3 2023 CAPP Databases Midterm A

This exam was given to Master's level students in the University of Chicago CAPP program in the Spring of 2023. There were two versions of the exam – this is version "A".

The following table contains information about campers at a camp. You can assume each name uniquely defines a camper and that a camper only appears once in the table.

- **name:** The name of the camper (string, NOT NULL).
- **age:** The camper's age, in years (integer, NOT NULL).
- **hgt:** The height of the camper (in centimeters) (float, NOT NULL).
- **state:** The state that the camper is from (string, NOT NULL).
- **program:** The specific program (elective) that the camper choose. You can assume this is all lower case (string, NOT NULL).
- **amt_paid:** The amount they paid to attend camp (float, NOT NULL).
- **allergy:** If a camper has a food allergy (if Null that means no allergy). You can assume this is all lower case. There will be only a SINGLE allergy listed (string, HAS NULLS).
- The name of the table / DataFrame is **camper**. No need to use a schema or load the DataFrame.
- Only use syntax covered in class.
- Interpret all inequalities as strict unless explicitly stated.

Figure D.1: *camper* Table: 12,345 Rows

name	age	hgt	state	program	amt_paid	allergy
Ringly Roberson	7	121.0	NY	basketball	987.54	
Crash Bandicoot	11	144.5	MS	basket-weaving	1128.75	peanuts
Alligator Reynolds	8	129.0	PA	skateboarding	585.46	

SQL Section

Please answer the following questions making sure to return *only* the information requested.

1. Using SQL, write a query which returns the names (name only) of the top-8 shortest campers who are 10 years old.

```
SELECT
        name
FROM
        campers
where age = 10
order by hgt asc limit 8;
```

2. Using SQL, write a query which returns all campers (name only) who are younger than 10 years old and are either from New Jersey ('NJ') or Wyoming ('WY'). Only include those campers who do NOT have a food allergy.

```
select name from campers where
        state in ('NY', 'AL')
        and age < 10
        and allergy is null;
```

3. Write an SQL query which returns the number of campers from each state who have food allergies. This should be two columns: one with the state and the other with the number of campers from that state who have food allergies.

```
SELECT
        state, count(1) as ct
from
        campers
where injury is not null
group by 1;
```

4. Write an SQL query which returns all rows and columns for campers who are taking "basketball" as their program (you can assume that all programs are lowercase). Sort them from tallest to shortest.

```
select * from campers
where program = 'basketball'
order by hgt desc;
```

5. We define an "allergy impacted" program as one with 10% (or more) of the campers in that program having a food allergy or any kind. Write an SQL query which returns a list of programs which are "allergy impacted". This should return 1 column with a list of "allergy impacted" programs.

```
select program from
        (select program
                , sum( case when allergy is not null then 1 else 0 end)::float / sum(1) as rat
        from campers
        group by 1 ) as innerQ
where rat >= .1
```

6. We calculate the age-adjusted height ("AAH") by taking a campers height and dividing it by their age squared ($\frac{height}{age^2}$). Write a query which returns all rows and three columns: age-adjusted height, name and program.

```
select hgt /age / age as aah, name, program
from campers;
```

7. Using SQL, write a query which returns three columns: name, program, and AAH_Flag. AAH_Flag should be equal to 0 if the AAH is less than or equal to 1, 1 if the AAH is greater than 1 and less than or equal to 3 and 2 otherwise. AAH is defined in the previous problem.

```
select
        name, program
        , case
                when hgt / age / age  <= 1 then 0
                when hgt / age / age <= 3 then 1
                else 2 end as AAH_Flag
from
        campers;
```

8. If a person's AAH is greater than or equal to 3 they are defined as "tall". Write a query which returns the *percentage* of campers of each program who are tall. This should have two columns: program and percentage of the campers in that program who are tall.

```
select
        program,
        sum( case when hgt / age / age >= 3 then 1 else 0 end )::float / count(1) as pctTall
from
        campers
group by 1;
```

9. Using SQL, write a query which returns one row and two columns. The first column should be the number of campers who are 10 years old (exactly) and signed up for the "basketball" program (call this column bb10). The second column should be the number of campers who are 7 years old (exactly) who are signed up for the "skateboarding" program (call this column sb7). You can assume that all programs are lower case.

```
select
        sum( case when program = 'basketball' and age = 10 then 1 else 0 end) as bb10
        , sum( case when program = 'skateboarding' and age = 7 then 1 else 0 end) as sb7
from campers;
```

Pandas Section

Please answer the following question, making sure to return only the information required. You can assume that a DataFrame named *campers* is already loaded. If a specific output is not specified you can return anything (DataFrame/Series/List/Array/etc.)

1. Using Pandas, return the name (as a Series) of the top-8 shortest campers who are 10 years old.

```
campers.loc[(campers.loc[:, 'age'] == 10), :].nsmallest(8, 'hgt').loc[:, 'name']
```

   OR:

```
(campers
        .loc[(campers.loc[:, 'age']== 10), :]
        .sort_values( 'hgt', ascending=True)
        .head(8)
        .loc[:, 'name']
)
```

2. Using Pandas, return a DataFrame with two columns: name and state. The dataset should only contain campers that are either (a) over 10 years and from Virginia ("VA") or (b) under 8 years old and from Michigan ("MI").

```
campers.loc[((campers.loc[:, 'age'] > 10) & (campers.loc[:, 'state'] == 'VA') )
        | ((campers.loc[:, 'age'] < 8) & (campers.loc[:, 'state'] == 'MI') )
        , ['name', 'state']]
```

3. Using Pandas, return a **DataFrame** which contains all campers (name only) who are younger than 10 years old and are either from New Jersey ('NJ') or Wyoming ('WY'). Only include those campers who do NOT have a food allergy.

```
campers.loc[(campers.loc[:, 'age'] < 10)
        & (campers.loc[:, 'allergies'].isna())
        & (campers.loc[:, 'state'].isin( ['AL', 'NY']))
        , ['name']]
```

4. Using Pandas, return all programs (this should be without duplicates) which have a camper who has an allergy to "peanuts". You can assume that all allergies in the table are lower case.

```
campers.loc[ (campers.loc[:, 'allergy'] == 'peanuts'), 'program'].unique()
```

5. Return all rows and columns for campers who are taking "basketball" as their program (you can assume that all programs are lower case). Sort the resulting DataFrame first by state (alphabetically) and then, within state, from tallest to shortest.

```
(campers.loc[ (campers.loc[:, 'program'] == 'basketball'), :]
        .sort_values( ['state', 'hgt'], ascending=[True, False])
        )
```

6. Please return a DataFrame which has all the original data and adds a column called "AAH" which is the age-adjusted-height (this is height divided by age squared, as in the previous problems).

```
campers.loc[:, 'aah'] = campers.loc[:, 'hgt'] / campers.loc[:, 'age'] / campers.loc[:,  'age']
```

7. Please return a DataFrame which has all the original data as well as adds a column called "hgt_flag" which is equal to 0 if the camper is greater than or equal to 140cm, 1 if they are greater than or equal to 110 and less than 140 and 2 otherwise.

```
campers.loc[:, 'hgt_flag'] = 2
campers.loc[ (campers.loc[:, 'hgt'] >= 140), 'hgt_flag'] = 0
campers.loc[  (campers.loc[:, 'hgt'] >= 110) & (campers.loc[:, 'hgt'] < 140), 'hgt_flag'] = 0
```

8. There was an error and students who were 10 years old all had their height recorded as 10 centimeters too high. Please return an updated version of the campers DataFrame which has this error fixed. Specifically the DataFrame should have all rows and columns, but the `hgt` column should have this error fixed.

```
campers.loc[ (campers.loc[:, 'age'] == 10), 'hgt'] = campers.loc[ (campers.loc[:, 'age'] == 10), 'hgt'] -10
```