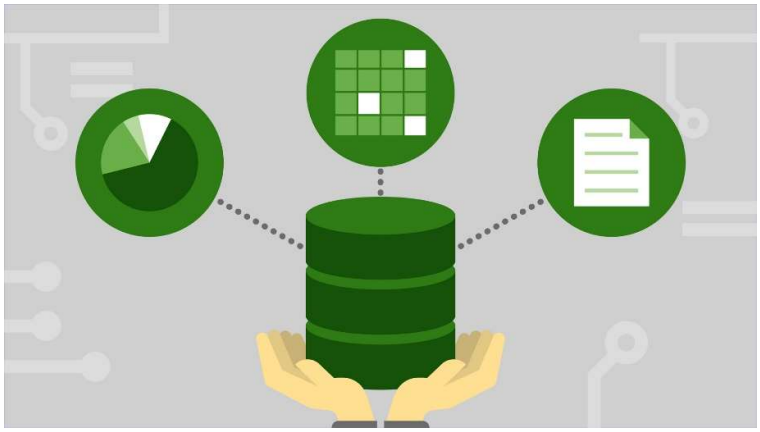
به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه‌داده

دستورکار شماره ۶

مهلت تحویل :

۱۴۰۱/۰۲/۱۰

مجتبی بنائی

آنچه خواهید آموخت

هدف اصلی از این تمرین، آشنایی عملی با سه مفهوم زیر در بانک‌های اطلاعاتی رابطه‌ای است:

توابع : مشابه با مفهوم تابع در زبان‌های برنامه‌نویسی، بسته به ورودی‌ها، خروجی مناسب را تولید میکند. این خروجی میتواند هر مقدار مجاز در SQL از جمله یک جدول (خروجی select) یا یک مقدار عددی باشد. مثلا در دیتابیس Northwind میتوانیم تابعی در پستگرس بنویسیم که شماره یک فاکتور را گرفته، کل اجناس و یا قیمت پرداخت شده مشتری به ازای آنرا برگرداند.

تریگر : با انجام هر تغییری در دیتابیس، میتوانیم تریگری را فعال کنیم و کار بخصوصی را انجام دهیم. مثلا فرض کنید به ازای هر تراکنش مالی (برداشت یا واریز به حساب)، نیاز داریم به دلیل مسائل امنیتی، مقدار قبلی حساب را در یک جدول جداگانه به نام ، transaction_history به صورت خودکار ذخیره کنیم (عملیات Audit Log).

این کار به راحتی توسط تریگرها قابل انجام است و کافی است **قبل** از عملیات **به روز رسانی** حساب کاربر، تابعی که برای این منظور نوشته‌ایم را به کمک تریگری که روی این عملیات، تنظیم میکنیم، فراخوانی کنیم.

نکته : در دنیای واقعی تولید نرم‌افزار، تا حدامکان از نوشتن مستقیم منطق کسب و کار در دیتابیس اجتناب می‌کنیم چون این امر، تست و تغییر برنامه‌ها را بسیار مشکل می‌کند. در صورت نیاز به این کار هم معمولاً با استفاده از ORM (کتابخانه‌های واسط برای کار با دیتابیس‌های رابطه‌ای) این مسایل را مدیریت می‌کنیم که اگر نیاز به تغییر خود دیتابیس داشتیم و یا منطق برنامه در بخشی از کار، نیاز به تغییر داشت، مجبور به تغییر مستقیم کدهای نوشته شده در دیتابیس‌ها که خارج از سیستم گیت و مدیریت پروژه و ... است، نشویم.

توابع پنجره‌ای : تاکنون با توابع تجمعی مانند میانگین و ماکزیمم و بخصوص count آشنا شده‌اید که بر روی مجموعه‌ای از سطرها کار کرده، یک مقدار را به ازای هر گروه برمیگردانند. اما در بسیاری از کاربردهای امروزی، نیاز داریم که *به ازای هر رکورد*، مقداری را بر اساس گروه‌بندی مرتبط با آن رکورد، محاسبه کنیم مثلا در جدول نمرات یک دانشگاه، بتوانیم رتبه دانشجو در یک کلاس خاص، اختلاف نمره هر دانش آموز از نمره اول آن کلاس، میانگین نمره دانشجویان هم ورودی و هم جنسیت هر دانشجو در آن کلاس و .... را محاسبه کنیم. این کارها را با توابع تجمعی عادی نمی‌توانیم محاسبه کنیم و برای این منظور نیاز به توابع پنجره‌ایWindow Function داریم.

با توجه به اینکه هدف از این تمرین، آشنایی عمومی با این مفاهیم است، این مطالب را در قالب دو آموزش تک صفحه‌ای و انجام عملی دستورات آنها، فراخواهید گرفت که زمان زیادی را از شما نخواهد گرفت.

دستورالعمل اجرایی

توابع و تریگرها

برای آشنایی با دو مفهوم توابع و تریگرها در پستگرس، به آدرس زیر مراجعه کنید:

https://severalnines.com/database-blog/postgresql-triggers-and-stored-function-basics

ابتدا کل آموزش را مرور کنید. سه تا از مثال‌ها (از بین ۶ مثال)را انتخاب کنید و آنها را در پستگرس پیاده سازی کنید.

به ازای هر مثال ابتدا توضیح دهید که قرار است چه کاری انجام شود و برای این منظور، چه مراحلی باید طی شود. سپس از خروجی نهایی آن از صفحه اسکرین شات ، گرفته و در گزارش نهایی درج کنید.

توابع پنجره‌ای

برای آشنایی با مفهوم توابع پنجره‌ای که امروزه نقش مهمی در تولید گزارش‌های تحلیلی برعهده دارند، به این سه منبع آموزشی مراجعه کنید :

- https://learnsql.com/course/window-functions/(دو بخش اول که رایگان است)
- yun.ir/ohbik5
- https://www.startdataengineering.com/post/6-concepts-to-clearly-understand-window-functions/

سپس به دلخواه، سه دستور **sql** برای دیتابیس **Northwind** (موضوع تمرین پنجم) با استفاده از توابع پنجره‌ای بنویسید.

دو سوال تکمیلی

سوال اول :

فرض کنید دیتابیس زیر را برای ذخیره کلیک‌های کاربران در یک سایت فروشگاهی طراحی کرده ایم و داده‌های زیر را هم درج نموده‌ایم (یک دیتابیس در پستگرس ایجاد کرده، این دستورات را اجرا کنید تا جدول مربوطه ایجاد شود) :

```
drop table if exists clickstream;
create table clickstream (
    eventId varchar(40),
    userId int,
    sessionId int,
    actionType varchar(8),
    datetimeCreated timestamp
```

```
);
insert into clickstream(eventId, userId, sessionId, actionTy
pe, datetimeCreated )
values
('6e598ae5-3fb1-476d-9787-175c34dcfeff',1 ,1000,'click','202
0-11-25 12:40:00'),
('0c66cf8c-0c00-495b-9386-28bc103364da',1 ,1000,'login','202
0-11-25 12:00:00'),
('58c021ad-fcc8-4284-a079-8df0d51601a5',1 ,1000,'click','202
0-11-25 12:10:00'),
('85eef2be-1701-4f7c-a4f0-7fa7808eaad1',1 ,1001,'buy',  '202
0-11-22 18:00:00'),
('08dd0940-177c-450a-8b3b-58d645b8993c',3 ,1010,'buy',  '202
0-11-20 01:00:00'),
('db839363-960d-4319-860d-2c9b34558994',10,1120,'click','202
0-11-01 13:10:03'),
('2c85e01d-1ed4-4ec6-a372-8ad85170a3c1',10,1121,'login','202
0-11-03 18:00:00'),
('51eec51c-7d97-47fa-8cb3-057af05d69ac',8 ,6,   'click','202
0-11-10 10:45:53'),
('5bbcbc71-da7a-4d75-98a9-2e9bfdb6f925',3 ,3002,'login','202
0-11-14 10:00:00'),
('f3ee0c19-a8f9-4153-b34e-b631ba383fad',1 ,90,  'buy',  '202
0-11-17 07:00:00'),
('f458653c-0dca-4a59-b423-dc2af92548b0',2 ,2000,'buy',  '202
0-11-20 01:00:00'),
('fd03f14d-d580-4fad-a6f1-447b8f19b689',2 ,2000,'click','202
0-11-20 00:00:00');

-- create fake geolocation data
drop table if exists geolocation;
create table geolocation (
    userId int,
    zipcode varchar(10),
    datetimeCreated timestamp
);

insert into geolocation(userId, zipCode, datetimeCreated )
values
(1 ,66206,'2020-11-25 12:40:00'),
(1 ,66209,'2020-11-25 12:00:00'),
(1 ,91355,'2020-11-25 12:10:00'),
(1 ,83646,'2020-11-22 18:00:00'),
(3 ,91354,'2020-11-20 01:00:00'),
(10,91355,'2020-11-01 13:10:03'),
(10,91355,'2020-11-03 18:00:00'),
(8 ,91355,'2020-11-10 10:45:53'),
(3 ,91355,'2020-11-14 10:00:00'),
(1 ,83646,'2020-11-17 07:00:00'),
(2 ,83646,'2020-11-20 01:00:00'),
```

```
(2 ,91355,'2020-11-20 00:00:00');
```

حال توضیح دهید خروجی دستور SQL زیر چیست ؟ نقش دستور Case در آنرا بیان کنید .

```
with purchasingUsers as (
    select userId,
        sum(
            case
                when actionType = 'buy' then 1
                else 0
            end
        ) as numPurchases
    from clickstream
    group by userId
    having numPurchases >= 1
),
movingUsers as (
    select userId
    from geolocation
    group by userId
    having count(distinct zipCode) > 1
),
userSessionMetrics as (
    select userId,
        sessionId,
        sum(
            case
                when actionType = 'click' then 1
                else 0
            end
        ) as numclicks,
        sum(
            case
                when actionType = 'login' then 1
                else 0
            end
        ) as numlogins,
        sum(
            case
                when actionType = 'buy' then 1
                else 0
            end
        ) as numPurchases
    from clickstream
    group by userId,
        sessionId
)
select usm.*
```

```
from userSessionMetrics as usm
    join movingUsers as mu on usm.userId = mu.userId
    join purchasingUsers as pu on usm.userId = pu.userId;
```

سوال دوم :

فرض کنید جدول زیر موجود باشد :

price	category_id	product_id
100000	89	1
2000	88	2
50000	89	3
...	...	...

نکته : این دیتابیس در قالب فایل sql_data_users.sqlite در اختیار شما قرار گرفته است. کافی است در نرم افزار DBeaver، یک کانکشن جدید ایجاد کرده، نوع دیتابیس را SQLite انتخاب کنید و این فایل را به عنوان ورودی به آن بدهید.

حال کوئری ای طراحی کنید که با اجرای آن، ده محصول گران و ده محصول ارزان در هر گروه (کتگوری) تولید شود. خروجی مد نظر، در فایل data_users_gold.csv تولید شده است و دستور SQL شما هم باید مشابه با آن را تولید کند.

اگر فایل اکسل را باز کنید، همانطور که در عکس زیر مشخص شده است به ازای کتگوری شماره ۲، ابتدا ده محصول ارزان و سپس ده محصول گران، اولی به ترتیب نزولی و دومی به ترتیب صعودی نمایش داده شده است .

نکته : حتما از توابع پنجره ای برای به دست آوردن خروجی این بخش استفاده کنید.

G7

	A	B	C	D
1	product_id	category_id	price	
2	112951	2	3115000	
3	174928	2	3257000	
4	161592	2	3361000	
5	178099	2	3433000	
6	174720	2	3757000	
7	128897	2	3814000	
8	102570	2	3937000	
9	171412	2	3992000	
10	149053	2	4007000	
11	101280	2	4018000	
12	106295	2	10589000	
13	108275	2	10613000	
14	158614	2	10668000	
15	198193	2	10677000	
16	180677	2	11009000	
17	175427	2	11414000	
18	166979	2	11486000	
19	136857	2	11969000	
20	173901	2	12719000	
21	138309	2	12755000	
22	146743	6	2522000	
23	144242	6	2689000	
24	122043	6	3096000	