Yandex

Yandex

Creating marketing funnels or how to calculate complex metrics in ClickHouse

Mary Mansurova, Yandex.Metrica, analyst

Analysts at Yandex

- Can code (usually Python or R)
- Answer manager's questions
- Make reports, alerts etc.

analysts + ClickHouse = 💛

ClickHouse

- Column-oriented DBMS
- SQL-like dialect
- Extremely fast
- Allows you to calculate rather complex metrics

The story of one task

AppMetrica

A free real-time ad tracking and mobile app analytics solution

- App can send custom events to AppMetrica
- Logs API to load raw app's data

Clients always want more analytics

Conversions and marketing funnels

Conversion

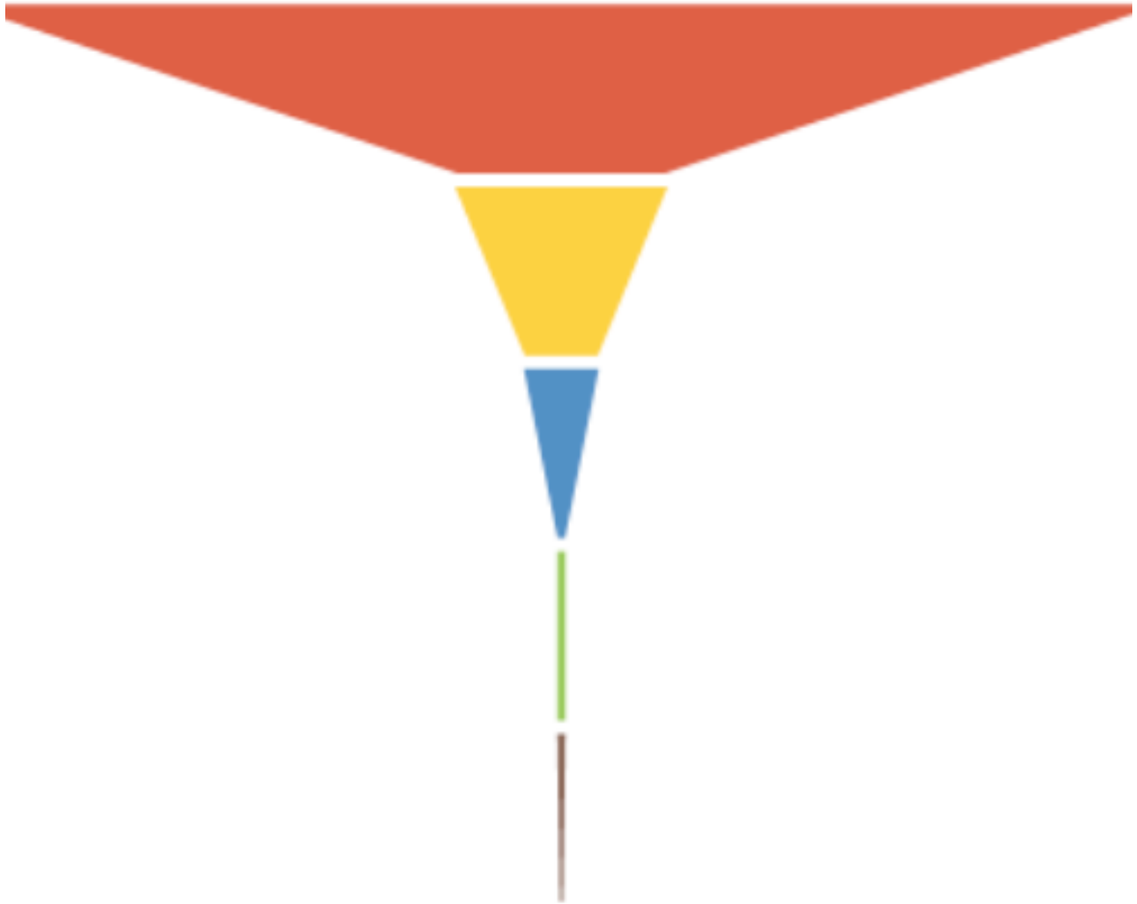
An app owner wants users to take certain actions. This is a conversion.

Examples

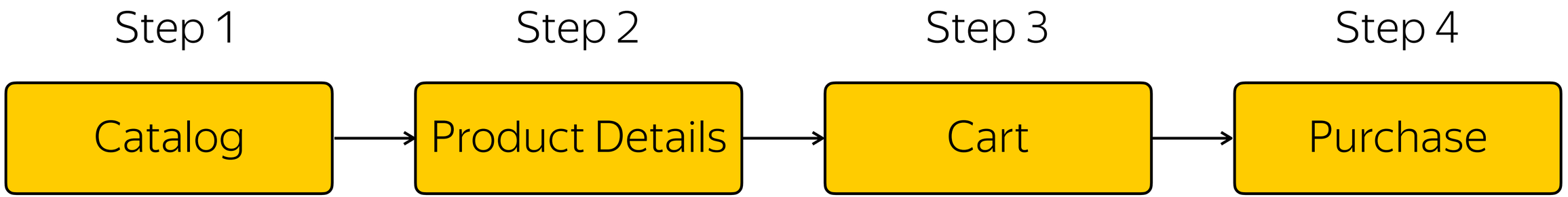
- Make a purchase
- Sign up
- Fill out a form

Marketing Funnel

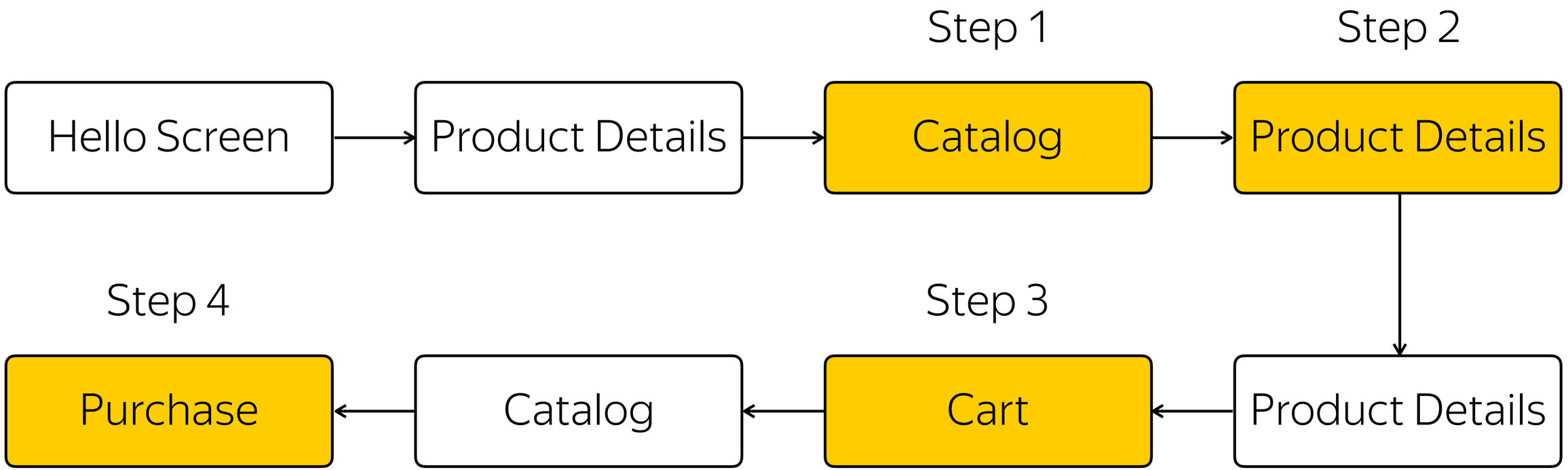
A funnel is the set of steps
a visitor needs to go through
before they can reach the conversion



Theory


Step 1
Step 2
Step 3
Step 4
Catalog
Product Details
Cart
Purchase

Practice


Step 1
Step 2
Hello Screen
Product Details
Catalog
Product Details
Step 4
Step 3
Purchase
Catalog
Cart
Product Details

Conversion rate = converted users / total users

Creating marketing funnels in ClickHouse

Data

DeviceID	EventName	EventTimestamp
6137721194148732620	Hello Screen	1506718845
6137721194148732620	Product Details	1506718845
6137721194148732620	Catalog	1506724083
6137721194148732620	Product Details	1506724093
6137721194148732620	Product Details	1506752725

Let’s start

Calculate a funnel of 3 events

1. Catalog
2. Product
3. Purchase

Straight-forward approach

```
sequenceMatch(pattern)
(time, cond1, cond2, …)
```

- Aggregate function
- Matches sequence of events to pattern
- Returns 1 or 0

First Approach

```
SELECT
  DeviceID,
  max((EventName = 'Catalog')) as step1_condition,
  sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,
  sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product'),
    (EventName = 'Purchase')) as step3_condition
FROM mobile.events_all
GROUP BY DeviceID
```

First Approach

```
SELECT
  DeviceID,
  max((EventName = 'Catalog')) as step1_condition,
  sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,
  sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product'),
    (EventName = 'Purchase')) as step3_condition
FROM mobile.events_all
GROUP BY DeviceID
```

First Approach

```
SELECT
  DeviceID,
  max((EventName = 'Catalog')) as step1_condition,
  sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,
  sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product'),
    (EventName = 'Purchase')) as step3_condition
FROM mobile.events_all
GROUP BY DeviceID
```

Data

DeviceID	step1_condition	step2_condition	step3_condition
6137721194148732620	1	1	0
6804205139374406489	1	0	0
5643727272341720158	1	1	1
227262599096481274 5	0	0	0
8307742578078641757	1	0	0

Sum everything up

```
SELECT
  sum(step1_condition) as step1_achieved,
  sum(step2_condition) as step2_achieved,
  sum(step3_condition) as step3_achieved
FROM (…)

┌─step1_achieved─┬─step2_achieved─┬─step3_achieved─┐
│          22757 │           9230 │           2353 │
└────────────────┴────────────────┴────────────────┘
```

Happy end?

Let’s add one more step

Calculate the funnel of 4 events

1. Catalog
2. Product
3. Cart
4. Purchase

Exception: Pattern application proves too difficult, exceeding max iterations

Arrays in ClickHouse

Arrays functionality

- `groupArray` – aggregation function to create arrays
- `arrayJoin` and `ARRAY JOIN` to split array into rows
- Higher order functions

Example: groupArray

```
SELECT
  DeviceID,
  groupArray(EventName) as event_names,
  groupArray(EventTimestamp) as event_timestamps
FROM mobile.events_all
GROUP BY DeviceID
```

```
┌─────────────DeviceID─┬─event_names──────────────────────────────────────┬─event_timestamps──────────────────────┐
│ 61377211941487326 20 │ ['Hello Screen']                                 │ [1506718845]                          │
│ 61377211941487326 20 │ ['Cart','Purchase']                              │ [1506762727,1506762735]               │
│ 68042051393744064 89 │ ['Catalog','Product Details','Product Details']  │ [1506762477,1506762523,1506762524]    │
└──────────────────────┴──────────────────────────────────────────────────┴───────────────────────────────────────┘
```

Example: arrayJoin

```
SELECT *, arrayJoin(Products) AS Product
FROM (
  SELECT
    ['coffee', 'cookie'] AS Products,
    toUInt64(now()) AS EventTimestamp,
    'Purchase' AS EventName
)
```

```
┌─Products─────────────┬─EventTimestamp─┬─EventName─┬─Product─┐
│ ['coffee','cookie'] │     1506805484 │ Purchase  │ coffee  │
│ ['coffee','cookie'] │     1506805484 │ Purchase  │ cookie  │
└──────────────────────┴────────────────┴───────────┴─────────┘
```

Example: `arrayJoin` with 2 arrays

```
SELECT *, arrayJoin(Products) AS Product, arrayJoin(ProductPrices) AS Price
FROM (
  SELECT
    ['coffee', 'cookie'] AS Products,
    [5, 3] AS ProductPrices,
    toUInt64(now()) AS EventTimestamp,
    'Purchase' AS EventName
)
```

```
┌─Products─────────────┬─ProductPrices─┬─EventTimestamp─┬─EventName─┬─Product─┬─Price─┐
│ ['coffee','cookie'] │ [5,3]         │     1506805729 │ Purchase  │ coffee  │     5 │
│ ['coffee','cookie'] │ [5,3]         │     1506805729 │ Purchase  │ coffee  │     3 │
│ ['coffee','cookie'] │ [5,3]         │     1506805729 │ Purchase  │ cookie  │     5 │
│ ['coffee','cookie'] │ [5,3]         │     1506805729 │ Purchase  │ cookie  │     3 │
└─────────────────────┴───────────────┴────────────────┴───────────┴─────────┴───────┘
```

Example: ARRAY JOIN

```
SELECT *, Product, Price
FROM (
  SELECT
    ['coffee', 'cookie'] AS Products,
    [5, 3] AS ProductPrices,
    toUInt64(now()) AS EventTimestamp,
    'Purchase' AS EventName)
ARRAY JOIN Products AS Product, ProductPrices as Price
```

```
┌─Products─────────────┬─ProductPrices─┬─EventTimestamp─┬─EventName─┬─Product─┬─Price─┐
│ ['coffee','cookie'] │ [5,3]         │     1506805976 │ Purchase  │ coffee  │     5 │
│ ['coffee','cookie'] │ [5,3]         │     1506805976 │ Purchase  │ cookie  │     3 │
└──────────────────────┴───────────────┴────────────────┴───────────┴─────────┴───────┘
```

Higher Order Functions

- Functions
  - `arrayMap`
  - `arrayFilter`
  - `arrayFirst`
  - etc.
- Lambda functions
  - `x -> x = 'Order Opened'`
  - `x, y -> pow(x, 2) + pow(y, 2)`

Example: arrayFilter

```
SELECT
  groupArray(EventName) as names,
  arrayEnumerate(EventNames) as indexes,
  arrayFilter(x -> x != 'Catalog', names) as names_without_catalog,
  arrayFilter(x, y -> y != 1, names, indexes) as names_without_first
FROM mobile.events_all
GROUP BY DeviceID
```

```
┌─names─────────────────────────────┬─indexes─┬─names_without_catalog─┬─names_without_first─┐
│ ['Catalog','Product','Catalog'] │ [1,2,3,4] │ ['Product']           │ ['Product','Catalog'] │
└─────────────────────────────────┴─────────┴───────────────────────┴─────────────────────┘
```

Getting back to the funnel calculation

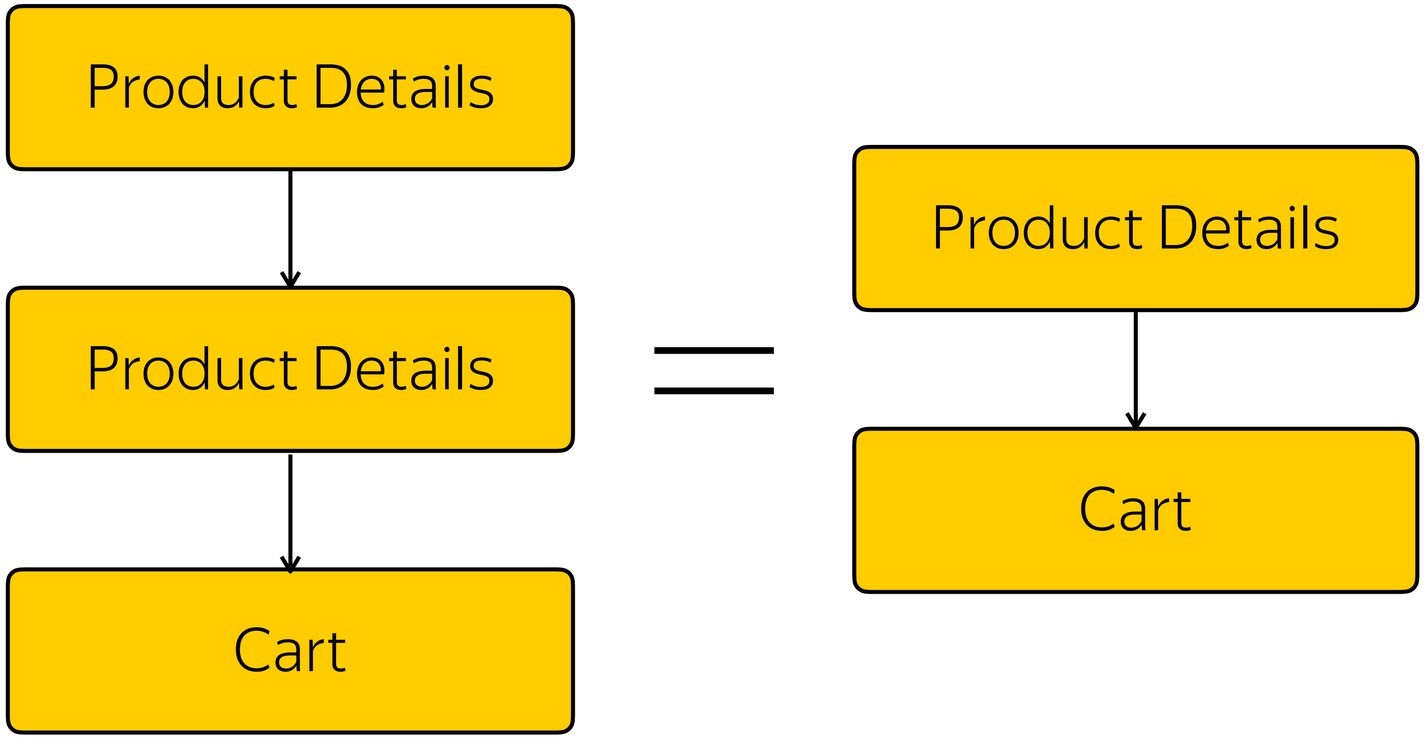
Where are we now?

- The straight-forward approach doesn’t work
- What’s next?

Can we make sequence processing easier?

Simplify procession of event sequences

Replace the same event sequences with just one event



Revisal: Straight-forward approach

```
SELECT
  DeviceID,
  max((EventName = 'Catalog')) as step1_condition,
  sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,
  sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),
    (EventName = 'Catalog'), (EventName = 'Product'),
    (EventName = 'Purchase')) as step3_condition
FROM mobile.events_all
GROUP BY DeviceID
```

Deduplicate event sequences

```
SELECT
  groupArray(EventName) as events,
  groupArray(EventTimestamp) as times,
  arrayEnumerate(events) as events_index,
  arrayFilter(name, index -> (index = 1) OR (events[index] != events[index - 1]),
    events, events_index) as events_filt,
  arrayFilter(time, index -> (index = 1) OR (events[index] != events[index - 1]),
    times, events_index) as times_filt
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
GROUP BY device_id
```

Make the initial table

```
SELECT
  device_id as DeviceID,
  EventName,
  EventTimestamp
FROM (SELECT …)
ARRAY JOIN
  events_filt as EventName,
  times_filt as EventTimestamp
```

It’s working!

… for 4 step funnel

The root cause

Calculation difficulty grows exponentially when we add steps due to `.*` (sequences of any events)

There's always a limit for current realization of `sequenceMatch`.

Calculating from scratch

What do we need?

The second step condition

- User viewed catalog at least once
- User viewed product at least once
- Time of first catalog view <= time of last product view

Algorithm

Step 1

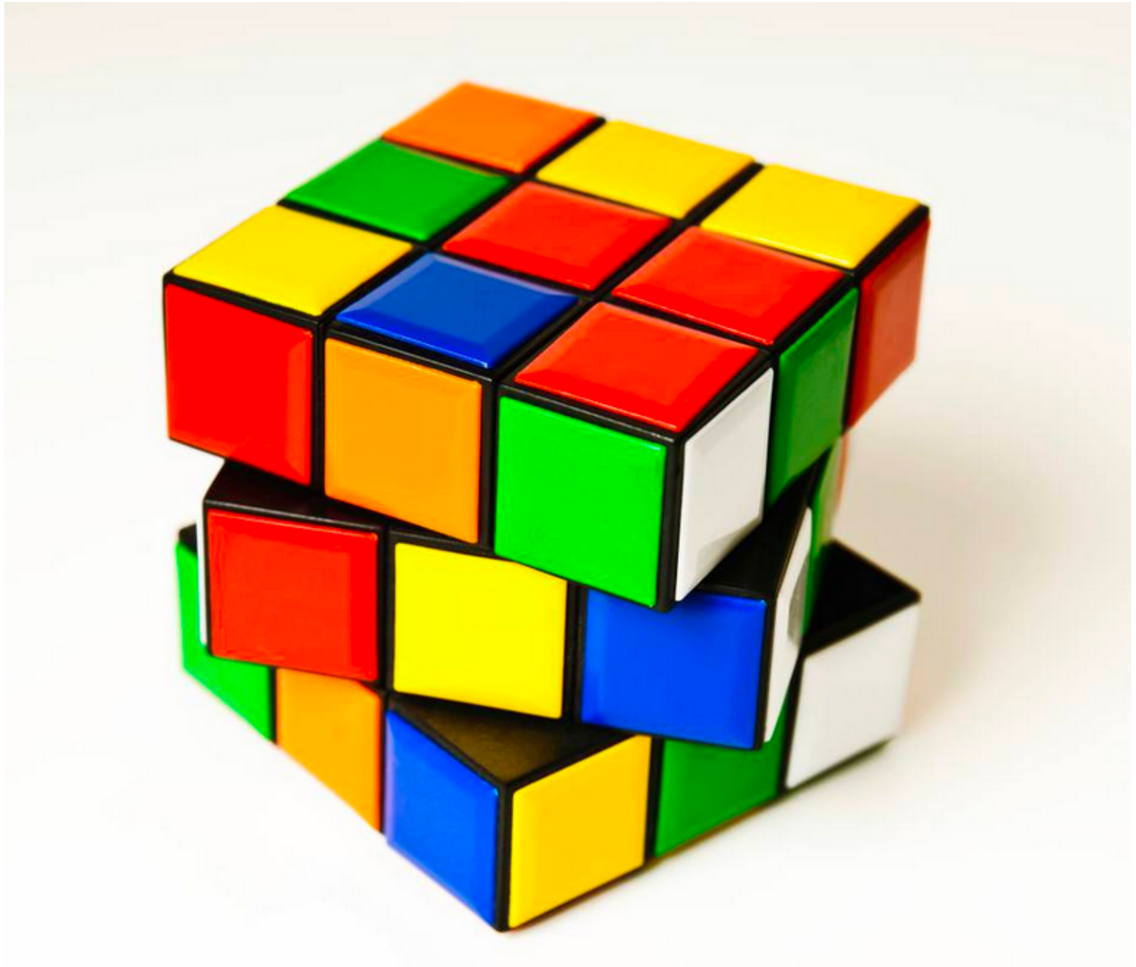
› calculate the first time of catalog view

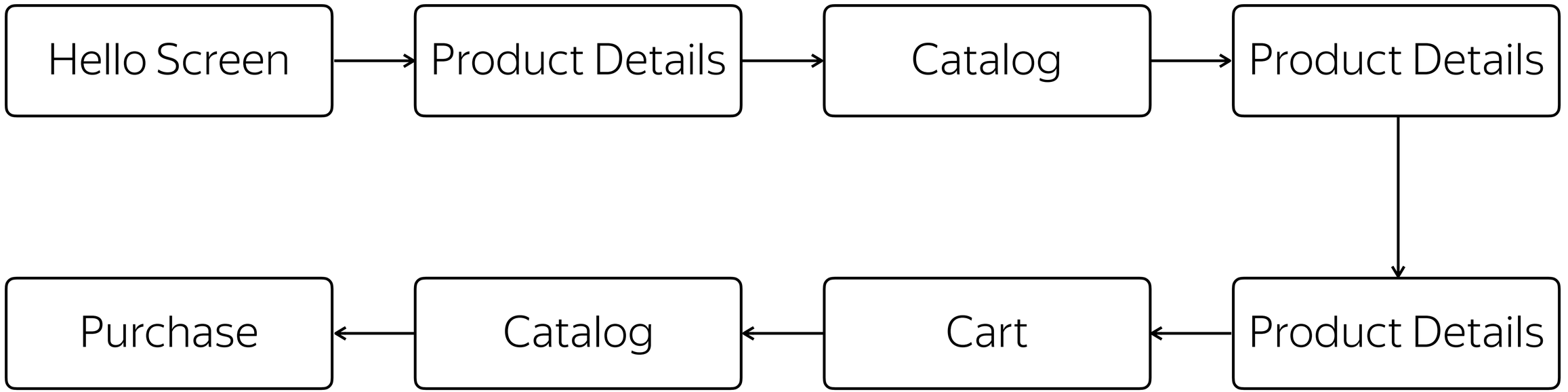
Step 2

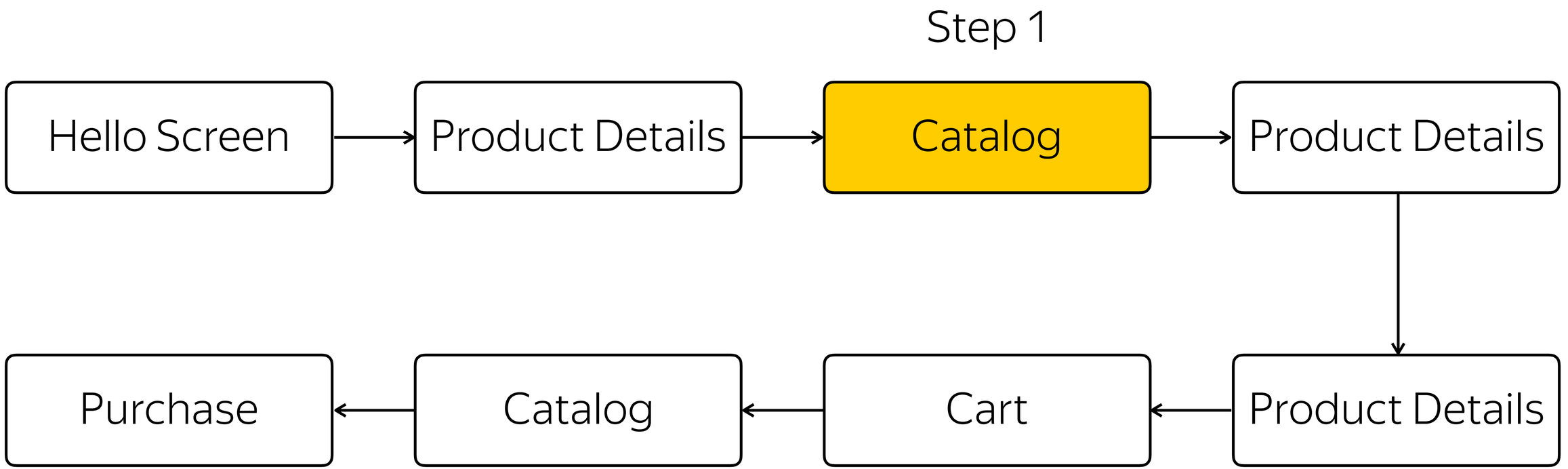
› calculate the first time of product view after catalog view

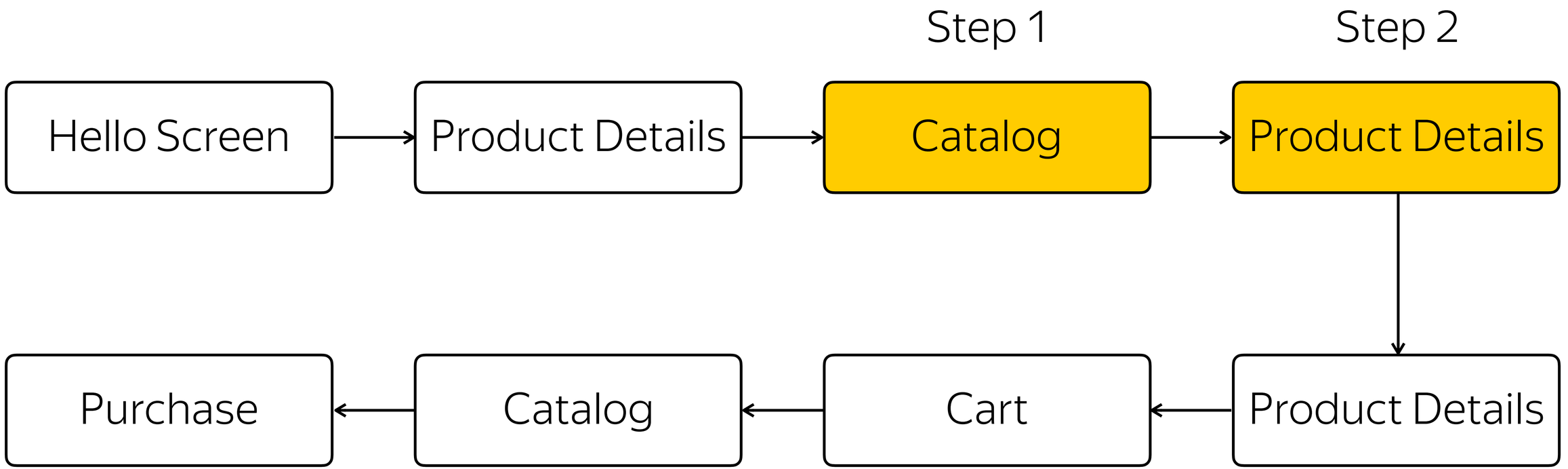
Step 3:

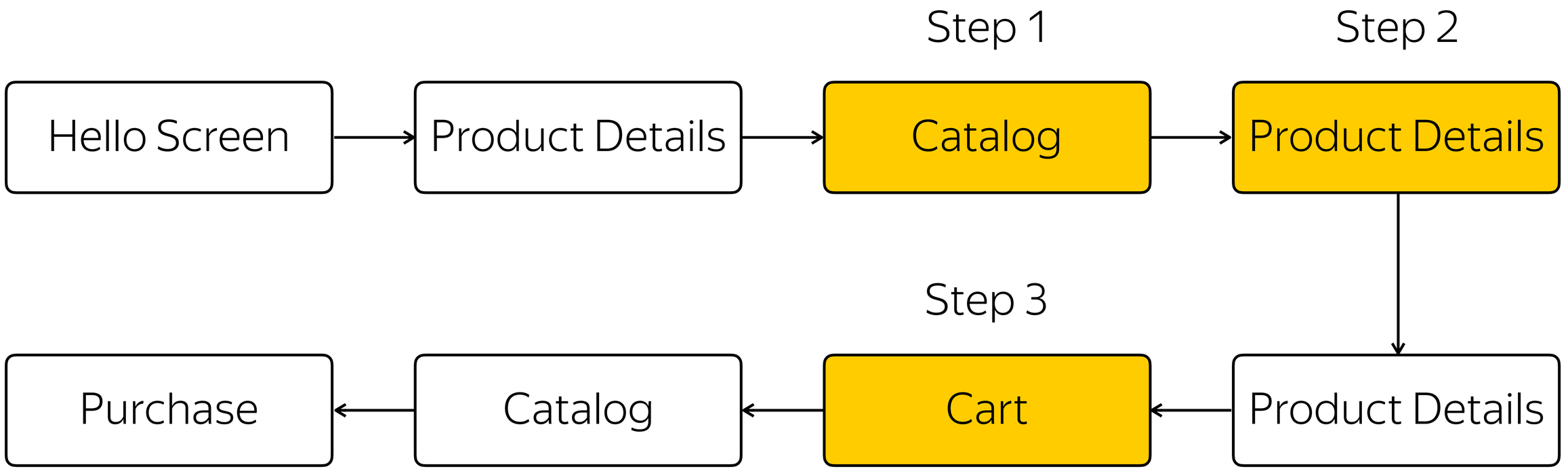
› calculate the first time of cart after catalog view, etc.

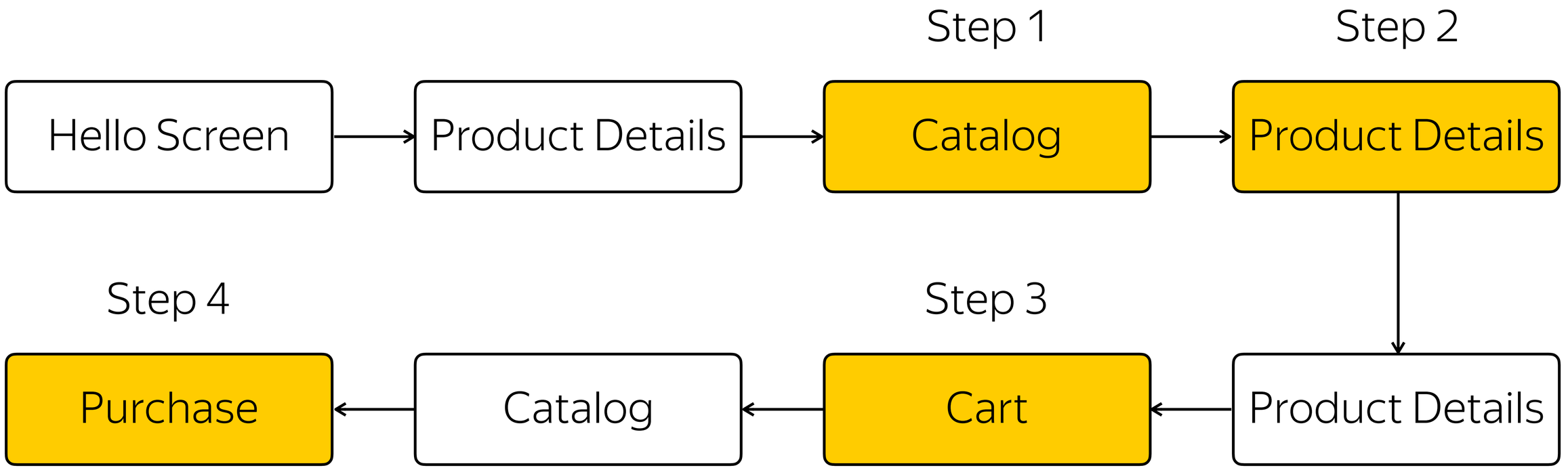



Hello Screen
Product Details
Catalog
Product Details
Product Details
Cart
Catalog
Purchase


Step 1
Hello Screen
Product Details
Catalog
Product Details
Product Details
Cart
Catalog
Purchase


Step 1
Step 2
Hello Screen
Product Details
Catalog
Product Details
Purchase
Catalog
Cart
Product Details


Step 1
Step 2
Hello Screen
Product Details
Catalog
Product Details
Step 3
Purchase
Catalog
Cart
Product Details


Step 1
Step 2
Hello Screen
Product Details
Catalog
Product Details
Step 4
Step 3
Purchase
Catalog
Cart
Product Details

Calculating achievement times

```
SELECT
  groupArray(EventName) as events,
  groupArray(EventTimestamp) as events_times,
  arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,
  arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)
    AND (step1_time != 0), events_times, events)[1] as step2_time,
  arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)
    AND (step2_time != 0), events_times, events)[1] as step3_time
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

Calculating achievement times

```
SELECT
  groupArray(EventName) as events,
  groupArray(EventTimestamp) as events_times,
  arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,
  arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)
    AND (step1_time != 0), events_times, events)[1] as step2_time,
  arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)
    AND (step2_time != 0), events_times, events)[1] as step3_time
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

Calculating achievement times

```
SELECT
  groupArray(EventName) as events,
  groupArray(EventTimestamp) as events_times,
  arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,
  arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)
    AND (step1_time != 0), events_times, events)[1] as step2_time,
  arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)
    AND (step2_time != 0), events_times, events)[1] as step3_time
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

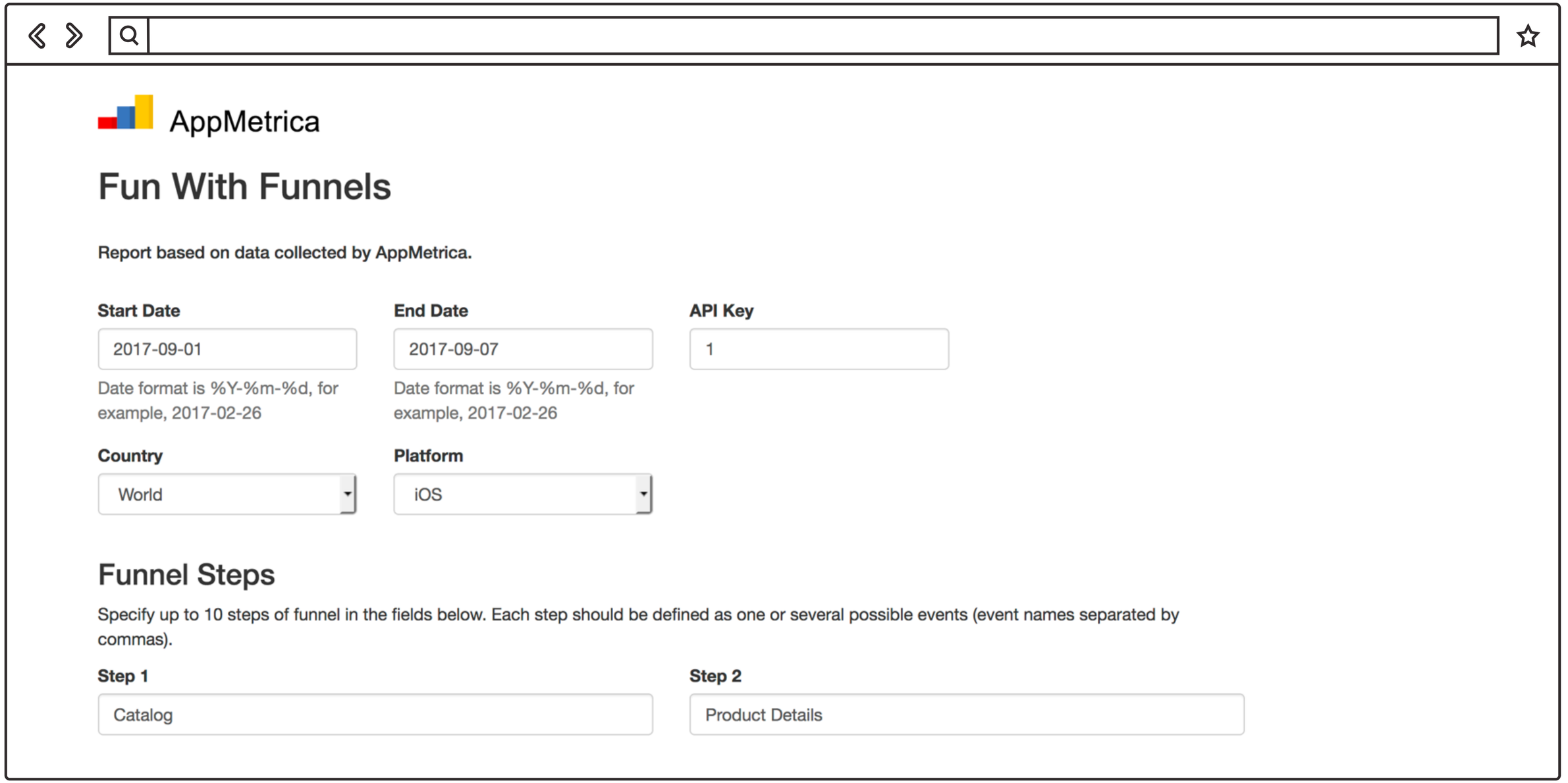
Marketing funnel

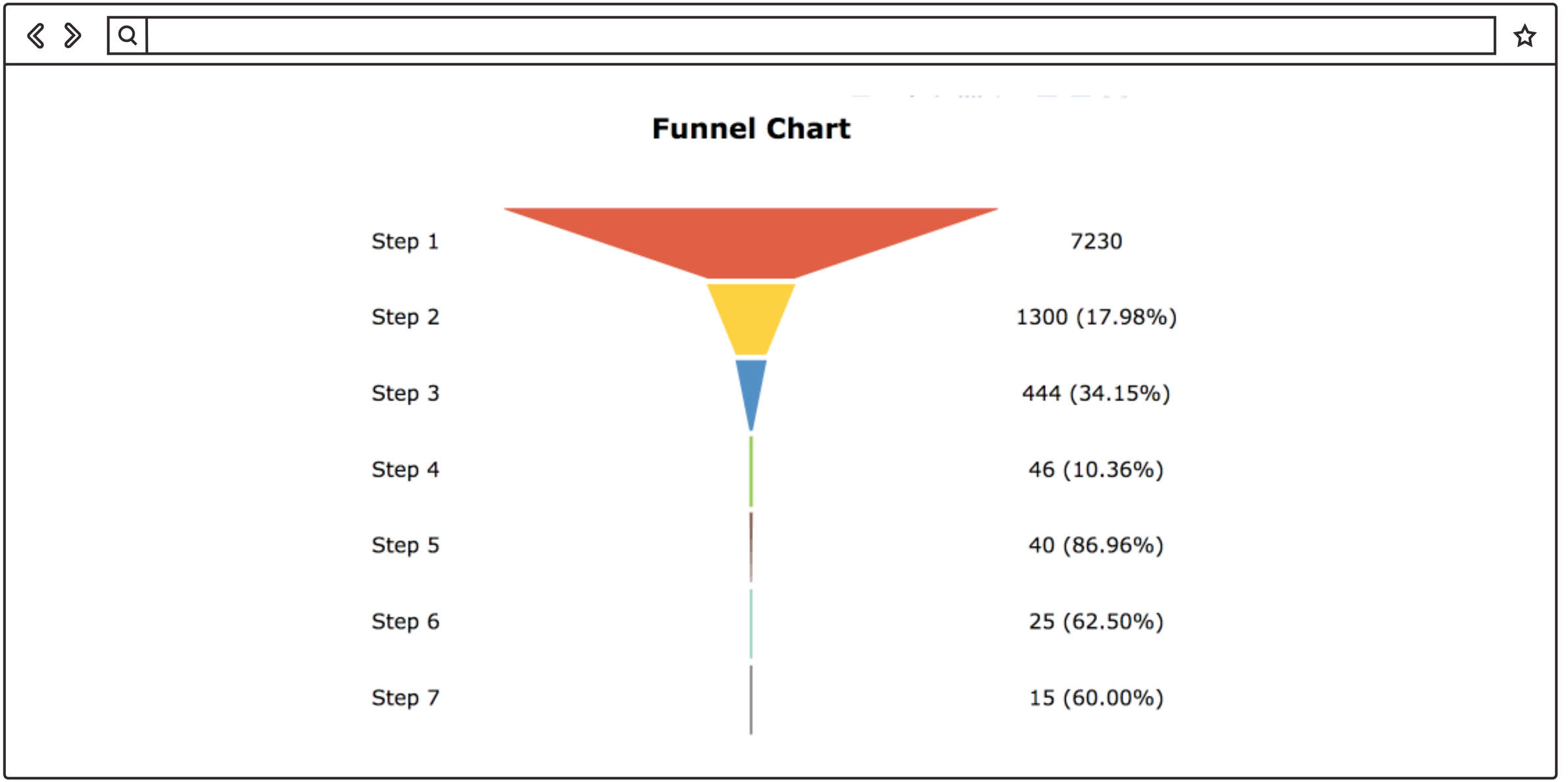
```
SELECT
  sum(step1_time != 0) as step1_achieved,
  sum(step2_time != 0) as step2_achieved,
  sum(step3_time != 0) as step3_achieved,
  sum(step4_time != 0) as step4_achieved
FROM (…)

┌─step1_achieved─┬─step2_achieved─┬─step3_achieved─┐
│          22757 │           9230 │           2353 │
└────────────────┴────────────────┴────────────────┘
```

It does work :)

Lessons learned

- Linear calculation difficulty growth
- It won't work for patterns like `(?1).*(?2).*(?1)`


AppMetrica
Fun With Funnels
Report based on data collected by AppMetrica.
Start Date
2017-09-01
Date format is %Y-%m-%d, for example, 2017-02-26
End Date
2017-09-07
Date format is %Y-%m-%d, for example, 2017-02-26
API Key
1
Country
World
Platform
iOS
Funnel Steps
Specify up to 10 steps of funnel in the fields below. Each step should be defined as one or several possible events (event names separated by commas).
Step 1
Catalog
Step 2
Product Details


Funnel Chart
Step 1
7230
Step 2
1300 (17.98%)
Step 3
444 (34.15%)
Step 4
46 (10.36%)
Step 5
40 (86.96%)
Step 6
25 (62.50%)
Step 7
15 (60.00%)

Technologies behind the project

- Docker container
- ClickHouse for storing and processing data
- Python as a main language
- Flask for web-server
- Plot.ly library as a visualization tool

https://hub.docker.com/r/yandex/appmetrica-funnels/

Wrapping it up

- You can calculate complex metrics with arrays and higher order functions
- Don’t give up and think differently
- You can make CH better even if you don’t know C++

Thank you!
Any questions?

Contacts


clickhouse.yandex


Google Group groups.google.com/forum/#!forum/clickhouse


clickhouse-feedback@yandex-team.ru


@clickhouse_en


github.com/yandex/ClickHouse


@ClickHouseDB