CS 220P: Introduction to Data Management – Fall 2022

Homework 6: More SQL (Hands-on) (100 points)

Due Date: **Monday, Nov 14 (11:59 PM Pacific)**

Submission

All HW assignments must be submitted online via the HW6 submission on Gradescope. See the table below for the HW 6 submission opportunities. Note that after 11:59 PM on Tuesday the 15th no further HW 6 submissions will be accepted. (We will be releasing the solution at that time.) Please strive to get all your work in on time! If possible, try to save the one dropped assignment for the end of the term when you are most likely to want/need it.

Date / Time	**Grade Implications**
Mon, Nov 14 (11:59 PM Pacific)	Full credit will be available
Tues, Nov 15 (11:59 PM Pacific)	10 points will be deducted

Student ID: 19683688

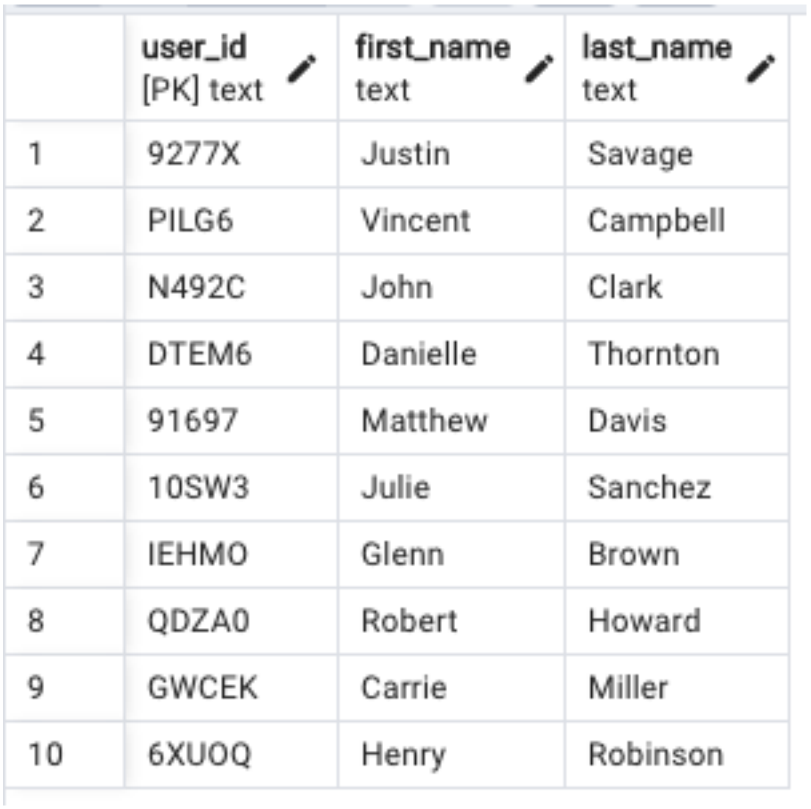
NAME: Jiaxin Li

1. [10pts] For all buyers who bought at least 3 items after the date 2022-07-24, list each buyer's user_id, first_name, and last_name.

a) [7pts] SQL Query:

```
WITH temp_bid AS (SELECT I.buyer_user_id, COUNT(*) AS scount
        FROM cs222p_interchange.user U, cs222p_interchange.buyer B, cs222p_interchange.item I
        WHERE U.user_id=B.user_id AND I.buyer_user_id=B.user_id AND I.purchase_date>'2022-07-24'
        GROUP BY I.buyer_user_id
        HAVING COUNT(*)>2)
SELECT U.user_id, U.first_name, U.last_name
FROM temp_bid, cs222p_interchange.user U
WHERE U.user_id=temp_bid.buyer_user_id
```

b) [3pts] Result:



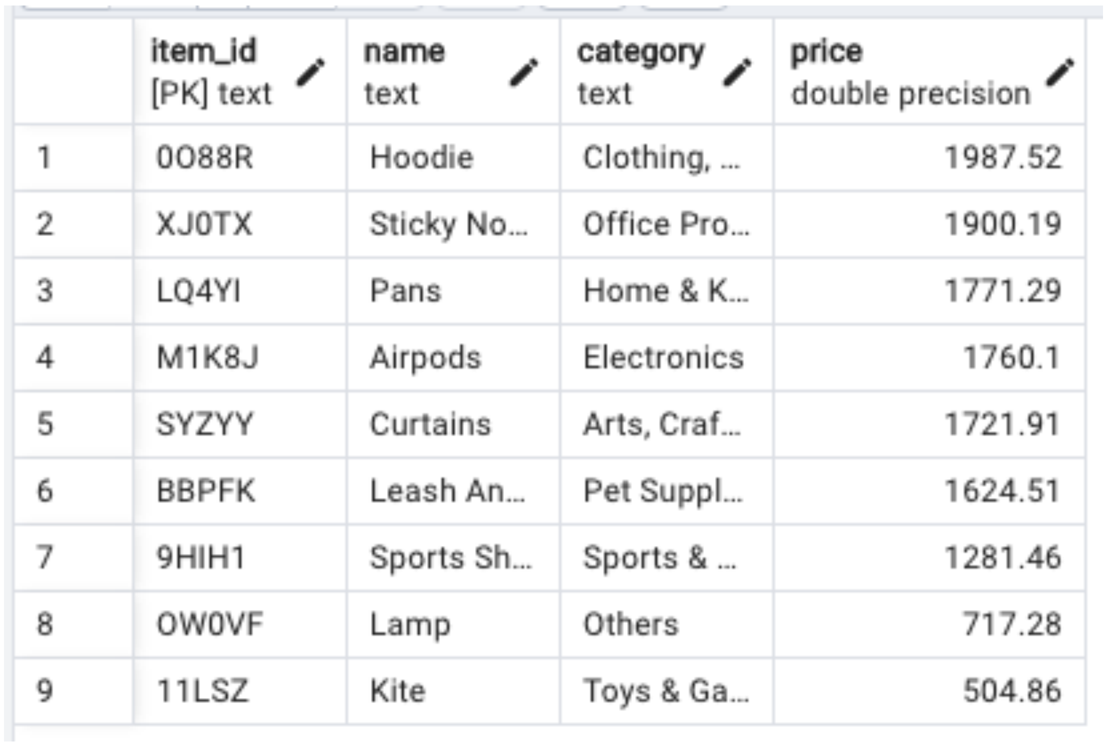
	user_id [PK] text	first_name text	last_name text
1	9277X	Justin	Savage
2	PILG6	Vincent	Campbell
3	N492C	John	Clark
4	DTEM6	Danielle	Thornton
5	91697	Matthew	Davis
6	10SW3	Julie	Sanchez
7	IEHMO	Glenn	Brown
8	QDZA0	Robert	Howard
9	GWCEK	Carrie	Miller
10	6XUOQ	Henry	Robinson

2. [10pts] Find the highest price for each item sold by the seller with user id 'S3AB0' for each category of item where they've had sales. Print the item_id, item_name, category, and price of these highest-price items. Rank the output by price from highest to lowest.

a) [7pts] SQL Query:

```
WITH temp_cat_group AS (
        SELECT I.category, MAX(I.price) AS max_price
        FROM cs222p_interchange.item I
        WHERE I.seller_user_id='S3AB0'
        GROUP BY I.category)
SELECT I.item_id, I.name, I.category, I.price
FROM temp_cat_group TG, cs222p_interchange.item I
WHERE TG.category=I.category AND TG.max_price=I.price
ORDER BY price DESC
```

b) [3pts] Result:



	item_id [PK] text	name text	category text	price double precision
1	0088R	Hoodie	Clothing, ...	1987.52
2	XJ0TX	Sticky No...	Office Pro...	1900.19
3	LQ4YI	Pans	Home & K...	1771.29
4	M1K8J	Airpods	Electronics	1760.1
5	SYZYY	Curtains	Arts, Craf...	1721.91
6	BBPFK	Leash An...	Pet Suppl...	1624.51
7	9HIH1	Sports Sh...	Sports & ...	1281.46
8	OW0VF	Lamp	Others	717.28
9	11LSZ	Kite	Toys & Ga...	504.86

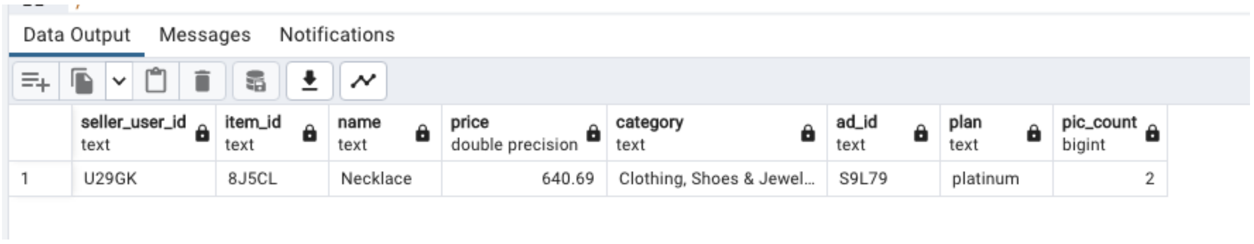
3. [10pts] For all unpurchased services that had an ad placed by its seller, list the seller's user_id and the item_id, item_name, price, category, ad_id, ad_plan, and number of pictures associated with the item. Limit your output to the top 10 results ordered from highest to lowest by price.

a) [7pts] SQL Query:

```
with temp_pic AS(
        SELECT pic.item_id, COUNT(*) AS pic_count
        FROM cs222p_interchange.picture pic
        group by item_id)

SELECT I.seller_user_id, I.item_id, I.name, I.price, I.category, ad.ad_id, ad.plan, pi.pic_count
FROM cs222p_interchange.ad ad, cs222p_interchange.item I, cs222p_interchange.service S, temp_pic Pi
WHERE I.buyer_user_id IS NULL
        AND ad.item_id=I.item_id
        AND S.item_id=I.item_id
        AND ad.seller_user_id=I.seller_user_id
        AND pi.item_id=I.item_id
ORDER BY I.price DESC
LIMIT 10
```

b) [3pts] Result:



	seller_user_id text	item_id text	name text	price double precision	category text	ad_id text	plan text	pic_count bigint
1	U29GK	8J5CL	Necklace	640.69	Clothing, Shoes & Jewel...	S9L79	platinum	2

4. **Views** [20 pts]

It's time to identify the highest rated sellers on the Interchange.com platform. To compute a seller's **overall rating** we will sum up the individual quality, price, and delivery ratings and compute their average. If a particular rating attribute (quality, price, or delivery) is NULL, we will set a default value of 2.5 for that particular rating in the computation of the seller's **overall rating**. We will classify a seller's rating as "High" if that seller's **overall rating** is at least 4.4 out of 5. If a seller's **overall rating** is at least 2.6 but under 4.4 we will classify that seller as "Medium". If a seller's **overall rating** is under 2.6 we will classify the seller as "Underdog". In order to ensure that the ratings are accurate (and not spam or the result of a grudge) we will also indicate the number of ratings from users who have actually purchased items from the seller and call it the valid rating count. To implement this we will create a view so that Interchange.com's data analysts don't have to deal with this complexity when working with the data. The view must include seller's id, overall rating, seller classification, and valid rating count.

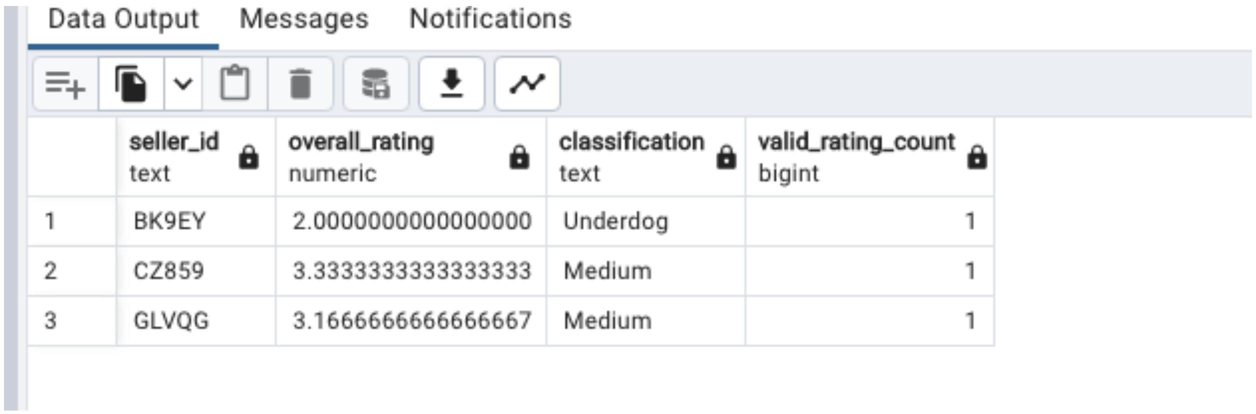
a) [15 pts] Create the desired view – *SellerOverallRating* – by writing an appropriate **CREATE VIEW** statement. [HINT: Check out COALESCE, CASE, and WITHs in the PostGreSQL documentation)

CREATE VIEW SellerOverallRating (seller_id, overall_rating, classification, valid_rating_count) **AS** ...;

b) [5 pts] Show the usefulness of your view by writing a **SELECT** query against the view that prints the seller_id, first_name, last_name, and website of all sellers, also including their classification and valid rating count. Rank your results by the number of valid ratings from the highest to the lowest and limit the results to 5.

```
CREATE VIEW SellerOverallRating(seller_id, overall_rating, classification, valid_rating_count)
        AS
SELECT R.seller_id,
        (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) AS overall_rating,

CASE
WHEN (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) >=0
        AND (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) <2.6 THEN
'Underdog'
WHEN (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) <4.4
        AND (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) >=2.6 THEN
'Medium'
WHEN (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) >=4.4
        AND (R.quality_sum + R.pricing_sum + R.delivery_sum)/(3*R.valid_rating_count) <=5 THEN 'High'
END classification,
R.valid_rating_count
FROM
        (SELECT R1.seller_id,
        (SUM(R1.modified_quality)+SUM(R1.modified_pricing)+
SUM(R1.modified_delivery))/(3*COUNT(*)) AS overall_avg,
        SUM(R1.modified_quality) AS quality_sum,
        SUM(R1.modified_pricing) AS pricing_sum,
        SUM(R1.modified_delivery) AS delivery_sum,
        COUNT(*) AS valid_rating_count
        FROM (select R.seller_id, R.buyer_id,
CASE WHEN R.quality is null THEN 2.5 ELSE R.quality END AS modified_quality,
CASE WHEN R.pricing is null THEN 2.5 ELSE R.pricing END AS modified_pricing,
CASE WHEN R.delivery is null THEN 2.5 ELSE R.delivery END AS modified_delivery
from cs222p_interchange.ratings R,cs222p_interchange.item I WHERE  I.buyer_user_id=R.buyer_id AND
I.seller_user_id=R.seller_id) R1
        GROUP BY R1.seller_id) R
```



	seller_id text	overall_rating numeric	classification text	valid_rating_count bigint
1	BK9EY	2.0000000000000000	Underdog	1
2	CZ859	3.3333333333333333	Medium	1
3	GLVQG	3.1666666666666667	Medium	1

5. **Stored Procedures** [20 pts]

a) [15 pts] Create and exercise a SQL stored procedure called *InsertServiceAndPlaceAd(...)* that the application developer can use to simultaneously add a new Service and place an Ad for it.

```
CREATE PROCEDURE InsertServiceAndPlaceAd(
        IN seller_user_id text,
        IN item_name text,
        IN item_id text,
        IN service_frequency cs222p_interchange.Frequency,
        IN price float,
        IN category text,
        IN description text,
        IN ad_id
        IN plan text,
        IN content text,
        IN picture_url text,
        IN picture_format cs222p_interchange.PictureFormat,
)
LANGUAGE SQL AS … ;
```

b) [5pts] Verify that your stored procedure works properly by calling it as follows to insert a new Service Item with an associated Ad and running a **SELECT** query (or queries) to show the stored procedure's after-effects:

CALL InsertServiceAndPlaceAd ('OE791', 'Yard Cleanup', 'yrdcleanup2022', 'weekly', 35.43, 'Paper, Cleaning, & Home', 'Cleanup services for yards done weekly', 'X2342YRD', 'Gold', 'Cleanup services for yards done weekly! Call Now!', 'https://yardworkforeveryone.net/pic1.png', 'png')

```
SELECT s.item_id, a.ad_id, i.seller_user_id, p.url
FROM cs222p_interchange.Service s, cs222p_interchange.Ad a, cs222p_interchange.Item i,
cs222p_interchange.Picture p
WHERE s.item_id = p.item_id AND s.item_id = a.item_id AND i.item_id = s.item_id AND
s.item_id='yrdcleanup2022';
```

Query Results:

6. **Alter Table** [10 pts]

a) [5 pts] Write and execute the **ALTER TABLE** statement(s) needed to modify the Ad table so that when an item associated with an Ad is deleted, the Ad will ***not*** also be deleted. It should now be retained instead.

b) [5 pts] Execute the following **SELECT** and **DELETE** statements to show the effect of your change. Report the COUNT query's result (just the number) returned by the **SELECT** statement both before and after running your **DELETE**.

```
SELECT COUNT(*) FROM cs222p_interchange.Ad a WHERE a.item_id = 'CBAGZ';

DELETE FROM cs222p_interchange.Item WHERE item_id = 'CBAGZ';

SELECT COUNT(*) FROM cs222p_interchange.Ad a WHERE a.item_id = 'CBAGZ';
```

Results:

a)

ALTER TABLE CS222P_INTERCHANGE.Ad DROP CONSTRAINT ad_pic_num_item_id_fkey;

b)

ALTER TABLE CS222P_INTERCHANGE.Ad DROP CONSTRAINT ad_item_id_fkey;

7. **Triggers** [20 pts]

a) [15 pt] Create a new table *TargetedAds*(user_id, ad_id, PRIMARY KEY(user_id, ad_id)) that stores the Ads curated for the users based on a user's indicated category of interests. Then write a CREATE TRIGGER statement (**by hand** of course!) to define a trigger that will do the following job: After a seller has placed an ad -- indicated by an insert into the Ad table -- if the category of the item for which the Ad was placed matches a user's category of interest – as indicated by the user in the Categories table – the ad_id and the user's user_id are added into the TargetedAds table. (The new table is only responsible for keeping the targeted ads for the user after the trigger is created.) Use the CREATE FUNCTION statement as well as needed. Your function should avoid inserting duplicate entries into the new table. (HINT: use "...ON CONFLICT..." to handle insertion conflicts.)

b) [5 pts] Execute the following INSERT and SELECT statements to show the effect of your trigger. Report the results.

```
SELECT *
FROM TargetedAds
WHERE buyer_id = 'NS804';
```

Result:

```
INSERT INTO cs222p_interchange.Ad(ad_id, plan, content, pic_num, item_id, seller_user_id,
placed_date)
VALUES ('ADT32457', 'Gold', ‘New games available!’, 0, ‘F7E1N’, ‘4Z5VC’, 2022-11-06);

SELECT *
FROM TargetedAds
WHERE ad_id = 'ADT32457';
```

Result:

```
INSERT INTO cs222p_interchange.Categories (user_id, category)
VALUES (‘YJLRR’, ‘Toys & Games’)

INSERT INTO cs222p_interchange.Ad(ad_id, plan, content, pic_num, item_id, seller_user_id,
placed_date)
VALUES ('ADT32458', 'Gold', ‘New games available!’, 0, ‘F7E1N’, ‘4Z5VC’, 2022-11-06);

SELECT *
FROM TargetedAds
WHERE buyer_id = 'YJLRR';
```

Result:

```
UPDATE cs222p_interchange.Item SET category = 'Pet Care' WHERE item_id = 'IRFRO';

INSERT INTO cs222p_interchange.Ad(ad_id, plan, content, pic_num, item_id, seller_user_id,
placed_date)
VALUES ('ADT32459', 'Gold', ‘Pet Care Kit!’, 0, ‘IRFRO’, ‘449OC’, 2022-11-06);

SELECT *
FROM TargetedAds
WHERE ad_id = 'ADT32459';
```

Result: