Usage ClickHouse for mobile app analytics

Dmitry Shemyatikhin, analyst

What do we want from mobile analytics?

Descriptive statistics

- DAU, MAU, usage (user events)
- Revenue

Marketing analytics

- Catch the traffic source

Cohort analysis

- LTV, Conversions, Retention

A/B Experiments

This story is about

How to arrange data

How to use ClickHouse for common metric calculations

- Aggregate functions
- Joins

Model application: app for taxi service

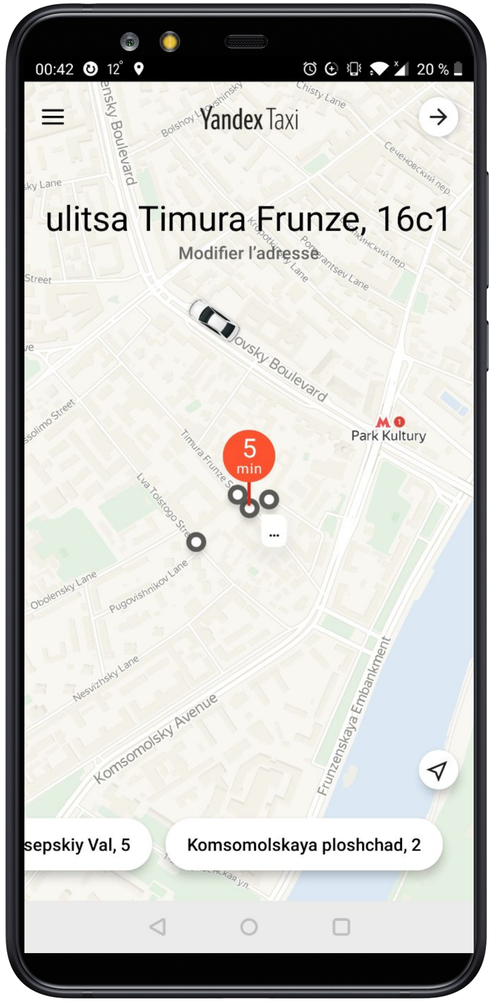
User steps in model taxi app

Install

Tell phone number

Attach credit card

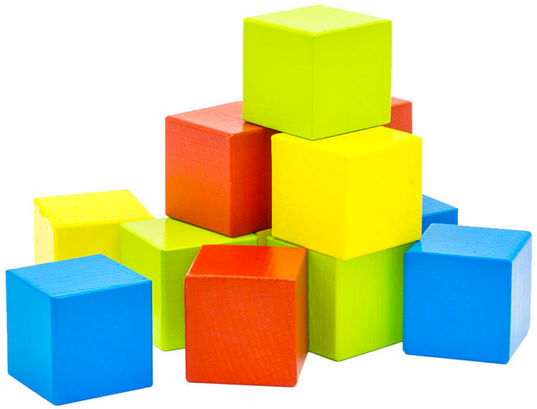
Make rides



Data model

Events-based analytics

One large table with events of five types:



1. Installs
2. Events with marketing source
3. Base events (like app open)
4. Custom user events (user tell us phone, create a ride)
5. Events with experiment values

Image from http://oxsid.ru/wa-data/public/shop/products/52/03/352/images/654/654.970.jpg

Custom user events

All events inside the app: usage of menu, navigations, buttons, etc.

- Always need to send additional data. E.g. the title of clicked menu item
- More than one parameter in one event
- Key-value storage

Custom user events

Key-value storage in three arrays:

- paramsKey
- paramsValue
- paramsFloat

Semantic unit number i

- paramsKey[i] = paramsValue[i]

eventName	paramsKey	paramsValue	paramsFloat
phone	['phone']	['79264101769']	[0]
ride	['price','**revenue**','tariff']	['','','econom']	[18.499384,**3.699877**,0]

Experiments

Event which describe experiment groups of current user

More than one at the same time

The same approach as key-value parameters

- paramsKey for experiment names
- paramsValue for experiment groups

paramsKey	paramsValue
['**button_color**', 'button_placement']	['**green**', 'up']
['**button_color**', 'button_placement']	['**red**', 'up']

Table scheme

Table scheme

```
CREATE TABLE IF NOT EXISTS events (
    `utc_dttm` DateTime,
    `userID` UInt64,
    `eventName` String,
    `eventType` Enum('install' = 1, 'install_source' = 2, 'base_event' = 3,
                     'custom_event' = 4, 'experiment' = 5),
    `paramsKey` Array(String),
    `paramsValue` Array(String),
    `paramsFloat` Array(Float32),
    `os` String
)
ENGINE = MergeTree()
ORDER BY (utc_dttm, eventType, intHash32(userID))
SAMPLE BY intHash32(userID)
```

Table scheme

```
CREATE TABLE IF NOT EXISTS events (
    `utc_dttm` DateTime,
    `userID` UInt64,
    `eventName` String,
    `eventType` Enum('install' = 1, 'install_source' = 2, 'base_event' = 3,
                     'custom_event' = 4, 'experiment' = 5),
    `paramsKey` Array(String),
    `paramsValue` Array(String),
    `paramsFloat` Array(Float32),
    `os` String
)
ENGINE = MergeTree()
ORDER BY (utc_dttm, eventType, intHash32(userID))
SAMPLE BY intHash32(userID)
```

In practice here would be much more columns, like

- App version
- OS version
- …

Events, used in examples

	eventType	eventName	paramsKey	paramsValue	paramsFloat
App install	install	install	-	-	-
User tells phone	custom_event	phone	['+79264101769']	[]	[]
User attaches card	custom_event	card	-	-	-
Taxi ride	custom_event	ride	['price', 'revenue', 'tariff']	['', '', 'econom']	[10.0, 1.0, 0.0]
Experiment info	experiment	experiment	['button', 'place']	['red', 'up']	-

Metrics

Simple metrics

Simple metrics

Users statistics:

- DAU (daily active users), WAU (weekly …), MAU (monthly …)
- Users with certain event by day, week, month

Event statistics:

- Event count (e.g. rides count)
- Revenue

Image from https://images.freeimages.com/images/large-previews/36b/graph-line-up-and-down-1-1237014.jpg

Simple metrics: weekly active users (WAU)

```
SELECT
    toMonday(utc_dttm) AS date,
    os,
    uniqExact(userID) AS wau
FROM events
WHERE (eventType IN ('install', 'base_event', 'custom_event'))
       AND (date > toDate('2019-05-01'))
GROUP BY
    date,
    os
```

Simple metrics: WAU, installs, users with ride

```
SELECT
    toMonday(utc_dttm) AS date,
    os,
    uniqExact(userID) AS wau,
    uniqExactIf(userID, eventType = 'install') AS installs,
    uniqExactIf(userID, eventType = 'custom_event' AND eventName = 'ride') AS clients
FROM events
WHERE (
    eventType IN ('install', 'base_event', 'custom_event'))
    AND (date > toDate('2019-05-01')
)
GROUP BY
    date, os
```

Simple metrics: add rides

```
SELECT
    toMonday(utc_dttm) AS date,
    os,
    uniqExact(userID) AS wau,
    uniqExactIf(userID, eventType = 'install') AS installs,
    uniqExactIf(userID, eventType = 'custom_event' AND eventName = 'ride') AS clients,
    countIf(eventType = 'custom_event' AND eventName = 'ride') AS rides
FROM events
WHERE (
    eventType IN ('install', 'base_event', 'custom_event'))
    AND (date > toDate('2019-05-01')
)
GROUP BY
    date, os
```

And how to add here values from array parameters?

Handling array parameters

```
SELECT
    eventName,
    paramsKey,
    paramsValue,
    paramsFloat
FROM events
WHERE (userID = 1419598) AND (eventName = 'ride')
```

```
┌─eventName─┬─paramsKey─────────────────────────┬─paramsValue─────────┬─paramsFloat──────────────┐
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [16.35786,3.271572,0]    │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [18.499384,3.699877,0]   │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [39.319595,7.863919,0]   │
└───────────┴───────────────────────────────┴────────────────────┴──────────────────────────┘
```

Handling array parameters

```
SELECT
    eventName,
    paramsKey,
    paramsValue,
    paramsFloat
FROM events
WHERE (userID = 1419598) AND (eventName = 'ride’)
```

```
┌─eventName─┬─paramsKey─────────────────────────┬─paramsValue─────────┬─paramsFloat──────────────┐
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [16.35786,3.271572,0]    │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [18.499384,3.699877,0]   │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [39.319595,7.863919,0]   │
└───────────┴───────────────────────────────┴────────────────────┴──────────────────────────┘
```

Handling array parameters

```
SELECT
    eventName,
    paramsKey,
    paramsValue,
    paramsFloat
FROM events
WHERE (userID = 1419598) AND (eventName = 'ride’)
```

```
┌─eventName─┬─paramsKey─────────────────────────┬─paramsValue─────────┬─paramsFloat──────────────┐
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [16.35786,3.271572,0]   │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [18.499384,3.699877,0] │
│ ride      │ ['price','revenue','tariff'] │ ['','','econom'] │ [39.319595,7.863919,0] │
└───────────┴───────────────────────────────┴─────────────────────┴──────────────────────────┘
```

Handling array parameters

Raw arrays:

```
┌─paramsKey──────────────────────────┬─paramsFloat──────────────────┐
│ ['price', 'revenue', 'tariff'] │ [116.74424, 23.348848, 0] │
└────────────────────────────────────┴──────────────────────────────┘
```

arrayFilter((n, v) -> v = 'revenue', paramsFloat, paramsKey)

```
┌─arrayFilter(lambda(tuple(n, v), equals(v, 'revenue')), paramsFloat, paramsKey)─┐
│ [23.348848]                                                                    │
└────────────────────────────────────────────────────────────────────────────────┘
```

arraySum(...)

```
┌─arraySum(arrayFilter(lambda(tuple(n, v), equals(v, 'revenue')), paramsFloat, paramsKey))─┐
│ 23.348848342895508                                                                       │
└──────────────────────────────────────────────────────────────────────────────────────────┘
```

Simple metrics: add revenue

```
SELECT
    toMonday(utc_dttm) AS date,
    os,
    uniqExact(userID) AS wau,
    uniqExactIf(userID, eventType = 'install') AS installs,
    uniqExactIf(userID, eventType = 'custom_event' AND eventName = 'ride') AS clients,
    countIf(eventType = 'custom_event' AND eventName = 'ride') AS rides,
    sumIf(arraySum(arrayFilter((n, v) -> v = 'revenue', paramsFloat, paramsKey)),
        eventType = 'custom_event' AND eventName = 'ride') as revenue
FROM events
WHERE (
    eventType IN ('install', 'base_event', 'custom_event'))
    AND (date > toDate('2019-05-01')
) GROUP BY date, os
```

Simple metrics: revenue for tariff ‘econom’

```
SELECT
    toMonday(utc_dttm) AS date,
    os,
    uniqExact(userID) AS wau,
    uniqExactIf(userID, eventType = 'install') AS installs,
    uniqExactIf(userID, eventType = 'custom_event' AND eventName = 'ride') AS clients,
    countIf(eventType = 'custom_event' AND eventName = 'ride') AS rides,
    sumIf(arraySum(arrayFilter((n, v) -> v = 'revenue', paramsFloat, paramsKey)),
        eventType = 'custom_event' AND eventName = 'ride' AND has(paramsValue, 'econom’))
        AS revenue
FROM events
WHERE (
    eventType IN ('install', 'base_event', 'custom_event’))
    AND (date > toDate('2019-05-01’)
) GROUP BY date, os
```

Cohort metrics

Cohort metrics

Conversions

- How much users add card in app?

Retention

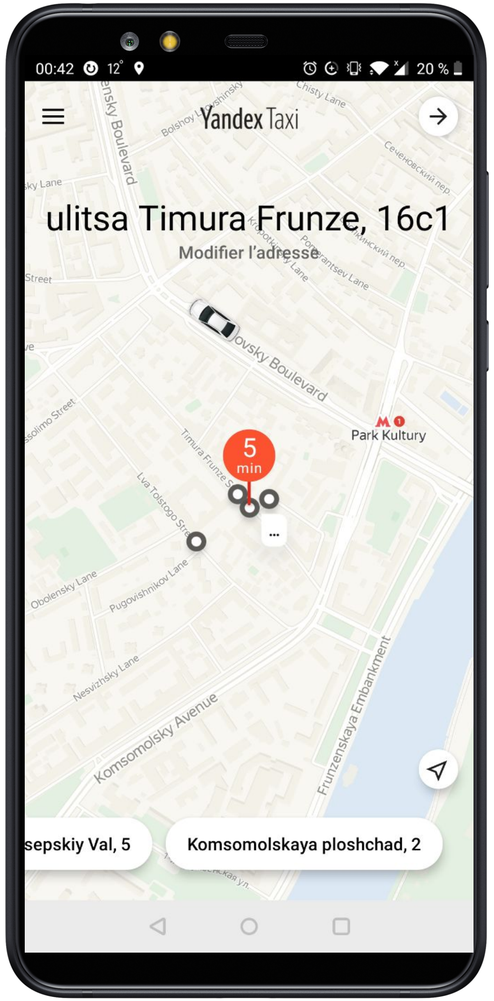
- How much users open app on X-th day after install?

Revenue

- Revenue per install
- Life time value

User experience

	eventType	eventName
Install	install	install
Tell phone number	custom_event	phone
Card attached	custom_event	card
Rides	custom_event	ride



Conversion

- How much users tell us their phone?
- How much users add card in app?
- How much users create ride?

Image from https://avatars.mds.yandex.net/get-pdb/1566941/dbaae2a0-bc2d-40d7-863b-a5bab0e807aa/s1200?webp=false

Conversions, common method (join)

```
SELECT
    sum(install) AS installs,
    sum(phone_event) AS phone_events,
    phone_events / installs AS phone_conversion,
    toMonday(install_date) AS date, os
FROM ( < ... Installs ... > )
ANY LEFT JOIN (
    SELECT
        userID,
        1 AS phone_event
    FROM events
    WHERE (eventType = 'custom_event') AND (eventName = 'phone')
        AND (utc_dttm > toDate('2019-05-01'))
) USING (userID)
GROUP BY date, os
```

Conversions, common method (join)

```
SELECT
    sum(install) AS installs,
    sum(phone_event) / installs AS phone_conversion,
    sum(card_event) / installs AS card_conversion,
    toMonday(install_date) AS date, os
FROM ( < ... Installs ... > )
ANY LEFT JOIN (
    SELECT
        userID,
        if(eventName = 'phone', 1, 0) AS phone_event,
        if(eventName = 'card', 1, 0) AS card_event
    FROM events
    WHERE (eventType = 'custom_event') AND (eventName in ('phone', 'card'))
        AND (utc_dttm > toDate('2019-05-01'))
) USING (userID)
GROUP BY date, os
```

Conversions, using ClickHouse functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    toMonday(install_dttm) AS date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        ???? AS install_dttm,
        ???? AS phone_dttm
    FROM ( < ... All events with basic filters ... > )
    GROUP BY userID
    HAVING install_dttm != 0
) GROUP BY date, os_system
```

Conversions, array functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    toMonday(install_dttm) AS date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'phone') AND (t >= install_dttm)), ts,
            event_list)[1] AS phone_dttm
    FROM ( < ... All events with basic filters order by utc_dttm... >  )
    GROUP BY userID
    HAVING install_dttm != 0
) GROUP BY date, os_system
```

Conversions, array functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    toMonday(install_dttm) AS date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'phone') AND (t >= install_dttm)), ts,
            event_list)[1] AS phone_dttm
    FROM ( < ... All events with basic filters order by utc_dttm... >  )
    GROUP BY userID
    HAVING install_dttm != 0
) GROUP BY date, os_system
```

```
┌─groupArray(utc_dttm)──────────────────────────────────────────────┐
│ ['2017-01-02 00:00:00','2017-01-02 01:00:00']                     │
└───────────────────────────────────────────────────────────────────┘
```

```
┌─groupArray(eventName)─┐
│ ['install','phone']   │
└───────────────────────┘
```

Conversions, array functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    toMonday(install_dttm) AS date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'phone') AND (t >= install_dttm)), ts,
            event_list)[1] AS phone_dttm
    FROM ( < ... All events with basic filters order by utc_dttm... >  )
    GROUP BY userID
    HAVING install_dttm != 0
) GROUP BY date, os_system
```

```
┌─groupArray(utc_dttm)──────────────────────────────────────────┐
│ ['2017-01-02 00:00:00','2017-01-02 01:00:00']                 │
└───────────────────────────────────────────────────────────────┘
```

```
┌─groupArray(eventName)─┐
│ ['install','phone']   │
└───────────────────────┘
```

Conversions, array functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    toMonday(install_dttm) AS date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'phone') AND (t >= install_dttm)), ts,
            event_list)[1] AS phone_dttm
    FROM ( < ... All events with basic filters order by utc_dttm... >  )
    GROUP BY userID
    HAVING install_dttm != 0
) GROUP BY date, os_system
```

Conversions, array functions

```
SELECT
    sum(phone_dttm != 0) / count(*) AS phone_conversion,
    sum(card_dttm != 0) / count(*) AS card_conversion,
    < ... >
FROM (
    SELECT < ... >
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'phone') AND (t >= install_dttm)), ts,
            event_list)[1] AS phone_dttm,
        arrayFilter((t, e) -> ((e = 'card') AND (t >= install_dttm)), ts,
            event_list)[1] AS card_dttm
    FROM ( < ... All events with basic filters order by utc_dttm ... >  )
    GROUP BY userID HAVING install_dttm != 0
) GROUP BY date, os_system
```

Conversions, windowFunnel

```
SELECT
    count() AS installs,
    countIf(conversion_depth > 1) AS phone_users,
    phone_users / installs AS phone_conversion,
    install_date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        toMonday(minIf(utc_dttm, eventType = 'install')) AS install_date,
        windowFunnel(604800)(utc_dttm, eventType = 'install', eventName = 'phone')
            AS conversion_depth
    FROM events WHERE < ... Basic filters ... > GROUP BY userID )
WHERE conversion_depth > 0
GROUP BY date, os_system
```

Conversions, windowFunnel

```
SELECT
    count() AS installs,
    countIf(conversion_depth > 1) AS phone_users,
    phone_users / installs AS phone_conversion,
    install_date,
    os_system
FROM (
    SELECT
        any(os) AS os_system,
        toMonday(minIf(utc_dttm, eventType = 'install')) AS install_date,
        windowFunnel(604800)(utc_dttm, eventType = 'install', eventName = 'phone')
            AS conversion_depth
    FROM events WHERE < ... Basic filters ... > GROUP BY userID )
WHERE conversion_depth > 0
GROUP BY install_date, os_system
```

Time window

First condition

Second condition,
- met with the first
- within time window

Conversions, windowFunnel. More steps

```
SELECT
    count() AS installs,
    countIf(conversion_depth > 1) AS phone_users,
    countIf(conversion_depth > 2) AS card_users,
    < ... >
FROM (
    SELECT
        < ... >
        windowFunnel(604800)(utc_dttm, eventType = 'install', eventName = 'phone',
            eventName = 'card') AS conversion_depth
    FROM events WHERE < ... Basic filters ... > GROUP BY userID )
WHERE conversion_depth > 0
GROUP BY install_date, os_system
```

Retention metrics

Retention

- How many users open app at 7-th (30-th) day after install?
- How many users open app after more then a week after install?

Image from https://roditeli.ua/uploads/images/1/10.jpg

Retention, common method (join)

```
SELECT
    sum(retention_user) / sum(install) AS retention7,
    install_date
FROM (
    SELECT
        userID,
        toMonday(utc_dttm) AS install_date,
        toDate(utc_dttm) + 7 AS dateX,
        1 AS install
    FROM events WHERE (eventType = 'install’)
) ANY LEFT JOIN
( < ... All events with basic filters ... > )
USING (userID, dateX)
GROUP BY install_date
```

Retention, arrays

```
SELECT
    sum(retention7_dttm != 0) / count() AS retention7,
    toMonday(install_dttm) AS install_date
FROM (
    SELECT
        groupArray(utc_dttm) AS ts,
        groupArray(eventType) AS event_list,
        arrayFilter((t, e) -> (e = 'install'), ts, event_list)[1] AS install_dttm,
        arrayFilter((t, e) -> ((e = 'custom_event') AND
            (toDate(t) = (toDate(install_dttm) + 7))), ts, event_list)[1] AS retention7_dttm
    FROM events
    WHERE < ... Basic filters ... > GROUP BY userID
    HAVING install_dttm != 0
)
GROUP BY install_date
```

Revenue metrics

Revenue metrics (LTV)

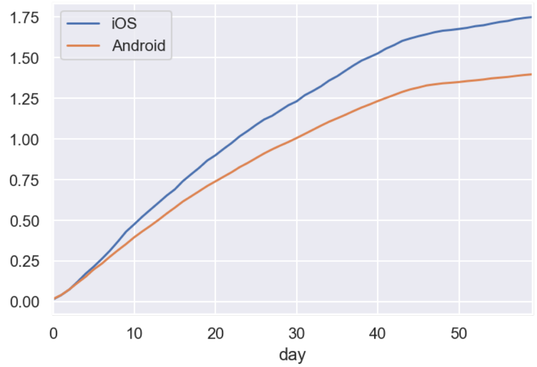
- How much revenue one installed app yields during the first X days.
  LTV = Life Time Value

- The same metric in dynamic

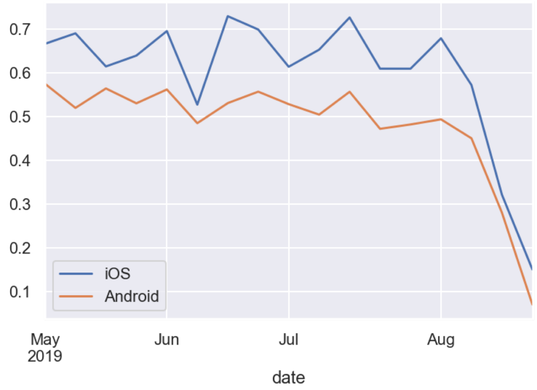
Image from https://3.bp.blogspot.com/-iPHuRIA9NhE/UVV_zZVTK3I/AAAAAAAAAEg/3zN0NbKucDM/s640/%D7%93%D7%99%D7%A0%D7%A8-%D7%96%D7%94%D7%91.jpg

Revenue metrics (LTV)

- How much revenue one installed app yields during the first X days.
  LTV = Life Time Value



- The same metric in dynamic

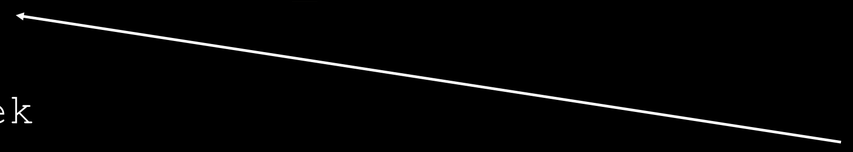


Life Time Value (LTV) by date

```
SELECT
    install_week,
    sum(revenue) / uniq(userID) AS ltv
FROM (
    SELECT
        revenue,
        toMonday(install_date) AS install_week
    FROM ( < ... Installs ... > )
    ALL LEFT JOIN ( < ... Revenue ... > ) USING (userID)
    WHERE (event_date - install_date) <= 14
)
GROUP BY install_week
```

Life Time Value (LTV) by date

```
SELECT
    install_week,
    sum(revenue) / uniq(userID) AS ltv
FROM (
    SELECT
        revenue,
        toMonday(install_date) AS install_week
    FROM ( < ... Installs ... > )
    ALL LEFT JOIN ( < ... Revenue ... > ) USING (userID)
    WHERE (event_date - install_date) <= 14
)
GROUP BY install_week
```



event_date might be '0000-00-00'!

Life Time Value (LTV) by day from install

```
SELECT
    day_from_install,
    sum(revenue) / uniq(userID) AS ltv
FROM (
    SELECT
        revenue,
        greatest(0, event_date - install_date) AS cohort_day,
        arrayMap(x -> (x + cohort_day), range(toUInt8(60 - cohort_day))) AS d
    FROM ( < ... Installs ... > )
    ALL LEFT JOIN ( < ... Revenue ... > ) USING (userID)
    WHERE (cohort_day <= 60) AND (install_date < toDate('2019-07-01'))
)
ARRAY JOIN d AS day_from_install
GROUP BY day_from_install
```

Life Time Value (LTV)

```
SELECT
    day_from_install,
    sum(revenue) / uniq(userID) AS ltv
FROM (
    SELECT
        revenue,
        greatest(0, event_date - install_date) AS cohort_day,
        arrayMap(x -> (x + cohort_day), range(toUInt8(60 - cohort_day))) AS d
    FROM ( < ... Installs ... > )
    ALL LEFT JOIN ( < ... Revenue ... > ) USING (userID)
    WHERE (cohort_day <= 60) AND (install_date < toDate('2019-07-01'))
)
ARRAY JOIN d AS day_from_install
GROUP BY day_from_install
```

An event on X-th day after install produces array
`[X-1, X, X+1, ..., 59]`

Life Time Value (LTV)

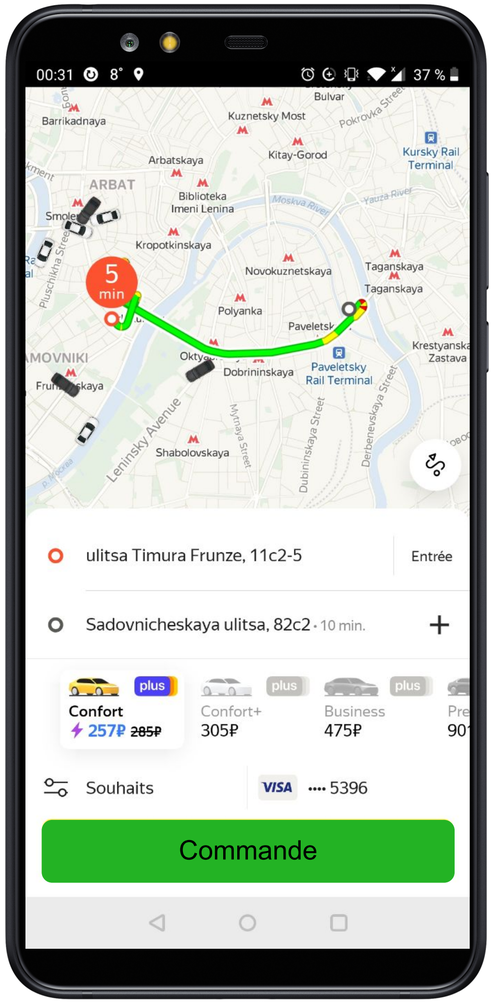
```
SELECT
    day_from_install,
    sum(revenue) / uniq(userID) AS ltv
FROM (
    SELECT
        revenue,
        greatest(0, event_date - install_date) AS cohort_day,
        arrayMap(x -> (x + cohort_day), range(toUInt8(60 - cohort_day))) AS d
    FROM ( < ... Installs ... > )
    ALL LEFT JOIN ( < ... Revenue ... > ) USING (userID)
    WHERE (cohort_day <= 60) AND (install_date < toDate('2019-07-01'))
)
ARRAY JOIN d AS day_from_install
GROUP BY day_from_install
```

An event on *X-th* day after install produces array
`[X-1, X, X+1, ..., 59]`

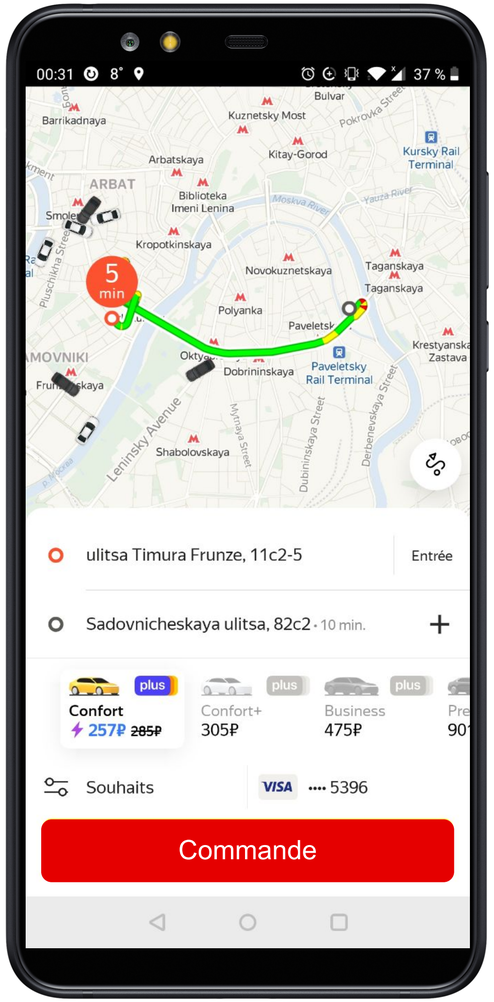
Group by unfolded array

Two words about handling A/B experiments

A/B experiments



VS



Events with experiment groups

Event describe experiment groups for current user

More than one at the same time

The same approach as key-value parameters

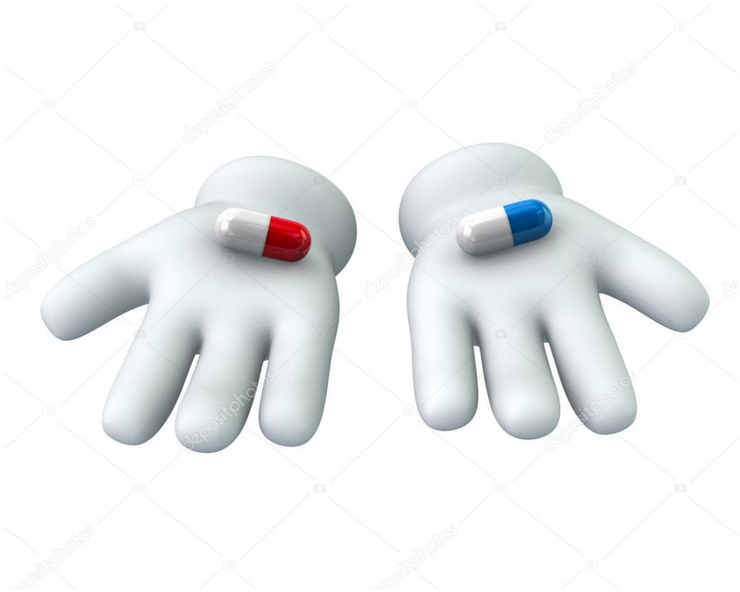
- paramsKey for experiment names
- paramsValue for experiment groups

paramsKey	paramsValue
['**button_color**', 'button_placement']	['**green**', 'up']
['**button_color**', 'button_placement']	['**red**', 'up']

Metrics in A/B experiments

All the metrics mentioned above:

- DAU, WAU
- Event count
- Conversions
- Revenue
- LTV



Two ways to calculate

- Additional join with (userID, experiment_group) subquery
- Get experiment group from aggregate functions

Image from https://st2.depositphotos.com/35543337/9423/i/950/depositphotos_94231422-stock-photo-hands-with-red-and-blue.jpg

Conversion, split by A/B

```
SELECT
    countIf(conversion_depth > 1) / countIf(conversion_depth > 0) AS conversion,
    date, exp_group
FROM (
    SELECT
        toMonday(minIf(utc_dttm, eventType = 'install')) AS date,
        anyIf(arrayFirst((value, title) -> (title = 'button'), paramsValue, paramsKey),
            eventType = 'experiment') AS exp_group,
        windowFunnel(604800)(utc_dttm, eventType = 'install', eventName = 'phone')
            AS conversion_depth
    FROM events
    WHERE < ... Basic filters ... > GROUP BY userID
)
WHERE conversion_depth > 0
GROUP BY date, exp_group
```

Summary

- One large table for all events
- Any key-value data can be kept in three arrays
- Array functions in some cases can replace joins
- ClickHouse has a lot of useful aggregate functions
- And it all works fast :)

Thank you

Dmitry Shemyatikhin

analyst

 dmshem@yandex-team.ru

 @dmshem