بسم الله الرحمن الرحیم

گزارش فاز دوم پروژه دیتابیس

اعضای گروه:

علیرضا فرشی

سیده فاطمه موسوی 400105252

زهرا زاهدی فر99101716

استاد:

دکتر مرتضی امینی

تاریخ : 28/9/1402

کوئری 1 :

ابتدا با استفاده از دستور set session، ورودی های خواسته شده کوئری را تعریف میکنیم. سپس تعداد سطر های JOIN دو جدول driving , travel را سلکت میکنیم. سپس محدودیت های زمان را تعیین میکنیم و شناسه راننده را مساوی شناسه ورودی قرار میدهیم. گروپ بای انجام شده، یک سفر را نشان میدهد یعنی با این ماشین، با این مبدا و مقصد و با این تاریخ، یعنی در این سفر ببین که اگر نسبت تعداد مسافران خانم به آقا بیشتر یا مساوی 60 درصد بود، تعداد سطر های چنین جدولی را بشمار.(برای محاسبه تعداد مسافران خانم و آقا هم چک شده که آیا citizen_id فرد مسافر، gender ای مساوی F دارد یا M.)

کوئری 2:

برای بدست آوردن خواسته این پرسمان، جدول citizen را با payment_reciept پیوند میدهیم و محدودیت زمانی خواسته شده در پرسمان را تعیین میکنیم. سپس شهروندان را طبق کسی که تکفل آن ها را برعهده دارد، گروه میکنیم یعنی هر خانواده را جدا محاسبه میکنیم. در هر خانواده چک میکنیم اگر مجموع هزینه رسید پرداخت در میان 5 تا بیشترین مجموع هزینه های رسید پرداخت بود، آن را چاپ میکنیم؛ به این صورت که مجموع هزینه رسیدهای پرداخت را سورت کرده و 5 تای اول آن را سلکت میکنیم.

کوئری 3:

این کوئری هم مانند کوئری قبلی است با این تفاوت که پس از تعیین بازه زمانی، هر راننده را به صورت جداگانه چک میکنیم که آیا مجموع مسافت طی شده توسط او، در میان 5 تا بیشترین مجموع مسافت های طی شده توسط رانندگان هست یا خیر. اگر بود، national_id و مجموع مسافت طی شده توسط او را چاپ میکنیم.

کوئری 4:

برای تقسیم بندی ماهانه، از تابع DATE_TRUNC استفاده میکنیم و بازه زمانی را تعیین میکنیم و چک میکنیم که آیا در جدول travel، مبدا یا مقصد ما ایستگاهی که در ورودی داده شده بوده است یا خیر. از آنجایی که اندازه distinct ماشین ها را می شماریم، پس ماشین های تکراری شمرده نمیشوند.

کوئری 5:

محاسبه فاصله دو نقطه را با استفاده از موقعیت جغرافیایی دو نقطه میدانیم:

$$\sqrt{(x2-x1)^2-(y2-y1)^2}$$

با توجه به فاصله ها کوئری ها را سورت کرده و اگر station_id داده شده در میان 5 تا فاصله اول بود، آن را چاپ میکنیم.

کوئری6:

ابتدا با استفاده از دستور SET SESSION، مقدار متغیرهای start_time و end_time تنظیم میشود، که از آنها برای محاسبه بازه زمانی استفاده میشود.

در کوئری از دو جدول Citizen و Citizen_Visit استفاده شده است.

با استفاده از INNER JOIN، اطلاعات شهروندانی که از ایستگاه ها بازدید کرده اند و اطلاعات بازدیدشان باهم متصل می شوند.

در بخش HAVING از زیرکوئری ای استفاده میشود که تعداد ایستگاه های بازدید شده هر شهروند در بازه زمانی داده شده را محاسبه میکند. سپس مقایسه میشود که آیا تعداد بازدید های هر شهروند بیشتر از دیگران است یا خیر. ORDER BY COUNT(*) برای مرتب سازی ردیف ها بر اساس تعداد تکرار آن ها در جدول می باشد و LIMIT 5 نشان دهنده این است که فقط 5 ردیف اول نشان داده شود.

کوئری7:

ابتدا با استفاده از دستور SET SESSION، مقدار متغیرهای start_time و end_time تنظیم میشود، که از آنها برای محاسبه بازه زمانی استفاده میشود.

سپس با استفاده از دستور WITH، دو زیر کوئری به نام های MetroTravelTimes و BusTravelTimes تعریف شده اند.

در هر کدام از این دو زیر کوئری، زمان های سفر شهروندان با مترو و اتوبوس محاسبه میشود. این زیرکوئری ها از جدول travel_receipt و travel برای یافتن زمان سفر ها استفاده میکنند. محاسبه زمان سفر با استفاده از SUM(EXTRACT(EPOCH FROM (trr.end_time - trr.start_time))) انجام میشود که در آن EPOCH مقدار هر زمان را تبدیل به ثانیه میکند.

در بخش WHERE هر دو زیر کوئری، محدوده زمانی سفرها از session variables محاسبه شده از (my.vars.start_time و my.vars.end_time) استفاده میکند.

در کوئری اصلی، اطلاعات شهروندان national_code,first_name,last_name از جدول Citizen با اطلاعات محاسبه شده از زیر کوئری های MetroTravelTimes و BusTravelTimes جوین شده اند. در شرط WHERE هم اختلاف زمان سفر با مترو و اتوبوس برای هر شهروند محاسبه شده و تنها شهروندانی که زمان سفر با مترو بیشتر از زمان سفر با اتوبوس است نمایش داده میشود.

کوئری8:

در این کوئری هم مانند کوئری های قبلی از SET SESSION استفادع شده است.
عملگر DISTINCT برای جلوگیری از شمارش تکراری استفاده شده است.
در اینجا با استفاده از INNER JOIN دو جدول car و parking_receipt ادغام شده اند و سپس به شرطی که شناسه رسید پارکینگ با مقدار تنظیم شده برابر باشد و زمان ورود و خروج هر دو خودرو در بازه زمانی تعیین شده قرار دارد محدود شده است.
در نهایت هم نتیجه بر اساس برند و رنگ خودروها گروه بندی میشوند و تعداد خودروهای مختلف شمرده میشود.

کوئری9:

در واقع ما برای این کوری از کد موجود در استک اورفلو برای بدست آوردن بیشترین تعداد به همراه زمان آن کمک گرفتیم و لینک در زیر گذاشته شده است:
sql - How to get maximum number of concurrent events in postgresql? - Stack Overflow

کوئری10:

همانند دیگر کدها، از set session استفاده شده است و مقدار citizen_Id را در متغیر my.vars.citizen_Id قرار میدهد.
سپس به قسمت اصلی کوئری میرسیم که از دو جدول payment_receipt و citizen اطلاعات را بازیابی میکند.
عملگر EXTRACT در این کد برای استخراج ماه از تاریخ صادر شدن تراکنش استفاده شده است. همچنین با استفاده از SUM مبلغ کل پرداختی را محاسبه و با نام total_payment اسم گذاری کرده ایم.
بخش WHERE شرایط مربوط به انتخاب رکوردها را تعیین میکند که ما citizen_Id را با national_code

مطابقت میدهیم.

بعد از این ها با GROUP BY نتایج را بر اساس ماه صادر شدن تراکنش ها گروه بندی میکنیم و سپس با ORDER BY نتایج را بر اساس ماه صادر شدن تراکنش ها مرتب میکنیم.

کوئری 11 :

برای پیاده سازی کوری خواسته شده ابتدا دو متغیر که در ورودی به ما معرفی شده است یعنی مبلغ و شناسه ایستگاه را تعریف کردیم و سپس با استفاده از کوری بازگشتی تمام ایستگاه های ممکن را از ایستگاه داده شده به همراه فاصله شان را پیدا می کنیم و ذخیره می کنیم و سپس روی آن یک فیلتر اعمال کرده به گونه ای که ایستگاه های مقصدی را پیدا کن به طوری که تاکسی باشند و با این روش ابتدا تمام ایستگاه های تاکسی مشخص شده و سپس برای محاسبه مبلغ فاصله ای که بین دو ایستگاه را در مبلغ پایه سامانه حمل و نقل تاکسیرانی ضرب می کنیم و این مبلغ باید کمتر از مقدار داده شده باشد و روی این نیز شرط می گذاریم تا در نهایت ایستگاه های معتبر ما پیدا شوند.

کوئری 12 :

ما ابتدا از رسید پرداخت های موجود رسید هایی که مربوط به سامانه حمل و نقل هستند را فیلتر میکنیم و همچنین شهروندانی که دارای ماشین هستند را از جدول car که یک ستون به نام car_attribute دارد استخراج می کنیم به طوری که اگر این مقدار null نباشد یعنی دارای ماشین است. حال یک group by بر اساس کد ملی شهروندان می کنیم و سپس میانگین پرداختی آن ها را خروجی می دهیم.

کوئری 13 :

در اینجا ابتدا جدول parking_receipt که رسید پارکینگ می باشد و دارای اطلاعاتی مانند زمان ورود و خروج است را در نظر می گیریم و آن را در درون خودش ضرب کارتزین می کنیم. حال به ازای سطر هایی که مربوط به یک کد ملی هستند ابتدا چک میکنیم که اول زمان ورود و خروج در بازه مشخص شده هستند و سپس با تابع Date_Add و با در نظر گرفتن بازه 1 روزه چک میکینم که آیا با تاریخ متناظرش در این سطر برابر است یا خیر و در صورتی که برابر بود بدین معناست که این شهروند در دو روز متوالی از پارکینگ استفاده کرده است و بنابراین باید اطلاعات او را خروجی داد.

کوئری 14 :

ابتدا تمام مسیر های مترو ممکن از مبدا داده شده به مقصد داده شده با استفاده از کوری بازگشتی محاسبه شده و اطلاعات مربوط به مسافت آن ها نیز در همین کوری محاسبه می شود. حال از بین تمام مسیرهایی که پیدا شدند کافی است min بگیریم تا کوتاه ترین مسیر را به ما بدهند.

کوئری 15 :

ابتدا با استفاده از کوری بازگشتی همانند مورد های قبل مسافت بین دو ایستگاهی که یک شهروند بین آن ها سفر کرده است را بدست می آوریم که برای این کار لازم است ایستگاه مبدا از جدول travel که حاوی ایستگاه های مبدا و مقصد است به آن کوری بازگشتی داده شود و در نهایت هم آن ایستگاه های نهایی ای قابل قبول اند که ایستگاه نهایی شهروندان می باشند. حال که تمام مسیرها بدست آمدند کافی است بر اساس کد ملی کاربر group by کنیم و با جمع distance هایی که طی کرده است چک می کنیم که آیا این جمع کمتر از مقدار داده شده هست یا خیر و اگر بود اطلاعات آن شهروند را خروجی می دهیم.

توضیح درباره چهار دید طراحی شده:

دید اول : در این دید ابتدا بین جدول driving و citizen جوین کرده ایم تا اطلاعات مربوط به تاریخ های رانندگی این شهروند را بدست بیاوریم سپس با استفاده از تابع date_trunc می آییم و در یک ماه گذشته تمام رانندگی ها را محاسبه می کنیم و سپس آن رانندگانی که برای اتوبوس هستند را نیز فیلتر کرده و سپس با group by بر اساس کد ملی شهروندان می توان با جمع distance هایی که پیموده اند مشخص کرد که آیا کمتر از مقدار مشخص شده است یا خیر.

دید دوم: در یک جدول جدا این حضور که نشان دهنده مبدا و مقصد است پیاده سازی شده است اما می توانستیم از همان جدول travel هم استفاده کنیم که حالا شاید در ادامه پروژه این قسمت را تغییر دادیم. در هر صورت در حال حاضر ابتدا می آییم ایستگاه های اتوبوسی که شهروند در آن بوده را با join زدن ایستگاه با مسیرش تا اینکه مشخص شود کدام یک مربوط به اتوبوس رانی است را انتخاب کرده و بازه زمانی الان تا 24 ساعت گذشته را انتخاب می کنیم و حال بر اساس شناسه ایستگاه group by می کنیم تا با count تعداد شهروندان در هر ایستگاه مورد خواسته شده بدست آورده شود.

دید سوم: در جدول سفر ها ابتدا مترو ها را فیلتر کرده و با گروه بندی بر اساس مترو ها و استفاده از دستور count distinct تعداد منحصر به فرد شهروندان را به ازای هر رام مترو بدست می آوریم.

دید چهارم: ابتدا جدول home که حاوی اطلاعات خانه ها می باشد را با جدول services_usage که جدول مربوط به استفاده از خدمات زیر ساخت شهری هستند join می کنیم و بر اساس برق فیلتر میکنیم و سپس مانند قبلی ها ابتدا با date_trunc مشخص می کنیم که مربوط به ماه قبل باشد و سپس بر اساس شناسه شهری خانه ها group by می کنیم و سپس محاسبه می کنیم که جمع میزان استفاده آن ها از حدی مشخص بیشتر بوده یا نه و سپس خروجی می دهیم.