# Aggregation Consistency Errors in Semantic Layers and How to Avoid Them

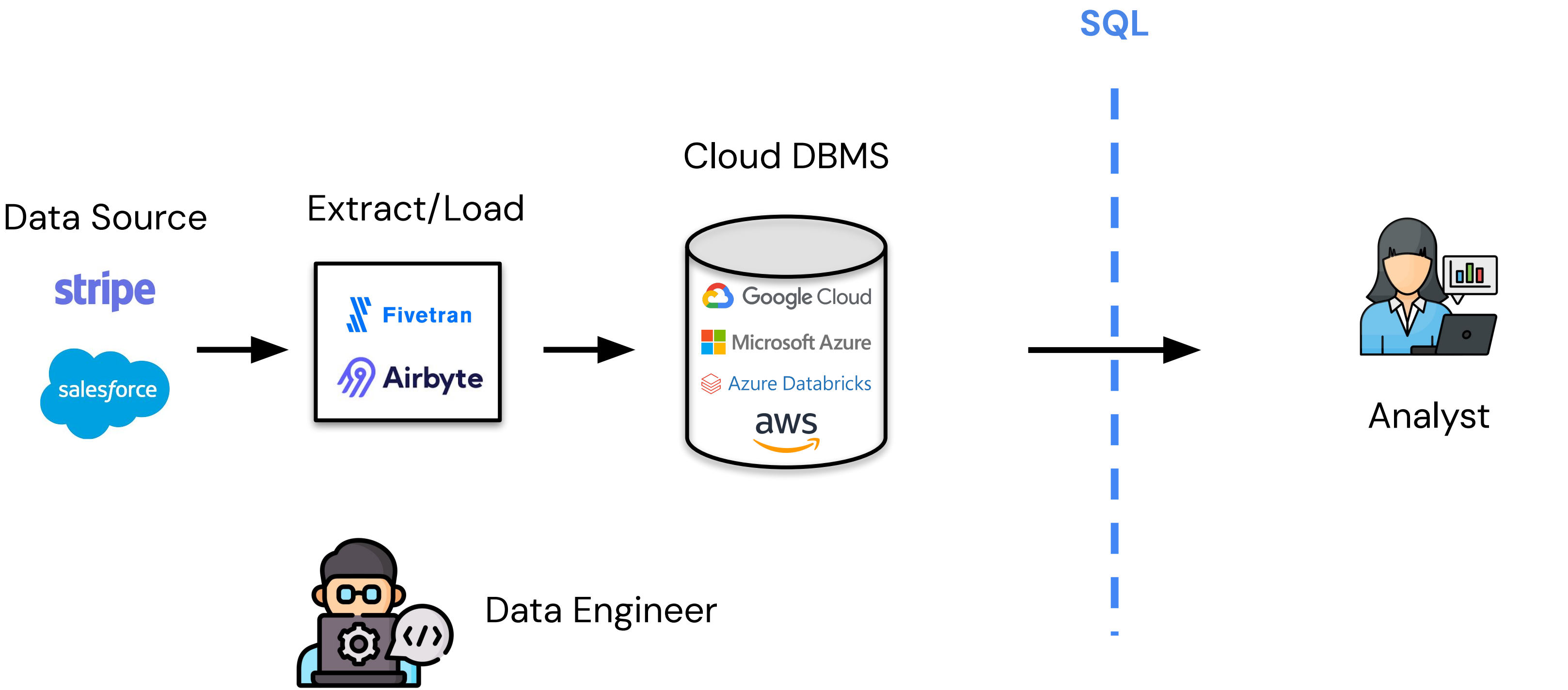
Zachary Huang, Pavan Kalyan Damalapati, Eugene Wu
Data Science Institute, Columbia University

COLUMBIA
UNIVERSITY

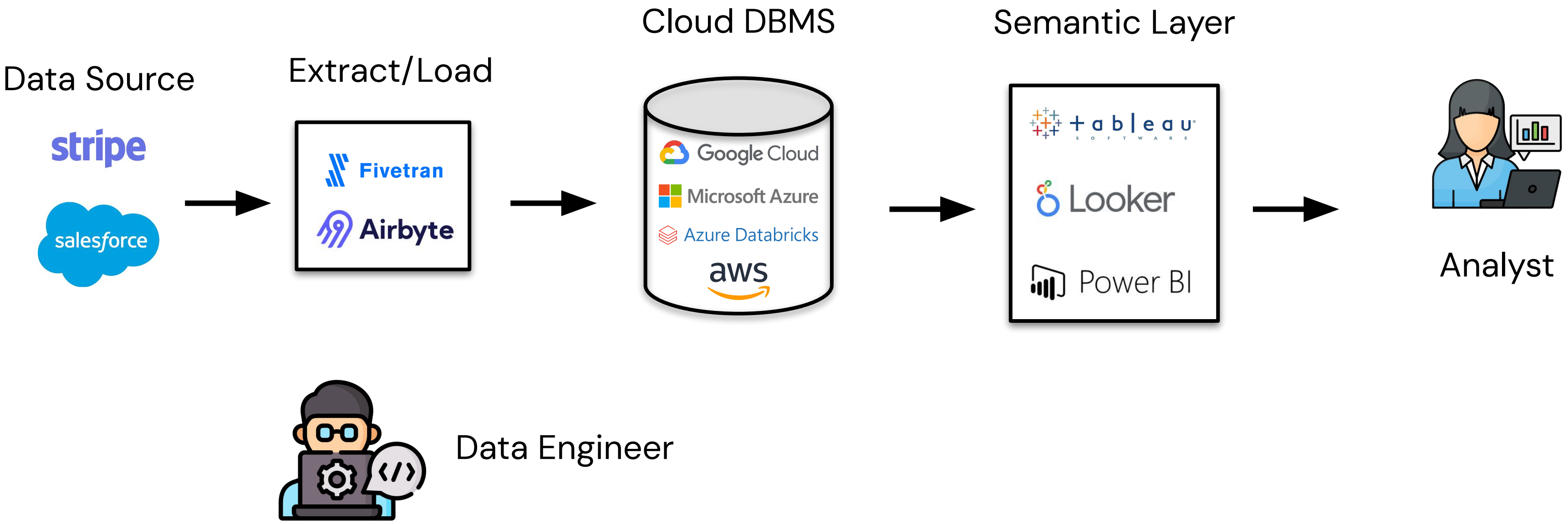
# Agenda

- Semantic Layer Background
- Aggregation Consistency Errors from Current Semantic Layer
- **Our solution:** Human-in-the-loop Weighing

# Semantic Layer Background



# Semantic Layer Background



Data Source

Extract/Load

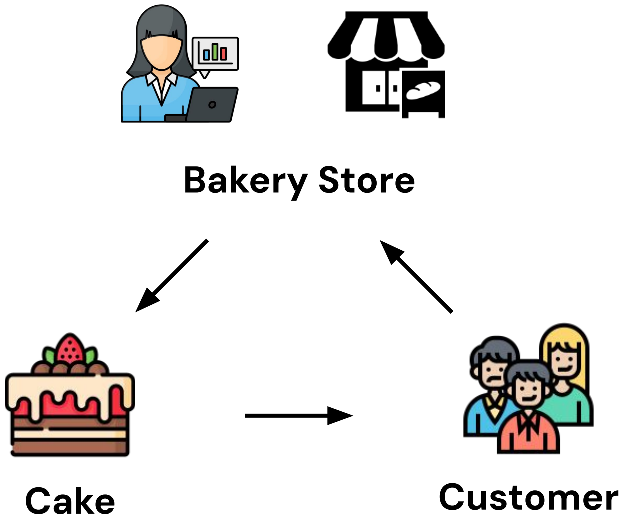
Cloud DBMS

Semantic Layer

Analyst

Data Engineer

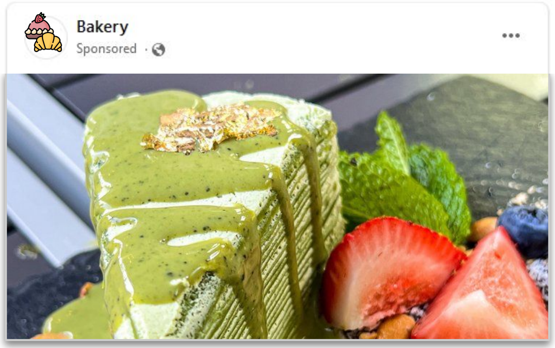
# Challenge: Complex Database Schema



Bakery Store

Cake

Customer



Bakery

Sponsored ·

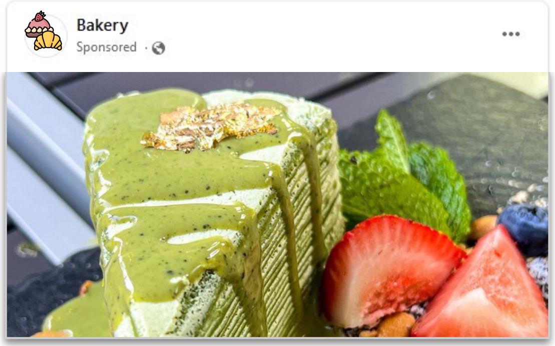
Ads

What's the return on Ad costs?

# Challenge: Complex Database Schema

Ad

AdID
Source
Cost



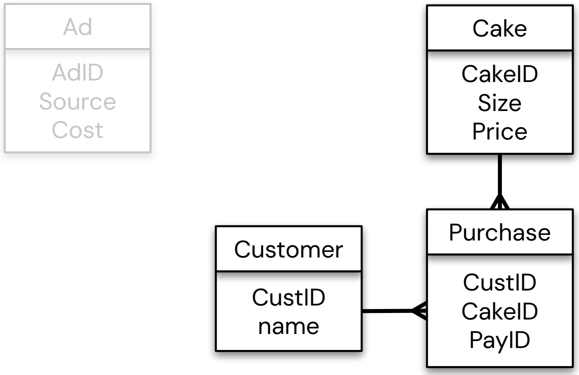
**Ads**

**What's the return on Ad costs?**

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

# Challenge: Complex Database Schema



Ad

AdID
Source
Cost

Cake

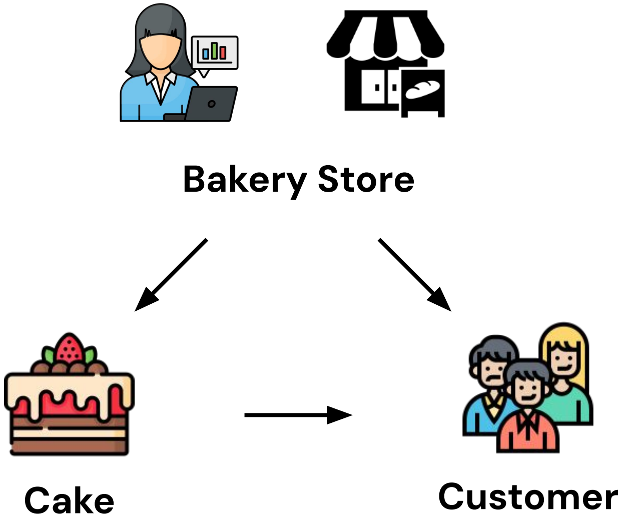
CakeID
Size
Price

Customer

CustID
name

Purchase

CustID
CakeID
PayID



Bakery Store

Cake

Customer

### Ad

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

### Customer

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

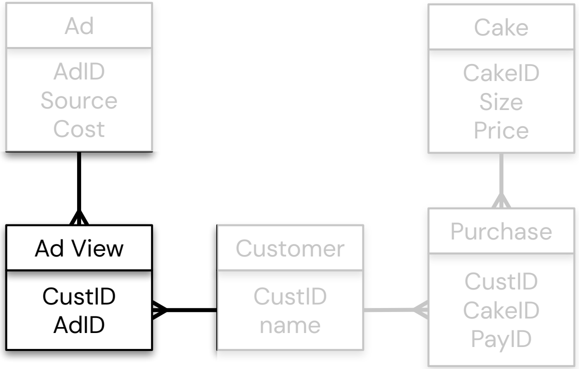
### Purchase

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

### Cake

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Challenge: Complex Database Schema



Ad
AdID
Source
Cost

Cake
CakeID
Size
Price

Ad View
CustID
AdID

Customer
CustID
name

Purchase
CustID
CakeID
PayID

Customer

view

Ads

### Ad

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

### Ad View

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

### Customer

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

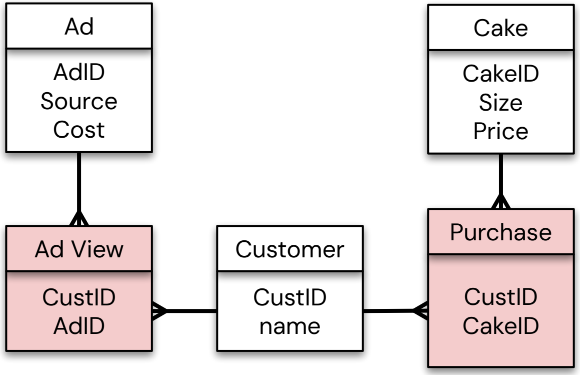
### Purchase

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

### Cake

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Challenge: Complex Database Schema



Analyzing five tables is hard.

Ideally, we just want to pick the attributes and not think about join.

### Ad

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

### Ad View

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

### Customer

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

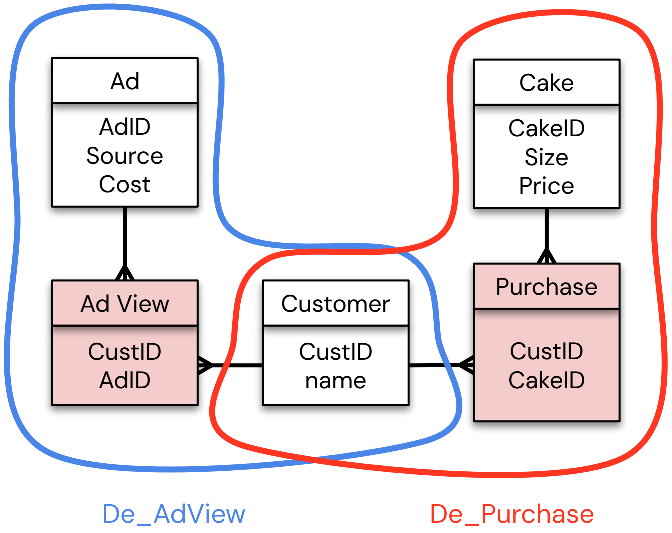
### Purchase

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

### Cake

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Challenge: Complex Database Schema



**Naive solution:** Denormalization

```
CREATE VIEW De_AdView AS
SELECT AdID, Source, Cost, CustID, name
FROM AdView JOIN Customer ON CustID
            JOIN Ad ON AdID;

CREATE VIEW De_Purchase AS
SELECT CustID, name, CakeID, Size, Price
FROM Purchase JOIN USER ON CustID
              JOIN Cake ON Cake
```

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Ad View**

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

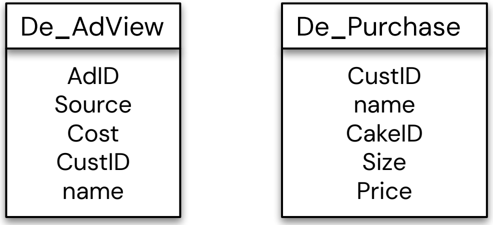
**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Challenge: Complex Database Schema



**However, Aggregation Consistency Errors**

**Naive solution:** Denormalization

```
CREATE VIEW De_AdView AS
SELECT AdID, Source, Cost, CustID, name
FROM AdView JOIN Customer ON CustID
                JOIN Ad ON AdID;

CREATE VIEW De_Purchase AS
SELECT CustID, name, CakeID, Size, Price
FROM Purchase JOIN USER ON CustID
                 JOIN Cake ON Cake
```

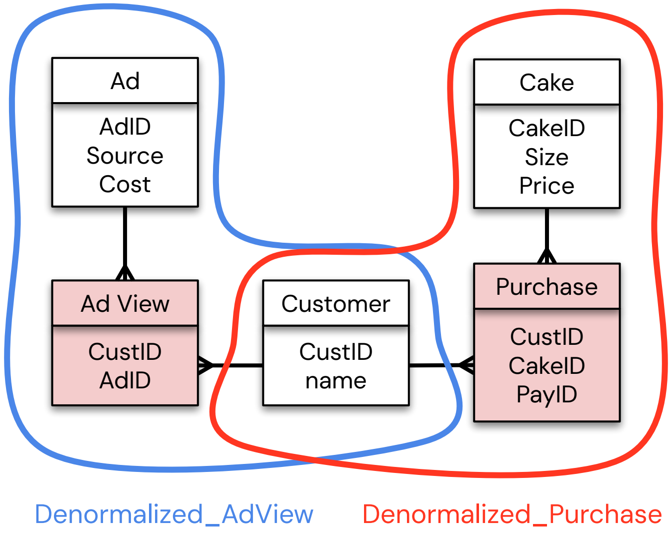
**De_AdView**

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | Google | 500 | 1 | Joe |
| 2 | Facebook | 600 | 1 | Joe |
| 1 | Google | 500 | 2 | Mary |

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | 20 |
| 2 | Mary | 1 | 1 | 20 |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

# Challenge: Complex Database Schema



**Naive solution:** Denormalization

```
CREATE VIEW Denormalized_AdView AS
SELECT *
FROM AdView JOIN USER ON CustID
             JOIN Ad ON AdID;

CREATE VIEW Denormalized_Purchase AS
SELECT *
FROM Purchase JOIN USER ON CustID
              JOIN Cake ON Cake
```

Therefore, single source of truth
However, Aggregation Consistency Errors

### Ad

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

### Ad View

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

### Customer

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

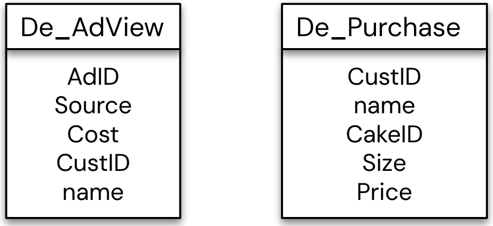
### Purchase

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

### Cake

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors



De_AdView

AdID
Source
Cost
CustID
name

De_Purchase

CustID
name
CakeID
Size
Price

Q1: What is the total cost of ads from all sources?

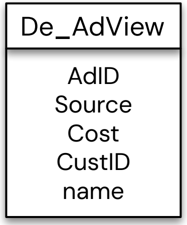
**De_AdView**

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | Google | 500 | 1 | Joe |
| 2 | Facebook | 600 | 1 | Joe |
| 1 | Google | 500 | 2 | Mary |

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | 20 |
| 2 | Mary | 1 | 1 | 20 |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

# Aggregation Consistency Errors



De_AdView

AdID
Source
Cost
CustID
name

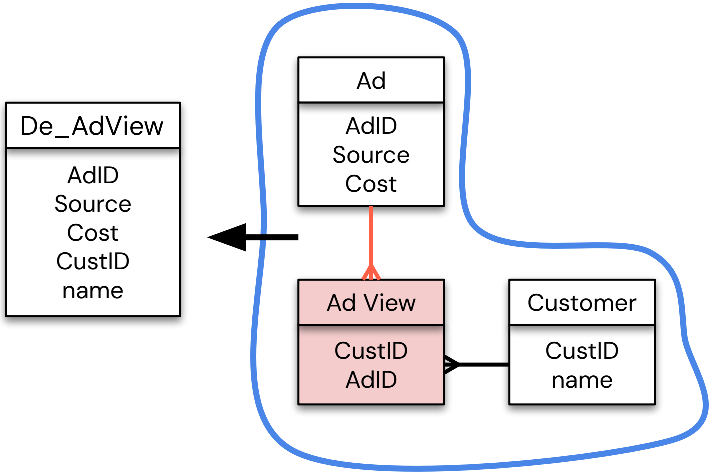
Q1: What is the total cost of ads from all sources?

```
SELECT SUM (cost) FROM Denormalized_AdView;
```

**De_AdView**

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | Google | 500 | 1 | Joe |
| 2 | Facebook | 600 | 1 | Joe |
| 1 | Google | 500 | 2 | Mary |

# Aggregation Consistency Errors



De_AdView

AdID
Source
Cost
CustID
name

Ad

AdID
Source
Cost

Ad View

CustID
AdID

Customer

CustID
name

N-1 join causes duplicates

Q1: What is the total cost of ads from all sources?
SELECT SUM (cost) FROM Denormalized_AdView;

To deduplicate, it seems a trivial fix:
Query normalized table
SELECT SUM (cost) FROM Ad;

That's what current semantic layers do.

**De_AdView**

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | Google | 500 | 1 | Joe |
| 2 | Facebook | 600 | 1 | Joe |
| 1 | Google | 500 | 2 | Mary |

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

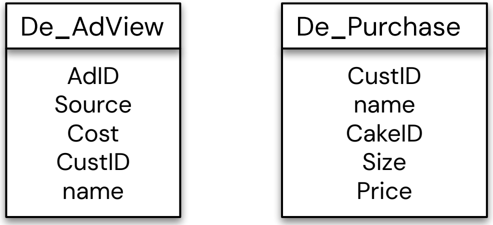
**Ad View**

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

# Aggregation Consistency Errors



Q2: What is the total revenue from the purchased items?

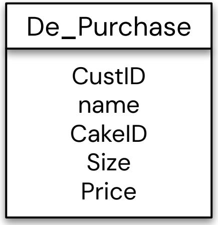
**De_AdView**

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | Google | 500 | 1 | Joe |
| 2 | Facebook | 600 | 1 | Joe |
| 1 | Google | 500 | 2 | Mary |

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | 20 |
| 2 | Mary | 1 | 1 | 20 |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

# Aggregation Consistency Errors



De_Purchase

CustID
name
CakeID
Size
Price

Q2: What is the total revenue from the purchased items?

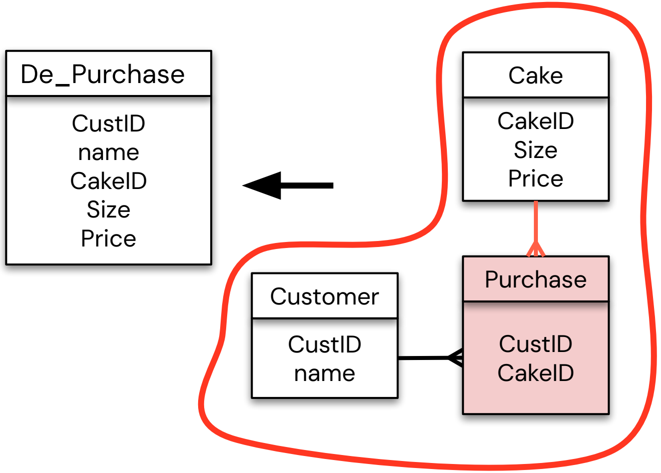
```
SELECT SUM (price) FROM Denormalized_Purchase;
```

**Shall we deduplicate, like Q1?**

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | 20 |
| 2 | Mary | 1 | 1 | 20 |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

# Aggregation Consistency Errors



Q2: What is the total revenue from the purchased items?
SELECT SUM (price) FROM Denormalized_Purchase;

We shall **not** deduplicate, as price is paid per purchase.
The choices depends on the query.

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | **20** |
| 2 | Mary | 1 | 1 | **20** |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

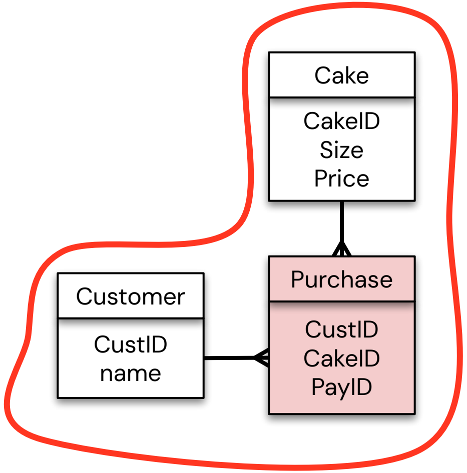
**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors



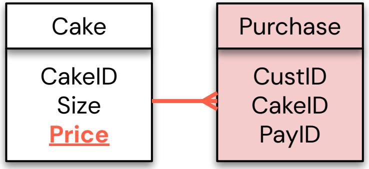
Denormalized_Purchase

Q2: What is the total revenue from the purchased items?

```
SELECT SUM (price) FROM Denormalized_Purchase;
```

**Denormalized_Purchase**

| CustID | CakeID | PayID | Price | ... |
|---|---|---|---|---|
| 1 | 1 | 1 | 20 | |
| 2 | 1 | 1 | 20 | |
| 2 | 2 | 1 | 30 | |
| 2 | 4 | 3 | 35 | |



Duplications Needed!

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Ad View**

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

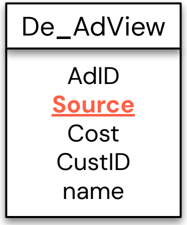
**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

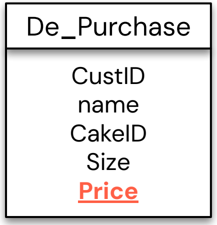
| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors



De_AdView

AdID<br>
**Source**<br>
Cost<br>
CustID<br>
name



De_Purchase

CustID<br>
name<br>
CakeID<br>
Size<br>
**Price**

Q3: What is the total revenue from different ad sources?

SELECT Ad.source, SUM(Cake.price) FROM ???

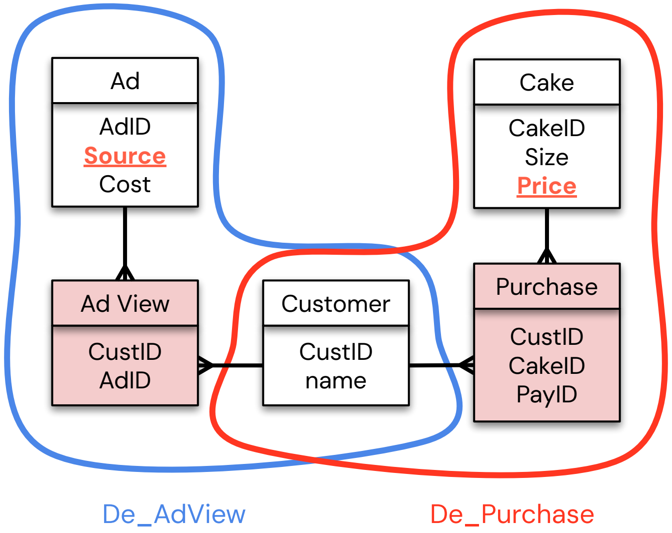
### De_AdView

| AdID | Source | Cost | CustID | Name |
|---|---|---|---|---|
| 1 | **Google** | 500 | 1 | Joe |
| 2 | **Facebook** | 600 | 1 | Joe |
| 1 | **Google** | 500 | 2 | Mary |

### De_Purchase

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | **20** |
| 2 | Mary | 1 | 1 | **20** |
| 2 | Mary | 2 | 3 | **30** |
| 2 | Mary | 3 | 5 | **35** |

# Aggregation Consistency Errors



Q3: What is the total revenue from different ad sources?

SELECT Ad.source, SUM(Cake.price) FROM ???

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Ad View**

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

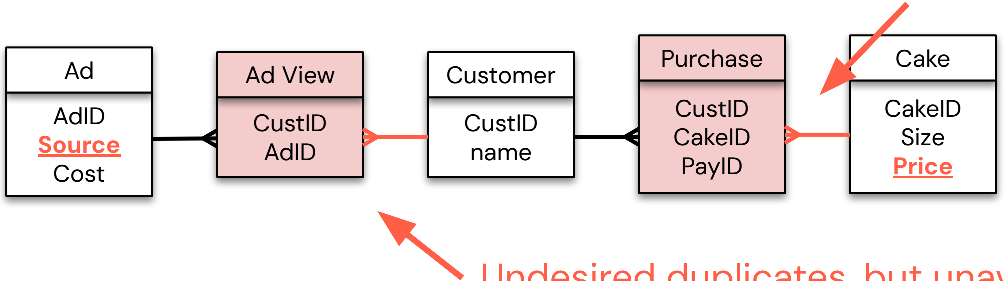
# Aggregation Consistency Errors

Q3. What is the total revenue from different ad sources?

SELECT Ad.source, SUM(Cake.price) FROM …

Price is duplicated along N-1 Path ≻—

Desired duplicates (from Q2)



Undesired duplicates, but unavoidable

**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Ad View**

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors **Are Hard!**

Correctness of Aggregates depends on

- Set of tables to join?
- deduplication methods (duplicate or not)
- Semantic meaning of attributes
- …

Some aggregation query like Q3 is fundamentally ambiguous.
**Impossible** to find a denormalized table as the “single source of truth” for all queries.

Next, I will discuss the (imperfect) solutions by current semantic layer

# Aggregation Consistency Errors Are Hard!

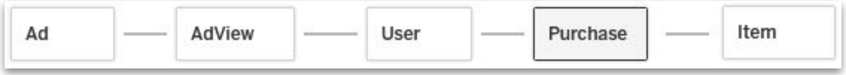
## (imperfect) solutions from industry

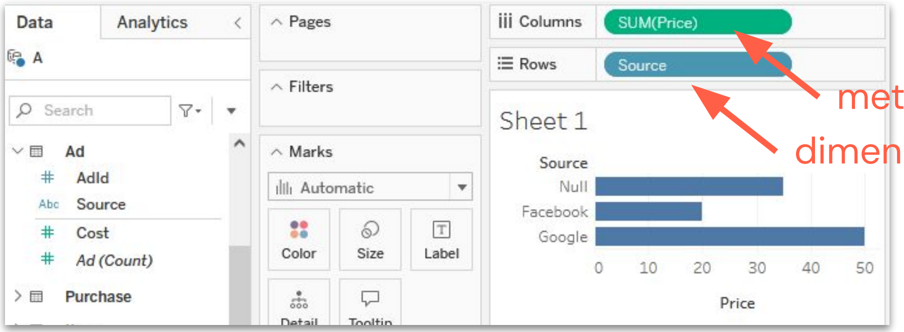
Tailor the decisions to query

**Offline:** build a “join graph”, and metrics (“aggregation”).

**Online:** analysts specify dimension attributes and metrics. BI tools decide the join and duplications.

**Tableau**





metrics

dimension

# Aggregation Consistency Errors Are Hard!

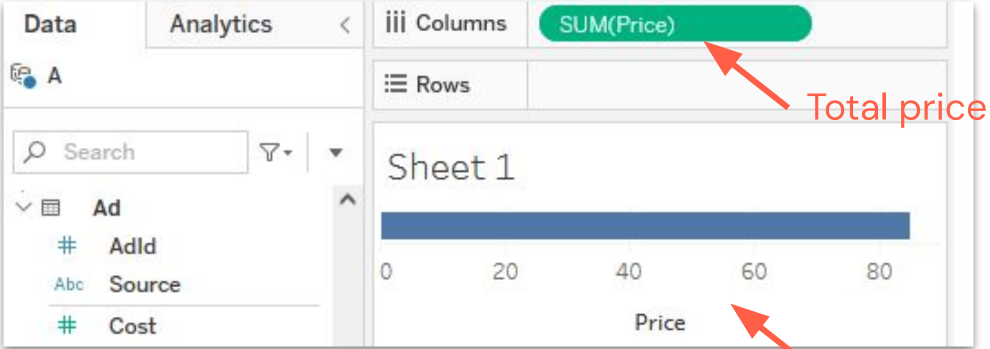
## (imperfect) solutions from industry

Q2: What is the total revenue from the purchased items? SELECT SUM (price)
Correct answer: 105

**De_Purchase**

| CustID | Name | CakeID | Size | Price |
|---|---|---|---|---|
| 1 | Joe | 1 | 1 | 20 |
| 2 | Mary | 1 | 1 | 20 |
| 2 | Mary | 2 | 3 | 30 |
| 2 | Mary | 3 | 5 | 35 |

tableau: 85 (without duplication)



Total price

85

Looker Power BI ... show same error.

Such errors are hard to notice!

**Customer**

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

**Purchase**

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

**Cake**

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors Are Hard!

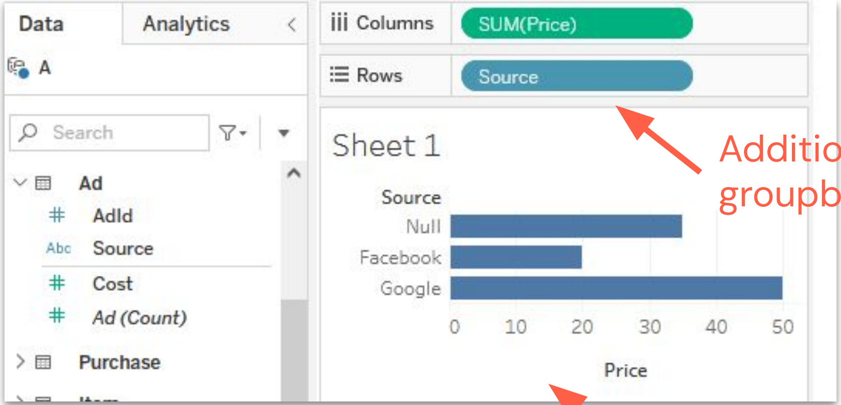
## (imperfect) solutions from industry

Q3: What is the total revenue from different ad sources? SELECT Ad.source, SUM(Cake.price)

Due to many-to-many joins, deduplication is ambiguous.

tableau : arbitrary heuristics.

Looker Power BI ... all decide arbitrarily, with different query results.



Additional groupby

Arbitrary deduplication

### Ad

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

### Ad View

| CustID | AdID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

### Customer

| CustID | Name |
|---|---|
| 1 | Joe |
| 2 | Mary |

### Purchase

| CustID | CakeID | PayID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 3 | 2 |

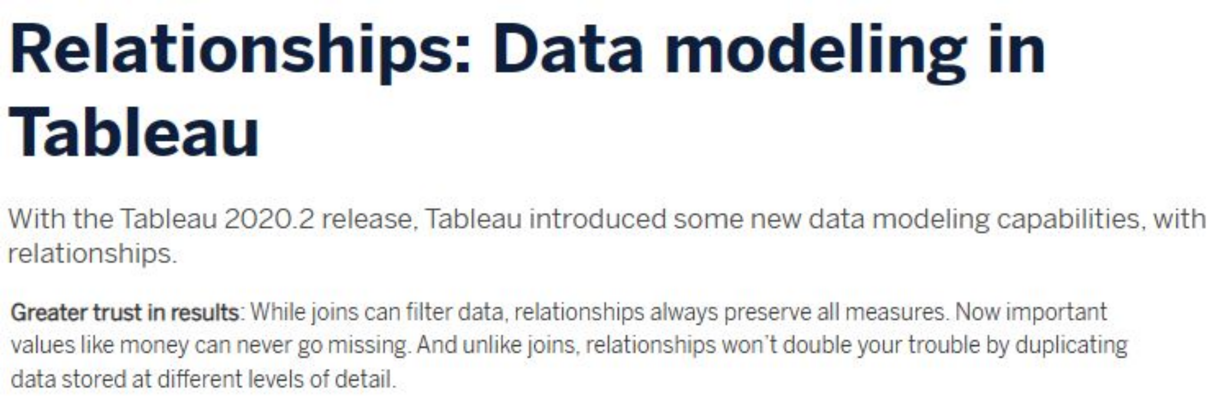
### Cake

| CakeID | Size | Price |
|---|---|---|
| 1 | 1 | 20 |
| 2 | 3 | 30 |
| 3 | 5 | 35 |

# Aggregation Consistency Errors Are Hard!

## (imperfect) solutions from industry

Current tools apply heuristics hidden from analysts, leads to unnoticed errors.



Relationships: Data modeling in Tableau

With the Tableau 2020.2 release, Tableau introduced some new data modeling capabilities, with relationships.

**Greater trust in results**: While joins can filter data, relationships always preserve all measures. Now important values like money can never go missing. And unlike joins, relationships won't double your trouble by duplicating data stored at different levels of detail.

Our survey shows: impossible to pre-define correct heuristics offline, as it depends on the query and analyst interpretation.

https://www.tableau.com/blog/relationships-tableau-data-model

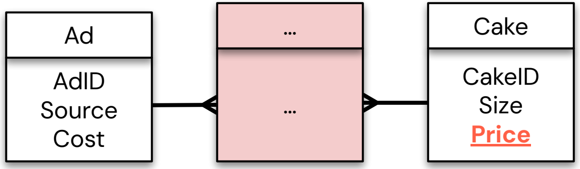
# Aggregation Consistency Errors Are Hard!

## (imperfect) solutions from academia

**Summarizability**: can we aggregate fine-grained values at coarser level?

However, "summarizability" is too strict for practical exploratory queries.

E.g., for Q3: “What is the total revenue from different ad sources?”
Many-to-many joins are "nonstrict" and thus not summarizable, but are important in applications

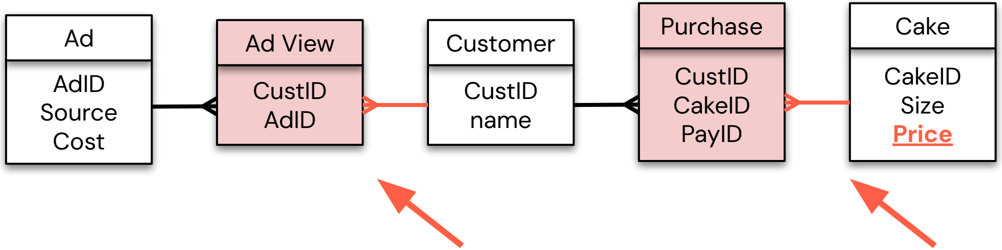


# Aggregation Consistency Errors Are Hard!

(imperfect) solutions from academia

**Pre-aggregation:** (e.g., average) before join to avoid N-N.
Used by ML over multiple relations, where N-N causes unbalanced training.

However, average of average is not average. This causes simpson paradox



Where to pre-aggregate?

# Aggregation Consistency Errors Are Hard!

## Our Solution

To solve the Errors, we

1. First formalize Aggregation Consistency Errors

2. Propose Weighing as the solution that requires human-in-the-loop

# Formalize Aggregation Consistency Errors

For Q3: What is the total revenue from different ad sources?

**Challenge: What** the query result should be consistent with?

**Use reference Query:** The total revenue as SPJA query (not base table)

$$Q = \gamma_{\text{SUM(item.price)}}(\text{Purchase} \bowtie \text{Cake})$$

**Exploration:** Analysts include more tables (e.g., for groupby)

$$Q^* = \gamma_{\text{Ad, SUM(item.price)}}(\text{Purchase} \bowtie \text{Cake} \bowtie \ldots)$$

**Consistency:** total revenue remains the same, even with additional tables

$$\boldsymbol{\gamma_{\text{SUM(item.price)}}(Q^*) = \gamma_{\text{SUM(item.price)}}(Q)}$$

# Formalize Aggregation Consistency Errors

In general

**Metric Definition:** We express metric as SPJA query

$$Q = \gamma_{AGG}(R1 \bowtie R2 \bowtie \ldots Rm)$$

**Exploration:** Analysts include more tables (e.g., for groupby A / selection σ)

$$Q^* = \gamma_{A,\ AGG}(\sigma(R1 \bowtie R2 \bowtie \ldots Rm \bowtie \ldots Rn))$$

**Consistency:** total revenue remains the same, even with additional tables

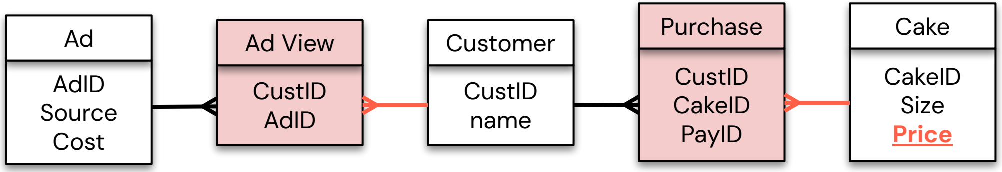
$$\boldsymbol{\gamma_{AGG}(Q^*) + \gamma_{AGG}(\neg Q^*) = \gamma_{AGG}(Q)}$$

where $\neg Q^* = \gamma_{A,\ AGG}(\neg\sigma(R1 \bowtie R2 \bowtie \ldots Rm \bowtie \ldots Rn))$ are not selected tuple

# Weighing as the core primitive

Q3. What is the total revenue from different ad sources?

SELECT Ad.source, SUM(Cake.price) FROM …



Undesired duplicates

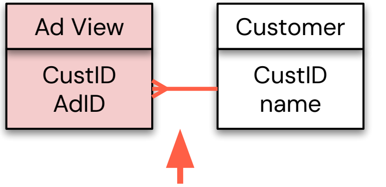
**Ad View**

| | CustID | AdID | SUM |
|---|---|---|---|
| action1 | 1 | 1 | 20 |
| action2 | 1 | 2 | 20 |
| action3 | 2 | 1 | 85 |

**Customer**

| CustID | Name | SUM |
|---|---|---|
| 1 | Joe | 20 |
| 2 | Mary | 85 |

# Weighing as the core primitive

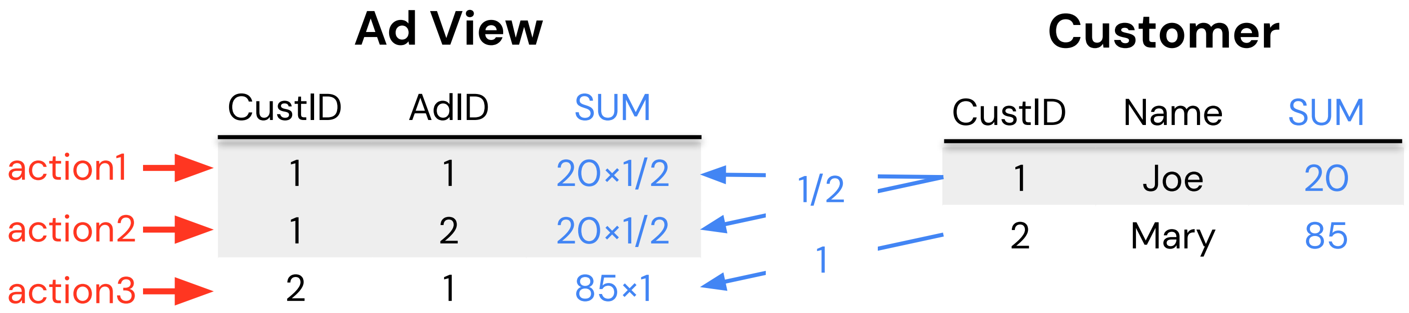


Undesired duplicates

**Solution Motivated by Marketing:** Attribution

Weigh customer actions (Ad View) that contribute to the outcome (Revenue), such that, for each customer, the total weights of actions add up to 1.

**Option 1:** Weigh the actions **equally**



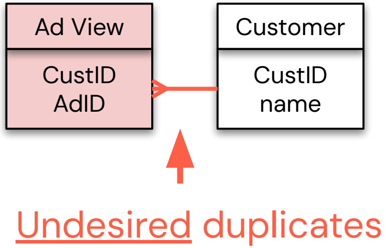
**Ad View**

| | CustID | AdID | SUM |
|---|---|---|---|
| action1 | 1 | 1 | 20×1/2 |
| action2 | 1 | 2 | 20×1/2 |
| action3 | 2 | 1 | 85×1 |

**Customer**

| CustID | Name | SUM |
|---|---|---|
| 1 | Joe | 20 |
| 2 | Mary | 85 |

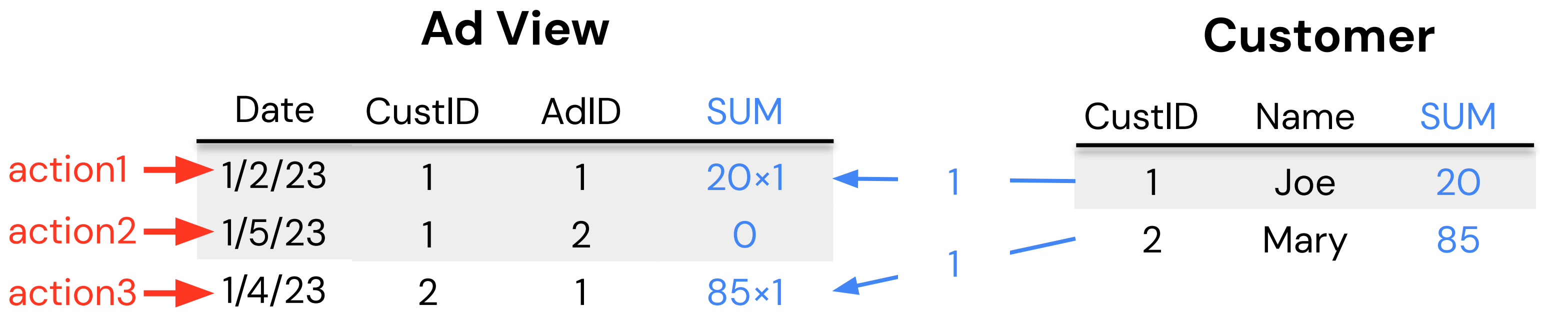
# Weighing as the core primitive



**Solution Motivated by Marketing:** Attribution

Weigh customer actions (Ad View) that contribute to the outcome (Revenue), such that, for each customer, the total weights of actions add up to 1.

**Option 2:** Attribute to the **first** action



**Ad View**

| | Date | CustID | AdID | SUM |
|---|---|---|---|---|
| action1 | 1/2/23 | 1 | 1 | 20×1 |
| action2 | 1/5/23 | 1 | 2 | 0 |
| action3 | 1/4/23 | 2 | 1 | 85×1 |

**Customer**

| CustID | Name | SUM |
|---|---|---|
| 1 | Joe | 20 |
| 2 | Mary | 85 |

# Weighing as the core primitive

**Weighing** generializes prior solutions across industry and academic

- Market Attributions, Order Management
- Causal Inference, Probabilistic graphical model
- ...

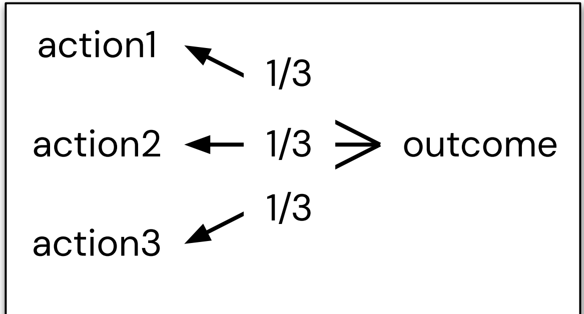
Supports broad range of aggregations

- Other aggregation functions like MIN/MAX/AVG...
- ML model like linear regression, k-means...

More technical details in the paper

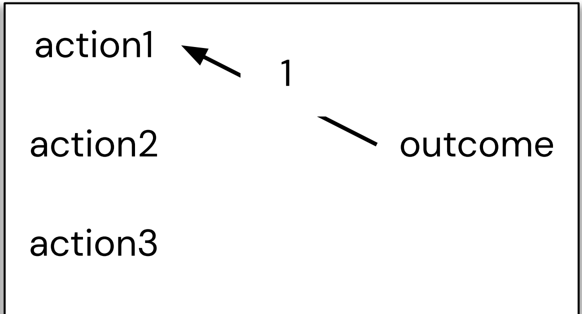
Maier, Marc, et al. "A sound and complete algorithm for learning causal models from relational data." arXiv preprint arXiv:1309.6843 (2013).
Koller, Daphne, and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

# Human-in-the-loop Weighing



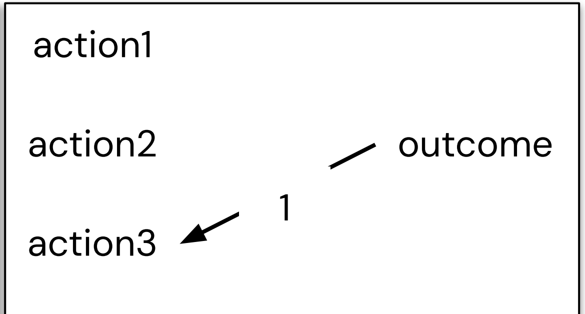
action1
1/3
action2
1/3
outcome
1/3
action3

**Equal Weighing**



action1
1
action2
outcome
action3

**First Touch**



action1
action2
outcome
1
action3

**Last Touch**

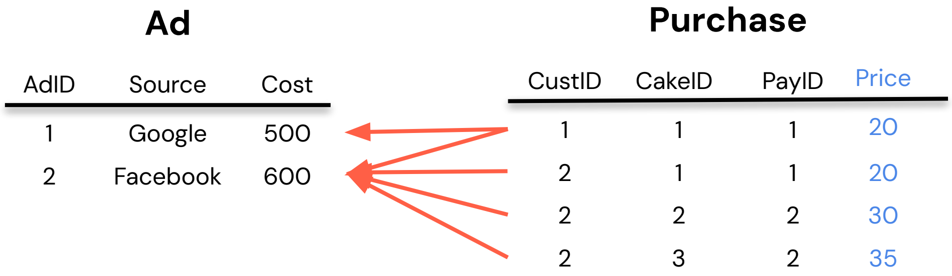
Fundamentally a human-in-the-loop problem.
**How to design framework for weighing?**

# Human-in-the-loop Weighing

**Usability challenges**:

- Presenting full databases to weigh is overwhelming
- Visualizing many-to-many relationships is hard

Customer purchase and Ad view is N-N
HARD to weight!



**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Purchase**

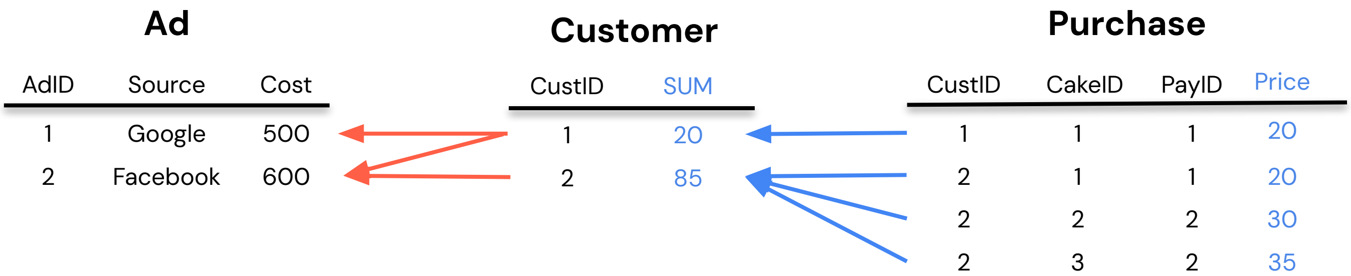
| CustID | CakeID | PayID | Price |
|---|---|---|---|
| 1 | 1 | 1 | 20 |
| 2 | 1 | 1 | 20 |
| 2 | 2 | 2 | 30 |
| 2 | 3 | 2 | 35 |

# Human-in-the-loop Weighing

**Usability challenges**:

- Presenting full databases to weigh is overwhelming
- Visualizing many-to-many relationships is hard

**Solution:** partially aggregate 1-N join and weigh N-1 join progressively



**Ad**

| AdID | Source | Cost |
|---|---|---|
| 1 | Google | 500 |
| 2 | Facebook | 600 |

**Customer**

| CustID | SUM |
|---|---|
| 1 | 20 |
| 2 | 85 |

**Purchase**

| CustID | CakeID | PayID | Price |
|---|---|---|---|
| 1 | 1 | 1 | 20 |
| 2 | 1 | 1 | 20 |
| 2 | 2 | 2 | 30 |
| 2 | 3 | 2 | 35 |

Weigh for N-1

Partial sum along 1-N

# Human-in-the-loop Weighing Interface

**Goal:** provide sufficient context while requesting minimum input

**Weighing for Q3:** What is the total revenue from different ad sources?

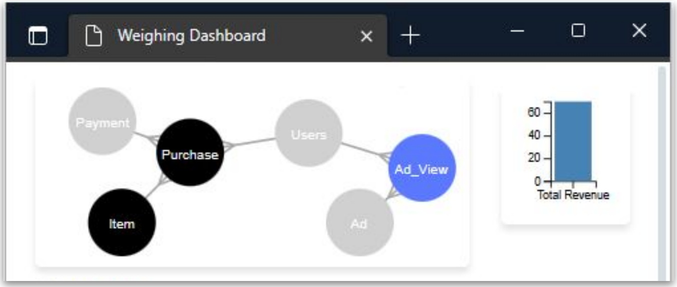
Metric: $Q = \gamma_{\text{SUM(item.price)}}(\text{Purchase} \bowtie \text{Cake})$

## Top Panel for Overview View

Join Graph to visualize progress.

- Blue are tables to weigh (Ad View)
- **Black** are tables in Q (Purchase, Cake)

Visualize the Q result.



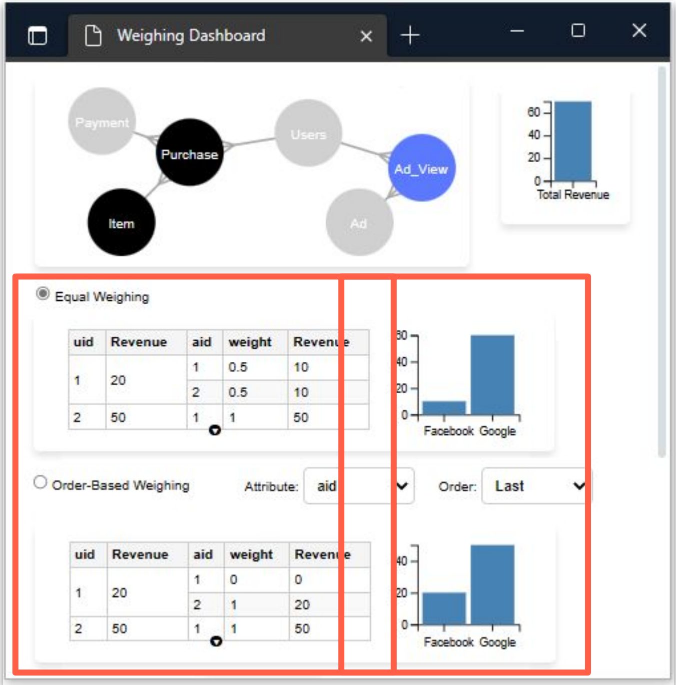
# Human-in-the-loop Weighing Interface

## Bottom Panel for Detailed Weights

Common policies as defaults

- Equal Weighing
- Order-Based Weighing
  (e.g., the first get the whole weight)
- Proportional Weighing
  (e.g., weigh freight based on item sizes)
- SQL interface for customized weighing

Visualizations for weighing results



● Equal Weighing

| uid | Revenue | aid | weight | Revenue |
|---|---|---|---|---|
| 1 | 20 | 1 | 0.5 | 10 |
| | | 2 | 0.5 | 10 |
| 2 | 50 | 1 | 1 | 50 |

○ Order-Based Weighing Attribute: aid Order: Last

| uid | Revenue | aid | weight | Revenue |
|---|---|---|---|---|
| 1 | 20 | 1 | 0 | 0 |
| | | 2 | 1 | 20 |
| 2 | 50 | 1 | 1 | 50 |

# Conclusion

- Study Aggregation Consistency Errors in Semantic Layer
- Propose Weighing as the core primitive
- Introduce framework for Human-in-the-loop Weighing

**Thank you!**