NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
NH 66, Srinivas Nagar, Surathkal, Mangaluru, Karnataka 575025, India

DBMS Assignment-6 on SQLViews

1. (a) **All students are requested to create the following three sample tables in your database and to implement the five questions given in 1(b).**

Sample table 1: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table 2: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London	null	5005

Sample table 3: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150	2012-10-05	3005	5002
70009	270	2012-09-10	3001	5005
70002	65	2012-10-05	3002	5001
70004	110	2012-08-17	3009	5003
70007	948	2012-09-10	3005	5002

70005	2400	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983	2012-10-10	3004	5006
70003	2480	2012-10-10	3009	5003
70012	250	2012-06-27	3008	5002
70011	75	2012-08-17	3003	5007
70013	3045	2012-04-25	3002	5001

(b) Implement the following queries in MySQL with the Tables generated in Question 1(a):

i. Write a query to create a view that shows for each order the salesman and customer by name.

ii. Write a query to create a view that finds the salesman who has the customer with the highest order of a day.

iii. Write a query to create a view to getting a count of how many customers we have at each level of a grade.

iv. Write a query to find the salesmen of the city New York who achieved the commission more than 13%.

v. Write a query to create a view that finds the salesman who has the customer with the highest order of a day.

2. (a) Create the following Tables in your database:

Schema:
Movie (mID, title, year, director)
Semantics: There is a movie with ID number mID, a title, a release year, and a director.
Reviewer (rID, name)
Semantics: The reviewer with ID number rID has a certain name.
Rating (rID, mID, stars, ratingDate)
Semantics: The reviewer rID gave the movie mID a number of stars rating (1-5) on certain ratingDate.

/* Delete the tables if they already exist */ (You just copy & paste)

```
drop table if exists Movie;
drop table if exists Reviewer;
drop table if exists Rating;
```

/* Create the schema for our tables */

```
create table Movie(mID int, title text, year int, director
text);
create table Reviewer(rID int, name text);
create table Rating(rID int, mID int, stars int, ratingDate
date);

/* Populate the tables with our data */
insert into Movie values(101, 'Gone with the Wind', 1939,
'Victor Fleming');
insert into Movie values(102, 'Star Wars', 1977, 'George
Lucas');
insert into Movie values(103, 'The Sound of Music', 1965,
'Robert Wise');
insert into Movie values(104, 'E.T.', 1982, 'Steven
Spielberg');
insert into Movie values(105, 'Titanic', 1997, 'James
Cameron');
insert into Movie values(106, 'Snow White', 1937, null);
insert into Movie values(107, 'Avatar', 2009, 'James
Cameron');
insert into Movie values(108, 'Raiders of the Lost Ark',
1981, 'Steven Spielberg');

insert into Reviewer values(201, 'Sarah Martinez');
insert into Reviewer values(202, 'Daniel Lewis');
insert into Reviewer values(203, 'Brittany Harris');
insert into Reviewer values(204, 'Mike Anderson');
insert into Reviewer values(205, 'Chris Jackson');
insert into Reviewer values(206, 'Elizabeth Thomas');
insert into Reviewer values(207, 'James Cameron');
insert into Reviewer values(208, 'Ashley White');

insert into Rating values(201, 101, 2, '2011-01-22');
insert into Rating values(201, 101, 4, '2011-01-27');
insert into Rating values(202, 106, 4, null);
insert into Rating values(203, 103, 2, '2011-01-20');
insert into Rating values(203, 108, 4, '2011-01-12');
insert into Rating values(203, 108, 2, '2011-01-30');
insert into Rating values(204, 101, 3, '2011-01-09');
```

```
insert into Rating values(205, 103, 3, '2011-01-27');
insert into Rating values(205, 104, 2, '2011-01-22');
insert into Rating values(205, 108, 4, null);
insert into Rating values(206, 107, 3, '2011-01-15');
insert into Rating values(206, 106, 5, '2011-01-19');
insert into Rating values(207, 107, 5, '2011-01-20');
insert into Rating values(208, 104, 3, '2011-01-02');
```

(b) Implement the following queries in MySQL. Each exercise asks you to create a view, and then write a query using that view, perhaps along with created views and/or the base Tables generated in Question 2(a):

i. Create a view called TNS containing title-name-stars triples, where the movie (title) was reviewed by a reviewer (name) and received the rating (stars). Then referencing only view TNS and table Movie, write a SQL query that returns the latest year of any movie reviewed by Chris Jackson. You may assume movie names are unique.

ii. Referencing view TNS from Exercise 1 and no other tables, create a view RatingStats containing each movie title that has at least one rating, the number of ratings it received, and its average rating. Then referencing view RatingStats and no other tables, write a SQL query to find the title of the highest-average-rating movie with at least three ratings.

iii. Create a view Favorites containing rID-mID pairs, where the reviewer with rID gave the movie with mID the highest rating he or she gave any movie. Then referencing only view Favorites and tables Movie and Reviewer, write a SQL query to return reviewer-reviewer-movie triples where the two (different) reviewers have the movie as their favorite. Return each pair once, i.e., don't return a pair and its inverse.

3. Put all your screenshots (query with output) in a single PDF file and upload. The PDF must comntain your name and roll no.

(10)