

## Assignment 6A Triggers and DB Performance Tuning

Total points: 25

**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.**

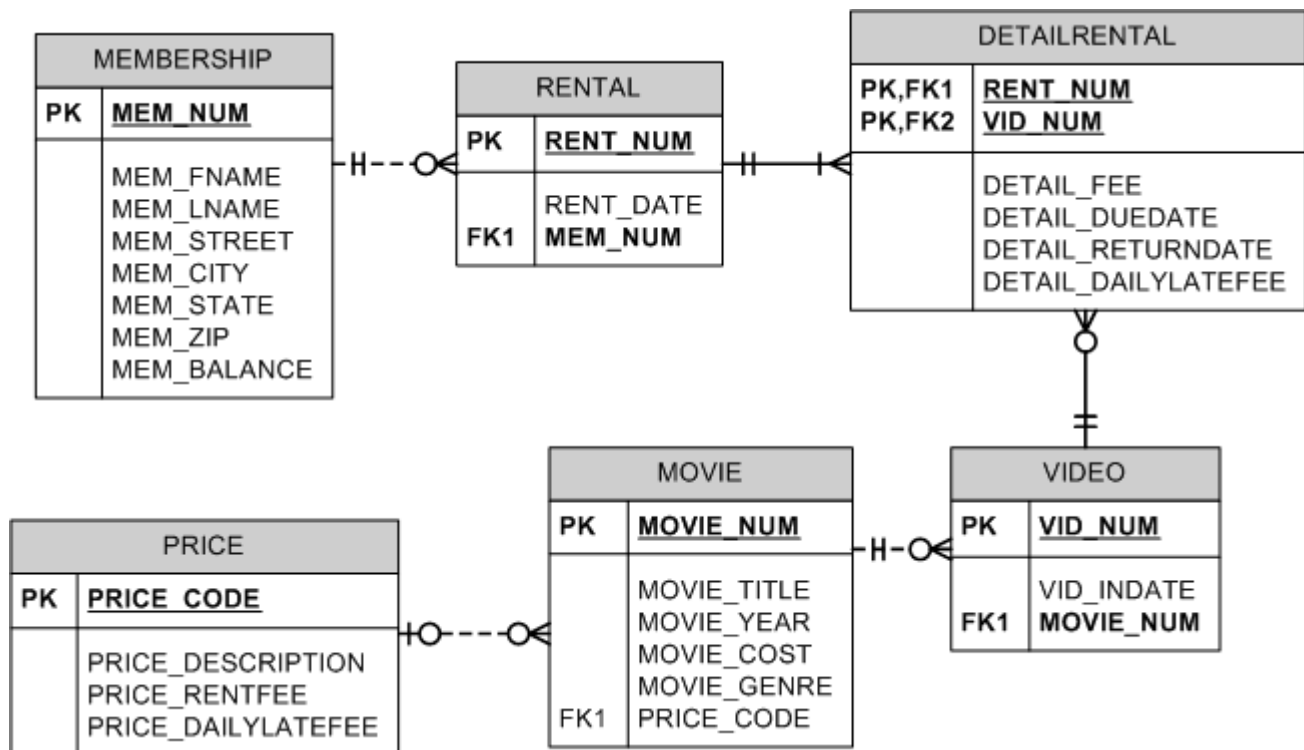
I recommend creating a new user and workspace, log in as that user and load the database script ourvideo\_A2.sql (provided in this week's assignment folder).

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



1. Alter the VIDEO table to include an attribute named VID\_STATUS to store character data up to 4 characters long. The attribute should not accept null values. The attribute should have a constraint to enforce the domain ("IN", "OUT", and "LOST"), and have a default value of "IN". (3 pts)

```
ALTER TABLE VIDEO  
ADD VID_STATUS CHAR(4) DEFAULT 'IN' NOT NULL CHECK(VID_STATUS IN  
( 'IN','OUT','LOST' ));
```

**ORACLE** Application Express

Home	Application Builder ▼	SQL Workshop ▼	Team Development ▼
Home > SQL Workshop > SQL Commands			
<input checked="" type="checkbox"/> Autocommit   Rows: 15 <input type="button" value="Save"/> <input type="button" value="Run"/>			
<pre>ALTER TABLE VIDEO ADD VID_STATUS CHAR(4) DEFAULT 'IN' NOT NULL CHECK(VID_STATUS IN ( 'IN','OUT','LOST' ));</pre>			
Results   Explain   Describe   Saved SQL   History			

Table altered.

0.04 seconds

2. Create a trigger named trg\_videorental\_up that will update the correct value VID\_STATUS in the VIDEO table whenever a video is checked out (OUT) or returned (IN). The trigger should execute as an AFTER trigger when the DETAIL\_DUEDATE or DETAIL\_RETURNDATE attributes are updated in the DETAILRENTAL table. The trigger should satisfy the following conditions:
  - a. If the DETAIL\_RETURNDATE in the detail rental table is set to NULL, the VID\_STATUS should be set to "OUT".
  - b. If the DETAIL\_RETURNDATE in the detail rental table is set to > than the current date, the VID\_STATUS should be set to "OUT".
  - c. If the DETAIL\_RETURNDATE in the detail rental table is set to < or = to the current date, the VID\_STATUS should be set to "IN".
  - d. If the DETAIL\_RETURNDATE in the detail rental table is set to "01/01/01", the VID\_STATUS should be set to "LOST".

```

CREATE OR REPLACE TRIGGER trg_videorental_up
AFTER INSERT OR UPDATE OF DETAIL_DUEDATE, DETAIL_RETURNDATE
ON DETAILRENTAL
FOR EACH ROW
BEGIN
    IF :new.DETAIL_RETURNDATE IS NULL THEN
        UPDATE VIDEO SET VID_STATUS = 'OUT' WHERE VID_NUM = :new.VID_NUM;
    ELSIF TO_CHAR(:new.DETAIL_RETURNDATE,'DD/MM/YY') = '01/01/01' THEN
        UPDATE VIDEO SET VID_STATUS = 'LOST' WHERE VID_NUM = :new.VID_NUM;
    ELSIF :new.DETAIL_RETURNDATE > SYSDATE THEN
        UPDATE VIDEO SET VID_STATUS = 'OUT' WHERE VID_NUM = :new.VID_NUM;
    ELSIF :new.DETAIL_RETURNDATE <= SYSDATE THEN
        UPDATE VIDEO SET VID_STATUS = 'IN' WHERE VID_NUM = :new.VID_NUM;
    END IF;
END;

```

**ORACLE** Application Express

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

Home > SQL Workshop > SQL Commands

---

☒ Autocommit    Rows:       

```

CREATE OR REPLACE TRIGGER trg_videorental_up
AFTER INSERT OR UPDATE OF DETAIL_DUEDATE, DETAIL_RETURNDATE
ON DETAILRENTAL
FOR EACH ROW
BEGIN
    IF :new.DETAIL_RETURNDATE IS NULL THEN
        UPDATE VIDEO SET VID_STATUS = 'OUT' WHERE VID_NUM = :new.VID_NUM;
    ELSIF TO_CHAR(:new.DETAIL_RETURNDATE,'DD/MM/YY') = '01/01/01' THEN
        UPDATE VIDEO SET VID_STATUS = 'LOST' WHERE VID_NUM = :new.VID_NUM;
    ELSIF :new.DETAIL_RETURNDATE > SYSDATE THEN
        UPDATE VIDEO SET VID_STATUS = 'OUT' WHERE VID_NUM = :new.VID_NUM;
    ELSIF :new.DETAIL_RETURNDATE <= SYSDATE THEN
        UPDATE VIDEO SET VID_STATUS = 'IN' WHERE VID_NUM = :new.VID_NUM;
    END IF;
END;

```

---

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

Trigger created.

0.10 seconds

--After you have created the trigger, test the trigger. Update a record for each scenario into the detail rental table. Show the update statements. To show that the trigger has run, show the output from the following query:

```
select dr.rent_num, dr.vid_num, v.movie_num, m.movie_title, v.vid_status, dr.detail_duedate,
dr.detail_returndate
from detailrental dr, video v, movie m
where dr.vid_num = v.vid_num and m.movie_num = v.movie_num
```

(9 pts for Trigger, 4 points for update statements, 5 points for output)

**UPDATE DETAILRENTAL SET DETAIL\_RETURNDATE = NULL WHERE RENT\_NUM = 1009;**

<input checked="" type="checkbox"/> Autocommit	Rows	20			Save	Run
UPDATE DETAILRENTAL SET DETAIL_RETURNDATE = NULL WHERE RENT_NUM = 1009;						
<pre>select dr.rent_num, dr.vid_num, v.movie_num, m.movie_title, v.vid_status, dr.detail_duedate, dr.detail_returndate from detailrental dr, video v, movie m where dr.vid_num = v.vid_num and m.movie_num = v.movie_num;</pre>						
Results	Explain	Describe	Saved SQL	History		
RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	VID_STATUS	DETAIL_DUEDATE	DETAIL_RETURNDATE
1001	34342	1235	Smokey Mountain Wildlife	IN	03/04/2011	03/02/2011
1001	34366	1236	Richard Goodhope	IN	03/04/2011	03/02/2011
1001	61353	1245	Time to Burn	IN	03/04/2011	03/03/2011
1002	59237	1237	Beatnik Fever	IN	03/04/2011	03/04/2011
1003	54325	1234	The Cesar Family Christmas	IN	03/04/2011	03/09/2011
1003	61369	1246	What He Doesn't Know	IN	03/06/2011	03/09/2011
1003	61388	1239	Where Hope Dies	IN	03/06/2011	03/09/2011
1004	34341	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/07/2011
1004	34367	1236	Richard Goodhope	IN	03/05/2011	03/07/2011
1004	44392	1237	Beatnik Fever	IN	03/05/2011	03/07/2011
1005	34342	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/05/2011
1005	44397	1237	Beatnik Fever	IN	03/05/2011	03/05/2011
1006	34366	1236	Richard Goodhope	IN	03/05/2011	03/04/2011
1006	61367	1246	What He Doesn't Know	IN	03/07/2011	-
1007	34368	1236	Richard Goodhope	IN	03/05/2011	-
1008	34369	1236	Richard Goodhope	IN	03/05/2011	03/05/2011
1009	54324	1234	The Cesar Family Christmas	OUT	03/05/2011	-

17 rows returned in 0.00 seconds

[Download](#)

**UPDATE DETAILRENTAL SET DETAIL\_RETURNDATE = TO\_DATE('01/01/01','DD/MM/YY')  
WHERE RENT\_NUM = 1009;**

☒ Autocommit
 Rows

```
UPDATE DETAILRENTAL SET DETAIL_RETURNDATE = TO_DATE('01/01/01','DD/MM/YY') WHERE RENT_NUM = 1009;
```

```
select dr.rent_num, dr.vid_num, v.movie_num, m.movie_title, v.vid_status, dr.detail_duedate, dr.detail_returndate
from detailrental dr, video v, movie m
where dr.vid_num = v.vid_num and m.movie_num = v.movie_num;
```

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

RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	VID_STATUS	DETAIL_DUEDATE	DETAIL_RETURNDATE
1001	34342	1235	Smokey Mountain Wildlife	IN	03/04/2011	03/02/2011
1001	34366	1236	Richard Goodhope	IN	03/04/2011	03/02/2011
1001	61353	1245	Time to Burn	IN	03/04/2011	03/03/2011
1002	59237	1237	Beatnik Fever	IN	03/04/2011	03/04/2011
1003	54325	1234	The Cesar Family Christmas	IN	03/04/2011	03/09/2011
1003	61369	1246	What He Doesn't Know	IN	03/06/2011	03/09/2011
1003	61388	1239	Where Hope Dies	IN	03/06/2011	03/09/2011
1004	34341	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/07/2011
1004	34367	1236	Richard Goodhope	IN	03/05/2011	03/07/2011
1004	44392	1237	Beatnik Fever	IN	03/05/2011	03/07/2011
1005	34342	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/05/2011
1005	44397	1237	Beatnik Fever	IN	03/05/2011	03/05/2011
1006	34366	1236	Richard Goodhope	IN	03/05/2011	03/04/2011
1006	61367	1246	What He Doesn't Know	IN	03/07/2011	-
1007	34368	1236	Richard Goodhope	IN	03/05/2011	-
1008	34369	1236	Richard Goodhope	IN	03/05/2011	03/05/2011
1009	54324	1234	The Cesar Family Christmas	LOST	03/05/2011	01/01/2001

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

**UPDATE DETAILRENTAL SET DETAIL\_RETURNDATE = TO\_DATE('01/01/17','DD/MM/YY')  
WHERE RENT\_NUM = 1009;**

☒ Autocommit
 Rows 20
Save Run

UPDATE DETAILRENTAL SET DETAIL\_RETURNDATE = TO\_DATE('01/01/17','DD/MM/YY') WHERE RENT\_NUM = 1009;

select dr.rent\_num, dr.vid\_num, v.movie\_num, m.movie\_title, v.vid\_status, dr.detail\_duedate, dr.detail\_returndate  
from detailrental dr, video v, movie m  
where dr.vid\_num = v.vid\_num and m.movie\_num = v.movie\_num;

Results Explain Describe Saved SQL History

RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	VID_STATUS	DETAIL_DUEDATE	DETAIL_RETURNDATE
1001	34342	1235	Smokey Mountain Wildlife	IN	03/04/2011	03/02/2011
1001	34366	1236	Richard Goodhope	IN	03/04/2011	03/02/2011
1001	61353	1245	Time to Burn	IN	03/04/2011	03/03/2011
1002	59237	1237	Beatnik Fever	IN	03/04/2011	03/04/2011
1003	54325	1234	The Cesar Family Christmas	IN	03/04/2011	03/09/2011
1003	61369	1246	What He Doesn't Know	IN	03/06/2011	03/09/2011
1003	61388	1239	Where Hope Dies	IN	03/06/2011	03/09/2011
1004	34341	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/07/2011
1004	34367	1236	Richard Goodhope	IN	03/05/2011	03/07/2011
1004	44392	1237	Beatnik Fever	IN	03/05/2011	03/07/2011
1005	34342	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/05/2011
1005	44397	1237	Beatnik Fever	IN	03/05/2011	03/05/2011
1006	34366	1236	Richard Goodhope	IN	03/05/2011	03/04/2011
1006	61367	1246	What He Doesn't Know	IN	03/07/2011	-
1007	34368	1236	Richard Goodhope	IN	03/05/2011	-
1008	34369	1236	Richard Goodhope	IN	03/05/2011	03/05/2011
1009	54324	1234	The Cesar Family Christmas	OUT	03/05/2011	01/01/2017

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

UPDATE DETAILRENTAL SET DETAIL\_RETURNDATE = TO\_DATE('01/01/12','DD/MM/YY')  
WHERE RENT\_NUM = 1009;

☒ Autocommit
 Rows 20
Save Run

```
UPDATE DETAILRENTAL SET DETAIL_RETURNDATE = TO_DATE('01/01/12','DD/MM/YY') WHERE RENT_NUM = 1009;
```

```
select dr.rent_num, dr.vid_num, v.movie_num, m.movie_title, v.vid_status, dr.detail_duedate, dr.detail_returndate
from detailrental dr, video v, movie m
where dr.vid_num = v.vid_num and m.movie_num = v.movie_num;
```

Results

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

RENT_NUM	VID_NUM	MOVIE_NUM	MOVIE_TITLE	VID_STATUS	DETAIL_DUEDATE	DETAIL_RETURNDATE
1001	34342	1235	Smokey Mountain Wildlife	IN	03/04/2011	03/02/2011
1001	34366	1236	Richard Goodhope	IN	03/04/2011	03/02/2011
1001	61353	1245	Time to Burn	IN	03/04/2011	03/03/2011
1002	59237	1237	Beatnik Fever	IN	03/04/2011	03/04/2011
1003	54325	1234	The Cesar Family Christmas	IN	03/04/2011	03/09/2011
1003	61369	1246	What He Doesn't Know	IN	03/06/2011	03/09/2011
1003	61388	1239	Where Hope Dies	IN	03/06/2011	03/09/2011
1004	34341	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/07/2011
1004	34367	1236	Richard Goodhope	IN	03/05/2011	03/07/2011
1004	44392	1237	Beatnik Fever	IN	03/05/2011	03/07/2011
1005	34342	1235	Smokey Mountain Wildlife	IN	03/07/2011	03/05/2011
1005	44397	1237	Beatnik Fever	IN	03/05/2011	03/05/2011
1006	34366	1236	Richard Goodhope	IN	03/05/2011	03/04/2011
1006	61367	1246	What He Doesn't Know	IN	03/07/2011	-
1007	34368	1236	Richard Goodhope	IN	03/05/2011	-
1008	34369	1236	Richard Goodhope	IN	03/05/2011	03/05/2011
1009	54324	1234	The Cesar Family Christmas	IN	03/05/2011	01/01/2012

17 rows returned in 0.01 seconds
[Download](#)

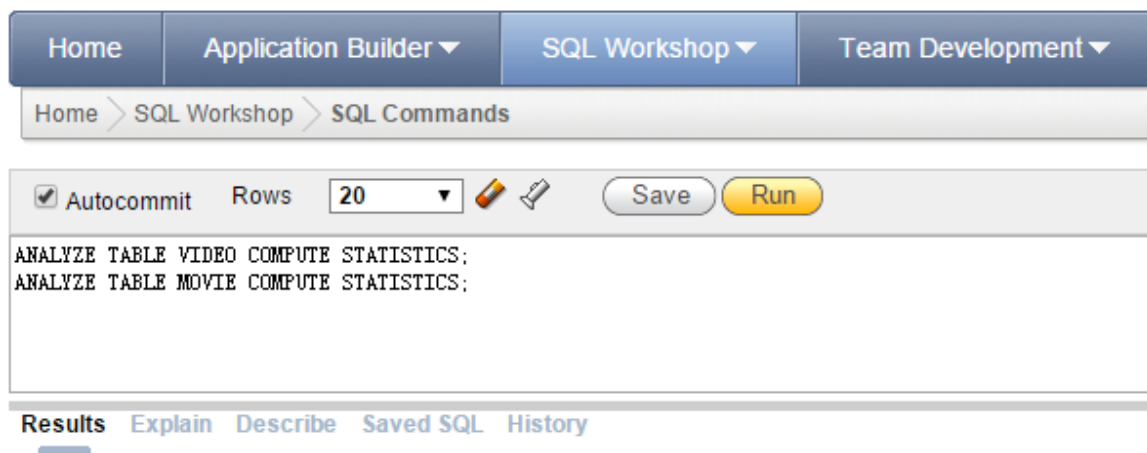
Problems 3-7 are based on the following query:

```
SELECT v.movie_num, movie_title, count(v.movie_num) as "Number of Videos"
FROM video v, movie
WHERE v.movie_num = movie.movie_num
GROUP BY v.movie_num, movie_title
ORDER BY "Number of Videos" DESC
```

3. Write the SQL command(s) to create statistics for these tables. (2 pts)

```
ANALYZE TABLE VIDEO COMPUTE STATISTICS;
ANALYZE TABLE MOVIE COMPUTE STATISTICS;
```

**ORACLE** Application Express



Statement processed.

0.01 seconds

4. Write the command to create an explain plan for this query. (2 pts)

```
EXPLAIN PLAN FOR
SELECT v.movie_num, movie_title, count(v.movie_num) as "Number of Videos"
FROM video v, movie
WHERE v.movie_num = movie.movie_num
GROUP BY v.movie_num, movie_title
ORDER BY "Number of Videos" DESC;
```



## ORACLE® Application Express

Home	Application Builder ▼	SQL Workshop ▼	Team Development ▼
Home > SQL Workshop > SQL Commands			
<input checked="" type="checkbox"/> Autocommit	Rows	20 ▼	Save Run
<pre>EXPLAIN PLAN FOR SELECT v.movie_num, movie_title, count(v.movie_num) as "Number of Videos" FROM video v, movie WHERE v.movie_num = movie.movie_num GROUP BY v.movie_num, movie_title ORDER BY "Number of Videos" DESC;</pre>			
Results Explain Describe Saved SQL History			

Statement processed.

0.03 seconds

5. Display the access plan given for this query. (3 pts) (Attach a screen shot of the results)

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

☒ Autocommit
 Rows 20
Save Run

```
SELECT * FROM TABLE(DEMS_XPLAN.DISPLAY);
```

**Results** Explain Describe Saved SQL History

PLAN_TABLE_OUTPUT						
Plan hash value: 503859967						
-----						
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
-----						
0	SELECT STATEMENT		32	896	8 (38)	00:00:01
1	SORT ORDER BY		32	896	8 (38)	00:00:01
2	HASH GROUP BY		32	896	8 (38)	00:00:01
3	MERGE JOIN		32	896	6 (17)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	MOVIE	8	168	2 (0)	00:00:01
5	INDEX FULL SCAN	SYS_C007223	8	1 (0)	1 (0)	00:00:01
* 6	SORT JOIN		17	119	4 (25)	00:00:01
7	TABLE ACCESS FULL	VIDEO	17	119	3 (0)	00:00:01
-----						
Predicate Information (identified by operation id):						
-----						
6 - access("V"."MOVIE_NUM"="MOVIE"."MOVIE_NUM")						
filter("V"."MOVIE_NUM"="MOVIE"."MOVIE_NUM")						

20 rows returned in 0.21 seconds
 [Download](#)

**6. Should you create an index? If so, what would the index column(s) be and why would you create that index? If not, explain your reasoning. (3 pts)**

According our textbook, we can know that a general rule for indexes are liked used:

- When an indexed column appears by itself in a search criteria of a WHERE or HAVING clause.
- When an indexed column appears by itself in a GROUP BY or ORDER BY clause.
- When a MAX or MIN function is applied to an indexed column.
- When the data sparsity on the indexed column is high.



In this question, we should create an index for movie\_num, because this column appears by itself in a search of WHERE or Having clause, and GROUP BY or ORDER BY.  
(The P\_CODE is foreign key, most DBMSs automatically index foreign key columns.  
Other columns not satisfied the first 2 rules addressed above. So, we should create an index on movie\_num).

- 7. Create an index on MOVIE\_NUM in the Video table. Show the SQL command to create the index. Update your statistics and rerun your explain plan. Create an updated explain plan for this query. Display the access plan given for this query. What difference did this change make? (3 pts)**

```
CREATE INDEX V_IDX ON VIDEO(MOVIE_NUM);  
ANALYZE TABLE VIDEO COMPUTE STATISTICS;  
ANALYZE TABLE MOVIE COMPUTE STATISTICS;
```

```
EXPLAIN PLAN FOR  
SELECT v.movie_num, movie_title, count(v.movie_num) as "Number of Videos"  
FROM video v, movie  
WHERE v.movie_num = movie.movie_num  
GROUP BY v.movie_num, movie_title  
ORDER BY "Number of Videos" DESC;
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

☒ Autocommit
 Rows 20


Save Run

```

CREATE INDEX V_IDX ON VIDEO(MOVIE_NUM);
ANALYZE TABLE VIDEO COMPUTE STATISTICS;
ANALYZE TABLE MOVIE COMPUTE STATISTICS;

EXPLAIN PLAN FOR
SELECT v.movie_num, movie_title, count(v.movie_num) as "Number of Videos"
FROM video v, movie
WHERE v.movie_num = movie.movie_num
GROUP BY v.movie_num, movie_title
ORDER BY "Number of Videos" DESC;

SELECT * FROM TABLE(DEMS_XPLAN.DISPLAY);
    
```

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

PLAN_TABLE_OUTPUT						
Plan hash value: 3417870536						
-----						
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
-----						
0	SELECT STATEMENT		32	896	5 (40)	00:00:01
1	SORT ORDER BY		32	896	5 (40)	00:00:01
2	HASH GROUP BY		32	896	5 (40)	00:00:01
3	NESTED LOOPS		32	896	3 (0)	00:00:01
4	TABLE ACCESS FULL	MOVIE	8	168	3 (0)	00:00:01
* 5	INDEX RANGE SCAN	V_IDX	4	28	0 (0)	00:00:01
-----						
Predicate Information (identified by operation id):						
-----						
5 - access("V"."MOVIE_NUM"="MOVIE"."MOVIE_NUM")						

17 rows returned in 0.03 seconds     [Download](#)

**We can see the cost decreased, from 8 to 5. So, the database performance is better.**