WEB PA Practice

Description

This doc contains some practice tasks. There is no dependency between them, so you can start anywhere.

Where you see `[something]` like this, it means this should be a parameter, received from the client via URL. And `this` is an inline code.

Please contribute! If you have a good idea in your mind, feel free to add it. You have suggestion mode, you can't make any harm 😎

Codecool Series Entry level (KATAs)

These should be done in under 30 minutes - if it is multiple hours for the first time but you can start again and again and it gets below 30 minutes then it is awesome.

1. **SQL**: List all shows and the number of episodes for each show.
   JS: On page load, add a light green background to shows with an even number of episodes and light blue for odds.
   CSS: Use CSS classes for the backgrounds.
   Python: Add an extra `is_long` key to all shows, which is true if the show contains at least 100 episodes.
   Jinja: Show the `fa-expand` Font Awesome icon when the `is_long` key is true, or `fa-compress` icon when it's false.

2. **SQL**: List the top 10 actors who played the most characters.
   Jinja/Python: Show these people in an ordered list.
   Jinja/Python: Add a golden, silver and bronze colored medal next to the 1st, 2nd and 3rd actors
   JS: When the user clicks on a medal, it should do some animation. Make it bigger for 2 seconds without modifying anything else on the page, or make the name blink, or something crazy. (a choo-choo train can go on the screen if you like)
   Python: calculate how many roles these 10 people acted in and pass it to the template as a separate variable. Show it before the list: "Here are the 10 most

active actors (they played xyz characters)”

3. **Jinja/Python**: Make a new page, display 2 number fields with “season” and “episode” labels.
   SQL: Search shows which have minimum the given amount of `[seasons]` and/or `[episodes]`.
   Jinja/Python: Display the number of shows matching this search.
   SQL + Jinja/Python: Modify your previous page and query to display the number of shows matching the search grouped by genres.

PA-like

Task combinations which should be possible in 1 hour.

1. **Stars for the stars**
   SQL: List the top 10 rated shows which implements the given `[genre]`! Display the show's title, starting date and rating in an ordered list. **Jinja/Python**: The rating should be shown as full stars, using FontAwesome icons. (You’ll need to round the values) ☆

   6.4 - ★ ★ ★ ★ ★ ★ ☆ ☆ ☆

   8.2 - ★ ★ ★ ★ ★ ★ ★ ★ ☆ ☆

   JS: Hovering the mouse on the stars should behave like on IMDB. When you hover on one of them, all of the stars on the left and the hovered one should be filled with stars, and the others on the right should not be filled-in stars. When there is no hover anymore, the rating should show the original state.

2. **What a year!**
   SQL: List all years between 1970 and 2010. Add a column of average rating of all shows started that year. Also add a column containing the number of shows started that year.
   Jinja/Python: Display all years and their data as cards next to each other. There should be many of them in a row. Display them like this plan:

   CSS: Display as many fixed size cards as many fits in a line.
   JS: Make a multiselect: On clicking on a card, change the background of that card, and display a rating aggregation above the year cards. So when you selected 3 cards, display the average rating of these 3 cards in a paragraph. (add

some text to it to make it user friendly)

3. **Longer show, more fun**
   SQL: List the top 10 longest shows by total runtime (how long would it take to watch all
   episodes)
   SQL: Collect all actors played in these shows ordered by name [separate query]
   Jinja/Python: Display the results from the 2 queries next to each other as 2 separate tables.
   JS: When the user hovers on one of the lines of any tables, highlight the related entries in the other table. So when you hover a show, highlight the actors in it, and when you hover an actor, highlight the related show.
4. **Have you seen my trailer?**
   Jinja/Python: On a new page, display a card with a form, containing a search field to allow the user to search shows by title.
   SQL: List all shows containing the given `[search string]`.
   Jinja/Python: Display the results of the query in a table, inside a card below the search possibility. Show the title, rating and start date of the show. Add a button, if the show has a trailer saved: “Watch trailer” which will be used in JavaScript.
   JS: When the user clicks on a “Watch trailer” button, display a modal with the embedded trailer video. Make the modal closable.

5. **Most active actors**
   SQL: Collect all actors with their name, age (or age at death, if deceased) and number of shows the actor played in. Order the most active actor on top of the result.
   Jinja/Python: Display all these actors on a new page as fix width cards: ...
   SQL: Extend your query to determine if the given actor is still alive.
   JS: When the user hovers the cursor on any of these actors do the following:
   If alive: Make the card background green
   If deceased: Make the card background dark and the text color white.

6. **Nice list of average actors**
   SQL: List all actors' name and birthday, who were born after “`[year]`-01-01”. For every actor list a number which means how many characters they played. For every actor list their shows' average rating.
   Jinja/Python: Create a page with an input field, a submit button and a well-formatted card-set with the results. (You can use Bootstrap if you want) The input field should serve as a filter: if you type a date and press the button, page should reload and show the actors who were born after the given date.

7.

   JS: Clicks on ratings increase them by 0.1, right clicks on them decrease by 0.1 and for right clicks context menu should remain hidden.

8. **Genrestic**
   SQL: List all shows using the given `[genre]`.
   Jinja/Python: Display each show title in separate cards.
   SQL: Extend the query with the number of characters playing in these shows.
   Jinja/Python: Display the character count in the same row as the title, but on the right side of the card.
   JS: When the user double clicks a card, the background should change according to the number of characters. If there are an odd number of characters, change it light green, otherwise it should become light blue. *(use CSS classes, not inline style)*

9. **Young actors in old shows**
   SQL: List all actors name and age, who played in shows released in the given `[year]`. Order them by age, and the older person should be on the top.
   Python: Add an extra parameter to the results, which indicates if the actor was older when the show was released than the shows current age.
   Jinja: Display the name of these actors in an ordered list in a new page. The page's welcome text should be "Actors of shows released in `[year]`".
   JS: When the page loads, highlight the people, who were older at release than the show's current age with a light yellow background.
   JS: By clicking on a card, display the age of the actor at show release and the shows current age in an alert (or nicer modal, if you like)

10. **Character fuzzy search**
    Python/Jinja: Make a new template with 2 card elements. The first one should have a form with one longer input field, and a "Search" button (Like Google). The second one will show our results later.
    Python: Handle the input of the user and add a "`%`" character between each word of the searched phrase.
    SQL: Query the characters matching the modified input.
    SQL: Extend the query with some more details: Title of the show and the real name of the actor
    Jinja: Show the results in an unordered list inside the second card element. Format it like this: *Emilia Clarke played **Daenerys Targaryen** in Game of Thrones*

JS: Modify the logic to handle searching with AJAX, using the `fetch` API. And make sure the page does not reload when the user searches.

11. **Season Ranger c**
    SQL: Calculate the highest number of seasons the database knows about.
    Python/Jinja: Add a *range input* to the top of the default index page, above the list of shows. It should allow the user to select a value between 1 and the highest season number.
    JS: As the user changes the value of the range input, hide all shows with less seasons than the selected value. *So when you select 5, only display shows with 5 or more seasons.*

12. **Longest shows in a genre**
    Python/Jinja/SQL: On a new page create a dropdown with all genres and a button next to it.
    SQL: Write a query that lists 50 shows from a genre (by `[genre.id]`) that have the most, but at least 20 episodes, ordered by the count of episodes.
    JS: When clicking the button next to the drop down, start an AJAX call with the `genre id` as a GET parameter. After receiving the data show it in a <table> with these columns: show title, season count, episode count

13. **Shows? Season? Episodes?**
    Python/Jinja: Create a new page on a new route from the main page. New page should contain two dropdown-lists with the following elements
    (1){shows/seasons/episodes}
    (2){ascending/descending}.
    The page shall display the selected objects {shows/seasons/episodes} in a selected order {ascending/descending} in a card view, 4 cards in a row.

    SQL: create a query to get the {shows/seasons/episodes} in a proper order from the database.

    Javascript: while the mouse pointer is on the card, the card should contain the *details* of the object {show/season/episode}. If there is no description “Overview not found!” shall be displayed.

    EXTRA: Try to hack the query with SQL injection. By that point you probably realized that these kinds of parameters require extra consideration. Are you sure sql.SQL creates a safe parameter? Try to hack, but at the same time be sure at the end of the task you have a safe query.

14. **Show of the year - NOT READY**
    SQL: Write a query that collects the earliest and latest years when a show was released.
    Jinja/Python: Make a new template with one card. It should contain a range input with the lowest release year on the left of the range and the highest on the right. Add a button “Get show of the year” in the next line.
    JS:
    SQL: Collect all shows titles, release date, rating and the average age of actors
    Jinja/Python: Store all shows and these details in the HTML without showing

Advanced

Do you aim for the green? Here are some exercises for you:

1. **SQL**: List all actors and their average rating. This is the average rating of all shows an actor worked on.
   Jinja/Python: Display all data. The rating should be rounded to the next half and displayed as stars:
   6.4 - ★★★★★★½☆☆☆
   8.2 - ★★★★★★★★☆☆
2. Do the **Character fuzzy search** exercise, then extend it with:
   JS: Highlight the letters of the original search inside the character name when the user hovers the pointer over any of the results. Make sure to highlight them in the order, as the user searched. (*hint: string.indexOf may help*)
   Like this:
   Search input: **nerarn**
   Good: *Emilia Clark played **Daenerys Targaryen** in Game of Thrones*
   Bad: *Emilia Clark played **Daenerys Targaryen** in Game of Thrones*

Other (not reviewed by mentors)

1. Írassuk ki a horror sorozatok azon színészeinek a nevét, akik már nem élnek, továbbá a születési és a halálozási dátumokat is.
   JS - ha rákattintasz a sorra, felugró ablakban írja ki, hogy mennyi ideig élt.
2. Írassuk ki minden vígjáték sorozat második évadának a 4. és az 5. részének a címét.
   JS - ha rákattintasz a sorra, tűnjön el.
3. Írassuk ki az összes műfajt, átlagolt rating alapján rendezzük sorba.
   JS - ha a ratingre kattintasz, felugró ablakban jelenjen meg a hozzá tartozó genre.

4. Írassuk ki azokat a színészeket akik már nem élnek, egy **[url-ből bekért]** genres-ből, a második oszlopban pedig évben megadva, hogy mennyit éltek.
   JS - a sorra kattintva színezze zöldre csak a színészek nevét.
5. Írassuk ki a színészeket, a második oszlopban pedig azt, hogy milyen műfajú dolgokban játszottak, vesszővel elválasztva, ABC szerint sorba rendezve.
   JS - a színészek nevére kattintva legyen text input a mező, benne a színészek nevével.
6. List all actors whose first or last name starts with an “a” letter.
7. List all actors whose first and last name start with the same letter.
8. List the top 10 deceased actors who played the most characters, show the character name, actors names, age of death, and the list of shows separated by commas.
9. List all tv-shows and their actors average age(compared to the tv-show release date, ! not to the actual date), which have an average age at least equal to 50.
   JS - clicking on actors column, all the belonging actors name should change to red, clicking again should change it back.
10. List the 20 oldest actors who played in at least 3 shows.
    JS - Clicking on an actor change the bg and write out the average age of the clicked actors somewhere.