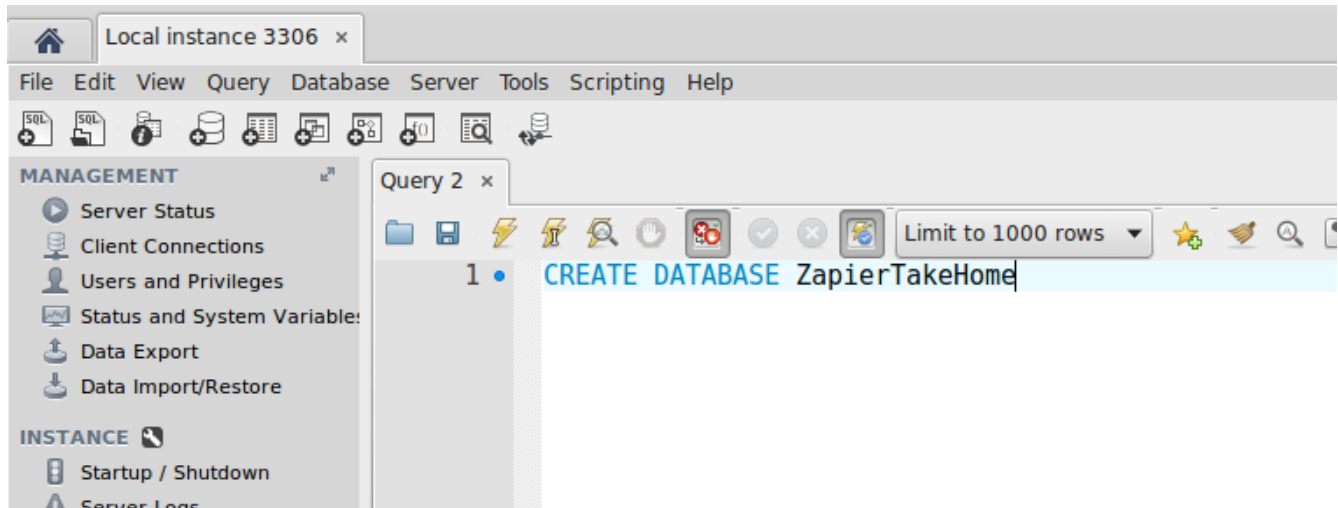


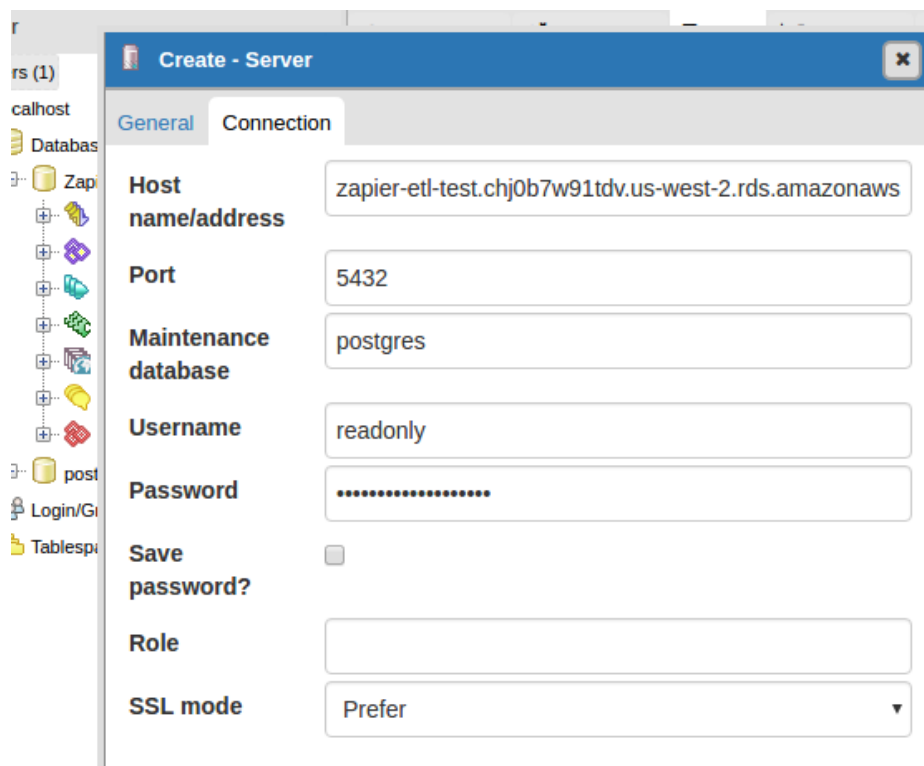
Part one Ingest the event data to your own DB

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.)

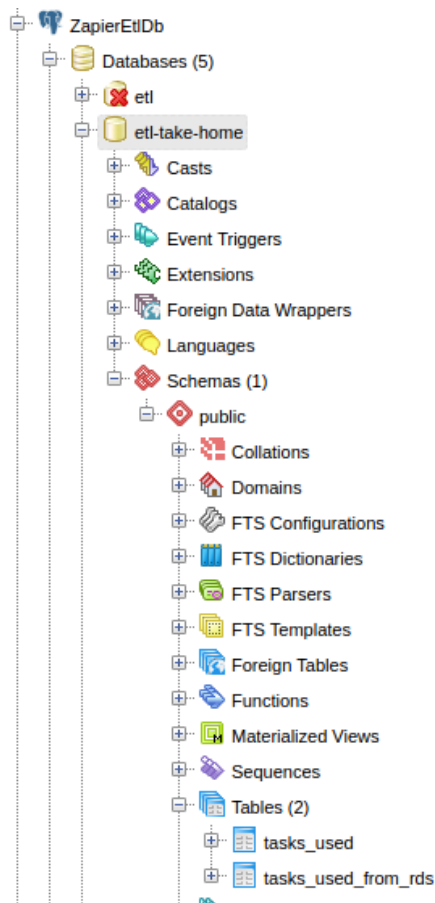
CREATE DATABASE ZapierTakeHome;



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



Step 3 – Verify the connection and object are valid -



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

General

Columns

Constraints

Advanced

Parameter

Security

SQL

Inherited from table(s)

Select to inherit from...

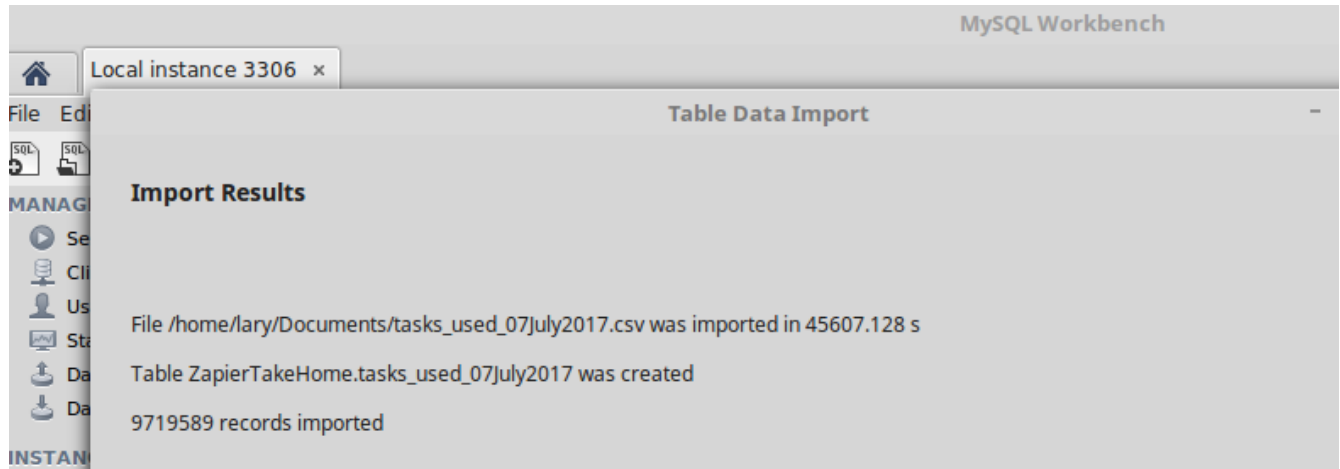
Columns

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

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.



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.

Query 3 x SQL File 3* x

Don't Limit

```

13 SELECT * FROM ZapierTakeHome.tasks_used_07July2017 AS TU
14 JOIN (SELECT user_id, date, COUNT(*)
15       FROM ZapierTakeHome.tasks_used_07July2017
16       GROUP BY user_id, date
17       HAVING COUNT(*) > 1) AS MyDups
18 ON TU.user_id = MyDups.user_id
19 AND TU.date = MyDups.date
20 ORDER BY TU.user_id, TU.date
21

```

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:

#	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 -

3

4

5

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. We will do this by Day so we can show counts by date and also aggregate by month..

```
SELECT Active.Monthname,Active.Day, Active.ActiveUsers, NULLIF(Churn.ChurnUsers,0) AS 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 MONTHNAME(TU.date) AS MonthName
           ,DATE(TU.date) AS Day
    ,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 MONTHNAME(TU.date), DATE(TU.date)
    ) AS Active
LEFT JOIN (
    /* A user is considered to be churn the 28 days following their last being considered active. */
    SELECT DATE(churnDate) AS Day
           ,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
```

```

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

```

Part three Visualize Monthly Active Users and churn over time

June data in set the set is incomplete and does not represent significant enough information to depict that month, so the data points for June 1-4 are being excluded for this analysis:

June	2017-06-01	77,559	1,593
June	2017-06-02	73,650	835
June	2017-06-03	50,708	846
June	2017-06-04	49,267	1,772

Placing the result set into a spreadsheet program and creating a pivot table allows us to derive some useful information. This information has been compiled and commented on in the included file:

ZapierTakeHomeStakeholderAnalysis.xlsx