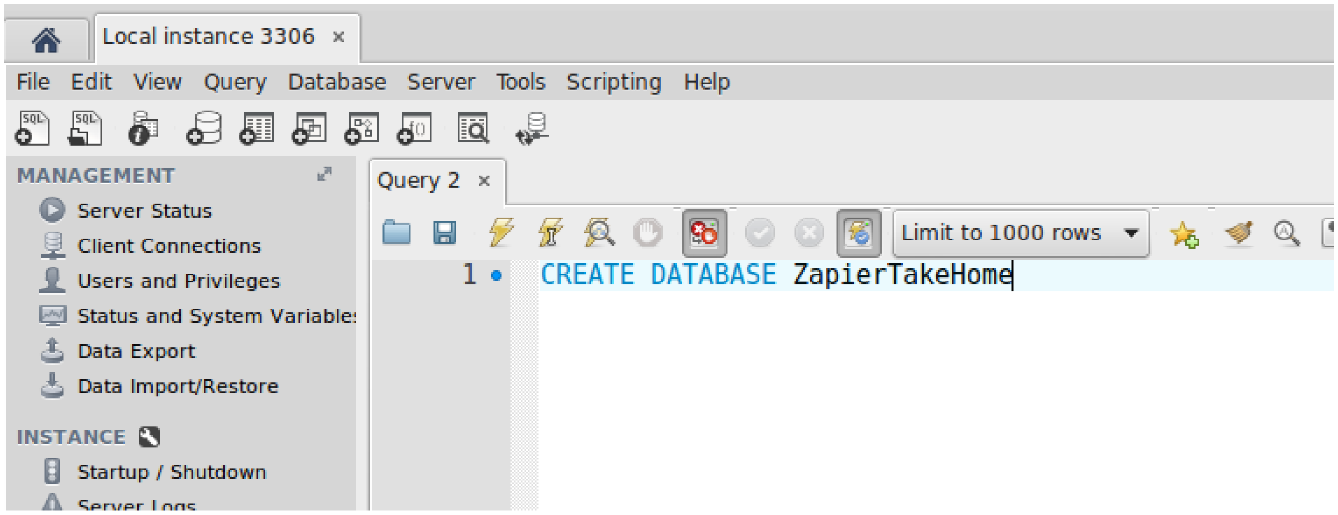
Lary Hartman – Zapier Data Engineer Take Home Assignment. July 2017.

Part one Ingest the event data to your own DB

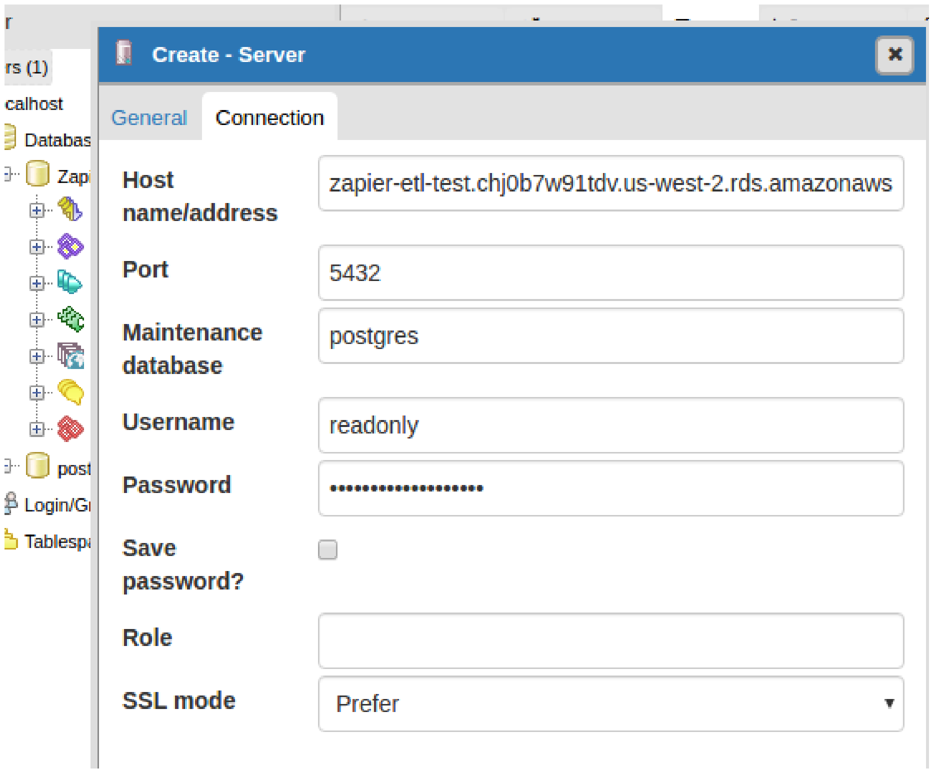
Step 1 – Create a Database to host the solution I ended up using MySql Workbench. (I Initially tried this with Postgres on Linux but had version issues as I mention below. I had success with MySQL so I went with that in the interest of moving on with the solution.)

CREATE DATABASE ZapierTakeHome;



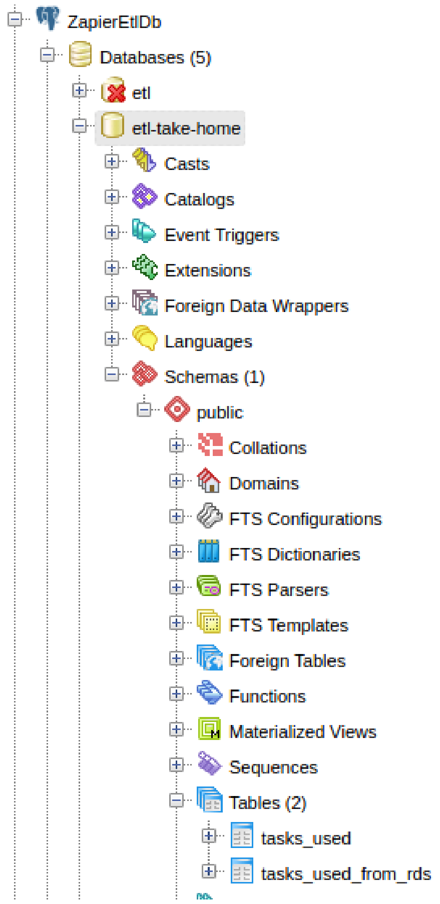
```
CREATE DATABASE ZapierTakeHome
```

Step2 – Create a connection to the AWS Instance (Postgres \ PgAdmin 4):



Host name/address	zapier-etl-test.chj0b7w91tdv.us-west-2.rds.amazonaws
Port	5432
Maintenance database	postgres
Username	readonly
Password	•••••••••••••••••••
Save password?	☐
Role	
SSL mode	Prefer

Step 3 – Verify the connection and object are valid -



```
ZapierEtlDb
  Databases (5)
    etl
    etl-take-home
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages
      Schemas (1)
        public
          Collations
          Domains
          FTS Configurations
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Sequences
          Tables (2)
            tasks_used
            tasks_used_from_rds
```

Step 4 – Get some table information so we know what we are dealing with -

SELECT COUNT(*) FROM tasks_used
result : 9,719,589

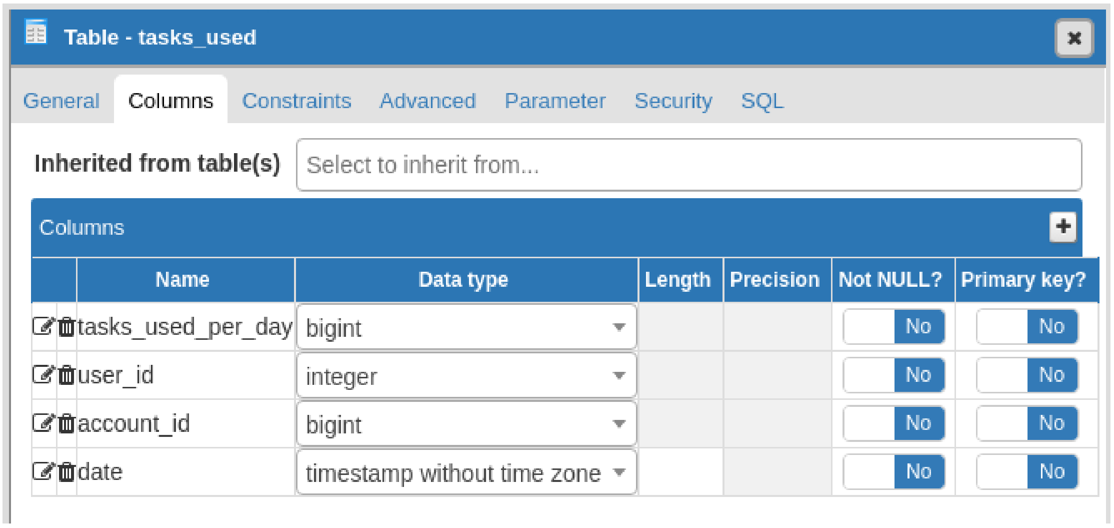


Table - tasks_used

Inherited from table(s): Select to inherit from...

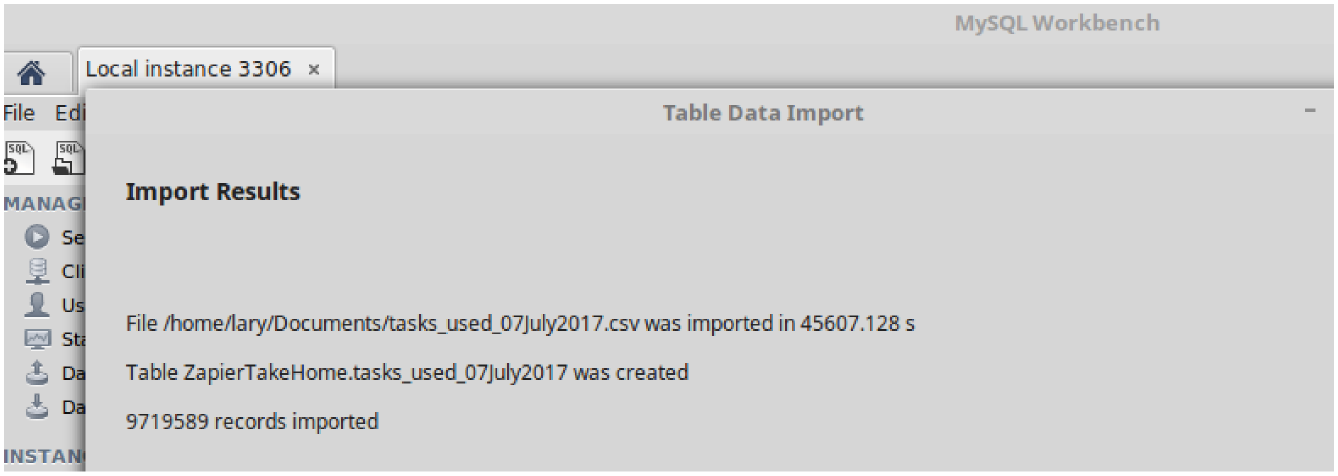
Columns

Name	Data type	Length	Precision	Not NULL?	Primary key?
tasks_used_per_day	bigint			No	No
user_id	integer			No	No
account_id	bigint			No	No
date	timestamp without time zone			No	No

Part two Extract the relevant data to a model.

Step 1 – Do a simple data import to a staging table in the solution database -

I had (500) errors using the native Postgres Import\Export tool. My suspicion is that this was due to using Linux and some of the binaries. In the interest of proceeding I exported a table SELECT into a .CSV, then performed a MySql file import from the CSV.



Import Results

```
File /home/lary/Documents/tasks_used_07July2017.csv was imported in 45607.128 s

Table ZapierTakeHome.tasks_used_07July2017 was created

9719589 records imported
```

Step 2 – Verify the rowcount we brought in from AWS in the staging table -

```
SELECT COUNT(*) FROM ZapierTakeHome.tasks_used_07July2017
```

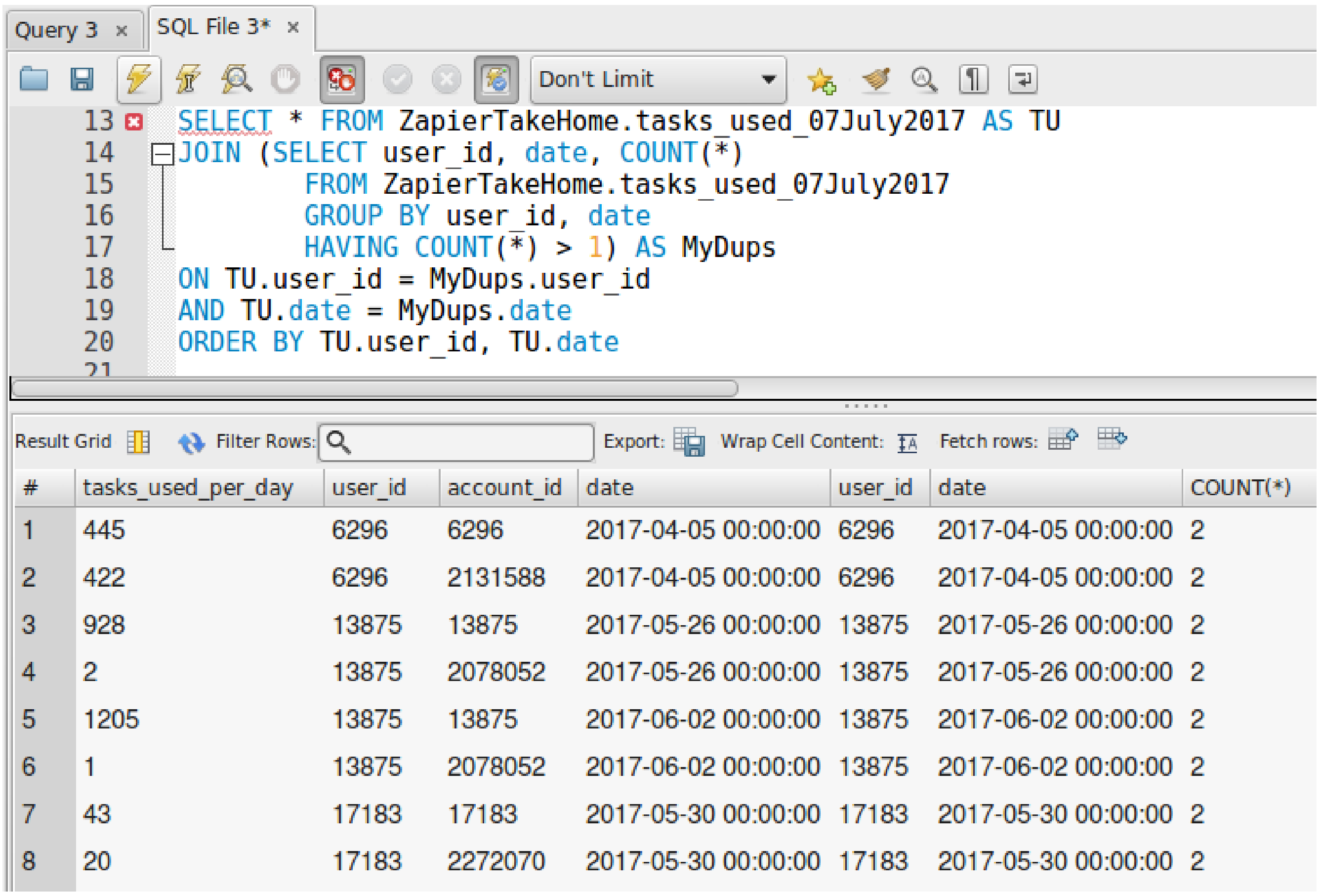
Result – 9,719,589. This matches the source count at the time we checked it.

Step 3 – Do some duplicate checking to see if there are any issues with the indexes we want to use:

```
SELECT user_id, date, COUNT(*)
FROM ZapierTakeHome.tasks_used_07July2017
GROUP BY user_id, date
HAVING COUNT(*) > 1
```

Result – 1096 rows.

Further investigation into the suspected duplicates shows that a user can have activity over multiple accounts per day.



```
SELECT * FROM ZapierTakeHome.tasks_used_07July2017 AS TU
JOIN (SELECT user_id, date, COUNT(*)
        FROM ZapierTakeHome.tasks_used_07July2017
        GROUP BY user_id, date
        HAVING COUNT(*) > 1) AS MyDups
ON TU.user_id = MyDups.user_id
AND TU.date = MyDups.date
ORDER BY TU.user_id, TU.date
```

#	tasks_used_per_day	user_id	account_id	date	user_id	date	COUNT(*)
1	445	6296	6296	2017-04-05 00:00:00	6296	2017-04-05 00:00:00	2
2	422	6296	2131588	2017-04-05 00:00:00	6296	2017-04-05 00:00:00	2
3	928	13875	13875	2017-05-26 00:00:00	13875	2017-05-26 00:00:00	2
4	2	13875	2078052	2017-05-26 00:00:00	13875	2017-05-26 00:00:00	2
5	1205	13875	13875	2017-06-02 00:00:00	13875	2017-06-02 00:00:00	2
6	1	13875	2078052	2017-06-02 00:00:00	13875	2017-06-02 00:00:00	2
7	43	17183	17183	2017-05-30 00:00:00	17183	2017-05-30 00:00:00	2
8	20	17183	2272070	2017-05-30 00:00:00	17183	2017-05-30 00:00:00	2

Conclusion – since the analysis is based on "User" and not "Account", the only relevant information is that a user has tasks used on a given day. For this reason account_id is really not needed.

Finally, let's check to see if there are any records with zero tasks, as these could affect the result negatively. If there are any we would want to throw those out on selection.

```
SELECT *
FROM ZapierTakeHome.tasks_used_07July2017 WHERE tasks_used_per_day = 0
        Result = 0
```

Conclusion – the number of tasks is not relevant to this analysis, rather just that the user had at least one task on any given day.

Step 4 – Create an "analysis" table which is indexed so we can get better performance. Also – only include the relevant data we will need for the analysis. Account_id and tasks_used are not needed to determine activity or attrition.

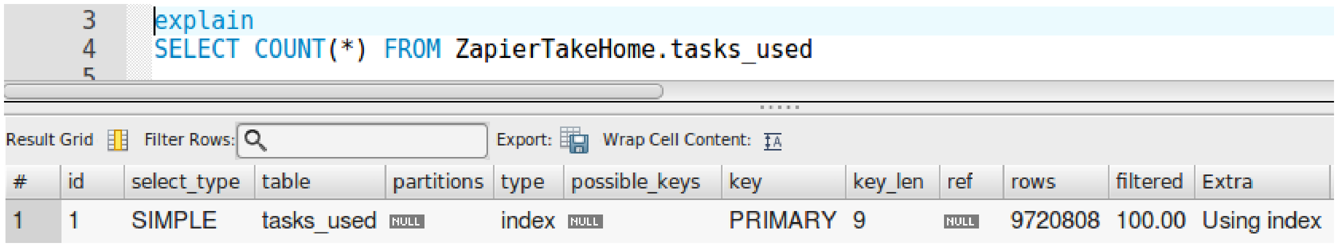
```
CREATE TABLE ZapierTakeHome.tasks_used (
  user_id int(11) NOT NULL,
  date datetime NOT NULL,
  PRIMARY KEY (user_id, date)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Step 5 - Insert the data into the analysis table:

```
INSERT INTO ZapierTakeHome.tasks_used(user_id, date)
SELECT DISTINCT user_id, date
FROM ZapierTakeHome.tasks_used_07July2017
```

Result - 9718493 row(s) affected Records: 9718493 Duplicates: 0 Warnings: 0   215.221 sec

Step 6 – Verify that the DB engine is using the index -



```
explain
SELECT COUNT(*) FROM ZapierTakeHome.tasks_used
```

Result Grid   Filter Rows: Export: Wrap Cell Content:

#	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	tasks_used	NULL	index	NULL	PRIMARY	9	NULL	9720808	100.00	Using index

Step 7 – For each record, we need to define what the active period and attrition dates are, then calculate the active periods, marry them up and aggregate them. The end result is Monthly counts of Active Users and Churned Users.

```
SELECT Active.Month, Active.Monthname, Active.ActiveUsers, Churn.ChurnUsers
FROM (
            /* A user is considered active on any day where they have at least one task is executed
            in the 28 days leading up to the event. */
            SELECT MONTH(TU.date) AS Month
                        ,MONTHNAME(TU.date) AS MonthName
      ,COUNT(DISTINCT TU.user_id) AS ActiveUsers
            FROM ZapierTakeHome.tasks_used TU
            LEFT JOIN (SELECT user_id
                                                ,date
                  FROM ZapierTakeHome.tasks_used
                  ) UserTasks /* A user is considered active on any day where they have at least one task
is executed
                                        in the 28 days leading up to the event. */
                  ON TU.user_id = UserTasks.user_id
                  AND UserTasks.date BETWEEN TU.Date AND DATE_SUB(TU.date, INTERVAL 28
DAY)
            GROUP BY MONTH(TU.date), MONTHNAME(TU.date)
   ) AS Active
LEFT JOIN (
            /* A user is considered to be churn the 28 days following their last being considered active. */
            SELECT MONTH(churnDate) AS Month
                        ,COUNT(DISTINCT user_id) AS ChurnUsers
   FROM (
                  SELECT TU.user_id
                              ,MAX(DATE_ADD(TU.date, INTERVAL 27 DAY)) AS churnDate
                  FROM ZapierTakeHome.tasks_used TU
                  LEFT JOIN
                        (SELECT DISTINCT user_id, date
                         FROM ZapierTakeHome.tasks_used
```

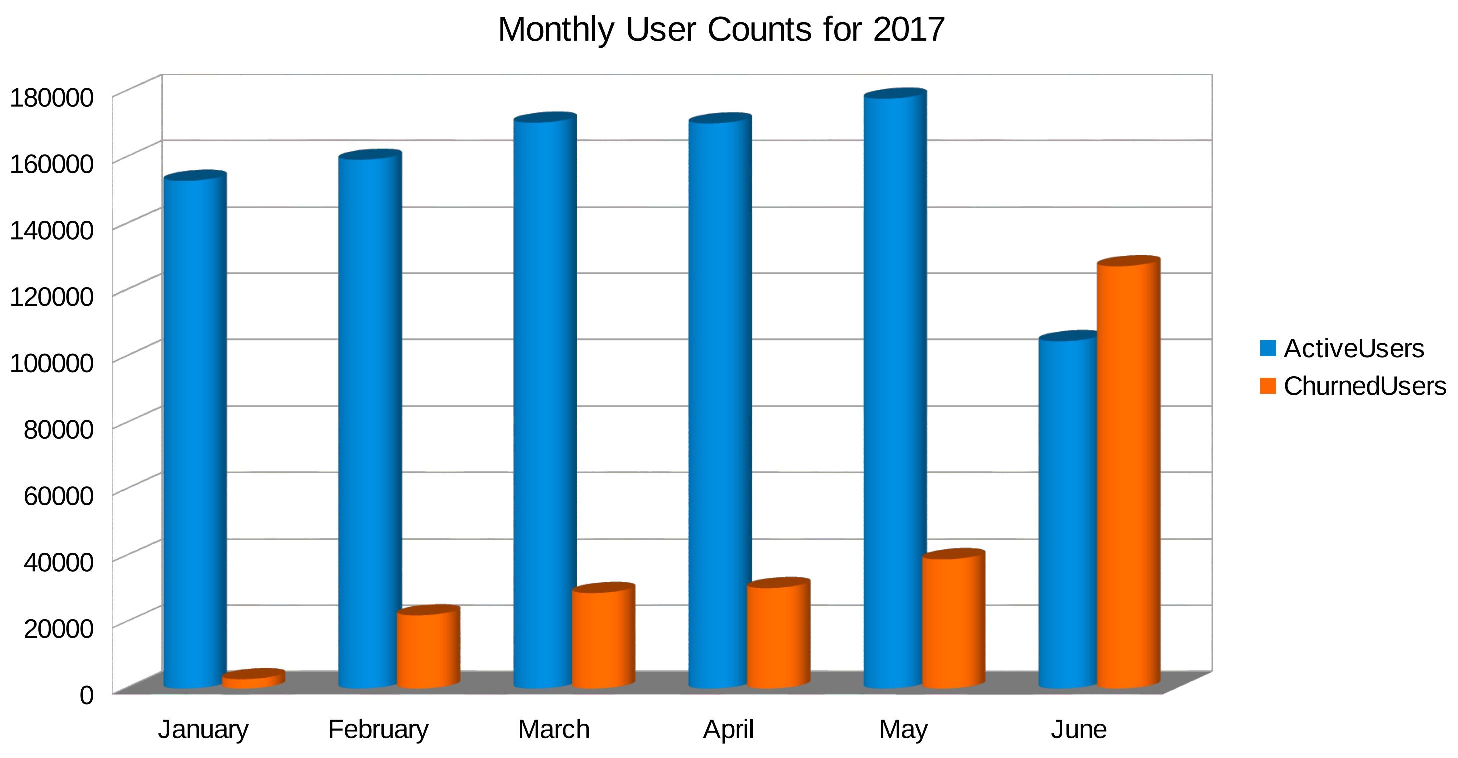
```
                                        ) AS ActiveDates /* A user is no longer part of churn if they become active
again.*/
                                        ON DATE_ADD(TU.date, INTERVAL 27 DAY) < ActiveDates.date
                                  AND ActiveDates.date IS NULL
                                  GROUP BY TU.user_id) AS ChurnDateByUser
                      GROUP BY MONTH(churnDate)
                      ) AS Churn ON Active.Month = Churn.Month
ORDER BY Active.Month;
```

The query produces the following result:

Month	ActiveUsers	ChurnedUsers
January	153018	2836
February	159366	22089
March	170551	28802
April	170317	30292
May	177764	38949
June	104677	127287

Part three Visualize Monthly Active Users and churn over time

Placing these into a spreadsheet allows us to visualize the data in the following way -


Monthly User Counts for 2017
January
February
March
April
May
June
ActiveUsers
ChurnedUsers