# Assignment 3: Using SQL to extract data to answer business questions

# Instructions:

* **Do your own work:** This is an individual assignment and you must do your own work and create your own SQL statements. If you are caught cheating on this or using someone else’s work, you will receive a zero on this assignment and be reported to the Dean of Students. Also, this homework prepares you for the exam coming up so doing the work now will help you learn and do well on when it counts more.
* **What to turn in**
  + Clearly separate your code for each question. Save your code into one SQL file with the naming format: LastName\_FirstName\_UTEid. Please make sure the lastname and firstname you use matches what is in Canvas.
  + **Save your file either as a .sql file or as a .txt file. If you need help doing this, refer to the page linked in the Canvas assignment. Files saved in a different format will be 50% and files in a different format that cannot be read into SQL (example: PDF) will result in a 0%.**
  + Submit your .sql file on Canvas before the deadline. Late submissions receive 50% off. No submissions will be accepted 24 hours after the deadline.
  + Do not include the DDL in your submission. If you do, you will lose 5 points. Only provide SQL with comments and nothing else. Do this going forward on all other assignments unless noted.
* The SQL problems below will be based on the DDL script that is posted on the Canvas instructions. Download that script and run it before you start.

# Problems:

1. Write a SELECT statement that returns following columns from the *User* table: first\_name, last\_name, email, birthdate. Then, run this statement to make sure it works correctly.

Add an ORDER BY clause to this statement that sorts the result set by last name in ascending order. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time.

1. Write a SELECT statement that returns one column from the *User* table named user\_full\_name that combines the first\_name and last\_name columns.

Format this column with the first name, a space, and last name like this:

John Doe

Sort the result set by first name in descending sequence.

Return only the users whose last name begins with letters of K, L, and M.

1. Write a SELECT statement that returns these columns from the *Video* table: title, subtitle, upload\_date, views, and likes. Return only the rows with an upload date between the beginning of this year and last week (i.e. between ’01-Jan-20’ and the date one week before this assignment is due). Use the BETWEEN operator. Sort the result set in descending sequence by the upload\_date column.
2. Create a duplicate of the previous query but this time update the WHERE clause to use only the following operators (<, >, <=, or >=). Keep the rest of the query the same.
3. Write a SELECT statement that returns these column names and data from the ***Video*** table:

video\_id The video\_id column

video\_size The video\_size column but with a column alias of ***video\_size\_MB***

likes The likes column with a column alias of ***Likes\_Earned***

video\_length The video\_length column with a column alias of **video\_length\_sec**

video\_length\_min The is a calculated column based on the logic below\*.

\*Divide the video\_length (currently in seconds) by the number of seconds in a minute to find video length in minutes.

Use the ROWNUM pseudo column so the result set contains only the first 3 rows from the table.

Sort the result set by the column alias ***Likes\_Earned*** in descending order.

NOTE: You should return only one decimal place for video\_length\_min. You *could* use ROUND but for this column we would prefer truncating (i.e. remove decimal) to see how many minutes are required at a minimum to watch most of the video.

1. Now copy the above statement, but adjust it to answer the following question: Which videos (by video name) have the highest number of likes and are at least 6 minutes long? Remove all non-relevant columns from your query.
2. Write a SELECT statement that returns these column names and data from the ***video***  table:

cc\_id The cc\_id column

video\_id The video\_id column

Popularity The likes column but with a column alias

Awards The likes column calculated as described below (ideally would be truncated)

Post\_date The upload\_date column but with a column alias

Only return rows with where the video has earned more than 10 awards. An award is earned every 5000 likes that accrued.

1. Now write a query that shows the above information and includes the full name of the user that created these videos. Do not return any column except the full user name.
2. Write a SELECT statement that uses the SYSDATE function to create a row with these columns:

today\_unformatted The SYSDATE function unformatted

today\_formatted The SYSDATE function in this format: MM/DD/YYYY

This displays a number for the month, a number for the day, and a four-digit year. Use a FROM clause that specifies the Dual table. *Hint: You will need to implement the TO\_CHAR function to format the sysdate in the format designated above.*

After you write this add the following columns to the row:

likes 1000

pay\_per\_like .0325

pay\_per\_video 10

pay\_sum The likes multiplied by the pay\_per\_like

video\_total The pay per video plus the pay\_sum

Your result table contains only one row.

1. Now return a table that shows these pay columns for the videos in the video table. Include upload date and format as MM/DD/YYYY. Show the videos that earn the most revenue first.
2. Write a SELECT statement that returns the first and last name of users, their birthdate, their status, and the comments they have made. We want to see the longest comments first.
3. Write a SELECT statement that pulls the user and their topic subscription info. Then display the following columns from the results:

user\_id The user\_id column

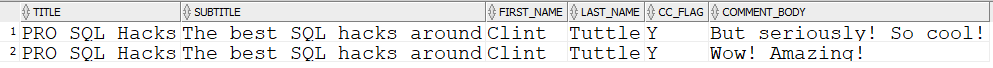
user\_name A concatenation of user first and last names.

topic\_id The topic\_id column

topic\_name The topic\_name column

Return on the subscription details for the user(s) subscribed to the “SQL” topic.

1. Write a SELECT statement that joins the appropriate tables in order to show video title, subtitle, user first\_name, last\_name , whether the CC\_Flag is checked or not, and comment\_body. Pull this for only video\_id 100000 and then sort by last\_name and first\_name. Your results look like something like this (for a different video):



1. Write a SELECT statement that pulls the distinct first name, last\_name, and email for users that have not commented on a video yet. Use a left outer join to accomplish this to show you understand how to properly use them. Sort them by last name.
2. Use the UNION operator to generate a result set consisting of three columns from the ***Video*** table:

video\_tier A calculated column that contains a value of ‘1-Top-Tier’, ‘2-Mid-Tier’, or ‘3-Low-Tier’

video\_id The video\_id column

revenue The revenue column

views The views column

If the video has at least 30000 views, the video\_tier column should contain a *literal string* value of ‘1-Top-Tier’. If the video has between 20000 views and 30000 views the video\_tier should contain a *literal* string value of ‘2-Mid-Tier’. Otherwise, it should contain a value of ‘3-Low-Tier’

Sort the final result set by *revenue* descending.

1. Write a SELECT statement to identify the most successful content creator (cc\_username) in terms of revenue. Is that the same person that is most successful in terms of rewards?
2. Show the distinct different card types that each content creator has with their first and last name.