2 2023 CAPP Databases Final B

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 "B".

The following tables contains information about a large painting company. The company has many painters who do jobs and then get reviews on those jobs. A job can, at most, have **one** review. Not all painters will have jobs (when they first start there is some time before they are assigned a job). A painter can have multiple jobs as most jobs last only a day or two and every job will be in the database.

- 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 *painters* table / DataFrame:
  - **painter_id:** The ID of the painter (INT, UNIQUE, NOT NULL).
  - **state:** The state that the painter works in (STRING, NOT NULL).
  - **work_exp:** The number of years of work experience (INT, NOT NULL).
- Columns in the *jobs* table / DataFrame:
  - **job_id:** The unique ID associated with the job (INT, UNIQUE, NOT NULL).
  - **job_date:** The date that the job occurred (DATE, NOT NULL).
  - **painter_id:** The ID of the painter (INT, NOT NULL).
  - **sprayer:** A flag (1/0) for if the painter used a sprayer or not (INT, NOT NULL).
  - **paint:** The amount of paint used, in gallons (FLOAT, NOT NULL).
- Columns in the *reviews* table / DataFrame:
  - **job_id:** The ID of the job being reviewed (INT, NOT NULL).
  - **review:** The review score on a 1 (worst) to 5 (best) scale (INT, NOT NULL)

painter_id	state	work_exp
1	CA	0
2	MN	12
3	CA	4
4	CA	11

Painters (1,234 Rows)

job_id	job_date	painter_id	sprayer	paint
1	1-1-2012	45	0	1.25
2	12-23-2012	45	1	23.5
3	7-6-2013	112	1	11.25
4	5-5-2014	1125	0	.75

Jobs (14,365 Rows)

job_id	review
1	5
23	4
35	4
45	1

Reviews (980 Rows)

SQL Section

1. Write a query which returns the 11 painters (*painter_id*) with the most experience (largest *work_exp*) from Hawaii ("HI").

```
select painter_id
from painters
where state = 'HI'
order by work_exp desc
limit 11
```

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

```
select
        state
        , count( distinct painters.painter_id)
        , count( jobs.job_id)
from
        painters
join
        jobs
using( painter_id )
group by state
```

3. Write a query which returns one row per job and four columns. The first column should be the state of the painter, the second should be the *painter_id*, the third should be *job_id* and the fourth should be the total amount of paint that the painter has used up to and including that job (cumulative sum of *paint*). Make sure that the cumulative sum is calculated by the date of the job from earliest to latest. If a painter does not have any jobs they should *not* be included in the results.

```
select
        painter_id
        , job_id
        , state
        , sum( paint ) over(
                partition by painter_id
                order by job_dt asc
                rows between unbounded preceding and current row
                ) as cum_sum
from
        painters
join
        jobs
using( painter_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 jobs from that state. Only include Alaska ("AK") and Hawaii ("HI").

```
select
        state,
        count( case when review = 5 then 1 else null end ) as num_five_stars
from
        (select * from painters where state in ('AK', 'HI')) as lhs
left join
        jobs
        using( painter_id)
left join
        review
        using( job_id)
group by state;
```

5. We want to return the average amount of paint used depending on if the painter used a sprayer or not. Write a query which return two rows and two columns. The first should be if the painter used a sprayer or not (the *sprayer* column 1/0 flag) and the second should be the average amount of paint used. Only include those observations from the year 2014.

```
select
        spray,
        avg( paint )
from
        jobs
where
        date_part('year', del_date) = 2014
group by 1
```

6. Please return one row with two columns. The first column should be the average amount of paint used when a sprayer is used ($sprayer = 1$) and the second column should be the average amount of paint used if a sprayer is not used ($sprayer = 0$). We want to calculate this on all jobs from July 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 sprayer = 0 then paint else null end) as spray_0_avg
        , avg( case when sprayer = 1 then paint else null end) as spray_1_avg
from
        jobs
where
        date_part('month', del_date) = 7;
```

7. What state (state only) has the most painters?

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

8. What was the largest (used the most paint) non-sprayer ($sprayer = 0$) paint job? Return the *job_id* only.

```
select
        job_id
from
        jobs
where sprayer = 0
order by paint desc
limit 1;
```

9. We call painters who have ever done a job of more than 25 gallons "large-scale" painters. What is the average review for "large-scale" painters? This should include *all* jobs from "large-scale" painters, even those jobs which are less 25 gallons. This should return only a single row and column.

```
select
        avg( review )
from
        jobs
left join
        reviews
using(job_id)
where
        painter_id in (select distinct painter_id from jobs where paint > 25)
```

Pandas Section

Please answer the following question, making sure to return only the information required. You can assume that DataFrames named *painters*, *jobs* 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 jobs completed by painters from that state.

```
pd.merge( painters, jobs, on='painter_id', how='left')
        .groupby('state', as_index=False)
        .agg({'job_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: "less3" and "greater20". The second column should be the average review for jobs which are (strictly) "less than 3 gallons" and "greater than 20 gallons" , respectively. In other words, one row should contain "Less3" and the average review for jobs which are less than 3 gallons and the other row should contain "greater20" and the average review for jobs which are more than 20 gallons.

```
mrg = pd.merge( jobs, reviews, on='job_id', how='inner')

mrg = (mrg
       .loc[(mrg.loc[:, 'paint'] < 3) | (mrg.loc[:, 'length'] > 20), :]
      )

mrg.loc[:, 'flag'] = 'less3'
mrg.loc[(mrg.loc[:, 'paint'] > 20), 'flag'] = 'greater20'

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

3. We call painters who have ever done a job of more than 25 gallons "large-scale" painters. What is the average review for "large-scale painters? This should include *all* jobs from "large-scale" painters, even those jobs which are less 25 gallons. This should return a single value (can be in a DataFrame, in a Series or as a number).

```
painter_id_list = jobs.loc[(jobs.loc[:, 'paint'] > 25), 'driver_id'].drop_duplicates()

mrg = pd.merge( reviews, jobs, on='driver_id', how='inner')
mrg.loc[ (mrg.loc[:, 'painter_id'].isin( painter_id_list), 'review'].mean()
```

4. What was the largest (most paint used) non-spray (*sprayer* = 0) job (job_id only)?

```
jobs.loc[ (jobs.loc[:, 'sprayer'] == 0), :].nlargest(1,'paint').loc[:, 'job_id']

OR

(job
        .loc[ (job.loc[:, 'sprayer'] == 0), :]
        .sort_values('paint', ascending=False)
        .loc[:, 'job_id']
)
```

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

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

6. How many painters are from New Mexico ("NM")?

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

Lots of ways to do this one.
```

7. Which date (*job_date*) had the largest number of 2-star jobs (count)?

```
mrg = pd.merge( jobs, reviews, how='left', on='job_id')

(mrg
        .loc[ (mrg.loc[:, 'review'] == 2), :]
        .groupby( 'job_date', as_index=False)
        .agg( {'job_id' : ['count']})
        .nlargest(1, ('job_id', 'count'))
        .loc[:, 'job_date']
)
```

8. Return all painters (*painter_id* only) from California ("CA") who have 12 years of experience as a *DataFrame*.

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