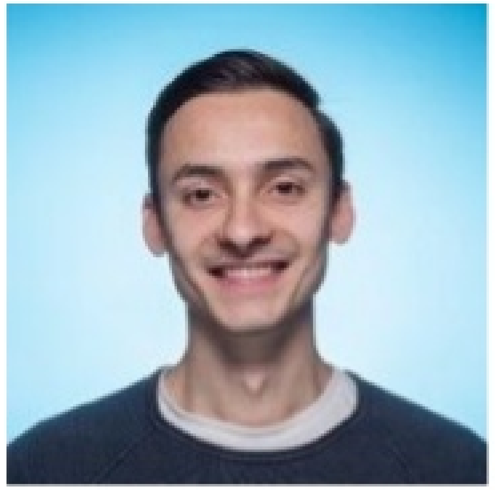
SPARK+AI
SUMMIT EUROPE

Attribution Done Right

Thiago Rigo, GetYourGuide

#SAISEnt13

Who am I?



Software engineer for the past 7 years, last 3 years focused on data engineering.

Data warehousing, data quality, and event processing.

Data Engineer, Data Platform

Agenda

1 Intro to GetYourGuide

2 What is Attribution?

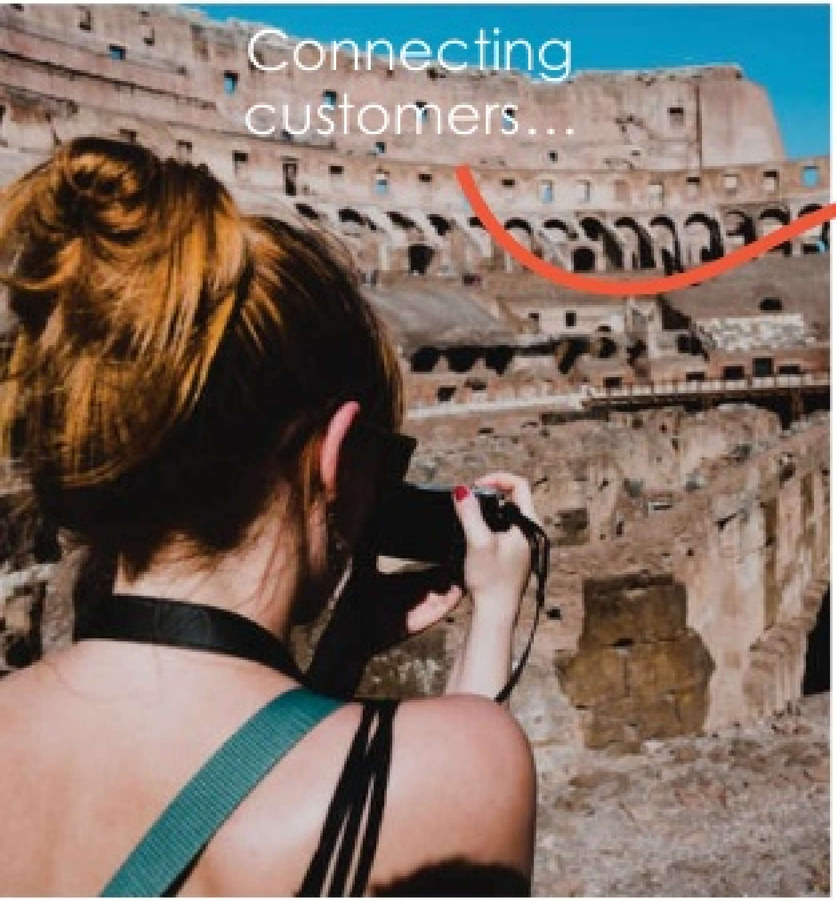
3 Attribution at GetYourGuide

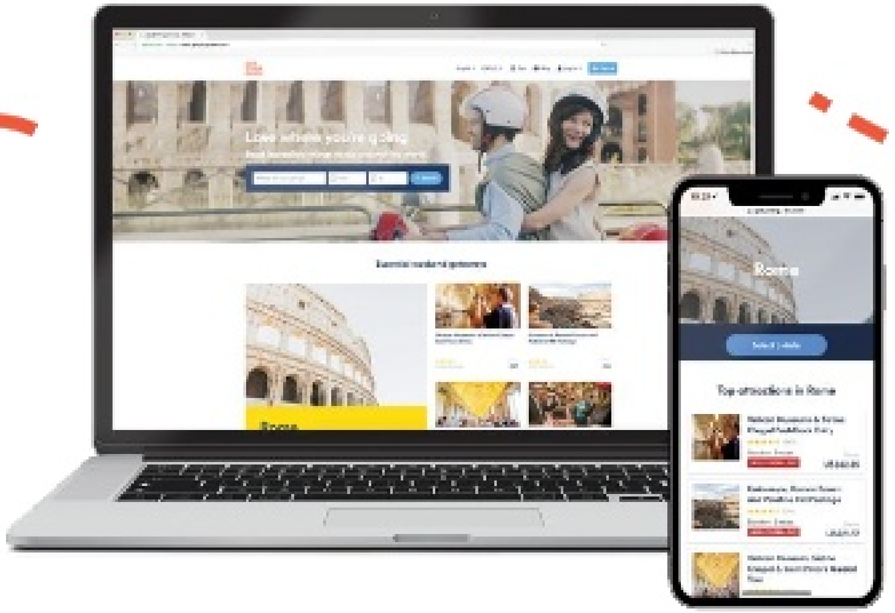
#SAISEnt13

Intro to GetYourGuide

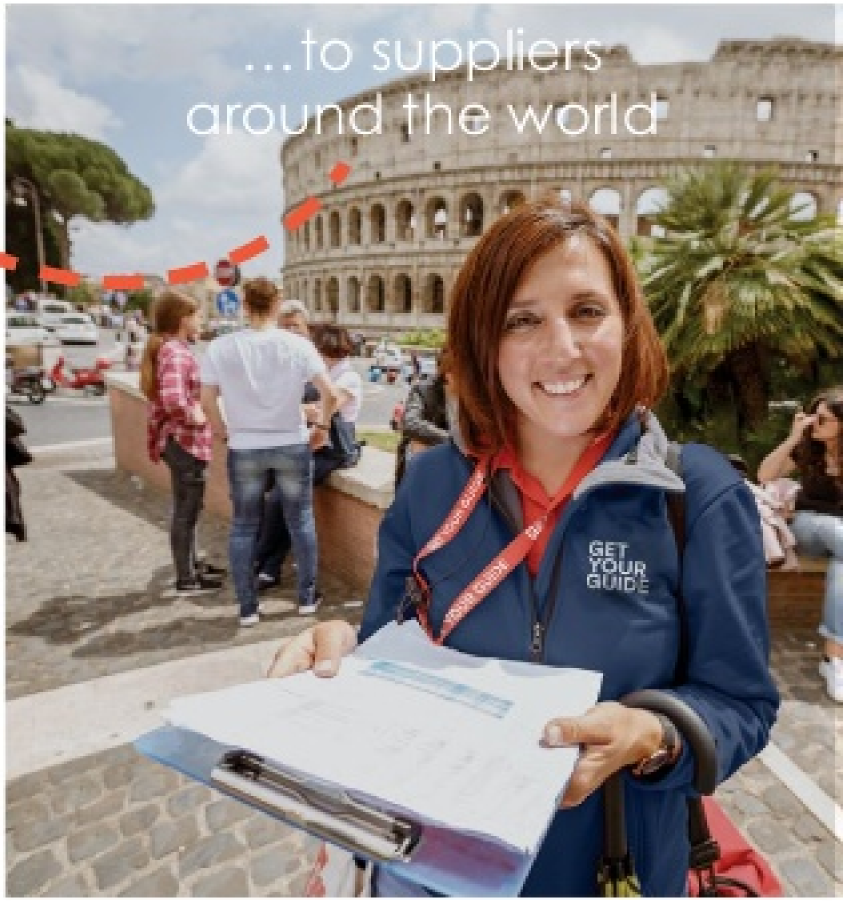
#SAISEnt13

We're the world's biggest marketplace for travel activities





GET YOUR GUIDE

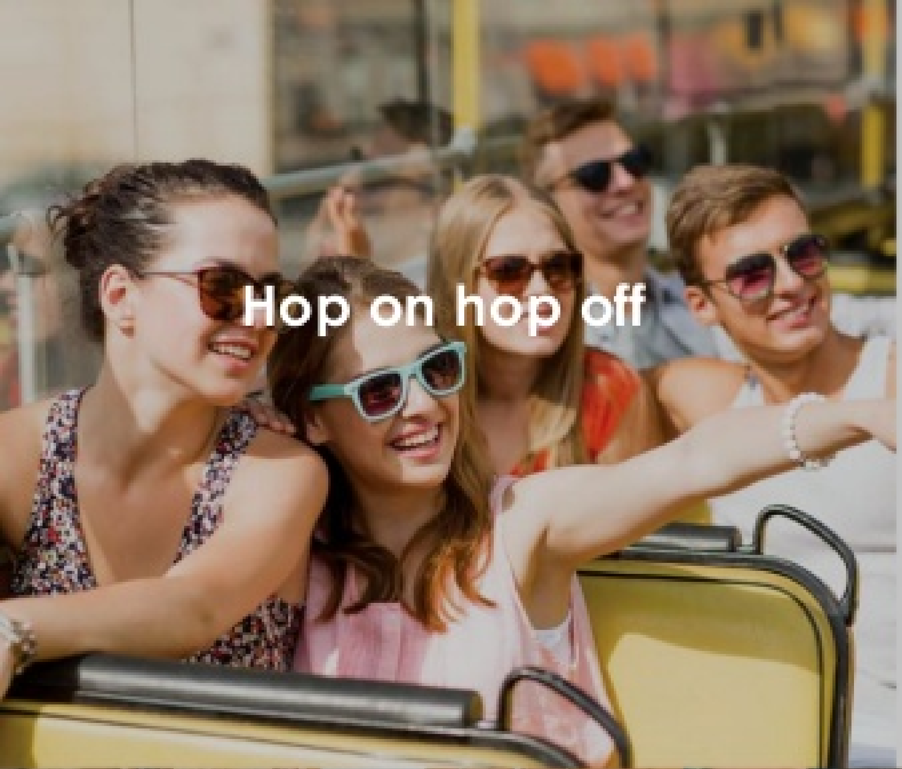


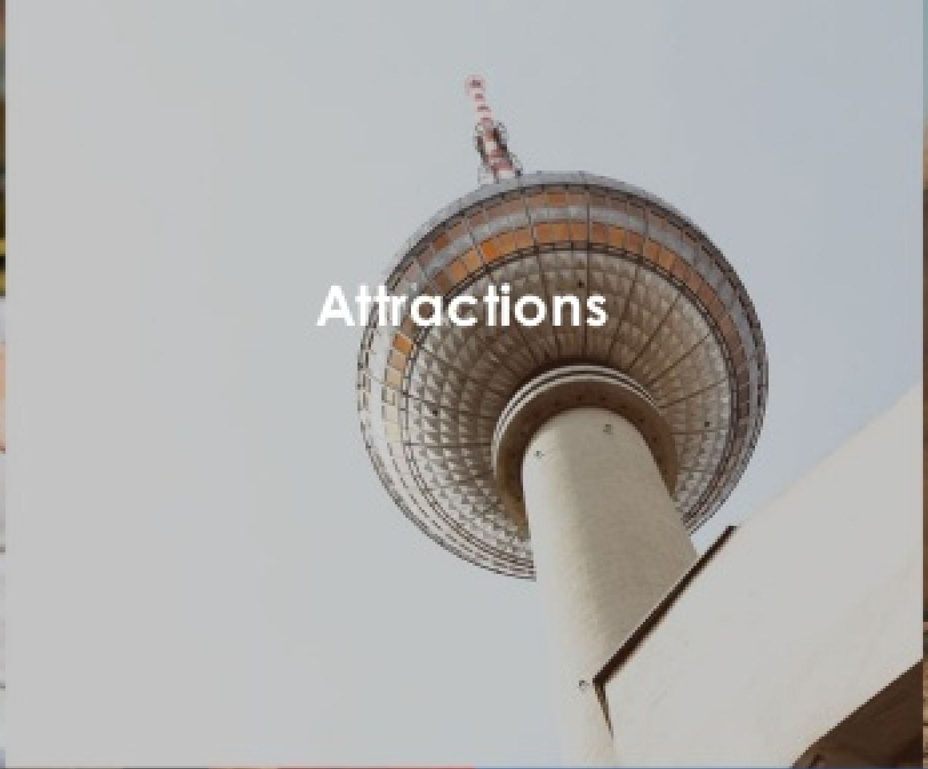
Millions of travelers use GetYourGuide every year

We facilitate the transaction

We offer more than 40,000 activities worldwide

#SAISEnt13


Hop on hop off


Attractions

Outdoor activities

Walking tours

Cooking classes

Amusement parks

Transfers

Our goal is to make sure "you love where you are going"

Discovery

With over 40,000 activities, we'll make your experience incredible wherever your travels take you.

Trust

Verified reviews and 24/7 customer service mean you can rest easy and enjoy the journey.

Transaction

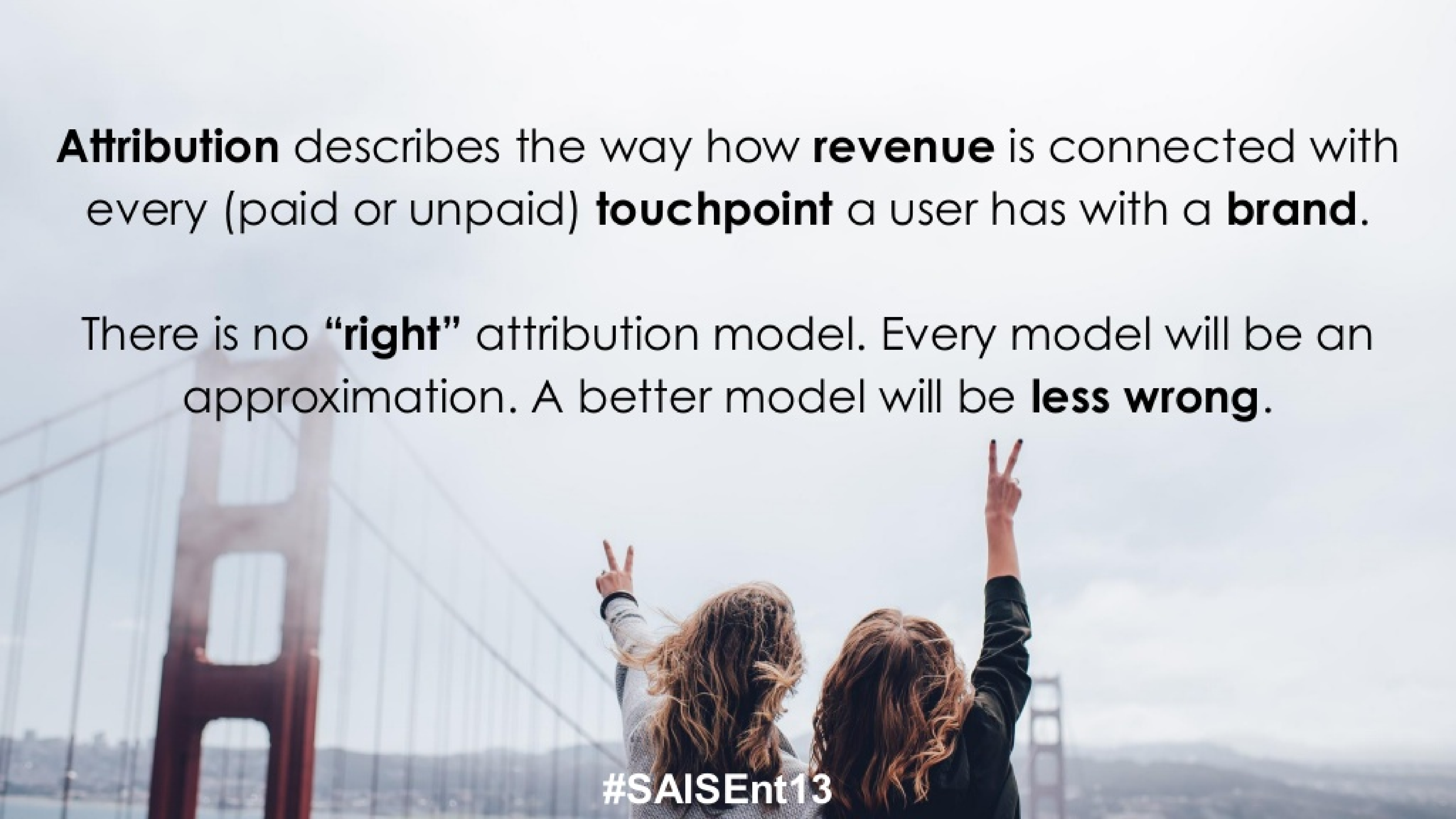
Book up to the last minute with the best price guaranteed. We've got you covered in 16 currencies.

Some Data

- **40k activities**
- Customers from **159 countries**
- Localized in **18 languages**
- **18 million** monthly active users
- **450** people in **13 offices** around the globe
- **175 million USD** raised in funding
- **50+ nationalities** working here

What is Attribution?

#SAISEnt13



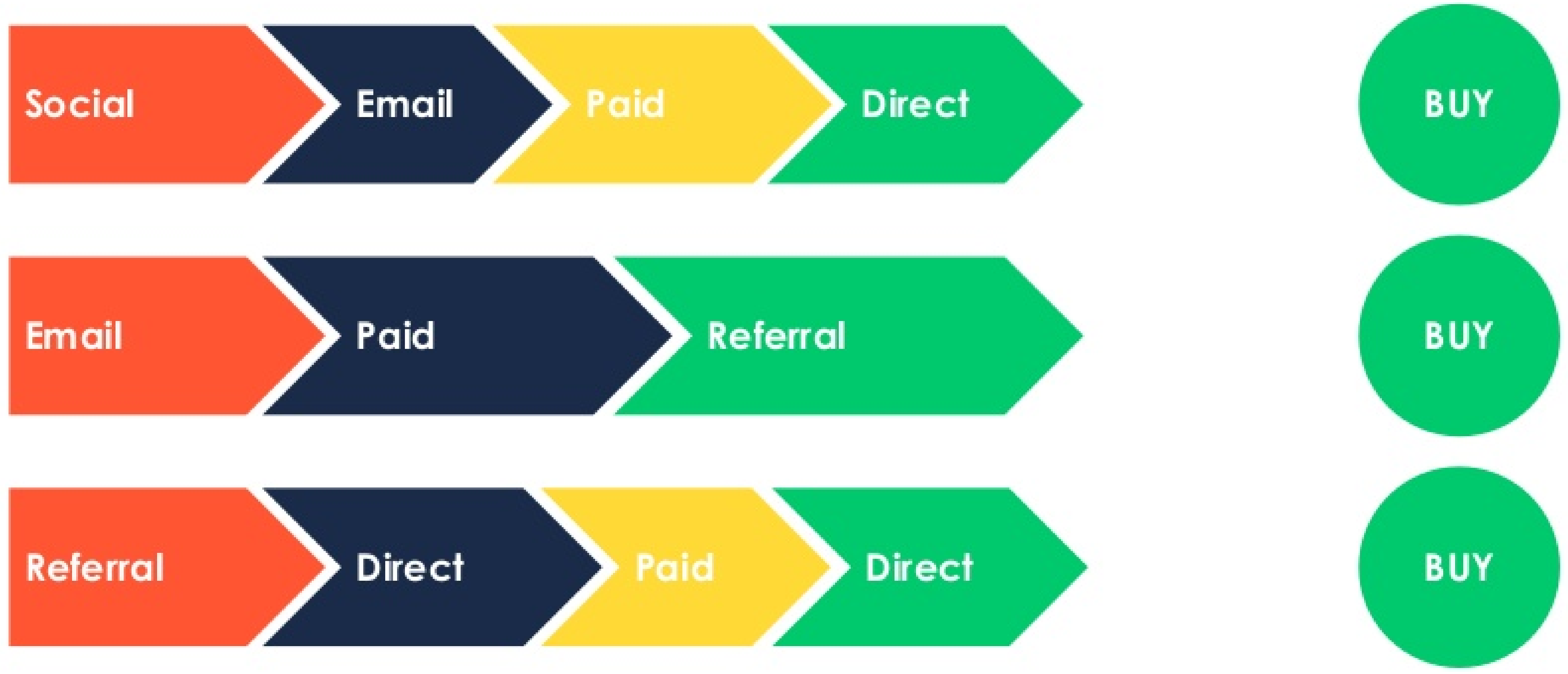
Attribution describes the way how **revenue** is connected with every (paid or unpaid) **touchpoint** a user has with a **brand.**

There is no **“right”** attribution model. Every model will be an approximation. A better model will be **less wrong**.

Why is Attribution Important?

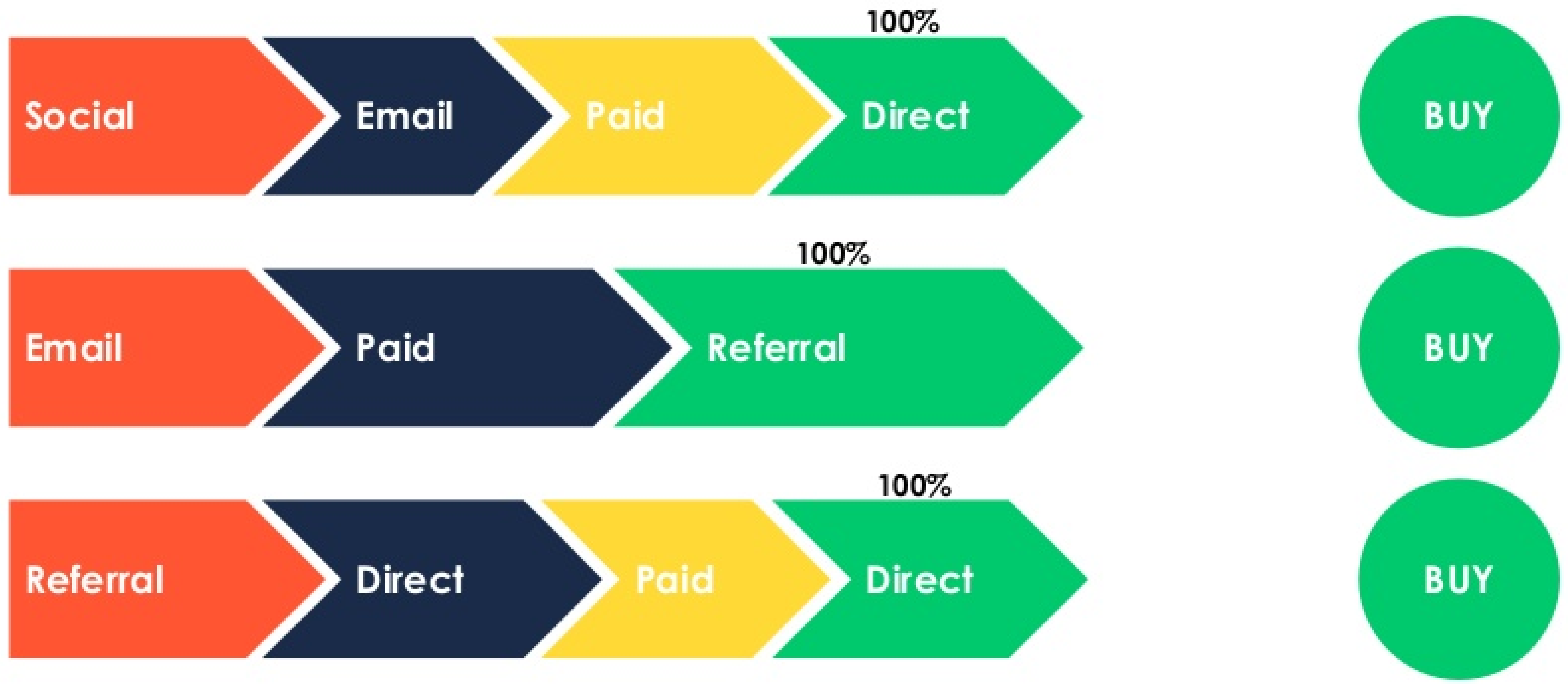
By identifying **touchpoints** that bring more **value**, we can allocate **marketing spend** better and generate **more revenue** at the **same cost**.

Conversion Path

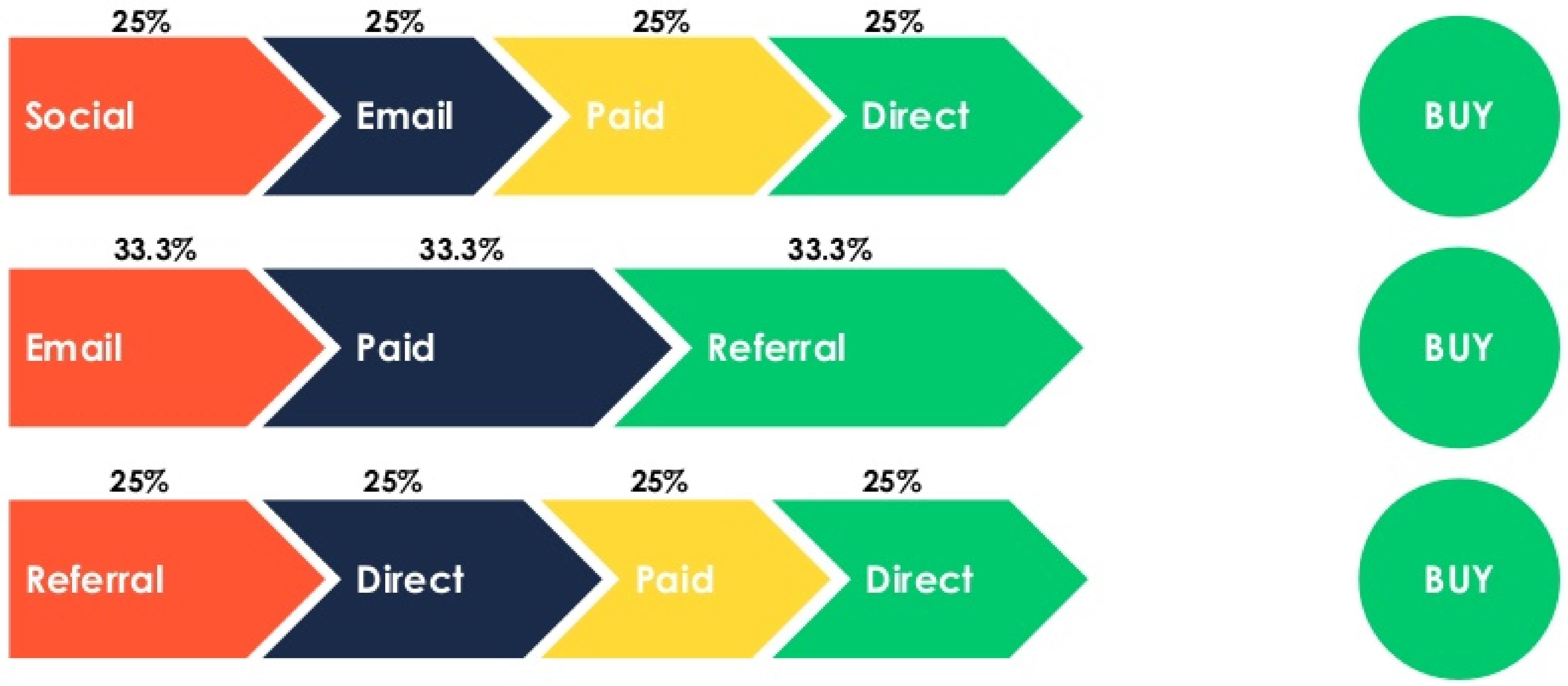


#SAISEnt13

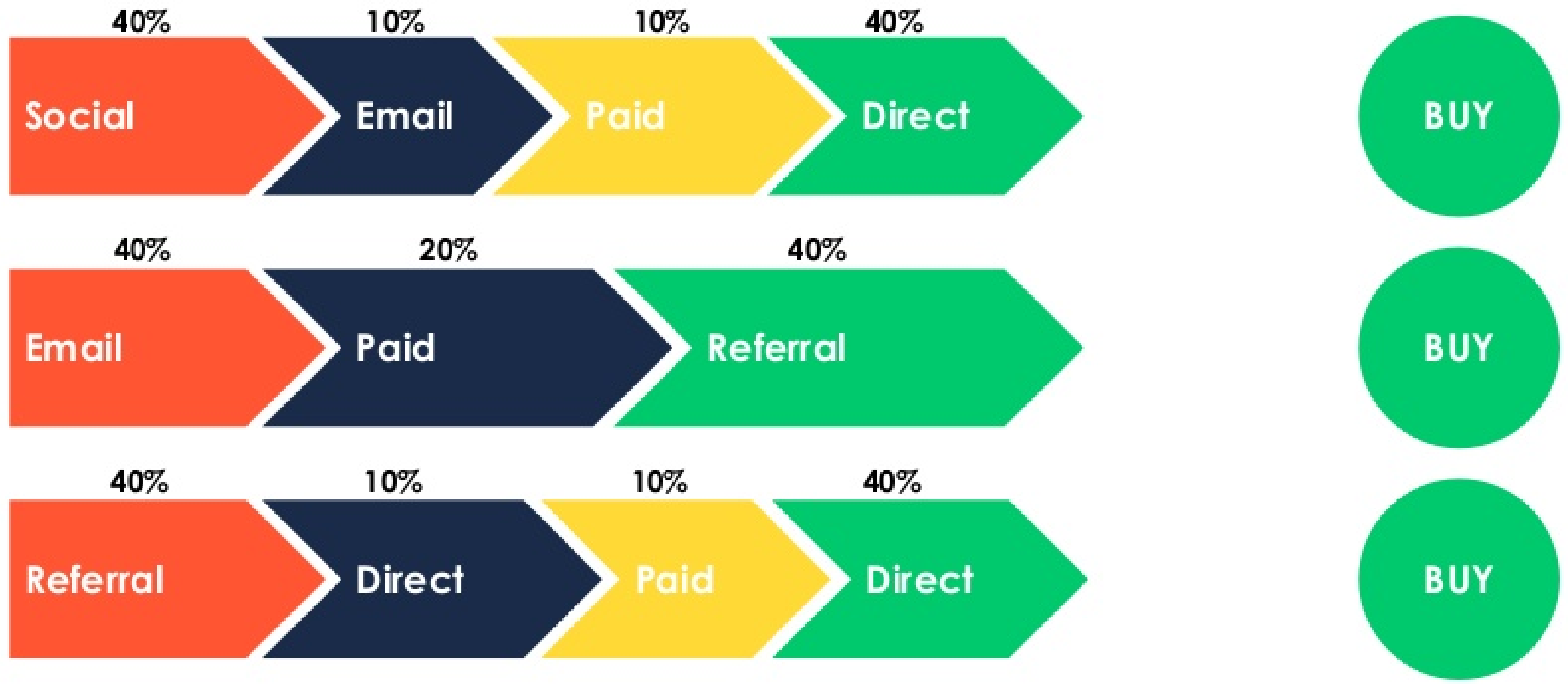
Last Click Attribution (Single Touch)



Linear Attribution (Multi Touch)



Position Based Attribution (Multi Touch)



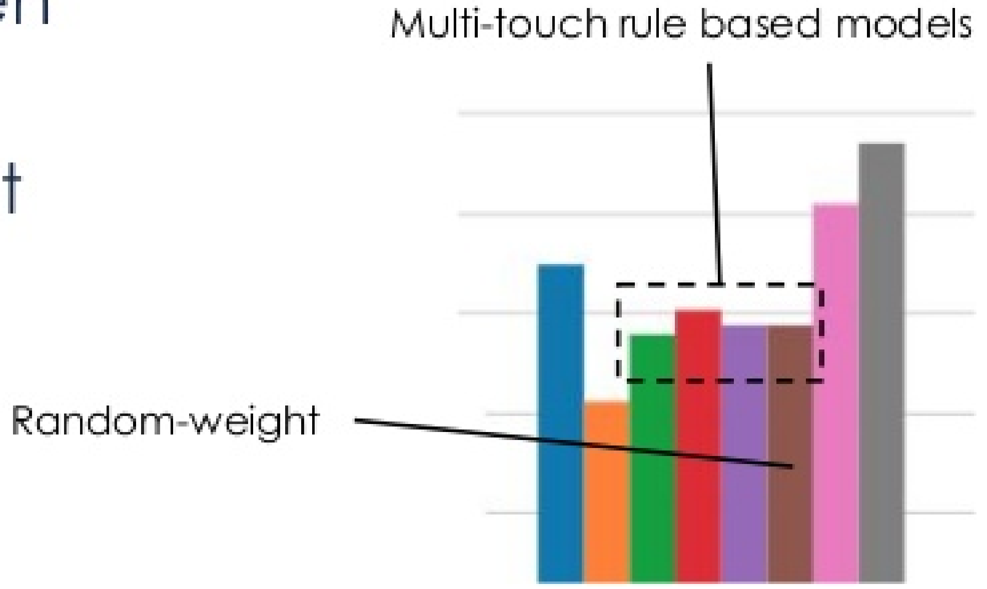
We use Position Based Attribution

Why?

- Simple rule based model
- Easy to understand and compare in third party tools

And...

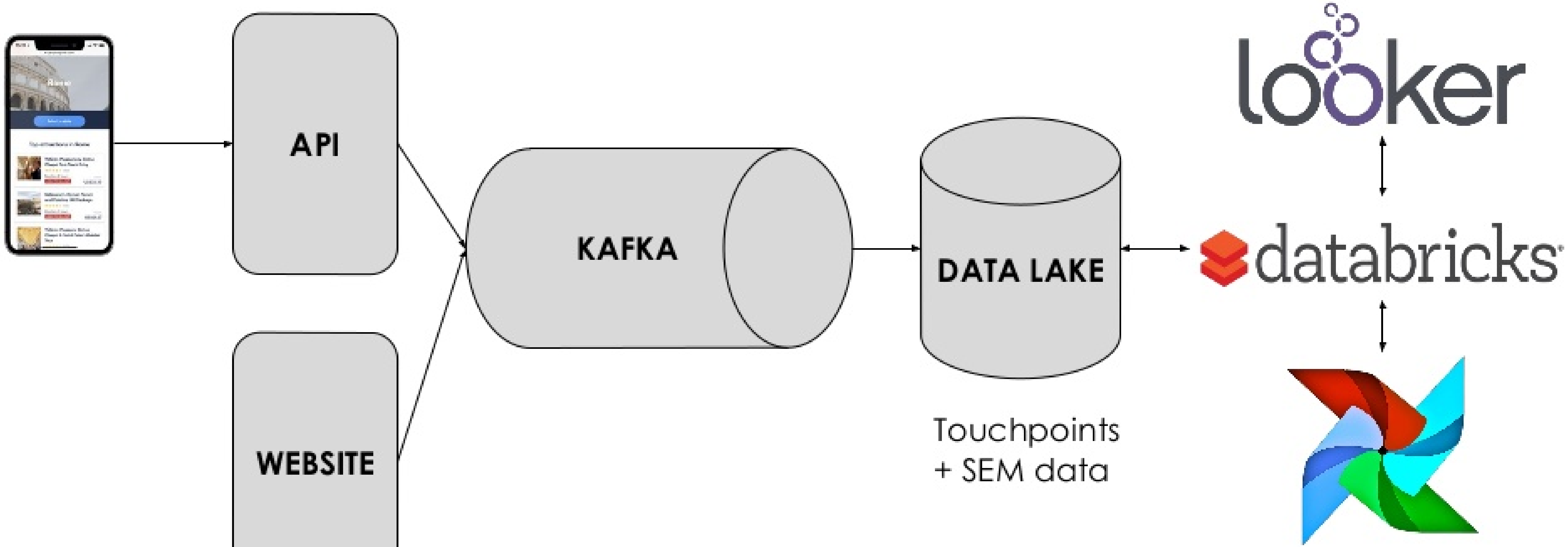
- We found that the key difference is between **one-touch** and **multi-touch models**
- **Random weight**, does not show a significant difference to other multi-touch models



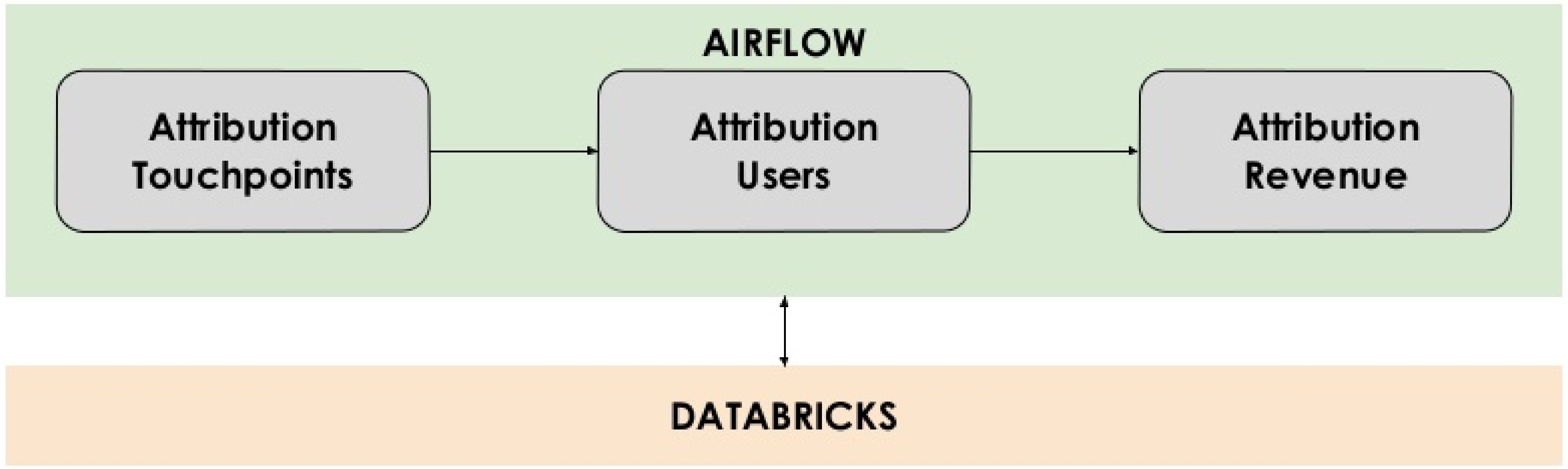
Attribution at GetYourGuide

#SAISEnt13

Architecture



Orchestration



Orchestration

```
cluster_config = cluster_auto_scale_config(min_workers=2, max_workers=8, enable_delta=True)

touchpoints = create_notebook_operator(
    task_id='touchpoints', notebook_path='/Attribution/Touchpoints', dag=dag, cluster_config=cluster_config
)

users = create_notebook_operator(
    task_id='users', notebook_path='/Attribution/Users', dag=dag, cluster_config=cluster_config
)

revenue = create_notebook_operator(
    task_id='revenue', notebook_path='/Attribution/Revenue', dag=dag, cluster_config=cluster_config
)

touchpoints >> users >> revenue
```

Touchpoints

Touchpoint events, in our case called **AttributionTracking**, are fired every time a user lands on one of our websites or one of our native apps.

They contain user and channel data which describe that event.

```
{
  "user": {
    "visitor_id": "P1AO52A3LJI3PK0D2CG8U36BWOPFVQPD"
  },
  "attribution": {
    "partner_campaign": "BING",
    "utm_campaign": "south africa:68|core|all|fr",
    "utm_medium": "paid_search",
    "utm_source": "bing",
    "utm_term": "visite cap town",
    "referral_visitor_id": "HVBTVOH34LOHUIMLNJRFC2G5QKGAV8ZO"
  },
  "sem_parameters": {
    "campaign_id": 285631588,
    "adgroup_id": 1249045283929190,
    "target_id": "kwd-78065385628556:loc-66",
    "ad_id": 78065343912280
  }
}
```

#SAISEnt13

Channel Assignment

```
def isPaidSocialBrand(event: AttributionTracking): Boolean = {
  event.attribution.utm_medium.contains(PaidSocialBrand.toString)
}

def isSocial(event: AttributionTracking): Boolean = {
  event.attribution.utm_medium.contains(Social.toString) ||
  (
    event.header.referrer.isDefined &&
    isFromSocial(event.header.referrer)
  )
}
```

Users

A user can generate a touchpoint in any device, so that means we need to be able to connect different visitor IDs.

Web to App, or Email.

#SAISEnt13

Web to App:
gyg://tickets/<ticket_id>?visitor_id=RFTV0QAF0 8PWVO12W4SM2IQ9J4A4P95T

Email:
https://www.getyourguide.com/booking/<booking_id>?visitor_id=EG86TZ1052WRBI6E2D9IAL8X B7OV1BMX

Users Table

Column Name	Type
visitor_a	String
visitor_b	String
update_timestamp	Timestamp

Users Table

visitor_a	visitor_b	update_timestamp
a	a	2018-08-01 10:00:00
a	b	2018-08-02 10:00:00
b	b	2018-08-02 10:00:00
b	a	2018-08-02 10:00:00

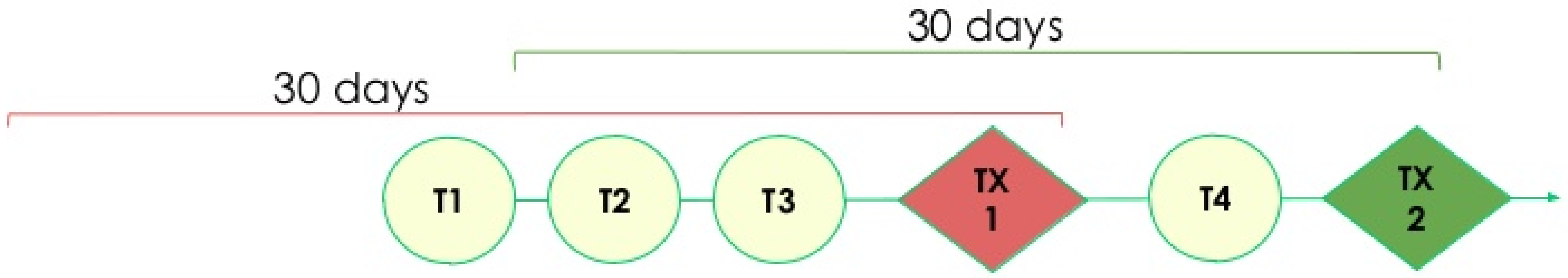
Users Table

```scala
case class VisitorMapping(visitor_id_a: String, visitor_id_b: String, update_timestamp: java.sql.Timestamp)

def addReversed(df: Dataset[VisitorMapping]): Dataset[VisitorMapping] = {
  df
  .union(
    df.select($"visitor_id_b" as "visitor_id_a", $"visitor_id_a" as "visitor_id_b", $"update_timestamp").as[VisitorMapping]
  )
  .distinct.as[VisitorMapping]
}

def addIdentity(df: Dataset[VisitorMapping]): Dataset[VisitorMapping] = {
  val identity = df.groupBy("visitor_id_a")
    .agg(min($"update_timestamp") as "update_timestamp")
    .select($"visitor_id_a" as "visitor_id_a", $"visitor_id_a" as "visitor_id_b", $"update_timestamp")
    .as[VisitorMapping]

  df.union(identity)
}
```

Revenue

Once we have all touchpoints and a mapping of visitor IDs, we can now attribute revenue to these touchpoints.

Revenue Table



1. All revenue on transaction level
2. All touchpoint info for a given transaction
3. Different attribution models (first click, last click, linear, **position-based** and time decay)

Touchpoint ID	Transaction ID	Timestamp Touchpoint	Timestamp Transaction	Position Based Weight	Revenue
T1	1	2018-02-10	2018-02-12	0.40	100
T2	1	2018-02-11	2018-02-12	0.20	100
T3	1	2018-02-12	2018-02-12	0.40	100
T1	2	2018-02-10	2018-02-20	0.40	50
T2	2	2018-02-11	2018-02-20	0.10	50
T3	2	2018-02-12	2018-02-20	0.10	50
T4	2	2018-02-20	2018-02-20	0.40	50

Revenue Table

```scala
val transactions = spark.read.table("transactions").as("tx")
val touchpoints = spark.read.table("touchpoints").as("tp").filter($"date" >= thirtyDaysAgo)
val visitor_to_visitor = spark.read.table("visitor_to_visitor").as("vtv")
val transactions_visitors = transactions
  .join(
    visitor_to_visitor,
    $"tx.visitor_id"===$"vtv.visitor_id_a",
    "inner"
  ).join(
    touchpoints,
    $"vtv.visitor_id_b"===$"tp.user.visitor_id" &&
    $"tp.header.timestamp" >= (
      // Join on touchpoints up to 30 days
      toMillisecondsUDF($"tx.date_of_checkout") - (attributionWindowNumDays * 24l * 60l * 60l * 1000l)
    ) && $"tp.header.timestamp" < toMillisecondsUDF($"tx.date_of_checkout"),
    "inner"
  )
```

Position-based UDF

```
def positionBased(position: Int, total: Int): Double = {
  total match {
    case 1 => 1.0
    case 2 => 0.5
    case _ => {
        if ((position == total) || (position == 1)) {
          0.4
        } else {
          0.2 * (1.0/(total-2))
        }
    }
  }
}
```

Revenue Table

```
val transactions_weights = transactions_visitors
  .withColumn("position_based_weight", positionBasedUDF($"touchpoint_order_desc", $"number_of_touchpoints"))

transactions_reseller_channel
  .repartition($"date_of_checkout")
  .write
  .format("delta")
  .mode("overwrite")
  .partitionBy("date_of_checkout")
  .option("replaceWhere", s"date_of_checkout >= '$yesterday' AND date_of_checkout < '$today'")
  .save("/mnt/data/attribution/revenue")
```

Looking Back

#SAISEnt13

Main Challenges

- Data quality is hard
  - Are events fired in the right place?
  - Do they contain all correct attributes?
  - Notebooks are very handy!
- Backfill historical data
  - Without historical data, you can't make good projections

We are hiring!

#SAISEnt13