Data 100, Spring 2024

Exam Prep Section #9

Pandas2SQL

1. Convert each `Pandas` expression to SQL assuming `data` represents a `Pandas DataFrame` containing 3 columns: `name`, `rank`, and `year`. Both the rank and year are stored as integers.

   (a) `data.loc[data["rank"] < 10, "name"].value_counts().reset_index()`

   Hint: Remember that `value_counts` returns a sorted output!

   (b) `data.merge(data, on = "name").sort_values(by = "name")`

   (c)
   ```
   data.groupby(["name", "year"]) \
       .filter(lambda sdf: len(sdf) > 5) \
       .groupby(["name", "year"])["rank"] \
       .min().reset_index().head(5)
   ```

Regular Ice, SQLite Sugar

2. Sean's Data 100 study group is holding a mid-semester social, and he is tasked with getting boba. Since it's a Data 100 event, he surveys staff members about the boba shops they've been to in Berkeley, and compiles the data into a table named `BobaReview`. Each row is a person's review of a boba shop; the first few rows are shown below.

reviewer	role	shop_name	rating	price
Srikar	staff	Boba_Ninja	2.0	5.0
Stephanie	staff	Asha	4.5	5.0
Parth	student	Asha	5.0	5.0
Rahul	staff	One_Plus	4.5	4.5

(a) Sean wants to find all the reviews that rated a shop at least 4 (out of a maximum rating of 5). Help Sean by writing a one-line SQL query that outputs all rows of `BobaReview` where a boba shop was rated **at least** 4 out of 5.

(b) Sean now wants to find the 5 top-rated boba shops in Berkeley. Help Sean by writing a SQL query which constructs a table `BestBobaShop` from `BobaReview` that selects the 5 boba shops with the highest average rating. Your table should have three columns labeled `shop_name`, `avg_rating`, and `avg_price`, which are the shop name, average rating, and average price paid for boba, respectively. Rows should be sorted by the average rating, with the highest rated boba shop as the first row. The first few rows of `BestBobaShop` are shown below:

shop_name	avg_rating	avg_price
Asha	4.75	5.0
One_Plus	4.5	4.5

Cartoon SQLs

3. The first 5 rows of a table `survey_data` are shown below

score	country	city	n_movies_per_yr	pref_cartoon_character
2	Italy	Rome	3	Popeye
2	Germany	Berlin	6	Popeye
3	Ireland	Clonakilty	8	Homer Simpson
3	India	Coimbatore	1	Popeye
3	United States	Los Angeles	5	Popeye

Each row represents one student response to the survey. `score` represents the student's score on the survey out of 3, `city` and `country` represent the self-reported location where the student currently resides, `n_movies_per_yr` represents the number of movies the student watches in a year, and `pref_cartoon_character` represents the student's favorite animated character.

(a) We want to find out which cartoon characters are most popular among Data 100 students in the United States. Write a SQL query that returns the most popular preferred cartoon character among all students living in the `United States`, in descending order of number of occurrences in the survey data.

The output should resemble the table shown below, with rows corresponding to each preferred cartoon character and respective counts. Note that the column names (aliases) must match.

pref_cartoon_character	count
Mickey Mouse	18
Bugs Bunny	18
Popeye	17
Homer Simpson	14
Fred Flintstone	14

(b) We wish to explore this relation further and load a table `cartoon_origin` showing the origin for each cartoon character. A sample of 5 rows from this table are shown below.

character	origin_country
Mickey Mouse	United States
Arthur	Canada
Franklin	Canada
Homer Simpson	United States
Mr. Bean	United Kingdom

Write a SQL query that shows the proportion of students living in each country from the survey data that chose cartoon characters originating from that country, ordered from greatest to least. For cartoon characters not listed in the cartoon origin table, assume by default they do not originate from the student's country.

A sample of the output is shown below, where rows correspond to each country and the desired proportion per the question specification. Note that the column names (aliases) must match.

country	prop
Germany	0.322222
United States	0.283186
South Korea	0.192661
Ireland	0.157025

(c) We further discover that Data 100 students are from all over the world! He decides to find using SQL which students' cities contain the word 'City' in them. If the city contains the word 'City', then we want to output 'yes'; otherwise, we want to output 'no'.

Fill in the blank below to accomplish this.

```
SELECT city, ________________________________________
_________________
_________________
_________________
FROM survey_data;
```