به نام خدا

گزارش فاز دوم پروژه

نام استاد: **امینی**

اعضای تیم:
نیما میری ۴۰۰۱۰۵۲۶۳
سروش پورنصیری ۴۰۰۱۰۴۸۵۶
عرفان محمدی ۴۰۰۱۷۰۳۵۲

آذر ماه ۱۴۰۲

جزئیات دیدها:

دید شماره ۱:

این دید شماره ملی‌هایی را باز بر می‌گرداند که نامشان در لیست نام راننده‌های واسایل نقلیه عمومی است و هم‌چنین جمع find_distance هایشان که از مسیرهایی که رفته و در رسید حمل و نقل ثبت شده است، از مقدار مشخص بیشتر است. هم‌چنین چک می‌کند که این رفت و آمدها در ماه اخیر باشد (یعنی AGE زمان حال و زمان آن رفت و آمدها، قسمت ماه و سالشان صفر باشد)
ما فرض کردیم که منظور سوال از متوسط مسافت، مسافت روزانه است بنابراین این مسافت را تقسیم بر ۳۰ کردیم.

دید شماره ۲:

این دید بررسی می‌کند که AGE زمان صدور رسید تا حال کمتر از یک روز باشد (یعنی قسمت روز و ماه و سال آن صفر باشد) و هم‌چنین ایستگاه مذکور حتما در ابتدا یا انتهای مسیر باشد.

دید شماره ۳:

در این دید با استفاده از count و group by transport_id (آیدی ترنسپورت همان شماره شناره هر رام است) تعداد اشخاص را می‌شمارد. قسمت DISTINCT باعث می‌شود تنها افراد منحصر به فرد شمرده شوند. بنابراین این دید باعث می‌شود تعداد افراد منحصر به فردی که در هر رام تا کنون سفر کرده اند شمرده شوند.

دید شماره ۴:

همانند روش دیدهای قبلی، ما شرط ماه اخیر را گذاشتیم (قسمت ماه و سال را صفر کردیم).
سپس از sum(price) استفاده نموده تا جمع قیمت‌هایی که برای هر خانه محاسبه می‌شود را حساب کند. با الحاق جداول مربوطه، به قیمت رسیدهای خدمات مربوط به برق می‌رسیم و آن‌ها را جمع می‌زنیم و در نهایت آن خانه‌هایی که جمع رسیدهای برقشان از ۵۰۰ بالاتر بود را نشان می‌دهیم.

جزئیات پرسش‌ها:

۱. از طریق رسید حمل و نقل به مسافران و از طریق مسافران به موجودیت شهروند و از طریق موجودیت شهروند به جنسیت او دسترسی داریم.
هم‌چنین مسافران یک سفر رسیدهای متفاوتی دارند اما چیزی که آن‌ها مشترکا دارند ترکیب وسیله سفرشان و زمان سفرشان است. یعنی هر دو شخصی که وسیله سفرشان و زمان سفرشان یکسان باشند، هم‌سفر تلقی می‌شوند.
از طریق نام راننده نیز به وسیله حمل و نقل عمومی او می‌توانیم برسیم.
حال کافی است یک count روی رسیدهای حمل و نقلی بزنیم که وسیله‌اش، وسیله شخص خاصی باشد و زمان شروع یا پایانشان با یکدیگر یکسان باشد. در نتیجه لیست هم‌سفران را خواهیم داشت. حال بین این هم‌سفران از طریق موجودیت شهروند تعداد مسافران زن و مرد را بدست آورده و نسبت بین آن‌ها را بررسی می‌کنیم. در نهایت تعداد سفرهایی که شرط مذکور را داشتند خروجی داده می‌شوند.

۲. این پرسش قیمت تمام رسیدها را با هم جمع می‌زند. منتهی رسیدهایی که برای حساب‌هایی‌است که در موجودیت شهروندشان، قسمت breadwinner یا همان سرپرستشان یکسان است، یعنی در یک خانواده هستند و در یک بسته قرار می‌گیرند. بنابراین روی ستون سرپرست، یک group by زده می‌شود و در نهایت با استفاده از sub قیمت تمامی این رسیدها با یکدیگر جمع می‌شوند. هم‌چنین یک شرط برای بررسی بازه زمانی هر قراره داده شده است.

۳. با استفاده از تابع find_distance که جلوتر توضیح داده می‌شود، می‌تواند مسافت هر سفر را به راحتی محاسبه کرد. حال این پرسش تنها روی راننده‌ها group by می‌زند و مسافت سفرهایشان را با هم جمع می‌کند. به این صورت که وارد رسیدهای حمل و نقل شده و روی راننده وسایل نقلیه یک group by زده و شناسه مسیر نیز که در همان رسید حمل و نقل قرار دارد.

۴. برای تفکیک ماهانه صرفا کافی است روی ماه و سپس روی سال یک group by بزنیم. با استفاده از ۵ limit خروجی را تنهای به نمایش ۵ داده محدود می‌کنیم.
هم‌چنین شرط بین بازه قرار گرفتن سفر را بررسی کرده و نهایتا بررسی می‌کنیم که آن ایستگاه‌ها آیا در جدول موقعیت-در-مسیر، در ابتدا یا انتهای آن سفرها قرار گرفته‌اند یا خیر.

۵. برای این کار فاصله تمامی ایستگاه‌ها را تا نقطه مذکور بررسی می‌کنیم. برای این‌کار از رابطه فیثاغورت استفاده کرده و طول و عرض صرفا موقعیت مکانی وارد شده را منهای طول و عرض موقعیت مکانی ایستگاه‌ها کرده و به توان دو رسانده و نهایتا زیر رادیکال می‌بریم. نهایتا آن‌ها را به صورت نزولی مرتب کرده (با استفاده از order by) و سپس با استفاده از ۵ limit تنها پنج خروجی را نمایش می‌دهیم.

۶. برای این‌کار رسیدهای حمل و نقل را بررسی می‌کنیم و آن را با جدول مسافران پیوند می‌زنیم. به این صورت از طریق مسیر سفر به شماره شناسه شهروند دسترسی خواهیم داشت.
بنابراین در رسیدهای حمل و نقل، روی شماره شناره افراد group by زده و تعداد منحصر به فرد ایستگاه‌ها (با استفاده از distinct) را از طریق مسیرها به دست آورده و با یکدیگر جمع می‌زنیم.

۷. از روی جدول رسید سفرها، به وسیله نقلیه و در نتیجه نوع وسیله نقلیه دسترسی خواهیم داشت. حال زمان پایان سفرها را منهای زمان شروع سفرها کرده و مقادیر آن‌ها را در هر دسته مترو یا اتوبوس جمع می‌زنیم و با یکدیگر مقایسه می‌کنیم. در نهایت مسافرین که مقدار زمان‌های مترویشان از اتوبوسشان بیشتر بود را در خروجی می‌دهیم. (البته بازه را نیز در قسمت where بررسی می‌کنیم)

۸. قسمت رنگ به برند را می‌چسبانیم و آن را یکتا قرار می‌دهیم. سپس شرط پارکینگ مشخص و بازه زمانی مشخص را مانند پرسش‌های قبلی بررسی کرده و تعداد خودروها را با count می‌شماریم.

۱۰. برای قسمت تفکیک ماهانه از این دستور استفاده کردیم:

```
GROUP BY TO_CHAR(issue_time, 'YYYY-MM')
```

که در واقع تمامی هزینه‌ها را بر طبق سال و ماه دسته‌بندی می‌کند.
قیمت تمام رسیدهایی که برای اکانتی است که شماره شناسه‌اش برای شخص خاصی است را جمع زدیم و در نهایت با استفاده از order by آن‌ها را طبق سال و ماه مرتب کردیم.

۱۱. از آنجایی که تابع find_distance را قبلا زده‌بودیم، کافی است مسافت مسیرها را ضرب در قیمت هر کیلومترشان بکنیم تا قیمت هر سفر به دست آید. سپس بررسی می‌کنیم که قیمت این سفرها از حد مشخص بالاتر نباشد. حال باید بررسی کنیم که این سفر از یک ایستگاه مشخص قابل انجام است یا خیر. کافی است که در جدول موقعیت-در-مسیر بررسی کنیم که این ایستگاه در شروع این مسیر قرار دارد یا خیر.

۱۲. این پرسش بسیار واضح است. از تمام رسیدهای موجودی که نوع آن‌ها حمل و نقل است و صاحب حساب رسید آن در شماره شناسه‌اس در جدول دارندگان ماشین موجود می‌باشد قیمتش با استفاده از تابع AVG میانگین زدیم.

۱۳. شرط بازه زمانی را که مانند قبلی‌ها بررسی کردیم. سپس گفتیم که اگر رسید پارکینگی موجود باشد که زمان شروعش + ۱ (یعنی یک روز بعد از زمان شروعش) نیز در بین رسیدهای پارکینگ همان شخص وجود داشته‌باشد، یعنی آن شخص در دو روز متوالی از امکانات سرویس استفاده کرده است.

۱۴. بین دو ایستگاه با استفاده از یک تابع بازگشتی تمامی مسیر ها را بدست می آوریم. حال مسیر با کمترین مسافت را انتخاب می کنیم.
برای این که در لوپ بی نهایت نیوفتد، فرض می کنیم طول کوتاه ترین مسیر بین هر دو ایستگاه از یه مقدار ثابت کمتر می باشد.

۱۵. با پیوند جدوال رسید حمل و نقل و جدول مسافران، از طریق مسیر می‌توانیم به شماره شناسه شهروند برسیم. حال بین تمام رسیدهای حمل و نقلی که از نوع اتوبوس هستند، مسافت‌های مسیرهایشان که در رسید حمل و نقلشان هست را با استفاده از find_distance پیدا کرده و با استفاده از sum جمع می‌زنیم. حال بررسی می‌کنید که این جمع از مقدار مشخصی کمتر باشد. هم‌چنین طبق روال بررسی می‌کنیم که سفرها در بازه مشخصی باشند.

جزئیات توابع:

اکثر توابع در این پروژه مربوط به بخش رهاناهاست که در همان بخش رهانا کارکرد آن‌ها توضیح داده شده است.

تابع `find_distance()`:
این تابع با جستجو در یال‌های موجود در جدول station_sequence، یال‌های داخل یک مسیر را می‌یابد. سپس با یک حلقه (loop)، یکی یکی اندازه این یال‌ها را با یکدیگر جمع می‌کند.

جزئیات رهاناها:

نکته: در ابتدا باید بگوییم که بین رسید پارکینگ، خدمات و حمل و نقل، و رسید رابطه is-a وجود دارد. بنابراین ما به صورت دستی رسیدهای فرزند را ایجاد و تغییر می‌دهیم و رسید پدر به صورت خودکار توسط رهاناها ساخته می‌شود.

رهانای parking_receipt_trigger:

به هنگام ورود ماشین به پارکینگ، یک رسید پارکینگ بدون زمان خروج ساخته خواهد شد (به صورت دستی).

این رهانا به هنگام ساخت یک رسید پارکینگ، تابع create_parking_receipt را فرا می‌خواند. این تابع قبل از ساخت یک رسید پارکینگ، یک رسید پدر با شماره‌حساب رانندهٔ صاحب ماشین می‌سازد. هم‌چنین یکی از ظرفیت پارکینگ کم کرده و بررسی می‌کند که شروع و پایان رسید پارکینگ، از شروع و پایان ساعات کاری پارکینگ به ترتیب کمتر و بیشتر نباشد. هم‌چنین بررسی می‌کند که اگر ظرفیت پارکینگ صفر باشد، اجازه ساخت رسید ندهد.

رهانای finish_parking_trigger:

ب به هنگام خروج ماشین به پارکینگ، زمان خروج رسید پارکینگ مربوط به آن ماشین بروزرسانی خواهد شد (به صورت دستی).

این رهانا به هنگام بروزرسانی رسید پارکینگ، در صورت هیچ‌مقدار نبود زمان خروج رسید پارکینگ، اطلاعات رسید، نظیر زمان صدور (که برابر زمان خروج ماشین در رسید پارکینگ) و قیمت (که به صورت خودکار محاسبه می‌گردد) تکمیل می‌گردد.

رهاناهای تابع check_time_order:

۳ رهانا مرتبط با تابع check_time_order وجود دارد. کار این تابع این است که در صورت بیشتر بودن هر نوع زمان شروعی نسبت به زمان پایان، یک ارور بدهد و جلوی ساخت رسید را بگیرد. رهاناها نیز پیش از ساخت رسید، این تابع را فرا می‌خوانند تا این موضوع را بررسی کنند.

رهانای deduct_credit_trigger:

این رهانا تابع deduct_credit_after_receipt را فرا می‌خواند که باعث می‌شود قیمت یک رسید را از اعتبار حساب‌بانکی آن شخص کم می‌کند. این رهانا این تابع را بعد از ساخت رسید پدر فرا می‌خواند.

رهانای check_credit_parking و check_credit_transportation:

این رهانا بررسی می‌کنند که اگر اعتبار شخصی کمتر یا مساوی صفر بود، جلوی ساخت رسید را بگیرد. هر کدام در حوزه خود با فراخوانی تابع مربوطه، این کار را می‌کنند.

رهانای create_acc_trigger:

از آنجایی که هر شخص الزاما حساب بانکی دارد، این رهانا بلافاصله بعد از ساخته شدن یه سطر به جدول شهروند، برای اون حساب بانکی می‌سازد و اعتبار او را برابر صفر قرار می‌دهد.

رهانای service_receipt_trigger:

این رهانا با استفاده از مقدار هزینه آب، برق، گاز و هم‌چنین با ردیابی صاحب خانه یک رسید می‌سازد. هم‌چنین مصرف آب، برق، گاز را صفر قرار می‌دهد.

نکته: برای یک سفر ابتدا به صورت دستی رابطه رسید سفر با شهروند ساخته می‌شود. بعد رسید سفر متناظر تکمیل می‌شود و تریگر مابقی کارها را انجام می‌دهد.

رهانای `passengers_receipt_trigger`:

پیش از ساخته شدن یک سطر در قسمت مسافران (= رابطه بین رسید سفر و شهروند) یک رسید سفر و یک رسید با شماره ملی آن شهروند می‌سازد. (الزام این تریگر در این است که یک ستون از جدول مسافران، کلید خارجی دارد به رسید سفر و بنابراین باید ابتدا رسید سفر به صورت خودکار ساخته شود)

رهانای `transportation_receipt_trigger`:

این رهانا پس از تکمیل رسید سفر (یعنی به اتمام رسیدن سفر و مشخص شدن زمان پایان سفر) اجرا می‌شود و با فراخوانی تابع مربوطه مقدار صدور رسید را وارد جدول رسید (در سطر مربوط به آن) می‌کند. هم‌چنین در تابع مربوطه تابعی به نام `find_distance` وجود دارد که به محاسبه مسافت یک مسیر می‌پردازد. سپس مسافت بدست آمده را ضربدر مقدار هزینه هر کیلومتر مربوطه می‌کند تا هزینه سفر بدست آید.

رهانای `receipt_id_trigger`:

این رهانا بررسی می‌کند که اگر شماره رسید یک رسید مثلا خدمات یا پارکینگ با عدد خاصی شروع نشود، اجازه ساخت رسید را ندهد. بنابراین برای مثال تمام رسیدهای پارکینگ‌ها باید با عدد ۲ شروع شوند.