SQL: VIews

Before you begin please do the following:

1. Watch the video lecture posted. Many of the commands I will show you on the video.
2. You can also look at the SQLCheatsheet in the "Resources" section for many of the common SQL commands.
3. The below exercises corresponds to the lecture titled SQL Views.

This homework continues from the previous one.

Submission:
You must create a SQL file with all your answers: The SQL file should have your name in it as a comment (using #) on the first line. The following line should have your "use" command preceded by a "#" comment. The next line will contain the date and number **that results** from: `SELECT @d:=NOW(),SHA(@d);` separated by a space or TAB. This line should also be preceded by a "#" comment, followed by a blank line.
Then, for every answer, first write a "#" followed by the question number and in the following lines write your answer. Then, skip a blank line and repeat the process for the next question.
If a question has a sub-item with further sub-items, only the top two levels comprise the question number. For example, if question 4 has two parts (4.1 and 4.2) and 4.1 has 3 bullets (a,b and c), paste your response under 4.1. Do not divide this into 4.1.a, 4.1.b and 4.1.c.
If you want to add additional comments, use the -- notation. You should be able to load your entire SQL file into your IDE, uncomment the second line and run your code without **syntax** errors, at least, before submitting. Remember to comment it back prior to submission.

You will need to submit the SQL file. **Other formats will not be graded**. The SQL file ends in **.sql** you can choose whatever name you like WITHOUT spaces. A good idea is your last and first name and homework number. For example: DoeJohnHW2.sql
Once your homework is formatted, please use the format checker to ensure that the very basics of the format are ok. The format checker is not completely precise, but can help you get rid of all major formatting mistakes.

For example, a file from John Doe with 2 questions, where the first question has 2 items, may look like this:

```
# John Doe
# use jdoe_db;
# 2017-01-05 17:38:27 41860934238f8b3a1fdd51dee11e2d4219c92068
#1.1
Nothing but SQL code goes
here as the answer to
The question. SELECTs, CREATEs, whatever.
#1.2
More SQL code answering
question number 1.2 (or I.2) and all its sub points (a,b,c,d,
etc.)
The answer can take up as many lines as needed in SQL only
-- non sql line to explain a result (if asked)
-- should be commented by preceding it with -- and a space.
#2.1
Yet more SQL commands
That answer
The first part of the second question
```

Exercise

1. **Views**
   1. You employ someone to help you with reports. However, you do not want this person to see your income information (that is, transactions made **to** accounts of type 0). Create a view named *JournalR* that will show all the information of the Journal table, and **to_acct_id** account description for all transactions, except those with **to_acct_id**s of accounts of type 0. *HINT: You need to join two tables and select your attributes.*
   2. Now you want your helper to change the amount paid for rent from $800 to $650. Paste the command to do this.
   3. How would you see whether your changes took place in the view? Paste your SQL command
   4. How would you see whether your changes took place in the source table? Please paste your SQL command.
   5. You want to empower your employee. You want this person to be able to see a quick report of all total expenses (to_acct_id of acct_ids of type 1) by to_acct_id. Please provide the SQL command to create a view named *TotalAcctDetail* that shows the account id, description, category and a new column called *total_spent* with the total spent on the each of the accounts.
   6. Provide the select command to see the name of the category of each account displayed in the previous command. If there are no categories, this

field should be null. *HINT: This is an outer join*. *Hint#2: you must use the view you just created.*

7. You decide to create another view, called *ExpensesVBudget*, that will provide information about total expenses versus budgeted money. Your view shows the following information: category name, budgeted money, total money spent, and whether there is a warning of overspending (using the **warn_percent**) The column with the warning should be a Boolean column called ***AtRisk***. Please paste the SQL commands to do this. *Hint: You can compute the amount for the warning by doing warn_amt*max_amt/100. The Boolean column is simply a comparison between this amount and the total spent on the category.*
8. Now, change the budget warning percent for clothing to 99%. Check whether the *ExpensesVBudget* view changes. Explained how it changes (if it does) and how.As always, use -- comments to answer.