Business Intelligence - Data Engineer Interview Task

Background:

MediaMath serves impressions, which are ads shown to users. In addition, MediaMath captures site visitor data points when users fire a pixel from the particular website. A conversion occurs when a user is served an impression and then continues to take a consumer action which is considered as an event (a form submission, purchase, etc). MediaMath stores this log level information (impressions, events, conversions) in a database for analysis.

Three tables available with log level data (schemas shown below):

* Impressions: a row is written to this table each time a user is served an ad.
* Conversions: a row is written every time a user who was served an impression has an attributed conversion (you can assume that a user can only purchase one time)
* Events: a row is written to the table each time a user has an onsite activity.

\*Note: the events table is not tied to media, this is purely on-site browsing activity.

**NOTE : I've used BigQuery SQL to solve the below tests**

1. Please write a SQL query to solve the following questions:

* How many users were served an impression per day during September?

SELECT DATE(timestamp) AS date, COUNT(DISTINCT(user\_id)) AS impression\_count

FROM impressions

WHERE timestamp >= '2023-09-01 00:00:00' AND timestamp < '2023-10-01 00:00:00'

GROUP BY DATE(timestamp);

*The query extracts date from the timestamp using date() function and groups the results by date, It then counts the distict number of user IDs as impression\_count to determine the number of users served as impression each day during september.*

* How many users were served an impression and fired pixel ID 101293 in September?

SELECT COUNT(DISTINCT impressions.user\_id) AS impression\_count

FROM impressions

INNER JOIN events

ON impressions.user\_id = events.user\_id

WHERE impressions.timestamp >= '2023-09-01 00:00:00' AND impressions.timestamp < '2023-09-01 00:00:00' AND events.pixel\_id = 101293;

*This query performs an inner join between the 'impressions' and 'events' tables based on 'user\_id' column. It filters the data to include only records with a timestamp in September and where the 'pixel\_id' in the 'events' table matches the value 101293. Finally, it counts the distinct number of 'user\_id's from the 'impressions' table that meets these conditions, giving the desired count of users served an impression and fired 'pixel\_id' 101293 in September.*

* How many attributed converters fired pixel ID 101293 and how many attributed conversions did not fire pixel ID 101293 before conversion (please write this in one query)?

SELECT COUNT(DISTINCT events.user\_id) AS attributed\_converters\_fired\_pixel, COUNT(DISTINCT conversions.user\_id) AS attributed\_conversiona\_no\_pixel

FROM events

JOIN conversions

ON events.user\_id = conversions.user\_id

LEFT JOIN impressions

ON conversions.user\_id = impressions.user\_id

AND impressions.timestamp < conversions.conversion\_timestamp

WHERE events.pixel\_id = 101293

AND impressions.user\_id IS NULL;

*The query uses a LEFT JOIN to connect the 'impressions' table with the 'conversions' table based on the 'user\_id', ensuring that we consider only conversions that occured after any related impressions. Then, it filters for the 'pixel\_id' 101293 in the 'events' table to count attrributed converters that fired the pixel. Finally, it includes a condition in the WHERE clause to select only those conversions that didn't have any matching impression (i.e, attributed conversions that did not fire pixel ID 101293 before conversion.*

2. Using the log tables and knowing that we have this data available:

* Write SQL that would output a frequency distribution for all converters before conversion.

SELECT conversions.pixel\_id, COUNT(\*) AS frequency

FROM conversions

LEFT JOIN impressions

ON conversions.user\_id = impressions.user\_id

AND impressions.timestamp < conversions.conversion\_timestamp

WHERE impressions.user\_id IS NULL

GROUP BY conversions.pixel\_id;

*The query retrieves the 'pixel\_id' column from the 'conversions' table and performs a LEFT JOIN with the 'impressions' table. It matches the 'user\_id' columns and checks if the impression timestamp is before the conversion timestamp. The condition 'impressions.user\_id IS NULL' filters for converters that did not have any matching impresion before conversion. Finally, the query uses the GROUP BY clause to group the conversions by 'pixel\_id', and the COUNT(\*) function calculated the frequency of each 'pixel\_id' giving the frequenct distribution, including the number of converters associated with each 'pixel\_id' before conversion.*

* Please write out how you would determine what the optimal frequency is and if there is a point that we should cap the frequency [You can create your own sample output to demonstrate]

\*frequency = the number of impressions served to a user

*- Determine what the optimal frequency is and if there is a point that we should cap the frequency?*

*To determine the optimal frequency and identify a point at which we should cap the frequency, we can analyse the distribution of impressions served to users. Here's a simple example:*

SELECT user\_id, COUNT(\*) AS frequency

FROM impressions

GROUP BY user\_id;

*The above calculates the frequency of impressions for each user by grouping the data by the 'user\_id' column.*

*We can examine statistical measures such as mean, median, mode and standard deviation to understand the central tendency and spread of the distribution. Additionally, we can plot a histogram or box plot to visualiase the frequency distribution graphically.*

*- Determine the optimal frequency*

*The optimal frequency will depend on various factors such as the specific campaign goals, target audience, and advertising strategy for example, wheather the campaign aims for high exposure or more controlled impressions or evaluate the potential impact on user experience and ad fatigue or monitor performance such as CTR and conversion rates at different frequency levels and also take associated costs into consideration. example we may learn that a moderate frequency range of 3-5 impressions per user yields better performance compared to lower or higher frequency levels.*

*If however, at any given point the frequency becomes excessive and negatively impacts campaign performance or user experience, we should consider capping the frequency and set a limit on the number of impressions served per user.*

Table Schemas:

|  |  |  |
| --- | --- | --- |
| IMPRESSIONS |  |  |
| **column\_name** | **column\_description** | **example\_field** |
| user\_id | unique user ID | ABC |
| auction\_id | auction ID that uniquely identifies an impression | 293842029 |
| advertiser\_id | unique ID associated with client name | 12345 |
| advertiser\_name | client name | Client A |
| total\_spend\_cpm | total spend on individual impression in CPM format (CPM = spend \* 1,000) | 5.92 |
| country | country where impression is served | United States |
| DMA | DMA where impression is served | New York |
| device | device of impression served | Desktop/Laptop |
| timestamp | timestamp of the impression | 9/10/18 9:08 |
|  |  |  |
| EVENTS |  |  |
| **column\_name** | **column\_description** | **example\_field** |
| user\_id | unique user ID | ABC |
| pixel\_id | unique ID for event pixel | 283928 |
| pixel\_name | event pixel name | Homepage |
| revenue | revenue associate with pixel (only populated if merit pixel) | 100 |
| country | country where pixel is fired | United States |
| DMA | DMA where pixel is fired | New York |
| device | device of pixel fire | Desktop/Laptop |
| timestamp | timestamp of pixel fire | 9/1/18 0:11 |
|  |  |  |
| CONVERSIONS |  |  |
| **column\_name** | **column\_description** | **example\_field** |
| user\_id | unique user ID | ABC |
| pixel\_id | unique ID for conversionn pixel | 283928 |
| pixel\_name | merit pixel name | Purchase |
| imp\_auction\_id | impression auction ID for attributed impression | 293842029 |
| conversion\_timestamp | timestamp for conversion | 9/15/18 10:09 |
| impression\_timestamp | timestamp for attributed impression | 9/10/18 9:08 |