Uber_DS_Challenge

Guanshengrui Hao

3/26/2019

University - Data Science - Product Excecise

There are 3 sections to this exercise > SQL with 3 parts AND Experimental Design with 2 parts AND Modeling with 3 parts. Please complete all sections.

SQL

Part A

Q: You have a table populated with trip information (named uber_trip) table with a rider_id (unique per rider), trip_id (unique per trip), trip_timestamp_utc (the UTC timestamp for when the trip began), and trip_status, which can either be 'completed' or 'not completed'.

rider_id, trip_id, begintrip_timestamp_utc, trip_status

Write a query to return the trip_id for the 5th completed trip for each rider. If a rider has completed fewer than five trips, then don't include them in the results.

A:

```sql
SELECT temp.rider_id AS rider_id, temp.trip_id AS trip_id
FROM (
    SELECT ut2.rider_id AS rider_id, ut2.trip_id AS trip_id
    FROM uber_trip AS ut1,
         uber_trip AS ut2
    WHERE ut1.rider_id = ut2.rider_id
        AND ut1.trip_status = 'completed'
        AND ut2.trip_status = 'completed'
        AND ut1.begintrip_timestamp_utc < ut2.begintrip_timestamp_utc
) AS temp
GROUP BY trip_id
HAVING COUNT(rider_id) = 4;
```

Part B

Q: You are given three separate tables (named trip_initiated, trip_cancel, trip_complete) of the form:

trip_initiated | trip_id, rider_id, driver_id, timestamp

trip_cancel | trip_id, rider_id, driver_id, timestamp

trip_complete | trip_id, rider_id, driver_id, timestamp

Each trip_id in these tables will be unique and only appear once, and a trip will only ever result in a single cancel event or it will be completed. Write a query to create a single table with one row per trip event sequence (trip initiated → cancel/complete):

dispatch_events | trip_id, rider_id, driver_id, initiated_ts, cancel_ts, complete_ts

There should only be a single row per trip with a unique trip_id.

A:

```
CREATE TABLE dispatch_events AS
    SELECT tic.trip_id AS trip_id, tic.rider_id AS rider_id,
           tic.driver_id AS driver_id, tic.initiated_ts AS initiated_ts,
           tic.cancel_ts AS cancel_ts, tp.timestamp AS complete_ts
    FROM (
        SELECT ti.trip_id AS trip_id, ti.rider_id AS rider_id,
               ti.driver_id AS driver_id, ti.timestamp AS initiated_ts,
               tc.timestamp AS cancel_ts
        FROM trip_initiated AS ti
        LEFT JOIN trip_cancel AS tc
        ON ti.trip_id = tc.trip_id
    ) AS tic
    LEFT JOIN trip_complete AS tp
    ON tic.trip_id = tp.trip_id;
```

Part C

Q: Write at least one test query to validate the data in the table you created in Part B. Indicate what you would expect the query to return if the data were valid.

A: There are a few things we should test to validate. * Each *trip_id* should only appear once in the new table *dispatch_events*. The number of distinct *trip_id*'s (*num_distinct_trips*) and the total number of *trip_id*'s (*num_total_trips*) in the new table *dispatch_events* should be the same.

```
  SELECT COUNT(DISTINCT trip_id) AS num_distinct_trips, COUNT(trip_id) AS num_total_trips
  FROM dispatch_events;
```

- The *trip_id*'s in the new table *dispatch_events* should be identical to those in the table *trip_initiated*. This could be validated in two steps:
  - On one hand, the table *trip_initiated* should contain all *trip_id*'s in the new table *dispatch_events*. So *num_trips_initiated* and *num_common_trips* in the first chunk of code below should be the same.
  - On the other hand, the new table *dispatch_events* should also contain all *trip_id*'s in the table *trip_initiated*. o *num_trips_in_dispatch* and *num_common_trips* in the second chunk of code below should be the same.

```
SELECT COUNT(ti.trip_id) AS num_trips_initiated,
       SUM(CASE de.trip_id IS NULL
           WHEN true THEN 0
           ELSE 1
           END
        ) AS num_common_trips
FROM trip_initiated AS ti
LEFT JOIN dispatch_events AS de
ON ti.trip_id = de.trip_id;
```

```
SELECT COUNT(de.trip_id) AS num_trips_in_dispatch,
       SUM(CASE ti.trip_id IS NULL
           WHEN true THEN 0
           ELSE 1
           END
        ) AS num_common_trips
```

```
FROM dispatch_events AS de
LEFT JOIN trip_initiated AS ti
ON de.trip_id = ti.trip_id;
```

- blabla