Candidate Name: **Mohamed Yasir M**

Contact No.: **4192695724**

Mail ID: [mohamedyasirmm@gmail.com](mailto:mohamedyasirmm@gmail.com)

Given the following tables:

(Please refer appendix for complete code)

**experiments**

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id | assignment\_ts | experiment\_id | experiment\_assignment |
| 1 | 20190106 12:00:05 | 1 | test |
| 1 | 20190304 01:45:23 | 2 | test |
| 2 | 20190107 03:45:12 | 1 | control |
| 3 | 20190311 01:45:23 | 2 | test |

And

**subscriptions**

|  |  |  |
| --- | --- | --- |
| user\_id | subscription\_event | event\_ts |
| 1 | trial\_start | 20190106 12:00:04 |
| 1 | subscription\_start | 20190115 10:03:05 |
| 2 | trial\_start | 20190107 03:45:10 |
| 3 | trial\_start | 20190311 01:45:20 |

Where

Experiments is a table in which we store whether a user is part of an experiment and if so whether they are in test or control (assume there is only one test variant per experiment). The fields are:

* user\_id - There are many users each of whom can be in many experiments
* assignment\_ts - timestamp of when the user was allocated to the experiment. A user is only allocated once per experiment.
* experiment\_id - An experiment has many users
* experiment\_assignment - Whether the user is in test or control. Assignments are immutable and there is only one assignment per user/experiment combo.

Subscriptions is a table of subscription related events. For each user, there will always be a trial start event however there will only be a subscription start event if the user subscribes. Assume a given user can only have one trial start and at most one subscription start. The subscription can start at any time after the trial start and times for either event type are captured in event\_ts.

**Questions**

Write queries to produce the following:

1. When did each experiment start? Use the first instance of an experiment assignment to either test or control for an experiment to equate to when the experiment started. Results should look like:

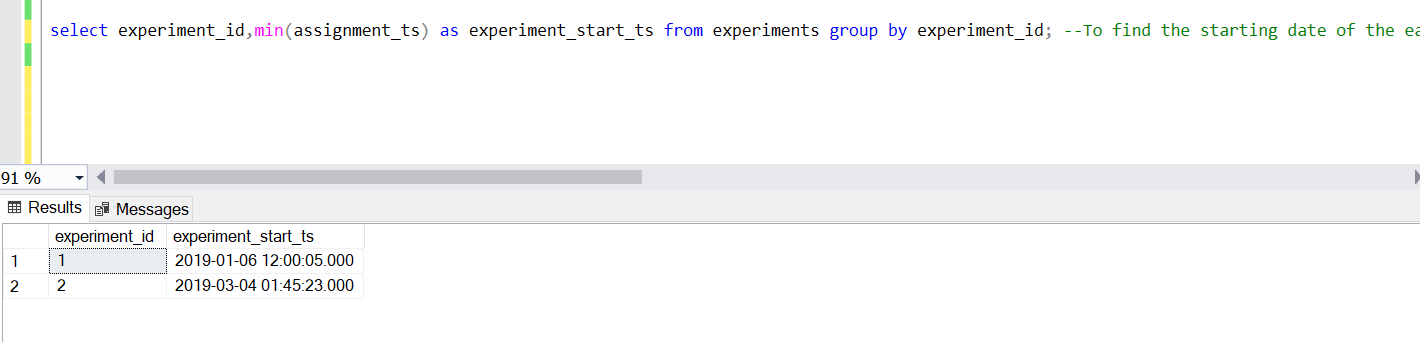
|  |  |
| --- | --- |
| experiment\_id | experiment\_start\_ts |
| 1 | 20190106 12:00:05 |
| 2 | 20190304 01:45:23 |

(and so on)

**Code:**

select experiment\_id,min(assignment\_ts) as experiment\_start\_ts from experiments group by experiment\_id;

**Result:**



1. How long did each experiment last, expressed in days? Assume the last instance of an experiment assignment to test or control for an experiment to equate to when the experiment ended. Results should look like:

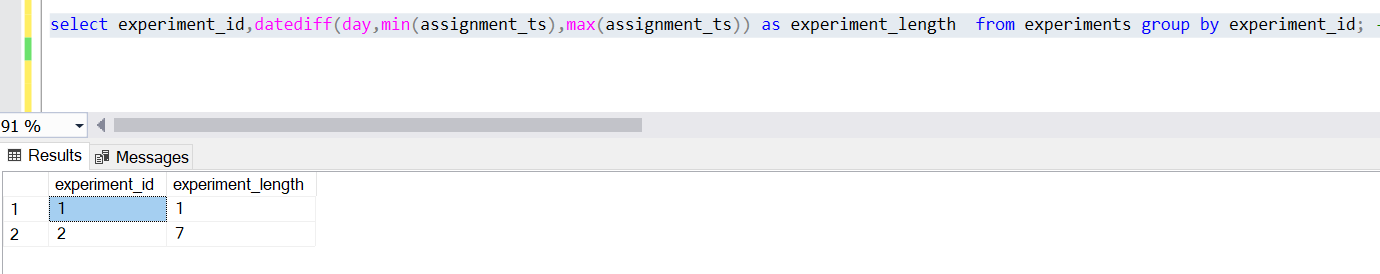
|  |  |
| --- | --- |
| experiment\_id | experiment\_length\_days |
| 1 | 2 |
| 2 | 7 |

(and so on)

**Code:**

select experiment\_id,datediff(day,min(assignment\_ts),max(assignment\_ts)) as experiment\_length from experiments group by experiment\_id;

**Result:**



Note: (24 hrs time-format, considered 1 day for experiment 1)

1. How many users are in test and control for each experiment? Result should look like:

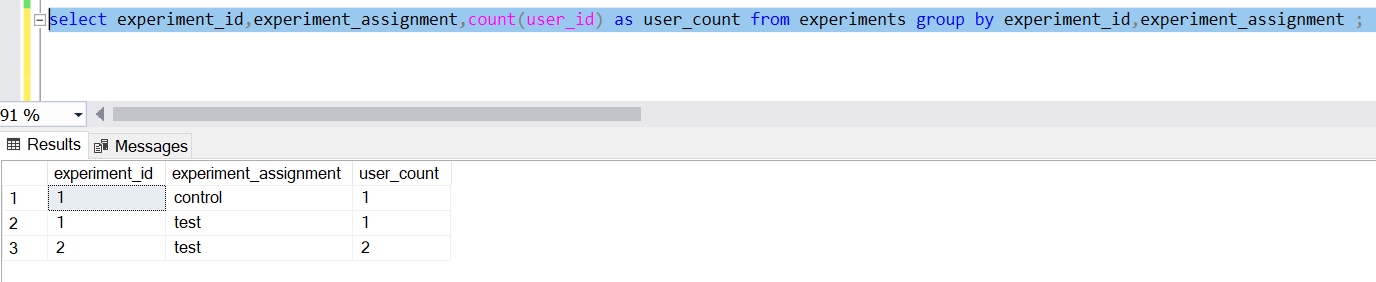
|  |  |  |
| --- | --- | --- |
| experiment\_id | experiment\_assignment | user\_count |
| 1 | test | 1 |
| 1 | control | 1 |
| 2 | test | 2 |

(and so on)

**Code:**

select experiment\_id,experiment\_assignment,count(user\_id) as user\_count from experiments group by experiment\_id,experiment\_assignment ;

**Result:**



1. What is the conversion rate by experiment assignment for each experiment? A conversion is any user for whom there is a subscription start event in addition to the trial start event (all users have a trial start event). If a user is in multiple experiments at the same time, it’s ok to count them towards the conversion rate of each experiment. We also want to only return one row per experiment. Result should look like:

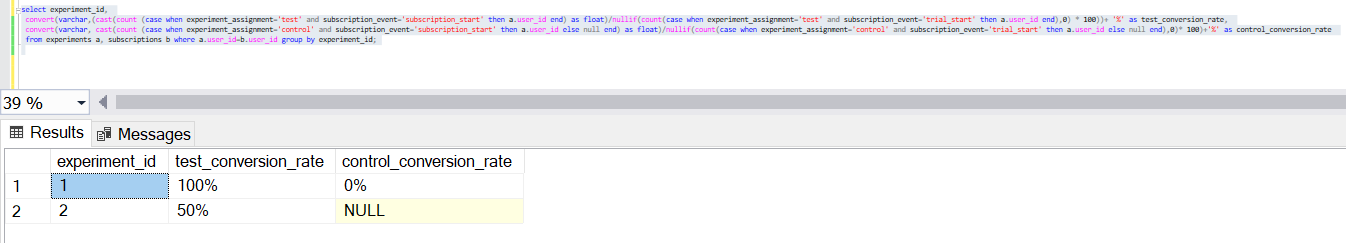
|  |  |  |
| --- | --- | --- |
| experiment\_id | test\_conversion\_rate | control\_conversion\_rate |
| 1 | 100% | 0% |
| 2 | 50% | (null) |

(and so on)

**Code:**

select experiment\_id,convert(varchar,(cast(count (case when experiment\_assignment='test' and subscription\_event='subscription\_start' then a.user\_id end) as float)/nullif(count(case when experiment\_assignment='test' and subscription\_event='trial\_start' then a.user\_id end),0) \* 100))+ '%' as test\_conversion\_rate,convert(varchar, cast(count (case when experiment\_assignment='control' and subscription\_event='subscription\_start' then a.user\_id else null end) as float)/nullif(count(case when experiment\_assignment='control' and subscription\_event='trial\_start' then a.user\_id else null end),0)\* 100)+'%' as control\_conversion\_rate from experiments a, subscriptions b where a.user\_id=b.user\_id group by experiment\_id;

**Result:**



5. For each experiment\_id, rank and list first 3 user\_ids who subscribed to the product. Output should look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| experiment\_id | Rank | user\_id | experiment\_assignment | assignment\_ts |
| 1 | 1 | 1 | test | 20190106 12:00:05 |
| 1 | 2 | 4 | control | 20190203 10:27:15 |
| 1 | 3 | 5 | test | 20190212 11:25:45 |
| 2 | 1 | 6 | test | 20190314 01:43:23 |

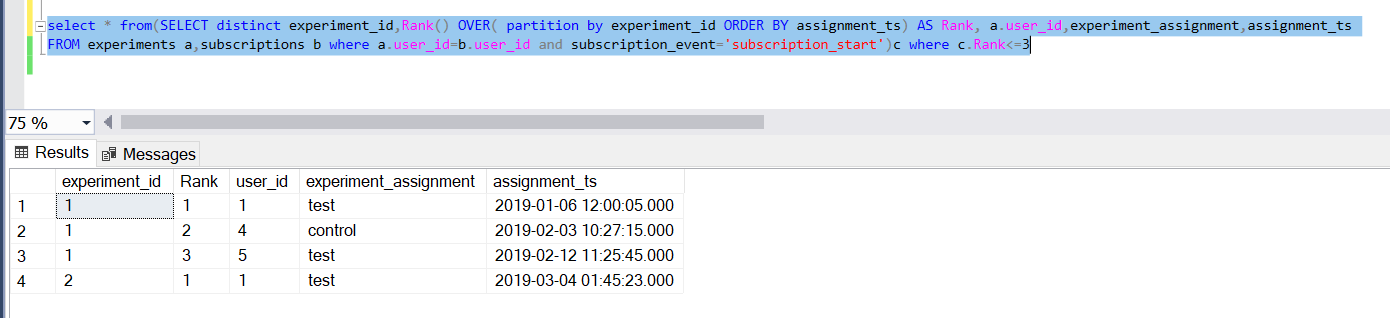
(and so on)

**Code:**

select \* from(SELECT distinct experiment\_id,Rank() OVER( partition by experiment\_id ORDER BY assignment\_ts) AS Rank, a.user\_id,experiment\_assignment,assignment\_ts

FROM experiments a,subscriptions b where a.user\_id=b.user\_id and subscription\_event='subscription\_start')c where c.Rank<=3

**Result:**



APPENDIX

Complete Code:

drop table if exists experiments;

drop table if exists subscriptions;

--Create table experiments to store users participate in experiment

create table experiments (

user\_id int not null,

assignment\_ts datetime default current\_timestamp,

experiment\_id int,

experiment\_assignment varchar(255)

);

ALTER TABLE experiments

ADD CONSTRAINT exps1 unique(user\_id,experiment\_id);

ALTER TABLE experiments

ADD CONSTRAINT exps check(experiment\_assignment='test' or experiment\_assignment='control'); --restrict the column to accept values test or control

insert into experiments

values(1,'20190106 12:00:05',1,'test');

insert into experiments

values(1,'20190304 01:45:23',2,'test');

insert into experiments

values(2,'20190107 03:45:12',1,'control');

insert into experiments

values(3,'20190311 01:45:23',2,'test');

--insert into experiments

--values(4,'20190203 10:27:15',1,'control');

--insert into experiments

--values(5,'20190212 11:25:45',1,'test');

--Create table subscriptions to record trial and subscription events

CREATE TABLE subscriptions(

user\_id int not null,

subscription\_event varchar(255) DEFAULT 'trial\_start' not null,

event\_ts datetime default current\_timestamp,

);

--Create function to check the subscription\_start date is greater than the trial\_start date

DROP FUNCTION IF EXISTS get\_date

go

create function dbo.get\_date(@user int,@event varchar(255),@dat datetime)

returns int as

begin

declare @ret int;

set @ret=1;

if(@event='subscription\_start')

if(SELECT event\_ts FROM subscriptions WHERE user\_id = @user and subscription\_event='trial\_start')<@dat

set @ret=1;

else

set @ret=0;

return @ret;

end;

ALTER TABLE subscriptions

ADD CONSTRAINT subs2 check(subscription\_event='trial\_start'or subscription\_event='subscription\_start');

ALTER TABLE subscriptions

ADD CONSTRAINT sub3 unique(user\_id,subscription\_event); --restricting user\_id and subscription\_event columns to be unique

ALTER TABLE subscriptions

ADD CONSTRAINT subs1 check(dbo.get\_date(user\_id,subscription\_event,event\_ts)=1); --call get\_date function to check subscription\_start date

insert into subscriptions

values(1,'trial\_start','20190106 12:00:04');

insert into subscriptions

values(1,'subscription\_start','20190115 10:03:05');

insert into subscriptions

values(2,'trial\_start','20190107 03:45:10');

insert into subscriptions

values(3,'trial\_start','20190311 01:45:20');

insert into subscriptions

--values(4,'trial\_start','20190204 08:25:20');

--insert into subscriptions

--values(4,'subscription\_start','20190211 10:30:20');

--insert into subscriptions

--values(5,'trial\_start','20190214 13:30:20');

--insert into subscriptions

--values(5,'subscription\_start','20190215 09:30:20');

select experiment\_id,min(assignment\_ts) as experiment\_start\_ts from experiments group by experiment\_id; --To find the starting date of the each experiments

select experiment\_id,datediff(day,min(assignment\_ts),max(assignment\_ts)) as experiment\_length from experiments group by experiment\_id; --To find how long each experiment lasted

select experiment\_id,experiment\_assignment,count(user\_id) as user\_count from experiments group by experiment\_id,experiment\_assignment ; --To calculate no. of users in each experiment events

--To calculate the subscription conversion rate of customers from each experiment event

select experiment\_id,

convert(varchar,(cast(count (case when experiment\_assignment='test' and subscription\_event='subscription\_start' then a.user\_id end) as float)/nullif(count(case when experiment\_assignment='test' and subscription\_event='trial\_start' then a.user\_id end),0) \* 100))+ '%' as test\_conversion\_rate,

convert(varchar, cast(count (case when experiment\_assignment='control' and subscription\_event='subscription\_start' then a.user\_id else null end) as float)/nullif(count(case when experiment\_assignment='control' and subscription\_event='trial\_start' then a.user\_id else null end),0)\* 100)+'%' as control\_conversion\_rate

from experiments a, subscriptions b where a.user\_id=b.user\_id group by experiment\_id;

--Ranking top 3 users who subscribed

select \* from(SELECT distinct experiment\_id,Rank() OVER( partition by experiment\_id ORDER BY assignment\_ts) AS Rank, a.user\_id,experiment\_assignment,assignment\_ts

FROM experiments a,subscriptions b where a.user\_id=b.user\_id and subscription\_event='subscription\_start')c where c.Rank<=3

\*\*\*\*\*