

Applying SQL #7

(45 points • estimated. time 2-3 hours)

Background

This draws on the material from chapters 13-15 of our Oracle SQL text. You will work with sequences, indexes, views, and the data dictionary. You will also create an *interactive* script that retrieves data based on a value the user enters.



By the end of the assignment, you will have developed a script that resembles the illustration. The top line is a comment containing your name. The remaining lines are pairs – a comment to identify the task number and the SQL statement which accomplishes the task.

Preliminaries

- Launch *SQL Developer* and connect to our *cis119do* database.
- Save your script (worksheet) as *applying_SQL#7_your_name.sql* to the **h:\cis119do** folder (if connected via mySCC) or to **c:\cis119do** (if connected via your own PC).
- Enter the top two lines (the comment with *your* name, and the comment to identify Task 1).

Tasks

Develop a statement for each task below. Build one statement at a time and run it to test that it works correctly. If not, tweak, run, and test the statement until it works correctly. *Save your script frequently.*

[**Note:** You won't run your entire final script in the end because it would attempt to make modifications that you've already made when you craft each statement as you accomplish each task. Just be sure each statement works correctly as you go.]

- Create a sequence named `countdown_seq` that generates values from 5 down to 1, then resumes at 5 and counts down to 1 (i.e., generates the values 5, 4, 3, 2, 1, 5, 4, 3, 2, 1, 5, 4,...).
- Use commands to generate the first seven (7) values of your `countdown_seq` sequence. (i.e., generate 5, 4, 3, 2, 1, 5, 4). **Hint:** generate the first value, and then run the same command 6 additional times to ensure it works as described.

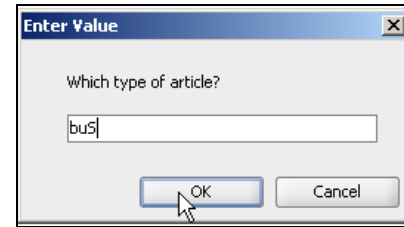
NOTE: The next two tasks must be done in the same session.

- Suppose you've been hired as a new instructor. Use an `INSERT` statement to add yourself to the `instructor` table. Your statement must use `instructor_id_seq` to generate the value for your `instructor_id`. It must also automatically determine the current date/time for the `created_date` and `modified_date` columns.
- Now that you've inserted yourself as an instructor, you've been assigned to teach `SECTION_ID` #122. Use an `UPDATE` statement to assign that section to you. The `UPDATE` statement must use the `instructor_id_seq` to determine the `instructor_id` that was generated when you were inserted into the `instructor` table in the previous task. It must also set `modified_date` to the current date/time.
- Use the data dictionary to confirm the existence of each sequence in your schema. Show each sequence's name and its minimum, maximum, increment, and last-value-generated values.
- Delete your `countdown_seq` sequence.
- Create a B-Tree index for *each* foreign key in your `article` table.

8. Use data dictionary views to show details about each index associated with your `article` table. For each index, show its name and uniqueness, along with the field(s) it is associated with. Accomplish this in one statement that employs a join.
9. Use data dictionary views to display, for only the `writer` table, each constraint's name, the field(s) the constraint pertains to, the text of the constraint (if applicable), and, depending on the type of constraint, either the phrase "Primary Key", "Check" or "Foreign Key". Accomplish this in one statement that employs a join.
10. Create a view named `business_writers_vu` that shows an unduplicated list of writers who have written a BUS article. Each writer who has written a BUS article should be shown only once, even if they've written multiple BUS articles. Show each writer's concatenated full name, phone number and last contact date. The view must use a *non-correlated* subquery, not an equijoin. Test your view and refine it until it works correctly.
11. Use the data dictionary to confirm that your `business_writers_vu` view exists and show the entire stored select statement the view represents. Show only the `business_writers_vu` view.
12. Use a command to allow user `ttrollen` to employ `SELECT` statements against your `business_writers_vu` view.
13. Use the data dictionary to identify users who have privileges on your `business_writers_vu` view. Show each user's name and the privilege(s) they possess.
14. At this point your `applying_SQL#7_your_name.sql` script is complete. Save and **print** your final script.
15. Close your script.

Interactive Script


The remaining Tasks create a new, *interactive* script that will prompt the user to enter a 3-byte code for a TYPE of article...



... and then display `TITLE`, `ISSUE` and `LENGTH` for articles of the specified type...

TITLE	ISSUE	LENGTH
\$100 Billion AOL Time Warner Merger Approved	01-JAN-01	1702
AT&T Antitrust Settlement	01-FEB-82	1600
Building Trade Outlook	01-APR-84	1437
Dot-Coms Go Bust	01-JAN-01	1830
The Economy Under Sub-Zero Population Growth	01-DEC-95	1020
Trans-Alaskan Oil Pipeline Opening	01-JUL-77	803

Your script must work whether the user enters the 3-character type code in uppercase, lowercase or mixed case (eg: **BUS** or **bus** or **bUs**). The rows should be sorted by `TITLE` and the substitution details must not be visible in the *Script Output* pane.

16. Open a new worksheet window either by clicking  or by pressing `[ALT] + [F10]`.
17. Enter comment lines at the top of the script to document the name of the script and your name.
18. Save the new script as **genre.sql** to the `h:\cis119do` folder (if connected via mySCC) or to `c:\cis119do` (if connected via your own PC).
19. Now enter `SQL*Plus` commands and an `SQL` statement to provide the described functionality.
20. Run and test your **genre.sql** script using **buS** as the type of article (as illustrated above). Modify and re-test until your script works as described.
21. Save and **print** the final version of your interactive **genre.sql** script.

Wrap Up

- A. If you submit your assignment via email, attach your `applying_SQL#7_your_name.sql` and `genre.sql` scripts.