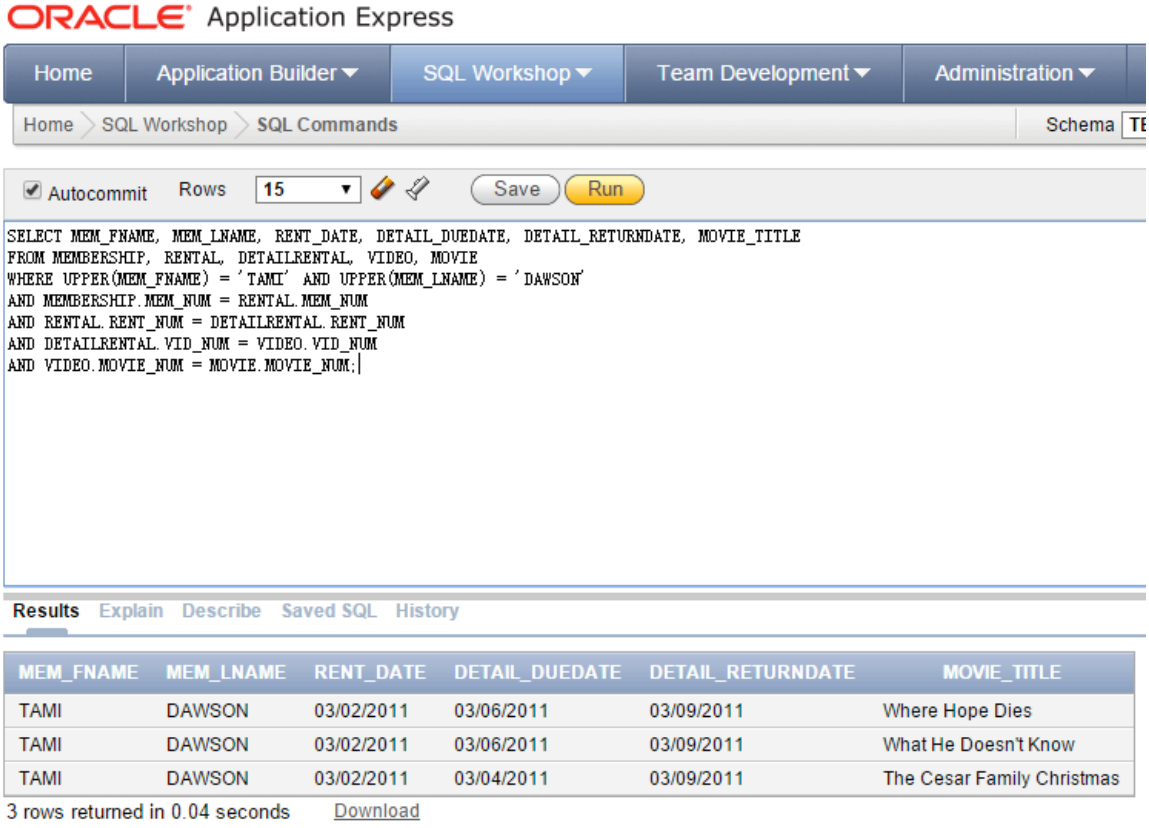


3. Display the rental history for Tami Dawson, display the first and last name, rent date, due date, return date, and movie title.

MEM_FNAME	MEM_LNAME	RENT_DATE	DETAIL_DUEDATE	DETAIL_RETURNDATE	MOVIE_TITLE
TAMI	DAWSON	03/02/0011	03/04/0011	03/09/0009	The Cesar Family Christmas
TAMI	DAWSON	03/02/0011	03/06/0011	03/09/0009	What He Doesn't Know
TAMI	DAWSON	03/02/0011	03/06/0011	03/09/0009	Where Hope Dies

```
SELECT MEM_FNAME, MEM_LNAME, RENT_DATE, DETAIL_DUEDATE,
DETAIL_RETURNDATE, MOVIE_TITLE
FROM MEMBERSHIP, RENTAL, DETAILRENTAL, VIDEO, MOVIE
WHERE UPPER(MEM_FNAME) = 'TAMI' AND UPPER(MEM_LNAME) = 'DAWSON'
AND MEMBERSHIP.MEM_NUM = RENTAL.MEM_NUM
```

```
AND RENTAL.RENT_NUM = DETAILRENTAL.RENT_NUM
AND DETAILRENTAL.VID_NUM = VIDEO.VID_NUM
AND VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM;
```

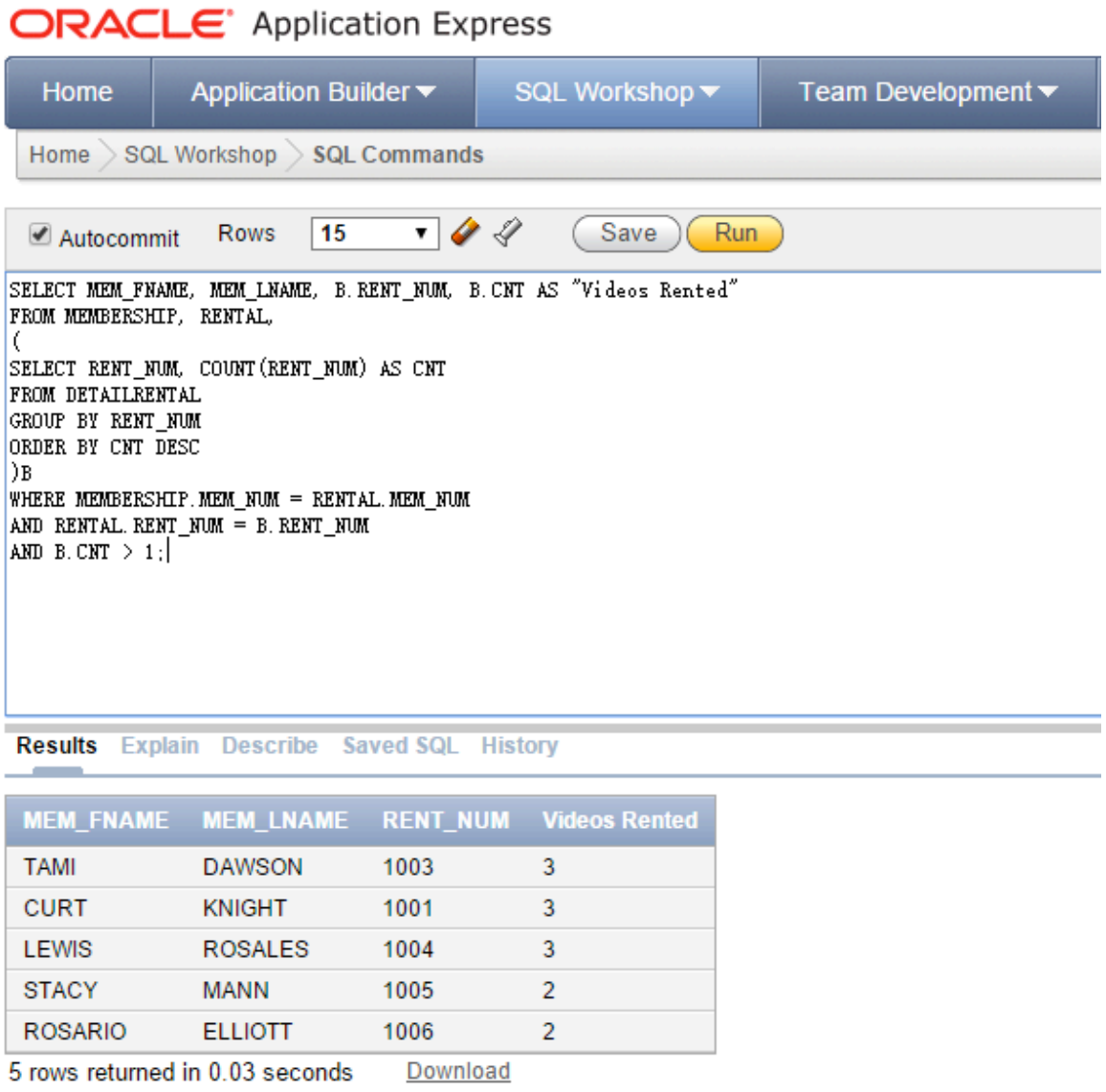


4. Which members have come to the store and rented more than one video at a time? Show the members, the rent number, and the number of videos rented (only display the members who have rented more than 1 video at a time).

MEM_FNAME	MEM_LNAME	RENT_NUM	Videos Rented
LEWIS	ROSALES	1004	3
CURT	KNIGHT	1001	3
TAMI	DAWSON	1003	3
STACY	MANN	1005	2
ROSARIO	ELLIOTT	1006	2

```
SELECT MEM_FNAME, MEM_LNAME, B.RENT_NUM, B.CNT AS "Videos Rented"
FROM MEMBERSHIP, RENTAL,
(
SELECT RENT_NUM, COUNT(RENT_NUM) AS CNT
FROM DETAILRENTAL
GROUP BY RENT_NUM
ORDER BY CNT DESC
)B
WHERE MEMBERSHIP.MEM_NUM = RENTAL.MEM_NUM
```

```
AND RENTAL.RENT_NUM = B.RENT_NUM
AND B.CNT > 1;
```



MEM_FNAME	MEM_LNAME	RENT_NUM	Videos Rented
TAMI	DAWSON	1003	3
CURT	KNIGHT	1001	3
LEWIS	ROSALES	1004	3
STACY	MANN	1005	2
ROSARIO	ELLIOTT	1006	2

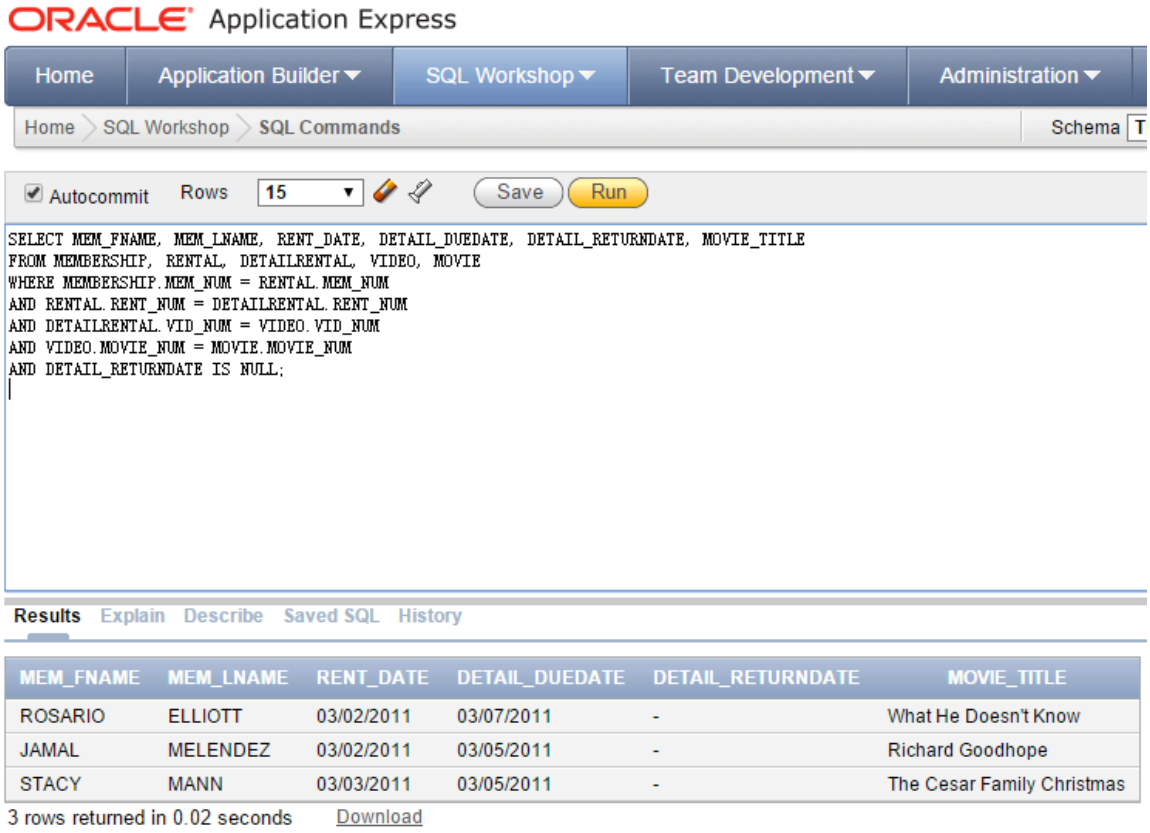
5 rows returned in 0.03 seconds Download

5. Which movies have not been returned? Display the movie title, member name, rental date and due date.

MEM_FNAME	MEM_LNAME	RENT_DATE	DETAIL_DUEDATE	DETAIL_RETURNDATE	MOVIE_TITLE
STACY	MANN	03/03/0011	03/05/0011	-	The Cesar Family Christmas
ROSARIO	ELLIOTT	03/02/0011	03/07/0011	-	What He Doesn't Know

JAMAL	MELENDEZ	03/02/0011	03/05/0011	-	Richard Goodhope

```
SELECT MEM_FNAME, MEM_LNAME, RENT_DATE, DETAIL_DUEDATE,
DETAIL_RETURNDATE, MOVIE_TITLE
FROM MEMBERSHIP, RENTAL, DETAILRENTAL, VIDEO, MOVIE
WHERE MEMBERSHIP.MEM_NUM = RENTAL.MEM_NUM
AND RENTAL.RENT_NUM = DETAILRENTAL.RENT_NUM
AND DETAILRENTAL.VID_NUM = VIDEO.VID_NUM
AND VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM
AND DETAIL_RETURNDATE IS NULL;
```

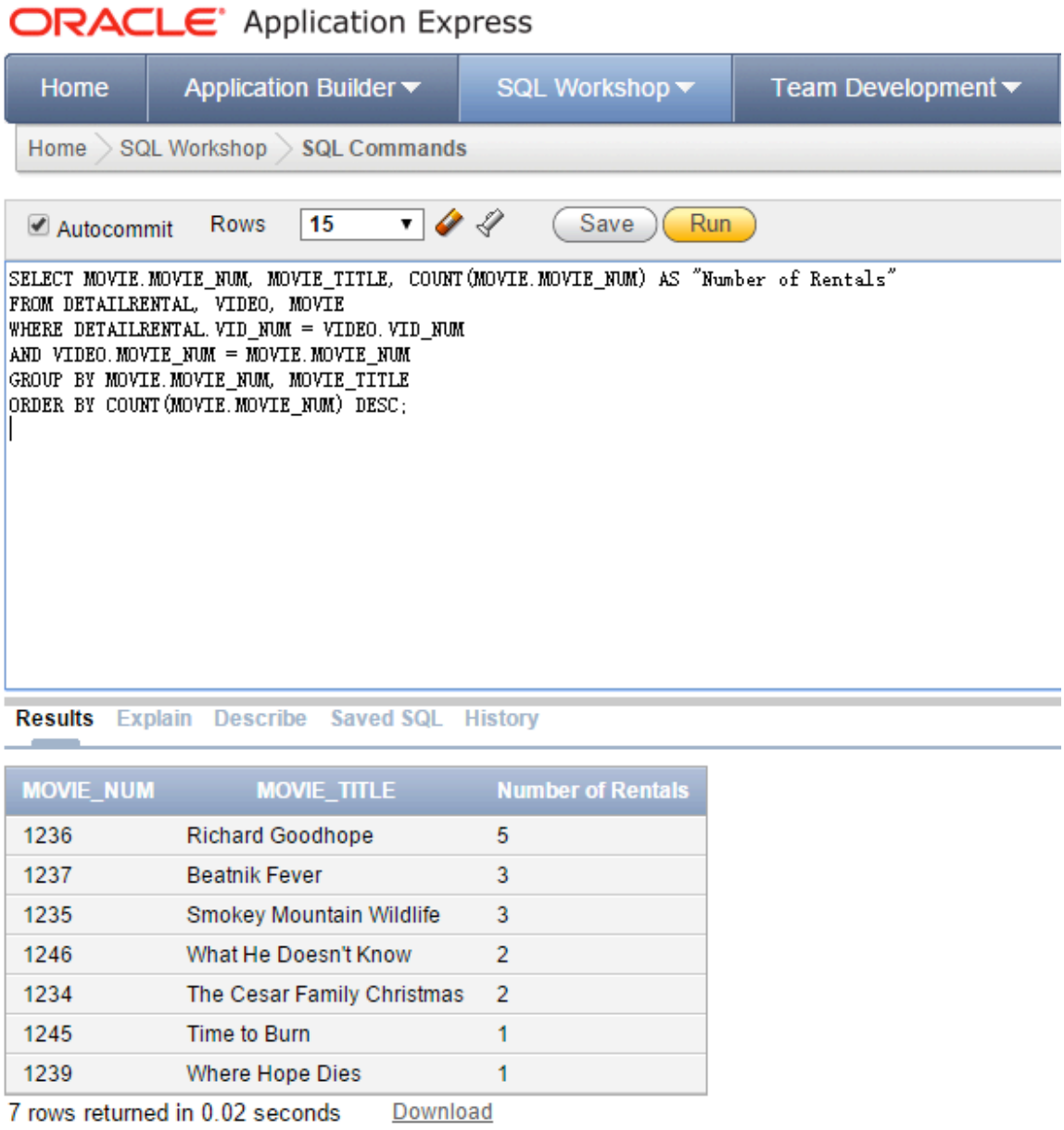


6. Which movie titles have been rented out the most?

MOVIE_NUM	MOVIE_TITLE	Number of Rentals
1236	Richard Goodhope	5
1237	Beatnik Fever	3
1235	Smokey Mountain Wildlife	3
1246	What He Doesn't Know	2
1234	The Cesar Family Christmas	2
1245	Time to Burn	1
1239	Where Hope Dies	1

```
SELECT MOVIE.MOVIE_NUM, MOVIE_TITLE, COUNT(MOVIE.MOVIE_NUM) AS "Number
of Rentals"
FROM DETAILRENTAL, VIDEO, MOVIE
WHERE DETAILRENTAL.VID_NUM = VIDEO.VID_NUM
```

```
AND VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM
GROUP BY MOVIE.MOVIE_NUM, MOVIE_TITLE
ORDER BY COUNT(MOVIE.MOVIE_NUM) DESC;
```



MOVIE_NUM	MOVIE_TITLE	Number of Rentals
1236	Richard Goodhope	5
1237	Beatnik Fever	3
1235	Smokey Mountain Wildlife	3
1246	What He Doesn't Know	2
1234	The Cesar Family Christmas	2
1245	Time to Burn	1
1239	Where Hope Dies	1

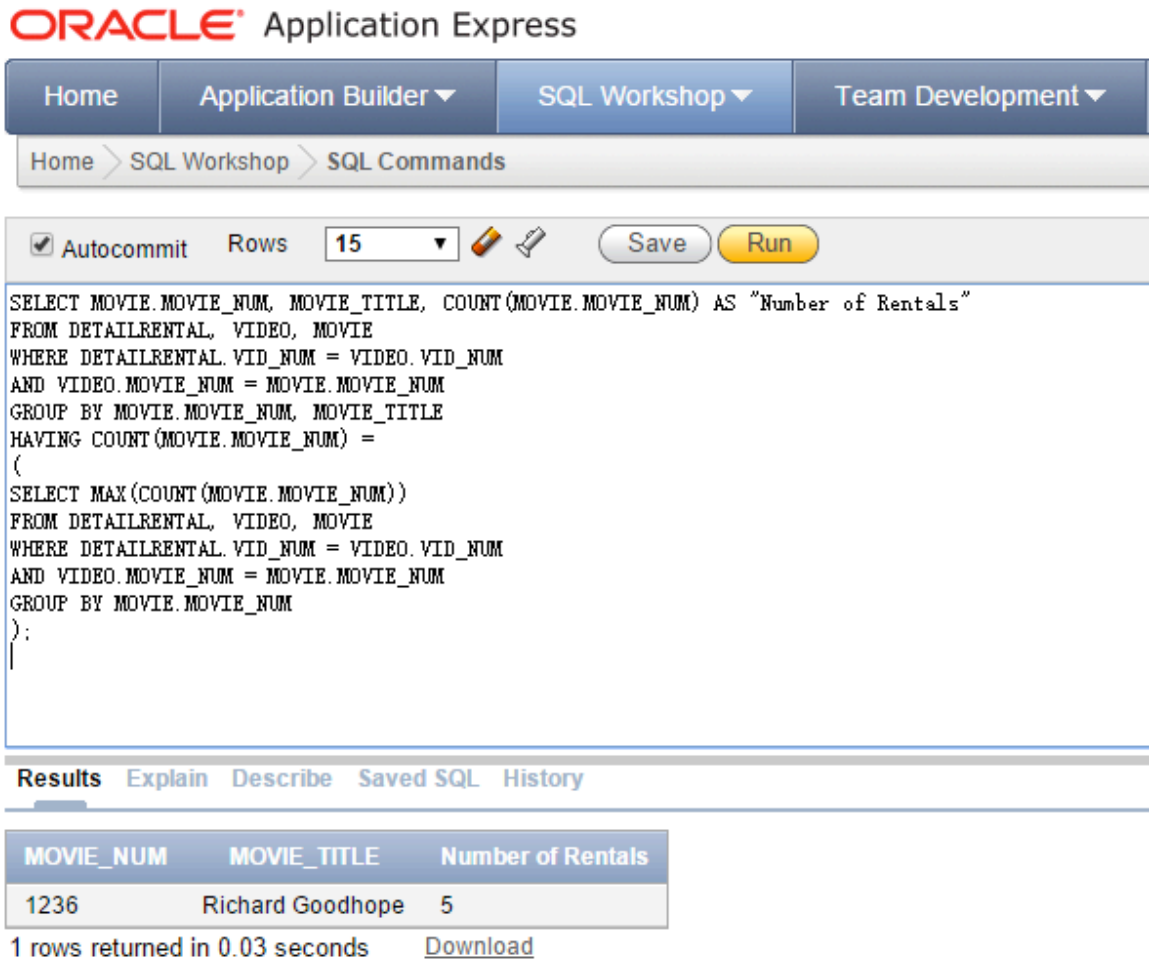
7 rows returned in 0.02 seconds Download

7. Modify the query in question #6 to only display the movie that has been rented the most times (Hint: you are looking for MAX Number of Rentals. You will need to use a subquery on the Having clause)

MOVIE_NUM	MOVIE_TITLE	Number of Rentals
1236	Richard Goodhope	5

SELECT MOVIE.MOVIE_NUM, MOVIE_TITLE, COUNT(MOVIE.MOVIE_NUM) AS "Number of Rentals"

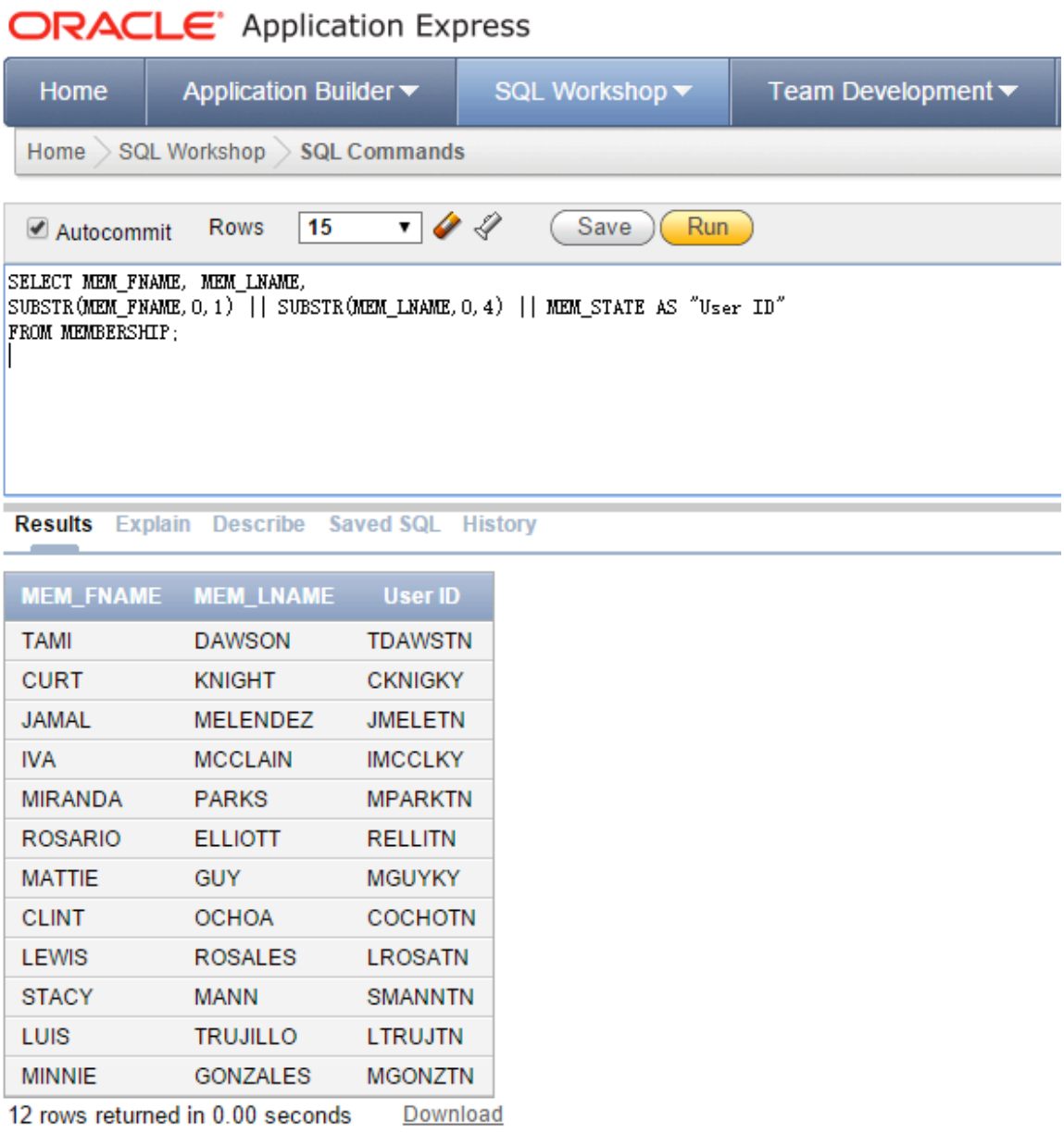
```
FROM DETAILRENTAL, VIDEO, MOVIE
WHERE DETAILRENTAL.VID_NUM = VIDEO.VID_NUM
AND VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM
GROUP BY MOVIE.MOVIE_NUM, MOVIE_TITLE
HAVING COUNT(MOVIE.MOVIE_NUM) =
(
SELECT MAX(COUNT(MOVIE.MOVIE_NUM))
FROM DETAILRENTAL, VIDEO, MOVIE
WHERE DETAILRENTAL.VID_NUM = VIDEO.VID_NUM
AND VIDEO.MOVIE_NUM = MOVIE.MOVIE_NUM
GROUP BY MOVIE.MOVIE_NUM
);
```



8. Generate a list of Member ids using the first character of the first name, the first 4 characters of the last name, and the 2 characters of member state.

MEM_FNAME	MEM_LNAME	User ID
TAMI	DAWSON	TDAWSOTN
CURT	KNIGHT	CKNIGHKY
JAMAL	MELENDEZ	JMELENTN
IVA	MCCLAIN	IMCCLAKY
MIRANDA	PARKS	MPARKSTN
ROSARIO	ELLIOTT	RELLIOTN
MATTIE	GUY	MGUYKY
CLINT	OCHOA	COCHOATN
LEWIS	ROSALES	LROSALTN
STACY	MANN	SMANNTN
LUIS	TRUJILLO	LTRUJITN
MINNIE	GONZALES	MGONZATN

```
SELECT MEM_FNAME, MEM_LNAME,
SUBSTR(MEM_FNAME,0,1) || SUBSTR(MEM_LNAME,0,4) || MEM_STATE AS "User ID"
FROM MEMBERSHIP;
```

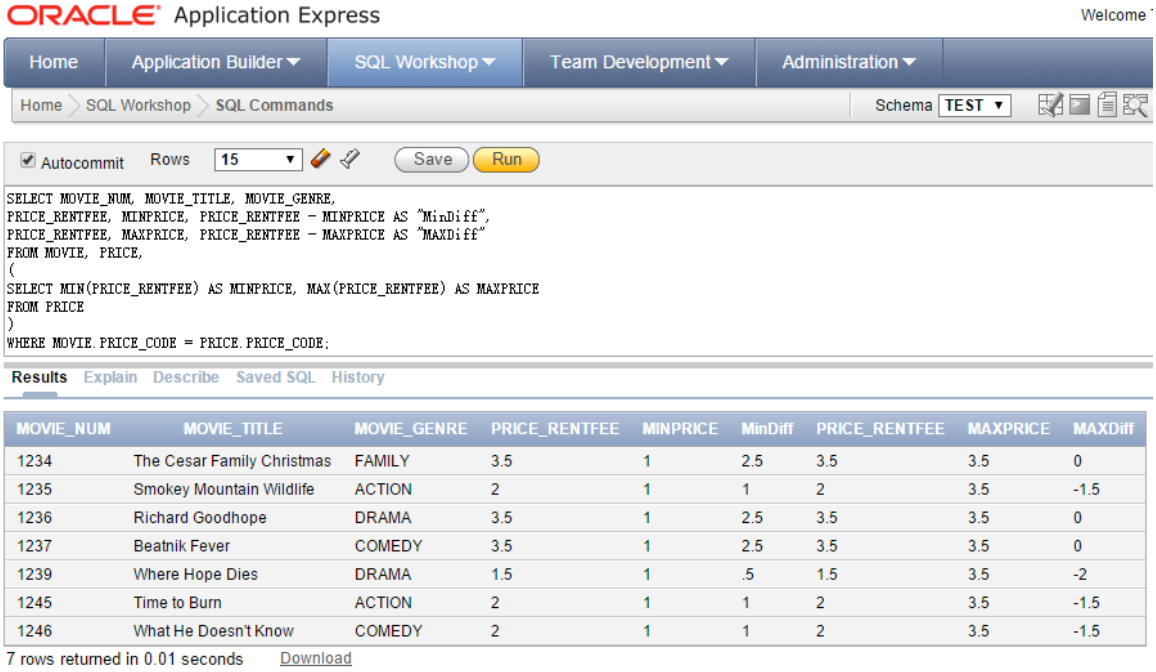


9. Show the difference between each movie's rental fee and the min movie price and the max movie price.

MOVIE_NUM	MOVIE_TITLE	MOVIE_GENRE	PRICE_RENTFEE	MINPRICE	MinDiff	PRICE_RENTFEE	MAXPRICE	MAXDiff
1245	Time to Burn	ACTION	2	1	1	2	3.5	-1.5
1246	What He Doesn't Know	COMEDY	2	1	1	2	3.5	-1.5
1235	Smokey Mountain Wildlife	ACTION	2	1	1	2	3.5	-1.5
1236	Richard Goodhope	DRAMA	3.5	1	2.5	3.5	3.5	0

1234	The Cesar Family Christmas	FAMILY	3.5	1	2.5	3.5	3.5	0
1237	Beatnik Fever	COMEDY	3.5	1	2.5	3.5	3.5	0
1239	Where Hope Dies	DRAMA	1.5	1	.5	1.5	3.5	-2

```
SELECT MOVIE_NUM, MOVIE_TITLE, MOVIE_GENRE,
PRICE_RENTFEE, MINPRICE, PRICE_RENTFEE - MINPRICE AS "MinDiff",
PRICE_RENTFEE, MAXPRICE, PRICE_RENTFEE - MAXPRICE AS "MAXDiff"
FROM MOVIE, PRICE,
(
SELECT MIN(PRICE_RENTFEE) AS MINPRICE, MAX(PRICE_RENTFEE) AS MAXPRICE
FROM PRICE
)
WHERE MOVIE.PRICE_CODE = PRICE.PRICE_CODE;
```



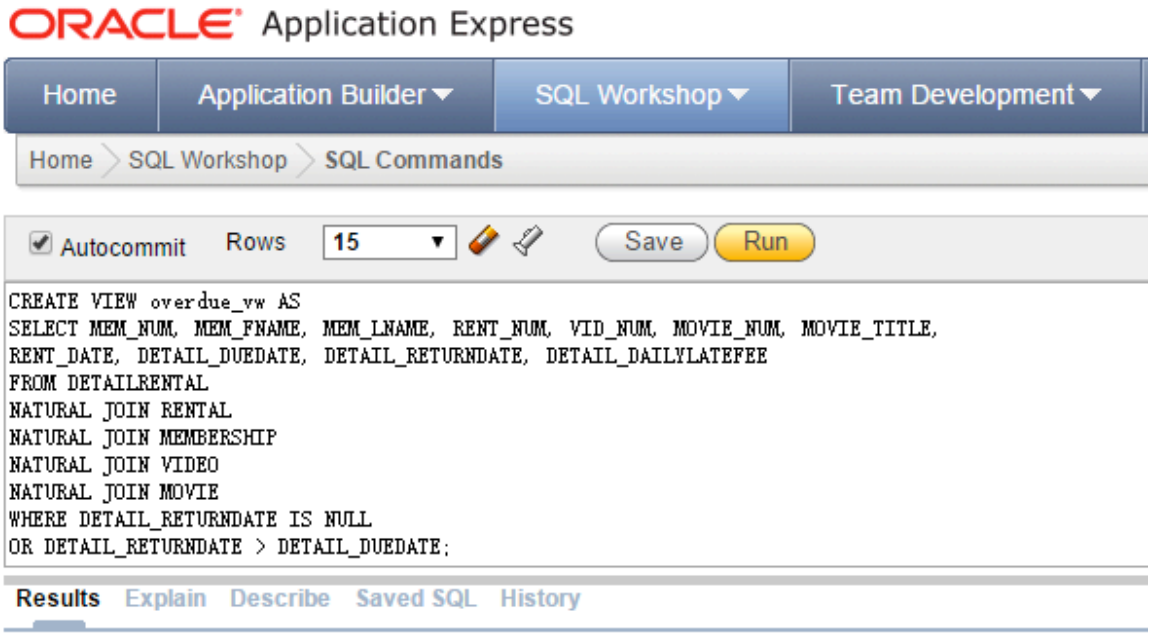
MOVIE_NUM	MOVIE_TITLE	MOVIE_GENRE	PRICE_RENTFEE	MINPRICE	MinDiff	PRICE_RENTFEE	MAXPRICE	MAXDiff
1234	The Cesar Family Christmas	FAMILY	3.5	1	2.5	3.5	3.5	0
1235	Smokey Mountain Wildlife	ACTION	2	1	1	2	3.5	-1.5
1236	Richard Goodhope	DRAMA	3.5	1	2.5	3.5	3.5	0
1237	Beatnik Fever	COMEDY	3.5	1	2.5	3.5	3.5	0
1239	Where Hope Dies	DRAMA	1.5	1	.5	1.5	3.5	-2
1245	Time to Burn	ACTION	2	1	1	2	3.5	-1.5
1246	What He Doesn't Know	COMEDY	2	1	1	2	3.5	-1.5

7 rows returned in 0.01 seconds Download

10. Create a view that will produce the result set shown below. Call the view overdue_vw. (5 pts)

MEM_NUM	MEM_LNAME	RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	RENT_DATE	DETAIL_DUEDATE	DETAIL_RETURNDATE
107	ELLIOTT	1006	61367	1246	What He Doesn't Know	03/02/0011	03/07/0011	-
104	MELENDEZ	1007	34368	1236	Richard Goodhope	03/02/0011	03/05/0011	-
111	MANN	1009	54324	1234	The Cesar Family Christmas	03/03/0011	03/05/0011	-

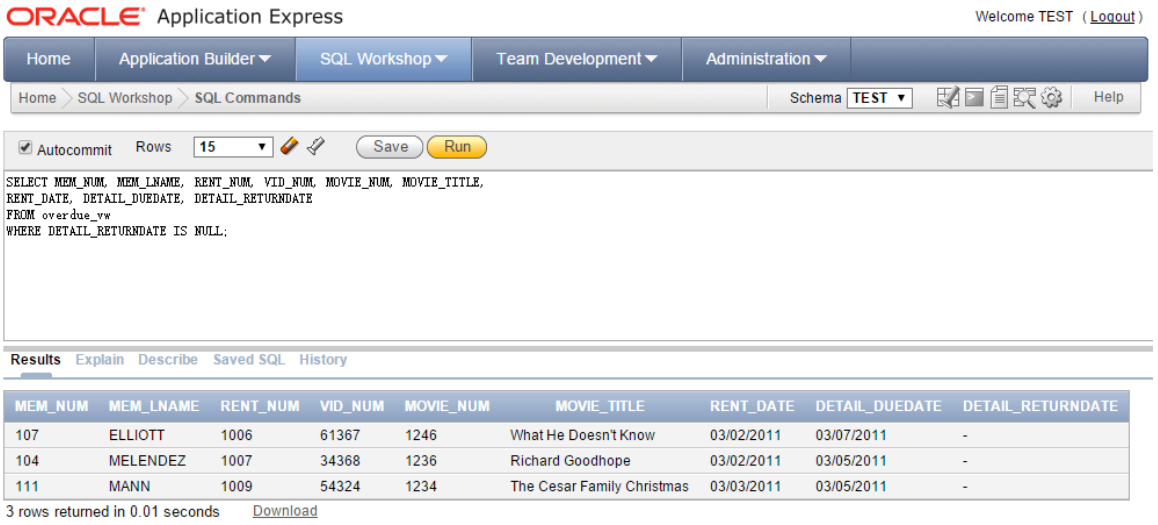
```
CREATE VIEW overdue_vw AS
SELECT MEM_NUM, MEM_FNAME, MEM_LNAME, RENT_NUM, VID_NUM, MOVIE_NUM,
MOVIE_TITLE,
RENT_DATE, DETAIL_DUEDATE, DETAIL_RETURNDATE, DETAIL_DAILYLATEFEE
FROM DETAILRENTAL
NATURAL JOIN RENTAL
NATURAL JOIN MEMBERSHIP
NATURAL JOIN VIDEO
NATURAL JOIN MOVIE
WHERE DETAIL_RETURNDATE IS NULL
OR DETAIL_RETURNDATE > DETAIL_DUEDATE;
```




```
View created.
```

0.02 seconds

```
SELECT MEM_NUM, MEM_LNAME, RENT_NUM, VID_NUM, MOVIE_NUM, MOVIE_TITLE,
RENT_DATE, DETAIL_DUEDATE, DETAIL_RETURNDATE
FROM overdue_vw
WHERE DETAIL_RETURNDATE IS NULL;
```

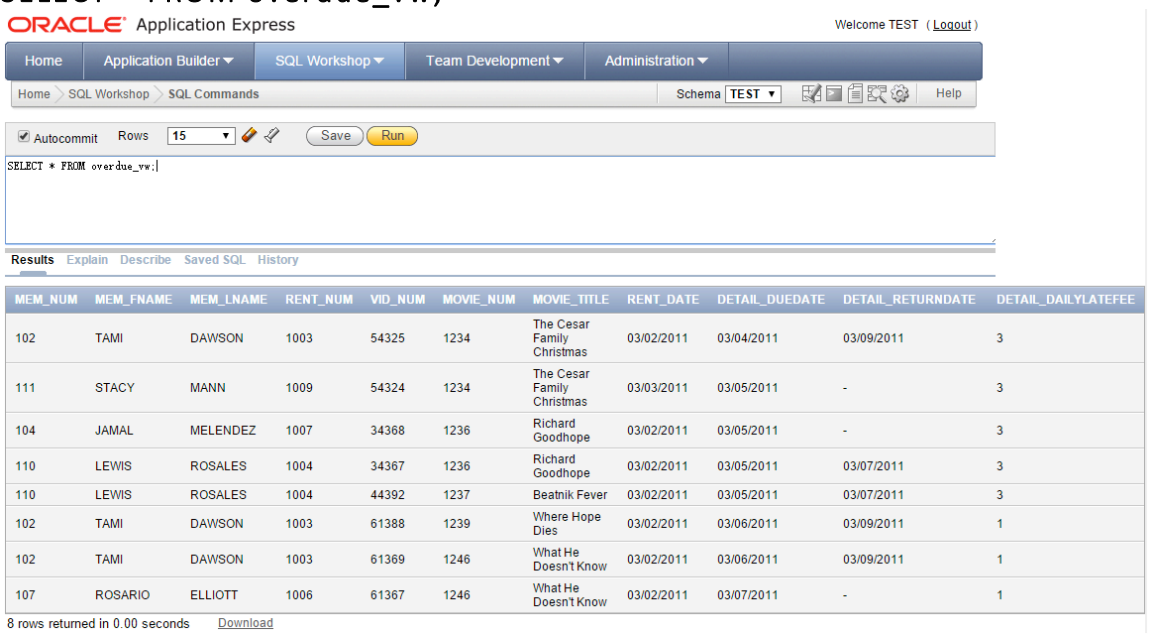


MEM_NUM	MEM_LNAME	RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	RENT_DATE	DETAIL_DUEDATE	DETAIL_RETURNDATE
107	ELLIOTT	1006	61367	1246	What He Doesn't Know	03/02/2011	03/07/2011	-
104	MELENDEZ	1007	34368	1236	Richard Goodhope	03/02/2011	03/05/2011	-
111	MANN	1009	54324	1234	The Cesar Family Christmas	03/03/2011	03/05/2011	-

3 rows returned in 0.01 seconds Download

--After you have created the view, test the view by selecting all rows and columns from the view. Show the query and the output.

SELECT * FROM overdue_vw;



MEM_NUM	MEM_FNAME	MEM_LNAME	RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	RENT_DATE	DETAIL_DUEDATE	DETAIL_RETURNDATE	DETAIL_DAILYLATEFEE
102	TAMI	DAWSON	1003	54325	1234	The Cesar Family Christmas	03/02/2011	03/04/2011	03/09/2011	3
111	STACY	MANN	1009	54324	1234	The Cesar Family Christmas	03/03/2011	03/05/2011	-	3
104	JAMAL	MELENDEZ	1007	34368	1236	Richard Goodhope	03/02/2011	03/05/2011	-	3
110	LEWIS	ROSALES	1004	34367	1236	Richard Goodhope	03/02/2011	03/05/2011	03/07/2011	3
110	LEWIS	ROSALES	1004	44392	1237	Beatnik Fever	03/02/2011	03/05/2011	03/07/2011	3
102	TAMI	DAWSON	1003	61388	1239	Where Hope Dies	03/02/2011	03/06/2011	03/09/2011	1
102	TAMI	DAWSON	1003	61369	1246	What He Doesn't Know	03/02/2011	03/06/2011	03/09/2011	1
107	ROSARIO	ELLIOTT	1006	61367	1246	What He Doesn't Know	03/02/2011	03/07/2011	-	1

8 rows returned in 0.00 seconds Download

Extra Credit:

Alter the query in #10 to display the total amount due per rental detail, per customer.

MEM_FNAME	MEM_LNAME	RENT_NUM	DETAIL_DUEDATE	Overdue Fee
TAMI	DAWSON	1003	03/06/0011	$6.00
LEWIS	ROSALES	1004	03/05/0011	$14.00
TAMI	DAWSON	1003	03/04/0011	$17.50

```
SELECT MEM_FNAME, MEM_LNAME, RENT_NUM, DETAIL_DUEDATE,
TO_CHAR(SUM((DETAIL_RETURNDATE-
DETAIL_DUEDATE)*DETAIL_DAILYLATEFEE),'$9999.99')
AS "Overdue Fee"
FROM overdue_vw
WHERE DETAIL_RETURNDATE IS NOT NULL
GROUP BY MEM_FNAME, MEM_LNAME, RENT_NUM, DETAIL_DUEDATE
ORDER BY DETAIL_DUEDATE DESC;
```

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, RENT_NUM, DETAIL_DUEDATE,
TO_CHAR(SUM((DETAIL_RETURNDATE-DETAIL_DUEDATE)*DETAIL_DAILYLATEFEE),'$9999.99')
AS "Overdue Fee"
FROM overdue_vw
WHERE DETAIL_RETURNDATE IS NOT NULL
GROUP BY MEM_FNAME, MEM_LNAME, RENT_NUM, DETAIL_DUEDATE
ORDER BY DETAIL_DUEDATE DESC;
```

Results Explain Describe Saved SQL History

MEM_FNAME	MEM_LNAME	RENT_NUM	DETAIL_DUEDATE	Overdue Fee
TAMI	DAWSON	1003	03/06/2011	$6.00
LEWIS	ROSALES	1004	03/05/2011	$12.00
TAMI	DAWSON	1003	03/04/2011	$15.00

3 rows returned in 0.02 seconds Download