Departments

> کدهای مربوط به هر بخش را در فایل‌هایی با نام‌های `part1.sql` و `part2.sql` و... قرار دهید و همه را با هم zip کنید و اپلود کنید. دقت کنید که پاسخ‌ها داخل فولدر دیگری نباشند و در root فایل zip قرار بگیرند.

ابتدا این فایل را دانلود کنید و آن را در یک دیتابیس PostgreSQL ران کنید تا تیبل‌ها و دیتاهای اولیه آن ساخته شود.

در این سوال، شما با مجموعه‌ای از جداول پایگاه داده مواجه خواهید شد که شامل اطلاعات مربوط به کارکنان، دپارتمان‌ها، مشتریان و فروش‌ها است. هر جدول نمایانگر بخشی از داده‌های مربوط به عملیات شرکت است که باید با استفاده از زبان SQL تجزیه و تحلیل شوند. در ادامه، جداول و داده‌های مختلف این پروژه توضیح داده شده‌اند:

۱. جدول departments: این جدول شامل اطلاعات دپارتمان‌های مختلف شرکت است. این اطلاعات برای انجام تجزیه و تحلیل‌هایی مانند دسته‌بندی دپارتمان‌ها بر اساس تقسیم‌بندی‌های سازمانی کاربرد دارد.
۲. جدول regions: این جدول اطلاعات جغرافیایی در مورد مناطق مختلف شرکت را شامل می‌شود. این اطلاعات می‌تواند برای تقسیم‌بندی داده‌ها یا محاسبه مجموع‌ها و میانگین‌ها بر اساس منطقه کاربرد داشته باشد.
۳. جدول employees : این جدول اطلاعات مربوط به کارکنان شرکت را شامل می‌شود. این داده‌ها برای انجام تحلیل‌هایی مانند تقسیم‌بندی کارمندان بر اساس حقوق یا محاسبه مجموع حقوق در دپارتمان‌ها استفاده می‌شوند.
۴. جدول customers: این جدول شامل اطلاعات مشتریان شرکت است. این اطلاعات برای تحلیل‌هایی نظیر شناسایی برترین مشتریان بر اساس تعداد خرید یا تجزیه‌وتحلیل بخش‌های مختلف مشتریان کاربرد دارد.
۵. جدول sales: این جدول اطلاعات مربوط به فروش‌های شرکت را شامل می‌شود. این داده‌ها برای تحلیل فروش‌ها، محاسبه مجموع فروش‌ها، و رتبه‌بندی مشتریان بر اساس خریدهایشان استفاده می‌شوند.

سوال ۱

با استفاده از عبارت WITH یک جدول میانی بسازید و با به‌کارگیری تابع NTILE از Window Function، کارمندان را به ۵ گروه مساوی بر اساس حقوق تقسیم کنید. سپس با استفاده از GROUP BY و AVG میانگین حقوق هر گروه را محاسبه نمایید.

ستون‌های خروجی:

rank_of_salary, avg_SALARY

سوال ۲

با استفاده از تابع SUM از Window Function و با بهره‌گیری از PARTITION BY و ORDER BY، مجموع حقوق (running_total) هر دپارتمان را بر اساس تاریخ استخدام کارمندان محاسبه کنید.

ستون‌های خروجی:

first_name, hire_date, department, salary, running_total

سوال ۳

با استفاده از یک زیرپرس‌وجو (Subquery) و JOIN، میانگین حقوق هر دپارتمان را محاسبه کرده و سپس کارمندانی را پیدا کنید که حقوقشان بیشتر از میانگین حقوق دپارتمان خودشان است.

ستون‌های خروجی:

employee_id, first_name, department, salary, avg_salary

سوال ۴ با استفاده از توابع FIRST_VALUE و LAST_VALUE از Window Functionها، برای هر division، نام اولین و آخرین department را بر اساس ترتیب الفبایی پیدا کنید.

ستون‌های خروجی:

division، department، first_department_in_division، last_department_in_division

سوال ۵

با استفاده از GROUP BY و تابع RANK از Window Function، یک Materialized View با نام mv_top_customers ایجاد کنید که مشتریان را بر اساس تعداد خریدهایشان رتبه‌بندی کند. سپس با یک

کوئری جداگانه، فقط سه مشتری برتر را از این نما بازیابی کنید.

ستون‌های خروجی:

customer_id, total_purchases, rank

سوال ۶

با استفاده از LATERAL JOIN، برای هر مشتری دسته‌بندی محصولی را بیابید که بیشترین تعداد خرید را از آن‌ها داشته است. در زیرپرس‌وجوی مربوط به LATERAL، از ترکیب GROUP BY برای گروه‌بندی خریدها بر اساس دسته‌بندی محصولات، ORDER BY به منظور مرتب‌سازی دسته‌ها بر اساس تعداد خرید به صورت نزولی LIMIT برای محدود کردن نتایج به دسته‌ی برتر استفاده کنید. هدف، ارائه‌ی لیستی از مشتریان به همراه دسته‌ی محبوب‌شان (از نظر تعداد خرید) است.

ستون‌های خروجی:

customer_id, customer_name, category, purchase_count

Library Management

> کدهای مربوط به هر بخش را در فایل‌هایی با نام‌های `part1.sql` و `part2.sql` و... قرار دهید و همه را با هم zip کنید و اپلود کنید. دقت کنید که پاسخ‌ها داخل فولدر دیگری نباشند و در root فایل zip قرار بگیرند.

ابتدا این فایل را دانلود کنید و آن را در یک دیتابیس PostgreSQL ران کنید تا تیبل‌ها و دیتاهای اولیه آن ساخته شود.

۱. تابعی به نام `calculate_lateness` بنویسید که به ازای هر کتابی که در تحویل امانت آن دیرکرد رخ داده، روزهای تاخیر را محاسبه کند و به ازای هر روز دیرکرد مبلغ 5000 تومان به عنوان جریمه محاسبه کند و سپس با استفاده ازwindow functions لیست اعضا را به همراه مجموع بدهی‌های ناشی از دیرکرد را به شکل نزولی مرتب کند و خروجی دهد.
۲. رویه‌ای به نام `membership_renewal` ایجاد کنید که کد عضویت شخص را دریافت کند و در صورتی که مجموع جریمه‌های پرداخت نشده شخص کمتر از 10000 باشد، عضویت او را 1 سال تمدید کند و پیام مناسب نشان دهد.
۳. یک Trigger جدید ایجاد کنید که هنگام درج رکورد جدید در جدول امانت‌ها، موارد زیر را چک کند و در صورت نقض هر کدام، عملیات را لغو کند:
    - کتاب مورد نظر باید در دسترس باشد.
    - بررسی شود که کاربر عضویت فعال داشه باشد.
۴. یک کوئری بنویسید که لیست کتاب‌هایی را که بیشترین تعداد امانت در سال جاری را داشتند اما در ماه گذشته حتی یک بار هم امانت داده نشده‌اند را برگرداند.
۵. یک assertion جدید بسازید که تضمین کند هیچ کتابی بیشتر از 5 بار به طور همزمان نمی‌تواند امانت داده شود (موجودی هر کتاب در کتابخانه ما 5 عدد است). برای پیاده‌سازی این بخش می‌توانید از Triggerها استفاده کنید.
۶. دو View با توضیحات زیر ایجاد کنید:
    - یک View به نام `Active_loans` که کتاب‌هایی که امانت داده شده‌اند را به همراه شخص امانت‌گیرنده لیست می‌کند. نام و نام‌خانوادگی شخص امانت‌گیرنده، نام کتاب و نام نویسنده کتاب را خروجی دهید.

- یک View به نام `Popular_books` که بر اساس تعداد دفعاتی که یک کتاب امانت گرفته شده، کتاب‌ها را امتیازدهی می‌کند و نمایش می‌دهد. نام کتاب، نویسنده کتاب و تعداد دفعات را خروجی دهید.

Anime

پیاده‌سازی API با ORM

در این تمرین یک فایل `SQL dump` به شما داده می‌شود که شامل داده‌های اولیه است. شما باید یک سرور HTTP پیاده‌سازی کنید که با استفاده از یک ORM داده‌ها را از پایگاه داده خوانده و پردازش کند. زبان برنامه‌نویسی پیاده‌سازی اهمیتی ندارد، اما باید از یک ORM برای انجام کوئری‌ها استفاده شود.

توجه: نیاز است تا مدیریت Session و اتصال به دیتابیس بر عهده خودتان باشد. **پس مجاز به استفاده از فریم‌ورک‌هایی نظیر Django** نیستید.

توجه: استفاده از `Direct SQL Querying` یا `Raw SQL Querying` مجاز نیست!

می‌توانید یکی از کتابخانه‌های زیر یا کتابخانه‌های مشابه را استفاده کنید:

- پایتون: Flask + SQLAlchemy یا FastAPI + SQLAlchemy
- جاوااسکریپت: Sequelize + Express یا Sequelize + Fastify یا Prisma + Express
- گولنگ: Gin + Gorm
- جاوا: JPA + Spring Boot

شما می‌توانید از این فایل Dump دیتابیس برای تست کوئری‌های خود استفاده کنید.

اندپوینت‌ها و کوئری‌ها

۱: دریافت لیست انیمه‌ها

```
GET /anime/top
```

کوئری: شناسه، عنوان، امتیاز و تعداد اپیزود ۱۰ انیمه با بیشترین تعداد اپیزود را به صورت نزولی بازگردانید.

۲: دریافت کاربران با میانگین امتیاز بالا

```
GET /users/top?page=1&offset=10&year=2017&gender=F
```

کوئری: دریافت username و میانگین امتیاز کاربران خانمی که میانگین رای آن‌ها بالاتر از ۸ است و بعد از سال ۲۰۱۷ در سایت ثبت‌نام کرده‌اند. (به ترتیب میانگین امتیاز نزولی با امکان pagination).

۳: دریافت لیست انیمه‌هایی که یک کاربر دیده است

```
GET  /users/:username/watched?count=10
```

کوئری: آیدی، عنوان، نمره‌ی کاربر و تعداد اپیزودهای انیمه‌هایی که یک کاربر خاص تماشا کرده است. (10تای اول به ترتیب صعودی براساس نمره‌ای که کاربر داده است)

۴: محبوب‌ترین ژانر در بین کاربران

```
GET  /anime/popular
```

کوئری: نام سه ژانر محبوب انیمه بر اساس دفعات دیده شدن توسط کاربران. (راهنمایی: هر رکورد در جدول user_anime_list را معادل یک بار دیده شدن در نظر بگیرید)

۵: کاربران فعال در سالی خاص و بیشترین تماشای انیمه

```
GET  /users/active/:year
```

کوئری: یوزرنیم و روزهای تماشای انیمه‌ی کاربرانی که بیشترین روز را صرف تماشای انیمه کرده‌اند. (۵ کاربر برتر)

۶: کاربران هم سلیقه با یک کاربر خاص

```
GET  /users/:username/similars
```

کوئری: یوزرنیم و تعداد انیمه‌های مشترک کاربرانی که انیمه‌ی مشترک را با یک کاربر خاص دارند. (به ترتیب نزولی تعداد انیمه‌ی مشترک)

۷: اضافه کردن به تعداد اپیزود انیمه

```
POST  /anime/:anime_id/episodes?value=1
```

کوئری: فرض کنید تعداد اپیزودهای یک انیمه تغییر کرده و می‌خواهید مقدار آن را تغییر دهید. (در صورتی که مقدار تعداد اپیزود مشخص نشده بود، به صورت پیشفرض عدد یک را به تعداد اپیزودهای انیمه مورد نظر اضافه کنید)

پاسخ:یک آیدی انیمه به همراه تعداد اپیزودهای فعلی آن (پس از تغییر) را برگردانید.

نحوه پاسخ‌دهی سرور

سرور باید در جواب به یک درخواست، یک آرایه برگرداند. آرایه مورد نظر باید شامل objectهایی باشد که به ازای هر column یک key دارند.

به عنوان مثال، به جدول و جیسون زیر توجه بفرمایید:

user_id	username	age	email
1	Alice	25	alice@example.com
2	Bob	30	bob@example.com
3	Charlie	22	charlie@example.com

```json
[
    {
        "user_id": 1,
        "username": "Alice",
        "age": 25,
        "email": "alice@example.com"
    },
    {
        "user_id": 2,
        "username": "Bob",
        "age": 30,
        "email": "bob@example.com"

```

```
    },
    {
        "user_id": 3,
        "username": "Charlie",
        "age": 22,
        "email": "charlie@example.com"
    }
]
```

نحوه ارسال پاسخ

از صحت کارکرد برنامه خود یک ویدیو کوتاه ضبط کنید و در آن تمامی APIها را تست کنید. همچنین به صورت مختصر راجع به کتابخانه‌های مورد استفاده در کد خود و نحوه کارکرد آن توضیح دهید.

ویدیو و کدهای خود را Zip کرده و در این بخش اپلود کنید.