قضاوت لینوکسی

در این سوال هدف پیاده‌سازی یک قضاوتگر کوچک برای کدهای bash است.

شما باید یک کد قضاوتگر بنویسید. در کنار کد شما کد دیگری بنام code.sh میباشد. حال باید تمام ورودی‌هایی را که در پوشه in با نام input1.txt تا input20.txt قرار دارند به نوبت به کد (code.sh) به عنوان ورودی بدهید و خروجی متناظر با آن را در پوشه out با نام output1.txt تا output20.txt قرار دهید.

یعنی هر یک فایل از ورودی را به کد مذکور داده و خروجی را در مسیر گفته شده با نام متناظر آن قرار می‌دهید.

حال با استفاده از دستور diff قرار است صحت خروجی را بسنجیم. در پوشه دیگر با نام ans تمام پاسخ‌های صحیح با نام‌های ans1.txt تا ans20.txt قرار دارد. باید دستور diff را به ازای هر خروجی تولید شده و پاسخ متناظر با آن اجرا کرده و خروجی جدید را در پوشه جدیدی به نام diff-res قرار دهیم. نام خرجی های جدید باید از res1.txt تا res20.txt باشد.

اگر تمام مراحل به درستی انجام شود و خروجی نهایی دستور diff درست باشد، نمره کامل به شما تعلق می‌گیرد.

جزئیات پروژه

کدی که شما آپلود می‌کنید با نام solution.sh در ساختار زیر قرار می‌گیرد:

```
.
├── in/
├── out/
├── ans/
├── diff-res/
├── code.sh
└── solution.sh
```

نکات بیشتر

- در شکل بالا in ، out ، ans و diff-res همگی پوشه هستند که فایل ها داخلشان قرار دارند. به نام فایل‌های هر پوشه دقت کنید.
- تعداد همه فایل‌های داخل پوشه‌ها دقیقا 20 تا است.
- مقادیر داخل فایل‌های جواب را تغییر ندهید. این امر موجب اشتباه در پاسخ شما میشود. این مقادیر صرفا برای مقایسه‌ی جواب شما با استفاده از دستور diff می‌باشد.
- در دستور diff آرگومان اول باید فایل جواب باشد و آرگومان دوم خروجی تولید شده در out .

آن‌چه باید آپلود کنید

پروژه‌ی اولیه را از این لینک دانلود کنید. پس از پیاده سازی کد قضاوتگر در فایل solution.sh ، بدون تغییر نام فایل، آن را آپلود کنید.

نمایش اصولی

در این تمرین قصد داریم با استفاده از دستور `awk` یک نمایشگر فایل ساده بنویسیم. برای این تمرین ما فایلی با نام `data.txt` داریم که چند خط ابتدایی آن به‌صورت زیر است:

data.txt

```
a|12|1
b|22|2
c|32|3
d|42|41
e|52|534
f|62|676
g|72|8
h|82|9
i|92|0
```

حال ما قصد داریم تا کل این فایل را به‌صورتی شکیل در ترمینال نمایش دهیم.

برای حل این تمرین شما باید در یک خط و با دستور `awk` محتوای این فایل را به‌صورت زیر نمایش دهید:

terminal

```
col1 - col2 - col3
a - 12 - 1
b - 22 - 2
c - 32 - 3
d - 42 - 41
e - 52 - 534
f - 62 - 676
g - 72 - 8
h - 82 - 9
i - 92 - 0
1274
```

همان‌طور که در قسمت بالا مشخص است، در خط اول خروجی باید ابتدا سه ستون `col1` ، `col2` و `col3` را نمایش دهید و بعد از نمایش محتوای فایل باید در خط آخر خروجی مجموع مقادیر ستون سوم را نمایش دهید.

توجه کنید

- شما باید دستور خواسته شده را فقط در یک خط بنویسید.
- مقادیر موجود در سوال فقط قسمتی از فایل اصلی را نمایش می‌دهد و فایل اصلی مقادیر زیادی دارد.

نحوه ارسال

برای ارسال جواب می‌توانید دستورات خواسته شده را درون فایلی با نام `solution.sh` وارد کنید و آن را انتخاب و سپس ارسال کنید.

کنترل دسترسی

کد شما باید دو کاربر `user1` و `user2` را بسازد. گروه مشترکی نیز به اسم `shared` ساخته شود که هردو کاربر به آن اضافه شوند. هر یک از این کاربران **باید** مشخصات زیر را داشته باشند:

- پوشه‌ی خانه‌ی مخصوص به خود را داشته باشند.
- نیازی به نوشتن کامنت ندارند.
- هیچکدام از یوزرها نیازی به پسورد ندارند.
- هر دو یوزر باید عضو گروه `shared` باشند.

سپس باید دایرکتوری‌ای با اسم `shared_files` در روت (`/`) بسازد. پس از ایجاد دایرکتوری، یک فایل با اسم `shared_file` در این دایرکتوری بسازد. در نهایت نیز گروهی که صاحب این فایل و این دایرکتوری است را گروه `shared` قرار دهد و کاربر صاحب فایل و دایرکتوری را `user1` قرار دهد. همچنین **تنها** دسترسی **خواندن و نوشتن** را برای گروه و کاربر صاحب این فایل بدهد و به سایر کاربران **هیچ** دسترسی‌ای ندهد. بعد از اتمام کار نیز کاربران و گروه و همه فایل‌های آنان را پاک کند!

توجه کنید

- شما می‌توانید هر تعداد دستوری که نیاز دارید را برای داوری ارسال کنید و محدودیتی در تعداد دستورات ندارید.
- شما حق استفاده از هر دستوری به غیر از `usermod` و `cd` را دارید ولی توصیه می‌کنیم **هیچ** پایپ‌لاینی تشکیل ندهید.
- نیازی به استفاده از `sudo` نیست.
- نیازی به استفاده از آپشن `--remove-all-files` برای حذف یوزر نیست.

نحوه ارسال

برای ارسال جواب می‌توانید دستورات خواسته شده را درون فایلی با نام `solution.sh` وارد کنید و آن را انتخاب و سپس ارسال کنید.

بهینه‌سازی جست و جو

کوئری‌های شما باید روی آخرین نسخه‌ی *MySQL* قابل اجرا باشند.

جزئیات پروژه

داده‌های اولیه برای تست نمونه را از این لینک دانلود کنید.

ساختار جداول به‌شرح زیر است:

جدول `users` : از این جدول برای نگه‌داری اطلاعات کاربران استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`id`	`BIGINT(20)`	شناسه‌ی کاربر
`username`	`VARCHAR(255)`	نام کاربری
`name`	`VARCHAR(255)`	نام کاربر
`password`	`VARCHAR(255)`	رمز عبور
`created_at`	`TIMESTAMP`	زمان عضویت کاربر

جدول `products` : از این جدول برای نگه‌داری اطلاعات محصولات استفاده می‌شود. فرض می‌شود که هر محصول تنها متعلق به یک دسته‌بندی است! ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`id`	`BIGINT(20)`	شناسه‌ی محصول
`category_id`	`BIGINT(20)`	شناسه‌ی دسته‌بندی محصول
`name`	`VARCHAR(255)`	نام محصول
`description`	`TEXT`	توضیحات محصول
`price`	`DECIMAL(15, 2)`	قیمت محصول
`created_at`	`TIMESTAMP`	زمان درج محصول

توجه داشته باشید که جدول دسته‌بندی‌های محصولات صرفاً جهت سادگی در سؤال نیامده است. نیازی به اطلاعات چنین جدولی نیست.

مطلوبات

کوئری‌های خواسته‌شده از شما، موارد زیر است:

۱. کوئری ساخت ایندکس جهت بهینه‌سازی حداکثری سرعت دریافت شناسه‌ی کاربرانی که نام‌شان با یک رشته‌ی خاص آغاز می‌شود:

```sql
SELECT id
FROM users
WHERE name LIKE 'Ali%'
```

۲. کوئری ساخت ایندکس جهت بهینه‌سازی حداکثری سرعت دریافت شناسه، نام و قیمت محصولاتی که قیمت‌شان بین دو عدد مشخص است و مربوط به یک دسته‌ی خاص هستند:

```sql
SELECT id, name, price
FROM products
WHERE price > 25000 AND price < 50000000 AND category_id = 5
```

نکات

- کوئری‌های شما باید روی آخرین نسخه‌ی *MySQL* قابل اجرا باشند.
- کوئری هر بخش باید تنها شامل یک *statement* باشد.
- هر کوئری امتیاز جداگانه دارد و اگر کوئری یک قسمت را نتوانستید بنویسید، کوئری‌هایی که حل کردید را بفرستید و قسمت آن کوئری را خالی بگذارید.

روش پیاده‌سازی

همانطور که در پیاده سازی زیر مشاهده می‌کنید، در فایل نهایی به تعداد Query های مورد انتظار در صورت سوال یکسری کامنت وجود دارند که نباید پاک بشوند.

شما باید دقیقا Query های مرتبط با هر بخش را در زیر کامنت‌های همان بخش در یک فایل با نام `code.sql` بنویسید و در نهایت آن را آپلود نمایید.

توجه کنید که هر کوئری نمره‌ای جداگانه دارد و اگر کوئری یک قسمت را نتوانستید بزنید، کوئری‌هایی که حل کردید را بفرستید و قسمت آن کوئری را خالی بگذارید.

```sql
-- Section1
   SELECT *
   FROM x
-- Section2


```

```
SELECT *
FROM y
```

ثبت‌نام کننده قلی

کد شما باید روی *PostgreSQL* قابل اجرا باشد.

در این سوال، پایگاه داده شرکت قلی در اختیار شما قرار گرفته است. او در شرکت خود برنامه‌ای نوشته است که وظیفه ثبت‌نام افراد را دارد. همچنین برنامه او این وظیفه را دارد که ثبت‌نام ها را به درخواست افراد نهایی کند و خب این امکان وجود دارد که برخی از این درخواست ها با timeout مواجه شوند پس او وظیفه دارد که bottleneck های سیستم خود را پیدا کرده و در راستای بهبود آنها تلاش کند.

جزئیات پایگاه‌داده

داده‌های اولیه برای **تست نهایی** را از این لینک دانلود کنید.

▼ **توضیحات در مورد داده‌های اولیه**

در فایل `registration.zip` فایلی به اسم `initial.sql` وجود دارد.

ابتدا پایگاه‌داده‌ای با نام `registration` در سیستم خود بسازید. با اجرای این فایل برروی این پایگاه‌داده، همه جداول و سطرهایی که برای **تست نهایی** مورد استفاده قرار می‌گیرد در سیستم شما ایجاد می‌شود.

▼ **توضیحات جداول دیتابیس**

ساختار جداول به‌شرح زیر است:

جدول `signups` : از این جدول برای نگه‌داری اطلاعات مربوط به ثبت نام افراد استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`user_id`	`integer`	شناسه‌ی کاربر
`time_stamp`	`date`	تاریخ و ساعت ثبت‌نام

جدول `confirmations` : از این جدول برای نگهداری اطلاعات درخواست های افراد در راستای نهایی کردن ثبت‌نام آن‌ها استفاده می‌شود.

نام ستون	نوع	تعریف
`user_id`	`integer`	شناسه‌ی کاربر
`time_stamp`	`date`	تاریخ و ساعت درخواست
`action`	`character varying(10)`	نتیجه درخواست (timeout, confirmed)

مطلوبات

۱. نرخ تایید هر کاربر به صورت اعشاری (`float`).

▼ *توضیحات مربوط به کوئری اول*

نرخ تایید یک کاربر، برابر با تعداد درخواست های تایید شده تقسیم بر تعداد کل درخواست های ارسال شده از طرف همان کاربر است.

شناسه کاربر را در ستونی با نام `user_id` و نرخ تایید مربوط به آن کاربر را در ستونی با نام `confirmation_rate` نمایش دهید.

▼

•

توجه *

توجه کنید خروجی شما باید به ترتیب **صعودی** بر حسب شناسه کاربری افراد باشد. همچنین دقت کنید که نرخ تایید به ازای هر کاربر باید تا **دورقم** اعشار گرد شده باشد.

**نکته : در صورتی که یک کاربر درخواست تایید نداده باشد نرخ تایید برای او برابر با 0.00 باید در نظر گرفته شود.

5 سطر اول خروجی شما باید به شکل زیر باشد.

user_id	confirmation_rate
1	0.00
2	0.67
3	0.00
4	1.00
5	1.00

۲. تعداد کل timeout ها در یک شبانه روز.

▼ *توضیحات مربوط به کوئری دوم*

قلی قصد دارد بفهمد بالاترین load بر روی سیستم در چه ساعتی اتفاق می‌افتد به همین منظور او تصمیم گرفته که تعداد timeout ها را به ازای هر ساعت از شبانه روز بدست آورد. هر ساعت از شبانه روز را در ستونی با نام `hour` و تعداد timeout های آن ساعت را در ستونی با نام `timeout_count` قرار دهید.

▼ **توجه**

توجه داشته باشید که بازه‌ی زمانی ساعت های شبانه روز از 0 - 23 در نظر گرفته شده است.

همچنین انتظار می‌رود ترتیب خروجی‌ها بر اساس ساعت به صورت **صعودی** باشد.

3 سطر اول خروجی شما باید به شکل زیر باشد.

hour	timeout_count
0	71
1	62
2	72

۳. تاریخ آخرین تایید به ازای هر کاربر.

▼ *توضیحات مربوط به کوئری سوم*

قلی قصد دارد تاریخ و ساعت مربوط به آخرین درخواست تایید هر کاربر را پیدا کند. به این منظور در جدول خروجی دو ستون قرار دارد یکی برای شناسه کاربر (`user_id`) و دیگری برای تاریخ و ساعت آخرین درخواست تایید (`latest_confirmation`) .

▼ توجه

درصورتی که یک کاربر هیچ درخواست تاییدی نداشت باید مقدار **null** در ستون latest_confirmation برای او نوشته شود.

خروجی ها باید بر اساس آخرین تاریخ تایید و به صورت **صعودی** درج شده باشند.

3 سطر اول خروجی شما باید به شکل زیر باشد.

user_id	latest_confirmation
1284	2024-10-01 10:32:00
1656	2024-10-01 12:01:00
1828	2024-10-01 13:37:00

روش پیاده‌سازی

همانطور که در پیاده سازی زیر مشاهده می‌کنید، در فایل نهایی به تعداد Query های مورد انتظار در صورت سوال یکسری کامنت وجود دارند که نباید پاک بشوند.

شما باید دقیقا Query های مرتبط با هر بخش را در زیر کامنت های همان بخش در یک فایل با نام `code.sql` بنویسید و در نهایت آن را آپلود نمایید.

```sql
-- Section1
   SELECT *
   FROM x
-- Section2
   SELECT *
   FROM y
-- Section3
   SELECT *
   FROM z
```

نرمال بهتره

کوئری‌های شما باید روی *PostgreSQL* قابل اجرا باشند.

قلی زبان SQL را بلد است و تجربه کار با دیتابیس‌ها را دارد، اما تاکنون خودش جداول یک دیتابیس را طراحی نکرده. پس به تمرین زیادی احتیاج دارد. او تصمیم گرفت سری به جداول خیلی قدیمی و خاک گرفته‌ی شرکت بزند و به عنوان تمرین آن‌ها را نرمال‌سازی کند.

جزئیات پروژه

قسمتی از جدول اولیه *سفرها* در این لینک قابل مشاهده است.

اما همان‌طور که مشخص است، طراحی این دیتابیس اصلاً جالب نیست و افزونگی داده دارد.

جدول موجود به نام `rides` به شکل زیر است:

نام ستون	نوع	تعریف
ride_id	bigint	آی‌دی سفر
passenger_name	varchar(255)	نام مسافر
passenger_phone	varchar(15)	شماره تلفن مسافر
driver_name	varchar(255)	نام راننده
driver_phone	varchar(15)	شماره تلفن راننده
car_model	varchar(255)	مدل خودرو
car_plate	varchar(10)	پلاک خودرو
pickup_location	text	محل شروع سفر
dropoff_location	text	محل پایان سفر
fare	bigint	کرایه سفر
ride_date	date	تاریخ سفر
discount_code	varchar(10)	کد تخفیف
vehicle_type	varchar(50)	نوع وسیله نقلیه
traffic_condition	varchar(50)	شرایط ترافیکی

مطلوبات

وظیفه شما این است که جدول `rides` را به چندین جدول مرتبط تقسیم کنید تا افزونگی داده‌ها حذف و یکپارچگی داده‌ها حفظ شود. شما باید نرمال‌سازی را تا جایی ادامه دهید که هیچ وابستگی تابعی نقض نشود و پایگاه داده حداقل در فرم نرمال سوم (3NF) باشد.

شما باید دستورات `CREATE TABLE` (*SQL DDL*) را برای ایجاد جداول نرمال‌سازی شده بنویسید.

یک نکته‌ی مهم این سؤال، نام‌گذاری جداول است. باید طبق جدول زیر از نام هر یک از جداول که احتیاج دارید، استفاده کنید. لازم به ذکر است که جدول شما می‌تواند ترکیبی از اسم های زیر باشد که با `_` به یکدیگر متصل شده اند.

کلید	نام جدول
مسافران	`passengers`
پرداخت‌ها	`payments`
مکان‌ها	`locations`
سفرها	`rides`
جزئیات سفر	`trip_details`
وسایل نقلیه	`vehicles`
تخفیف‌ها	`discounts`
وسایل نقلیه و رانندگان	`drivers-vehicles`
شرایط ترافیک	`traffic_conditions`
رانندگان	`drivers`
شاخص‌های افزایشی	`surge_multipliers`
امتیازات	`ratings`
انواع وسایل نقلیه	`vehicle_types`

نکات

- **دقت‌کنید** که نیاز به استفاده از همه‌ی جداول ذکر شده در جدول بالا نیست و فقط باید از جداولی که به آن‌ها احتیاج دارید استفاده کنید.
- تمام اطلاعات ستون‌ها (مانند نام و...) باید مشابه موارد ذکر شده در اول سؤال باشند و نباید تغییر کنند.
- امکان `NULL` بودن در هیچ یک از ستون‌ها مجاز نیست.
- کلیدهای خارجی و اصلی باید به درستی ایجاد شوند نام آن‌ها نام جدول به علاوه `_id` است.
- توجه کنید که هر راننده فقط یک خودرو دارد. در واقع راننده و خودرو از هم جدا نیستند.
- هر کد تخفیف می‌تواند توسط چندین مسافر استفاده شود اما هر مسافر برای هر سفر تنها می‌تواند از یک کد تخفیف استفاده کند.
- شرایط ترافیکی تأثیر مستقیم روی کرایه سفر دارد. جدول `traffic_conditions` فقط نوع شرایط و ضریب مربوطه را نگهداری می‌کند.
- انواع وسایل نقلیه به صورت مستقل در جدول `vehicle_types` ذخیره می‌شوند و طبیعتاً هر وسیله نقلیه فقط یک نوع دارد.
- نرمال‌سازی باید به گونه‌ای انجام شود که هیچ وابستگی تابعی نقض نشود و هیچ گونه داده تکراری و افزونگی وجود نداشته باشد. تمامی روابط بین موجودیت‌ها باید دقیقاً طبق دستورالعمل انجام شود.
- دیتابیس تست شامل هیچ جدولی نیست و شما فقط باید با دستورات خود جدول‌ها را ایجاد کنید.
- در نهایت در جواب شما باید جدول rides وجود داشته باشد و جداول دیگر در کنارش هستند.

آن‌چه باید آپلود کنید

پس از طراحی دستورات، کد خود را در قالب یک فایل با پسوند `.sql` آپلود کنید.

فوتبالیستا

کد شما باید روی *PostgreSQL* قابل اجرا باشد.

در این سوال، بخشی از پایگاه داده مربوط به مسابقات فوتبال اروپا در اختیار شما قرار گرفته است.

جزئیات پایگاه‌داده

داده‌های اولیه برای **تست نهایی** را از این لینک دانلود کنید.

▼ **توضیحات در مورد داده‌های اولیه**

در فایل `football.zip` فایلی به اسم `initial.sql` وجود دارد.

ابتدا پایگاه‌داده‌ای با نام `football` در سیستم خود را بسازید و با اجرای این فایل برروی این پایگاه‌داده، همه جداول و سطرهایی که برای **تست نهایی** مورد استفاده قرار می‌گیرد در سیستم شما ایجاد می‌شود.

▼ **توضیحات جداول دیتابیس**

ساختار جداول به‌شرح زیر است:

جدول `players` : از این جدول برای نگه‌داری اطلاعات بازیکن‌ها استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`player_id`	`integer`	شناسه‌ی بازیکن
`current_club_id`	`integer`	شناسه‌ی باشگاه فعلی بازیکن
`player_code`	`character varying(64)`	کد بازیکن (نام کامل بازیکن)
`country_of_birth`	`character varying(32)`	کشور تولد بازیکن
`city_of_birth`	`character varying(64)`	شهر تولد بازیکن
`country_of_citizenship`	`character varying(32)`	کشور ملیت بازیکن
`date_of_birth`	`date`	تاریخ تولد بازیکن
`sub_position`	`character varying(32)`	تخصص دوم بازیکن
`position`	`character varying(16)`	تخصص اول بازیکن
`foot`	`character varying(8)`	پای تخصصی بازیکن
`height_in_cm`	`integer`	قد بازی کن به سانتی متر
`contract_expiration_date`	`date`	تاریخ انقضای قرارداد بازیکن

جدول `clubs` : از این جدول برای نگه‌داری اطلاعات باشگاه‌ها استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`club_id`	`integer`	شناسه‌ی باشگاه
`name`	`character varying(64)`	نام باشگاه
`domestic_competition_id`	`character varying(4)`	لیگ باشگاه
`squad_size`	`integer`	اندازه تیم
`foreigners_number`	`integer`	تعداد افراد خارجی تیم
`national_team_players`	`integer`	تعداد بازیکن‌های تیم ملی
`stadium_name`	`character varying(64)`	نام استادیوم اختصاصی باشگاه
`stadium_seats`	`integer`	کشور ملیت بازیکن
`net_transfer_record`	`character varying(16)`	ارزش خالص باشگاه

جدول `competitions` : از این جدول برای نگه‌داری اطلاعات مسابقات استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
`competition_id`	`character varying(4)`	شناسه مسابقات
`name`	`character varying(64)`	نام مسابقات
`type`	`character varying(32)`	نوع مسابقات
`country_name`	`character varying(16)`	کشور مسابقات

جدول `games` : از این جدول برای نگه‌داری اطلاعات بازی‌ها استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
game_id	integer	شناسه‌ی بازی
competition_id	character varying(4)	شناسه سری مسابقات
season	integer	فصل برگذاری بازی
date	date	تاریخ بازی
home_club_id	integer	شناسه‌ی تیم (باشگاه) میزبان
away_club_id	integer	شناسه‌ی تیم مهمان
home_club_goals	integer	اتعداد گل تیم میزبان
away_club_goals	integer	تعداد گل تیم مهمان
stadium	character varying(64)	نام استادیوم بازی
attendance	integer	تعداد تماشاگرها

جدول `apearances` : از این جدول برای نگه‌داری اطلاعات حضور بازیکن‌ها در بازی استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
appearance_id	character varying(16)	شناسه‌ی جدول
game_id	integer	شناسه‌ی بازی
player_id	integer	شناسه‌ی بازیکن
yellow_cards	integer	تعداد کارت زردهای بازیکن
red_cards	integer	تعداد کارت قرمزهای بازیکن
goals	integer	تعداد گل‌های بازیکن
assists	integer	تعداد پاس گل‌های بازیکن
minutes_played	integer	تعداد دقایقی که بازیکن بازی کرده‌است

جدول `game_events` : از این جدول برای نگه‌داری اطلاعات اتفاق‌های مسابقه (گل، موقیت گل، پنالتی و ...)استفاده می‌شود. ساختار این جدول به‌صورت زیر است:

نام ستون	نوع	تعریف
game_event_id	integer	شناسه‌ی جدول
game_id	integer	شناسه‌ی بازی
minute	integer	زمان(دقیقه) اتفاق
type	character varying(16)	نوع اتفاق
player_id	integer	شناسه‌ی بازیکن اصلی اتفاق
player_in_id	integer	شناسه‌ی بازیکن که از گل جلوگیری کرده است
player_assist_id	integer	شناسه‌ی بازیکنی که پاس گل داده است

مطلوبات

۱. برای هر باشگاه، تعداد بازیکنانی را که ملیت آن‌ها با کشوری که مسابقات داخلی باشگاه در آن برگزار می‌شود مطابقت ندارد، به دست آورید.

▼ *توضیحات مربوط به کوئری اول*

نام باشگاه را در ستونی با نام `club_name` و تعداد بازیکن‌های خارجی آن را در ستونی با نام `foreign_player_count` نمایش دهید. توجه کنید خروجی شما باید به ترتیب **نزولی** بر حسب تعداد بازیکن خارجی و در صورتی که این مقدار برابر بود بر اساس نام باشگاه به‌صورت **صعودی** مرتب شود.

3 سطر اول خروجی شما باید به شکل زیر باشد.

foreign_player_count	club_name
64	Portimonense Futebol SAD
51	Makina ve Kimya Endüstrisi Ankaragücü Spor Kulübü
50	Aris Thessalonikis

۲. سری مسابقات را بر حسب مجموع تعداد اتفاق‌های آن (تعداد گل‌ها، موقیت‌های گل، کارت‌ها و ...) رتبه بندی کنید.

▼ *توضیحات مربوط به کوئری دوم*

رتبه سری مسابقات بر اساس مجموع تعداد اتفاق‌های آن (پر حادثه‌ترین رتبه اول) را در ستونی با نام `ranking` و نام سری مسابقات را در ستونی با نام `name` و تعداد اتفاق‌های در آن سری مسابقات را در ستونی با نام `events` قرار دهید. ستون‌ها را ابتدا برحسب رتبه، سپس براساس نام سری مسابقات به‌صورت **صعودی** مرتب کنید.

▼ *توجه*

اگر رتبه دو سری مسابقات با بیشترین تعداد حوادث، یکسان بود، هردو رتبه یک می‌شوند و سری مسابقات بعدی رتبه‌اش دو می‌شود، این قانون برای تمامی رتبه‌ها صادق است.

3 سطر **آخر** خروجی شما باید به شکل زیر باشد.

ranking	name	events
39	johan-cruijff-schaal	47
40	trophee-des-champions	45
41	ukrainian-super-cup	33

۳.لیستی از بازیکنانی که قرارداد آن‌ها قبل از تاریخ ۱ ژانویه ۲۰۲۵ به اتمام می‌رسد و در فصل اخیر بازی کرده‌اند، همراه با عملکرد آن‌ها (شامل تعداد گل‌ها، تعداد پاس گل‌ها و مجموع دقایق بازی) ارائه دهید.

▼ *توضیحات مربوط به کوئری سوم*

نام یا همان کد بازیکن را در ستونی به نام `player_code` (این ستون را یونیک فرض کنید)، تاریخ انقضای قرارداد را در ستونی با نام `contract_expiration_date` ، تعداد گل‌ها را در ستونی به نام `total_goals` ، تعداد پاس گل‌هایی که داده را در ستونی با نام `total_assists` و تعداد دقایقی که بازی کرده را در ستونی به نام `total_minutes` قرار دهید. ستون‌ها را ابتدا برحسب تاریخ انقضای قرارداد به صورت **صعودی** ، سپس برحسب تعداد گل‌های بازیکن به صورت **نزولی** و سپس برحسب پاس گل‌هایی که داده به‌صورت **نزولی** و در نهایت براساس تعداد دقایقی که بازی کرده به‌صورت **نزولی** مرتب کنید.

3 سطر اول خروجی شما باید به شکل زیر باشد.

player_code	contract_expiration_date	total_goals	total_assists	total_minutes
pedrinho	2023-06-30	0	0	88
abdallah-sima	2024-05-31	14	2	1868
simon-murray	2024-05-31	11	4	2607

روش پیاده‌سازی

همانطور که در پیاده سازی زیر مشاهده می‌کنید، در فایل نهایی به تعداد Query های مورد انتظار در صورت سوال یکسری کامنت وجود دارند که نباید پاک بشوند.

شما باید دقیقا Query های مرتبط با هر بخش را در زیر کامنت‌های همان بخش در یک فایل با نام `code.sql` بنویسید و در نهایت آن را آپلود نمایید.

توجه کنید که هر کوئری نمره‌ای جداگانه دارد و اگر کوئری یک قسمت را نتوانستید بزنید، کوئری‌هایی که حل کردید را بفرستید و قسمت آن کوئری را خالی بگذارید.

```sql
-- Section1
   SELECT *
   FROM x
-- Section2
   SELECT *
   FROM y
-- Section3
   SELECT *
   FROM z
```

آبگیر

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

یک روز قلی به جنگلی رفت که پر از تپه‌های کوچک و بزرگ بود. این تپه‌ها، یکی پس از دیگری به دنبال هم چیده شده بودند، درست مثل ستون‌های یک نقشه ارتفاع. قلی با خودش فکر کرد: *اگه بارون بیاد، بین این تپه‌ها چقدر آب گیر میفته؟* اما محاسبه‌ی این کار آسان نبود!

قلی که همیشه عاشق حل مسائل سخت بود، شروع کرد به بررسی نقشه ارتفاع این تپه‌ها. بعضی تپه‌ها بلند بودند، بعضی کوتاه، و برخی مثل چاله‌های کوچکی بودند که می‌شد آب در آن‌ها جمع شود. قلی فهمید که باید بفهمد بین هر دو تپه بلند، چقدر آب می‌تواند به دام بیفتد.

قلی به خودش گفت: *باید از دو طرف نقشه‌ی این تپه‌ها شروع کنم و هر بار ارتفاع بلندترین تپه رو با تپه‌های کناری مقایسه کنم. اینطوری می‌تونم مقدار آبی که بینشون گیر میفته رو محاسبه کنم.*

به شما آرایه‌ای از n عدد صحیح غیر منفی داده شده است که نمایانگر نقشه‌ی ارتفاع است، که عرض هر ستون برابر ۱ است. حالا باید محاسبه کنید که پس از بارش باران، چه مقدار آب بین این ستون‌ها می‌تواند به دام بیفتد. برای درک بهتر به شکل مثال یک توجه کنید.

ورودی

در خط اول عدد n می آید که نشانگر تعداد تپه ها است. در n خط بعدی به ترتیب ارتفاع تپه ها از چپ به راست می آید.

$$1 \leq n \leq 20000$$

$$0 \leq h \leq 100000$$

خروجی

در خروجی باید تعداد بلوک هایی که در آن ها آب جمع می شود چاپ شود.

مثال

ورودی نمونه ۱

```
12
0,1,0,2,1,0,1,3,2,1,2,1
```

خروجی نمونه ۱

```
6
```

ورودی نمونه ۲

```
6
4,2,0,3,2,5
```

خروجی نمونه ۲

```
9
```

آن‌چه باید آپلود کنید

یک فایل `main.go` که شامل پیاده‌سازی کامل راه حل باشد.