▫Data Test Directions

Part I

Background: You have been provided a small dataset of impressions and activities for an online advertising campaign. Each row contains the User ID, Event Type and the associated Timestamp. Event Type is either an impression or an activity. Impression indicates that the user was shown an ad for the campaign. Activity indicates that the user performed an activity on the advertiser's website. The timestamp is provided in unix format.

In addition, you have been given a dataset containing user information. In this simple example the user data contains a column for the user id and a region code.

Lastly you will be asked to create a table with the following region information:

1000	Northeast
2000	Northwest
3000	Southeast
4000	Southwest
5000	Middle of Nowhere

1. Please write sql statements to create table for these datasets. Postgresql is preferred, however if you'd like to use a different RDBMS please identify the database you are using, and do not use database specific functionality that is not present in Postgres. Two requirements for the tables are that a timestamp data type should be used for readability for the impression and activities, and that duplicate User ID/Event Type/Timestamp are not allowed.

```
DROP TABLE IF EXISTS EventTypes, RegionTypes, Users, Events;

CREATE TABLE EventTypes
(
    EventTypeID     serial PRIMARY KEY,
    EventTypeDesc   varchar(64)
);

CREATE TABLE RegionTypes
(
    RegionID        serial    PRIMARY KEY,
    RegionDesc      varchar(64)
);
```

```
CREATE TABLE Users
(
    UserID          char(25) PRIMARY KEY,
    RegionID        integer,
    FOREIGN KEY(RegionID) REFERENCES RegionTypes(RegionID)
);

CREATE TABLE Events
(
    EventID         serial PRIMARY KEY,
    UserID          char(25),
    EventTypeID     integer,
    EventTime       timestamp without time zone,
    UNIQUE (UserID, EventTypeID, EventTime),
    FOREIGN KEY(UserID) REFERENCES Users(UserID),
    FOREIGN KEY(EventTypeID) REFERENCES EventTypes(EventTypeID)
);

INSERT INTO EventTypes(EventTypeDesc) VALUES('impression');
INSERT INTO EventTypes(EventTypeDesc) VALUES('activity');

INSERT INTO RegionTypes(RegionID, RegionDesc) VALUES(1000, 'Northeast');
INSERT INTO RegionTypes(RegionID, RegionDesc) VALUES(2000, 'Northwest');
INSERT INTO RegionTypes(RegionID, RegionDesc) VALUES(3000, 'Southeast');
INSERT INTO RegionTypes(RegionID, RegionDesc) VALUES(4000, 'Southwest');
INSERT INTO RegionTypes(RegionID, RegionDesc) VALUES(5000, 'Middle of Nowhere');
```

2. Show the code you use to load each of the tables you've created. The data should be cleansed of invalid data and duplicates.

   See the file CodingTest.java which is included in the appendix. In particular, the tables are loaded in the constructor.

3. Write both a sql query and java code that would count the number of users that were shown an advertisement a specific number of times. (i.e. event_type=impression).

```
SELECT NUM_IMPRESSIONS, COUNT(UserID) AS NUM_USERS
FROM
(
        SELECT UserID, COUNT(*) AS NUM_IMPRESSIONS
```

```
            FROM Events
            WHERE EventTypeID = (SELECT EventTypeID FROM EventTypes WHERE EventTypeDesc =
'impression')
            GROUP BY UserID
) AS SUB_QUERY
GROUP BY NUM_IMPRESSIONS
ORDER BY NUM_IMPRESSIONS
```

Number of ads they saw	**Number of users who saw that number of ads**
1	19507
2	5329
3	1052
4	682
5	282
6	227
7	126
8	90
9	63
10	63
11	40
12	28
13	18
14	19
15	16
16	9
17	12
18	13
19	9

20	11
21	4
22	2
23	1
24	3
25	6
26	3
27	4
28	2
29	2
30	3
31	5
33	4
34	2
35	2
36	1
37	1
39	1
41	1
42	1
43	1
44	1
45	1
46	1

47	1
48	1
50	1
54	1
56	1
63	1
64	1
65	1
68	1
79	1

I understood the question (based on the example output) to not want the number of users who saw zero ads, although this would have been relatively straightforward to determine.

4. Use SQL to produce a table of the number of times a user was presented an impression, with the regions as column headings and the number of impressions shown as the row title.

```
SELECT
        NUM_IMPRESSIONS AS Impressions,
        SUM(NE) AS Northeast,
        SUM(NW) AS Northwest,
        SUM(SE) AS Southeast,
        SUM(SW) AS Southwest,
        SUM(NOWHERE) AS Middle_of_Nowhere
INTO
        Impressions_By_Region
FROM
        (
                SELECT
                        NUM_IMPRESSIONS,
                        CASE WHEN RegionID=1000 THEN NUM_USERS ELSE 0 END AS NE,
                        CASE WHEN RegionID=2000 THEN NUM_USERS ELSE 0 END AS NW,
                        CASE WHEN RegionID=3000 THEN NUM_USERS ELSE 0 END AS SE,
                        CASE WHEN RegionID=4000 THEN NUM_USERS ELSE 0 END AS SW,
                        CASE WHEN RegionID=5000 THEN NUM_USERS ELSE 0 END AS
NOWHERE
```

```
            FROM
                (
                    SELECT
                        NUM_IMPRESSIONS,
                        RegionID,
                        Count(UserID) AS NUM_USERS
                    FROM
                        (
                            SELECT
                                Events.UserID,
                                RegionID,
                                COUNT(*) AS NUM_IMPRESSIONS
                            FROM
                                Events INNER JOIN Users ON
    Events.UserID = Users.UserID
                            WHERE
                                EventTypeID = (SELECT EventTypeID
    FROM EventTypes WHERE EventTypeDesc = 'impression')
                                AND
                                RegionID IS NOT NULL
                            GROUP BY
                                Events.UserID, RegionID
                        ) AS SUBQUERY_1
                    GROUP BY
                        RegionID,
                        NUM_IMPRESSIONS
                ) AS SUBQUERY_2
        ) AS SUBQUERY_3
    GROUP BY
        Impressions
    ORDER BY
    Impressions;

SELECT * FROM Impressions_By_Region;
```

Number of ads they saw	**Northeast**	**Northwest**	**Southeast**	**Southwest**	**Middle of Nowhere**
1	3895	3873	3898	3915	0
2	1463	1061	675	233	0
3	316	141	70	5	0
4	185	75	23	5	0

5	79	22	3	0	0
6	37	16	0	0	0
7	25	3	0	0	0
8	14	1	0	0	0
9	6	0	0	0	0
10	11	0	1	0	0
11	4	0	0	0	0
12	2	0	0	0	0
13	1	0	0	0	0
14	1	0	0	0	0
16	1	0	0	0	0

Obviously a significant number of impressions are associated with users that do not have a region specified. It was my understanding based on the example output that the above table should not include a "No Region" column.

Also, given sufficient time and using the crosstab function, we could probably have generated the column headers dynamically based on the various regions in the RegionTypes table.

5. Activities are only counted for the campaign if a user performs an activity after seeing an impression. Write sql query and java code to count the number of attributed activities. An activity should only be counted once. For example if a user sees two impressions before performing a single activity, then there is only 1 attributed activity for the user.

   Please provide your output as well as the code/query statement.

I understood an "attributed" activity to an activity that directly follows an impression (not another activity) for any given user.

```
SELECT
        COUNT(*) AS NUM_ATTRIBUTED_ACTIVITIES
FROM
        (
                SELECT
                        *,
                        (SELECT EventTypeID FROM Events WHERE UserID = E1.UserID AND
EventTime = E1.LAST_EVENT_TIME) AS LAST_EVENT_TYPE
                FROM
                        (
                                SELECT
                                        UserID,
                                        ACTIVITY_TIME,
                                        LAST_EVENT_TIME
                                FROM
                                        (
                                                SELECT
                                                        UserID,
                                                        EventTime AS ACTIVITY_TIME,
                                                        (SELECT MAX(EventTime) FROM Events
WHERE EventTime < E1.EventTime AND UserID = E1.UserID) AS LAST_EVENT_TIME
                                                FROM
                                                        Events AS E1
                                                WHERE
                                                        E1.EventTypeID = (SELECT EventTypeID
FROM EventTypes WHERE EventTypeDesc = 'activity')
                                        ) AS SUB_QUERY
                                WHERE
                                        LAST_EVENT_TIME IS NOT NULL
                        ) AS E1
        ) AS E2
WHERE
        LAST_EVENT_TYPE = (SELECT EventTypeID FROM EventTypes WHERE EventTypeDesc =
'impression')
```

See the file CodingTest.java included in the appendix. In particular, I called the method GetNumAttributedActivities to determine the number of attributed activities.

Number of Attributed Activities = 16

6. Attributed impression is an impression that is shown any time prior to an activity. For example, if a user saw 5 impressions prior to performing an activity, there would be 5 attributed impressions. If a user has multiple activities after seeing an impression, then the impression should only be counted once (i.e. I,I,I,A,A where I=impression and A=action would result in 3 attributed impressions and not 6). Write sql query and java code to count the number of attributed impressions in this dataset. Please provide your output as well as the code/query statement.

```
SELECT
        COUNT(*)
FROM
        (
                SELECT
                        CASE WHEN EXISTS (SELECT EventTime FROM Events WHERE EventTime
> E1.EventTime AND UserID = E1.UserID AND EventTypeID = (SELECT EventTypeID FROM
EventTypes WHERE EventTypeDesc = 'activity')) THEN 1 ELSE 0 END AS IS_ATTRIBUTED
                FROM
                        Events AS E1
                WHERE
                        E1.EventTypeID = (SELECT EventTypeID FROM EventTypes WHERE
EventTypeDesc = 'impression')
        ) AS SUB_QUERY
WHERE
        IS_ATTRIBUTED = 1
```

See the file CodingTest.java. In particular, I called the method GetNumAttributedImpressions to determine the number of attributed impressions.

Number of Attributed Impressions = 39

7. How would you go about tuning the statements you have provided for better performance? You can assume the data sets will grow much larger.

SQL:

This is a very open-ended question, so there are obviously a myriad of ways we could improve the performance of the SQL. However, here is what immediately comes to mind:

1) The SQL statements for questions 5 and 6 are reading the EventTime, UserID, and EventTypeID quite a bit. We should consider indexing the Events table by these fields and combinations thereof.

2) If we are performing certain select statements often in relation to how often we need to update the database, then we could periodically create tables that correspond to particular queries. For example, if we are very often retrieving the data in the table we created in question 4, then it would probably be a good idea to create/update the table we created in question 4 periodically.

3) We could consider creating a table similar to the table we created in question 4. Instead of updating it periodically, we could use database triggers to keep the data constantly up to date. This would clearly give us better select statement performance when querying the data shown in this table. However, we would need to weigh that against the overhead of such database triggers.

4) We could add columns to the Events table that would simplify our select statements. For example, we could add a "DIRECTLY_PREVIOUS_IMPRESSION" column to the Events table that stores the primary key of a row representing an impression that directly precedes an activity for a particular user. Clearly this would simplify the select statement for question 5.

Java:

1) I stored the data in 2 different HashMaps which each provided fast access to the Events for each of the users. The only reason I used 2 different HashMaps is because one was easier to use for questions 1-3 and the other was easier to use for questions 5-6. If this were production code we'd want to settle on one HashMap implementation to save on memory.

2) As the dataset grows large it will at some point become unreasonable to store the entire dataset for every user in memory on one machine. Fortunately this problem lends itself very well to using a Hadoop cluster (or similar MapReduce framework) to perform many of these operations. One reason it strikes me that many of these problems would be well suited for a Hadoop cluster is that it strikes me as unlikely that the data processing that we need to do for one particular user will become too large to process on a single machine. Thus, we could look into only storing the data in memory (of a machine in a Hadoop cluster) related to a particular user and using the cluster to aggregate our output.

Part II

Background: A business has asked you to design a database for them that will provide phone numbers to a widget that the UI team is developing. This widget will be used to provide listings for things like an employee phone directory, a customer contact list, and the contact numbers for various departments in the corporate web site. The widget will always provide at a minimum the name (of the organization or person), purpose of phone number (fax, cell, work, home, day, evening, etc), and the phone number itself.

The purpose of phone number may be different based on the type of listings being requested, and new purposes may be added at any time. For example, a listing may ask for 'Office', 'Fax. and 'Cell' for an employee, while a personal listing may list 'Day' and 'Evening', and 'Cell'.

In addition to the basic fields of name, purpose, and number, the widget will also return specific descriptive information about the listing depending the type of listing requested. For organizations (e.g. the corporate customer service department) it will also list the geographic region (e.g. Northeast) and the employee who is the contact for that organization. For an employee listing it will also list their business title. For a personal listing it will also list an occupation.

Provide the DDL and supporting documentation as you see fit (e.g. ERD's or Physical Data Models) for your design.

I don't have a nice tool to build an ER diagram handy. However, I think the following description combined with the DDL found below should make the data model self-explanatory.

We have 3 tables that store entities: PhoneNumbers, Persons, and Organizations. We have 3 tables that store many-to-many relationships between said entities: PersonsToPhoneNumbers, PersonsToOrganizations , and OrganizationsToPhoneNumbers. A person's business title is stored in the PersonsToOrganizations table so that a person can have multiple titles in case they are involved with multiple Organizations.

```
CREATE TABLE Purposes
(
        PurposeID       serial PRIMARY KEY,
        PurposeDesc     varchar(64)
);

CREATE TABLE Regions
(
        RegionID        serial PRIMARY KEY,
        RegionDesc      varchar(64)
);

CREATE TABLE PhoneNumbers
(
        PhoneNumberID           serial PRIMARY KEY,
        PhoneNumber     varchar(32) NOT NULL,
        PurposeID       integer,
        UNIQUE(PhoneNumber),
        FOREIGN KEY(PurposeID) REFERENCES Purposes(PurposeID)
);

CREATE TABLE Persons
(
        PersonID        serial PRIMARY KEY,
        FirstName       varchar(64),
	LastName	varchar(64),
	Occupation	varchar(64)
);

CREATE TABLE Organizations
(
	OrganizationID		serial PRIMARY KEY,
	OrganizationName	varchar(128) NOT NULL,
	RegionID		integer,
	ContactEmployee			integer,
	FOREIGN KEY(RegionID) REFERENCES Regions(RegionID)
);

CREATE TABLE PersonsToPhoneNumbers
(
	PersonID	integer,
	PhoneNumberID		integer,
	PRIMARY KEY(PersonID, PhoneNumberID),
	FOREIGN KEY(PersonID) REFERENCES Persons(PersonID),
	FOREIGN KEY(PhoneNumberID) REFERENCES PhoneNumbers(PhoneNumberID)
);

CREATE TABLE PersonsToOrganizations
(
	PersonID	integer,
	OrganizationID	integer,
	BusinessTitle	varchar(64),
	PRIMARY KEY(PersonID, OrganizationID),
	FOREIGN KEY(PersonID) REFERENCES Persons(PersonID),
	FOREIGN KEY(OrganizationID) REFERENCES Organizations(OrganizationID)
);

CREATE TABLE OrganizationsToPhoneNumbers
(
	OrganizationID	integer,
	PhoneNumberID		integer,
	PRIMARY KEY(OrganizationID, PhoneNumberID),
	FOREIGN KEY(OrganizationID) REFERENCES Organizations(OrganizationID),
	FOREIGN KEY(PhoneNumberID) REFERENCES PhoneNumbers(PhoneNumberID)
);

ALTER TABLE Organizations ADD FOREIGN KEY(ContactEmployee,OrganizationID) REFERENCES
PersonsToOrganizations(PersonID, OrganizationID);
```