1 2023 CAPP Databases Final 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 tables contains information about Uber Eats drivers, their deliveries and reviews. Keep in mind that (a) not all deliveries will have reviews. A delivery can have, at most, one review. Not all drivers will have deliveries. When a driver first signs up they will not have any deliveries or reviews

- Only use syntax covered in class. Do not create any views.
- Interpret all inequalities as strict unless explicitly stated.
- If there is no specified return format (DataFrame/Series/etc.) than any format will be accepted.
- If you provide more than one answer, the lower of the two scores will be counted.
- Any two columns with the same name can be assumed to match.
- Columns in the *drivers* table / DataFrame:
  - **driver_id:** The ID of the driver (INT, UNIQUE, NOT NULL).
  - **state:** The state that the driver lives in (STRING, NOT NULL).
  - **age:** The age of the driver in years (INT, NOT NULL).
- Columns in the *reviews* table / DataFrame:
  - **del_id:** The ID of the delivery being reviewed (INT, NOT NULL).
  - **review:** The review score on a 1 (worst) to 5 (best) scale (INT, NOT NULL).
- Columns in the *delivery* table / DataFrame:
  - **del_id:** The unique ID associated with the delivery (INT, UNIQUE, NOT NULL).
  - **del_date:** The date that the delivery occurred (DATE, NOT NULL).
  - **driver_id:** The ID of the driver (INT, NOT NULL).
  - **car:** A flag (1/0) for if the driver used a car to make the delivery (INT, NOT NULL).
  - **length:** The distance that the delivery took, in km (FLOAT, NOT NULL).

driver_id	state	age
1	CA	27
2	MN	37
3	CA	28
4	CA	32

Drivers (1,234 Rows)

del_id	review
1	5
23	4
35	4
45	1

Reviews (980 Rows)

del_id	del_date	driver_id	car	length
1	1-1-2012	45	0	1.25
2	12-23-2012	45	1	23.45
3	7-6-2013	112	1	11.17
4	5-5-2014	1125	0	.75

Deliveries (14,365 Rows)

SQL Section

1. Write a query which returns the 7 oldest drivers (*driver_id* only) from the state of Michigan ("MI").

```
select driver_id
from drivers
where state = 'MI'
order by age desc
limit 7
```

2. Write a query which returns three columns: (1) the state, (2) the total number of *drivers* (count) from that state (only including drivers who have one or more deliveries) and (3) the total number of *deliveries* (count) from that state. This should return one row per state.

```
select
        drivers.state
        , count( distinct drivers.driver_id)
        , count( deliveries.del_id)
from
        drivers
join
        deliveries
using( driver_id)
group by state
```

3. Write a query which returns one row per delivery and four columns. The first column should be the state of the driver, the second should be the *driver_id*, the third should be *del_id* and the fourth should be the total length that the driver has travelled up to and including that delivery (cumulative sum of *length*). Make sure that the cumulative sum is calculated by the date of the delivery from earliest to latest. If a driver does not have any deliveries they should *not* be included in the results.

```
select
        driver_id
        , del_id
        , state
        , sum( length) over(
                partition by driver_id
                order by del_dt asc
                rows between unbounded preceding and current row
                ) as cum_sum
from
        drivers
join
        deliveries
using( driver_id )
```

4. Write a query which returns two rows and two columns. The first column should be state and the second should be the number of 5-star reviews for deliveries from that state. Only include Michigan ("MI") and Pennsylvania ("PA").

```
select
        state,
        count( case when review = 5 then 1 else null end ) as num_five_stars
from
        (select * from drivers where state in ('MI', 'PA')) as lhs
left join
        deliveries
        using( driver_id)
left join
        review
        using( del_id)
group by state
```

5. We want to return the average length of deliveries depending on if the driver used a car or not. Write a query which returns two rows and two columns. The first column should be if the delivery person used a car or not (the *car* column 1/0 flag) and the second column should be the average length of a delivery with that particular flag value. Only include those observations from the year 2012.

```
select
        car,
        avg( length)
from
        deliveries
where
        date_part('year', del_date) = 2012
group by 1
```

6. Please return one row and two columns. The first column should be the average length of deliveries when a car is used ($car = 1$) and the second column should be the average length of a delivery when a car is not used ($car = 0$). We want to calculate this on all deliveries from February in any year. Note that this is similar to the last problem, but the data shape and date filters are different.

```
select
        avg( case when car = 0 then length else null end) as car_0_avg
        , avg( case when car = 1 then length else null end) as car_1_avg
from
        deliveries
where
        date_part('month', del_date) = 2;
```

7. What state (*state* only) has the most drivers?

```
select state
from drivers
group by 1
order by count(1) desc
limit 1;
```

8. What was the longest (by *length*) non-car ($car = 0$) delivery (*del_id* only)?

```
select
        del_id
from
        deliveries
where car = 0
order by length desc
limit 1;
```

9. We call drivers who have ever done a delivery of more than 60 km a "long-driver". What is the average review score for "long-drivers"? This should include *all* deliveries from "long-drivers", even those less than 60 km. This should return only a single row and column.

```
select
        avg( review )
from
        deliveries
left join
        reviews
using(del_id)
where
        driver_id in (select distinct driver_id from deliveries where length > 60)
```

Pandas Section

Please answer the following question, making sure to return only the information required. You can assume that DataFrames named *drivers*, *deliveries* and *reviews* are already loaded. Unless otherwise specified you may return either a Series or DataFrame.

1. Return a DataFrame which has two columns and a row for each state. The first column should be the state and the second should be the number of deliveries completed by drivers from that state.

```
pd.merge( drivers, deliveries, on='did', how='left')
        .groupby('state', as_index=False)
        .agg({'del_id' : ['count']})

-- Since this is number of jobs it could be inner or left or outer join
```

2. Return a DataFrame with two rows and two columns. The first column should be a flag which takes one of two values: "LT10" and "MT15". The second column should be the average review for deliveries which are (strictly) "Less Than 10 km" and "More Than 15 km" , respectively. In other words, one row should contain "LT10" and the average review for deliveries which are less than 10 km and the other row should contain "MT15" and the average review for deliveries which are more than 15 km.

```
mrg = pd.merge( deliveries, reviews, on='del_id', how='inner')

mrg = (mrg
       .loc[(mrg.loc[:, 'length'] < 10) | (mrg.loc[:, 'length'] > 15), :]
      )

mrg.loc[:, 'flag'] = 'LT10'
mrg.loc[(mrg.loc[:, 'length'] >15), 'flag'] = 'MT15'

mrg.groupby('flag', as_index=False).agg({'review' : ['mean']})
```

3. We call drivers who have ever done a delivery of more than 60 km a "long-driver". What is the average review score for "long-drivers"? This should include *all* deliveries from "long-drivers", even those less than 60 km. This should return a single value (can be in a DataFrame, in a Series or as a number).

```
driver_id_lst = deliveries.loc[(deliveries.loc[:, 'length'] > 60), 'driver_id'].drop_duplicates()

mrg = pd.merge( reviews, deliveries, on='del_id', how='inner')
mrg.loc[ (mrg.loc[:, 'driver_id'].isin( driver_id_lst), 'review'].mean()
```

4. What was the longest (by length) non-car ($car = 0$) delivery (del_id only)?

```
deliveries.loc[ (deliveries.loc[:, 'car'] == 0), :].nlargest(1,'length').loc[:, 'del_id']

OR

(deliveries
        .loc[ (deliveries.loc[:, 'car'] == 0), :]
        .sort_values('length', ascending=False)
        .loc[:, 'del_id']
)
```

5. Which year had the largest number of deliveries (count)?

```
(deliveries
        .assign(yr = deliveries.loc[:, 'del_date'].dt.year)
        .groupby( yr, as_index=False)
        .agg( {'del_id' : ['count']})
        .nlargest(1, ('del_id', 'count'))
        .loc[:, 'yr']
)
```

6. How many drivers are from California ("CA")?

```
drivers.loc[ (drivers.loc[:, 'state'] == 'CA'), :].shape[0]

Lots of ways to do this one.
```

7. Which date (*del_date*) had the largest number of 3-star deliveries (count)?

```
mrg = pd.merge( deliveries, reviews, how='left', on='del_id')

(mrg
        .loc[ (mrg.loc[:, 'review'] == 3), :]
        .groupby( 'del_date', as_index=False)
        .agg( {'del_id' : ['count']})
        .nlargest(1, ('del_id', 'count'))
        .loc[:, 'del_date']
)
```

8. Return all drivers (*driver_id* only) from Texas ("TX") who are 32 years old as a *DataFrame*.

```
drivers.loc[ (drivers.loc[:, 'state'] == 'CA') & (drivers.loc[:, 'age'] == 32), ['driver_id']]
```