Data Analysis

CSCI 220: Database Management and Systems Design

Slides adapted from
Simon Miner
Gordon College

Today you will learn…

- How to analyze data stored in a relational database

Agenda

- Online Transaction Processing vs Online Analytical Processing
- Privacy Tools Study
- If time: Map-Reduce Pattern

Online Analytical Processing

Online Transaction Processing (OLTP)

- Transactional data: database concerned with maintaining single focused end-user interactions
  - Examples
    - Customer placing an order on an e-commerce website
    - Account holder making a deposit at a bank
  - Can be comprised of several rows/records of data
    - Example: An order has records for the order itself, each line item, address, payment method, etc.
  - Lots of data can accumulate quickly for numerous transactions
    - Needed for its own sake (i.e. shipping orders, order history, monthly account statements, etc.
    - Also useful for analysis…

- OLTP databases built and optimized for speed of transactions (both in the ACID and interaction contexts)
  - e.g. Provisioned with smaller block sizes to facilitate more precise (and maybe quicker) read and write operations

Online Analytical Processing (OLAP)

- Decision support systems (DSS) to help organizations determine longer term courses of action
  - Example: Not many orders for a certain product, so adjust product offerings to better match customer desires
  - Work with summaries and aggregations of raw transaction data
- OLAP user needs to have specific queries in mind
  - Example: Give me a cross-tab of item type vs. color …
- Data mining: reveal patterns in data and system usage
- OLAP databases designed to handle large amounts of data
  - e.g. Provisioned with larger block sizes to store and retrieve more data in read and write operations

Data Warehouse

- Unified repository for an organization’s historical OLAP data
  - Supports trending, analysis, and decision making
- Gathered from numerous disparate sources via ETL processes
  - **Extract:** get data from individual source(s) owned or managed by various parties
  - **Transform:** manipulate data so that it fits into the data warehousing schema – i.e. de-duplication, summarization
  - **Load:** store the transformed data in the data warehouse
- Data is loaded at regular intervals
  - Slightly out of date, which is fine for analytical tasks the data warehouse is used for

OLAP Concepts

- Attribute types
  - Dimension attribute: values to analyze on
    - Explicit: color, size, price, customer type, etc.
    - Derived: age (computed from DOB), ranges (years of experience)
  - Measurement attribute: value summarized or aggregated over various dimensions (sum, count, average, etc.)
- Cross-tab (pivot table): tool allowing easy analysis of data along various dimensions
  - Available in tools like spreadsheets
  - Basic SQL is not an effective tool to produce this kind of structure (lots of dynamic “group by” queries needed)

Example: Sales Data

```
  id  | item_name | color  |  size  | number
------+-----------+--------+--------+--------
    1 | dress     | pastel | medium |      4
    2 | skirt     | dark   | large  |      3
    3 | skirt     | dark   | large  |      1
    4 | dress     | pastel | small  |      1
    5 | pants     | dark   | large  |      2
    6 | shirt     | white  | medium |      4
    7 | skirt     | pastel | medium |      4
    8 | dress     | dark   | medium |      2
    9 | pants     | pastel | large  |      1
   10 | pants     | dark   | large  |      4
  ...
```

Sales Data Crosstab

Color

Item Name

	dark	**pastel**	**white**	**total**
dress	198	177	217	592
pants	228	187	196	611
shirt	233	191	241	665
skirt	208	194	226	628
total	867	749	880	2496

OLAP Operations

- Basic SQL
  - Aggregate functions, like sum(), count(), average()
  - Group by / having clause
- SQL-99 added analytics processing operations
  - Cube
  - Rollup
  - Rank / dense rank

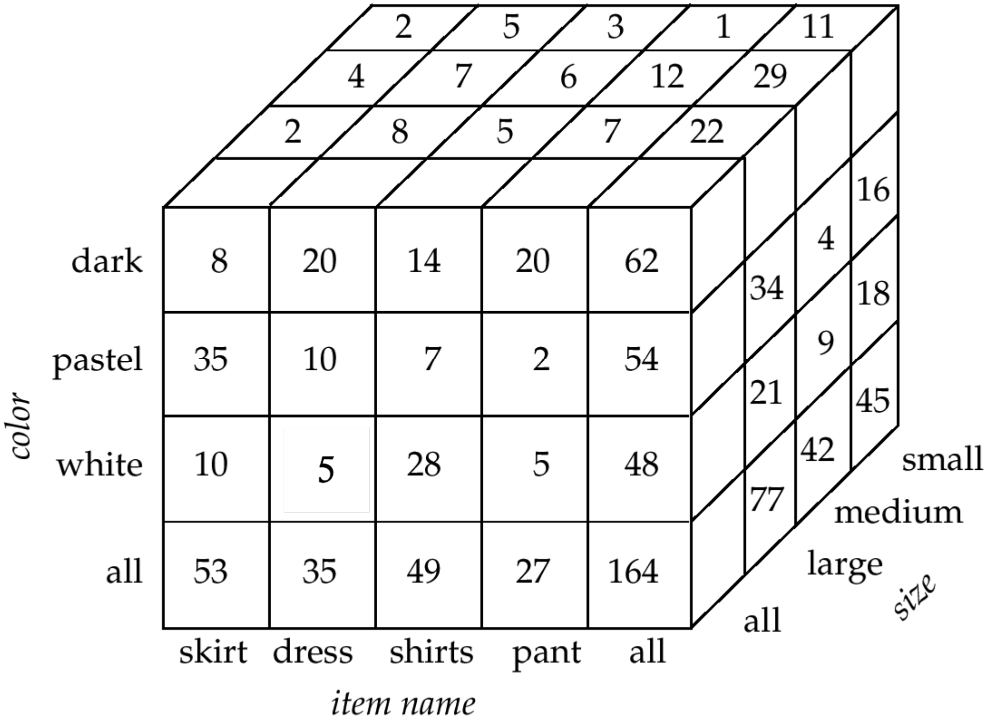
Crosstab Data from CUBE

```
 item_name | color  | sum
-----------+--------+------
 dress     | dark   |  198
 dress     | pastel |  177
 dress     | white  |  217
 dress     |        |  592
 pants     | dark   |  228
 pants     | pastel |  187
 pants     | white  |  196
 pants     |        |  611
 shirt     | dark   |  233
 shirt     | pastel |  191
 shirt     | white  |  241
 shirt     |        |  665
 skirt     | dark   |  208
 skirt     | pastel |  194
 skirt     | white  |  226
 skirt     |        |  628
           | dark   |  867
           | pastel |  749
           | white  |  880
           |        | 2496
```

Cube

- Structure to aggregate a single measurement attribute across numerous dimensions
  - Includes all possible combinations of dimension values
  - Number of cube dimensions = number of dimensional attributes
  - Each dimension "row" includes a summary value for the aggregate of all possible values of that dimension
- User slices cube for specific dimension values

Cube Example

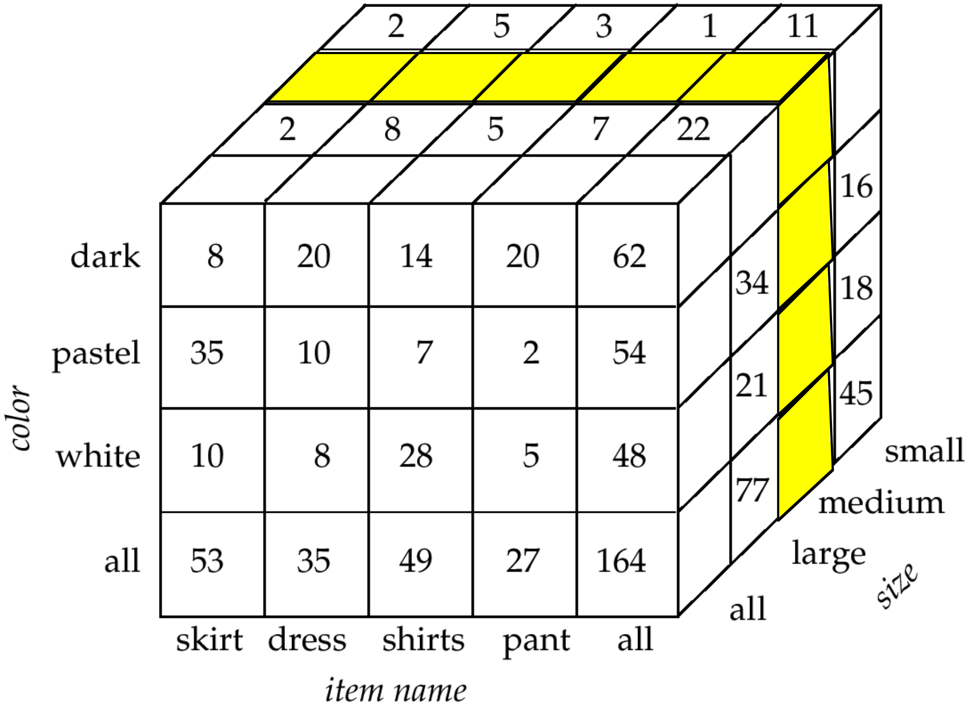


	skirt	dress	shirts	pant	all
dark	8	20	14	20	62
pastel	35	10	7	2	54
white	10	5	28	5	48
all	53	35	49	27	164

Cube showing sales for various combinations of item_name, color and size - including summaries for all item_names, colors, and/or sizes

Figure 18.3 in book

Cube Slice Example

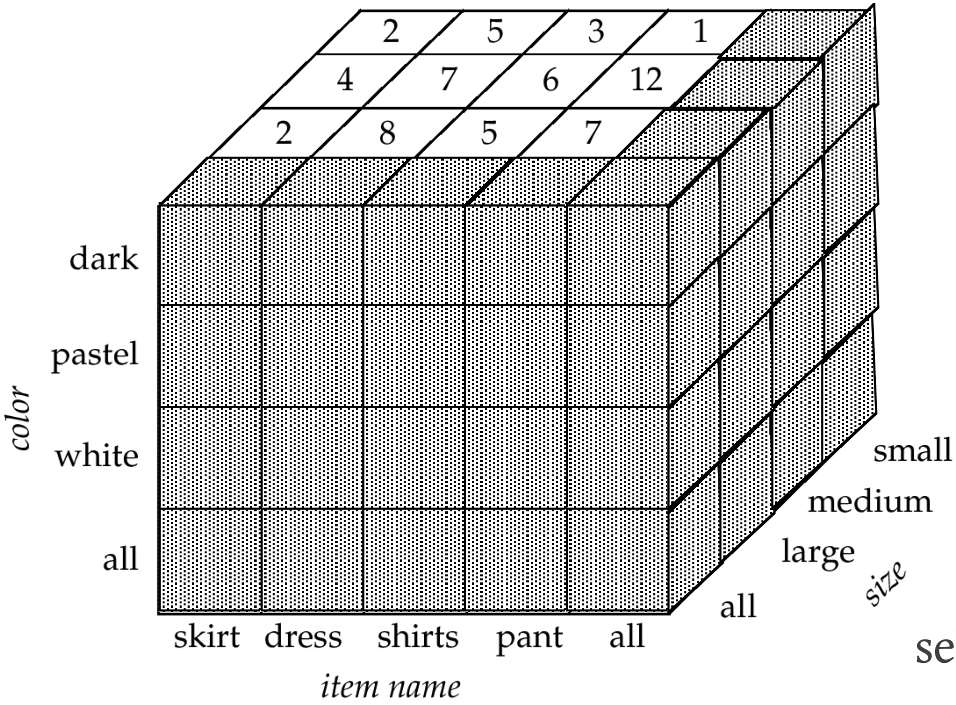


	skirt	dress	shirts	pant	all
dark	8	20	14	20	62
pastel	35	10	7	2	54
white	10	8	28	5	48
all	53	35	49	27	164

Slice showing sales for various combinations of item_name and color for size = medium

Slice from figure 18.3 in book

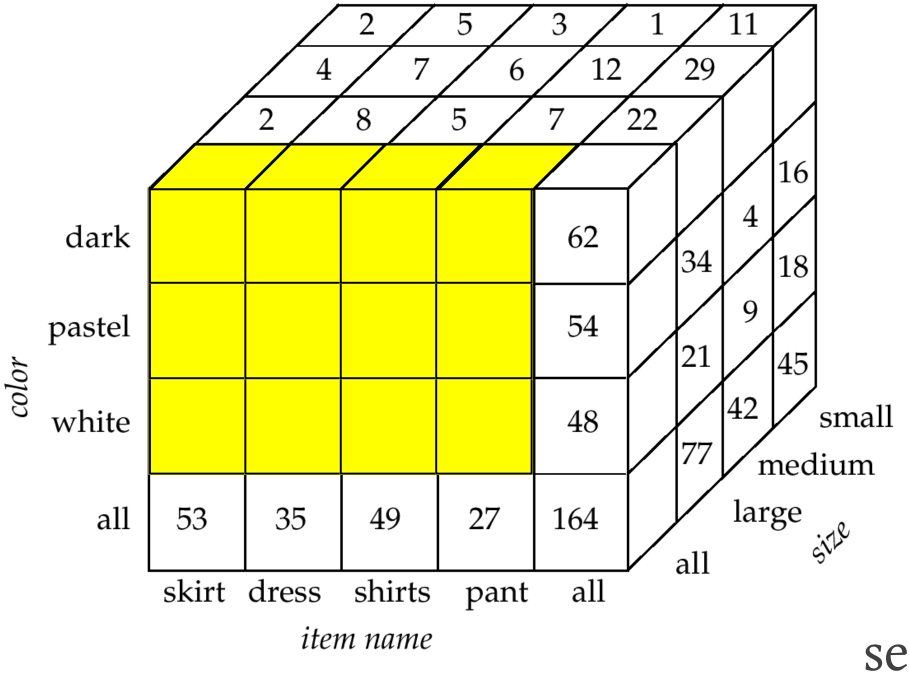
Cube Without Summaries



select item_name, color, size, sum(number)
    from sales
    group by item_name, color, size;

Sales for all possible combinations of item_name, color, size <u>without summaries</u> - everything <u>but</u> shaded squares and squares on bottom. (What would result from a standard SQL query using `group by`).

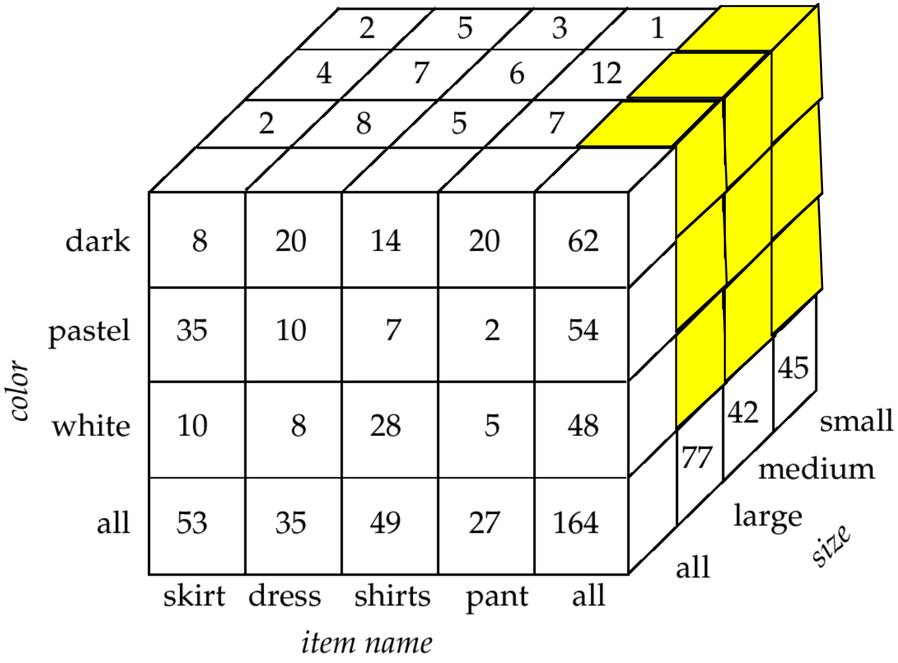
Another Cube Slice



Sales by item_name and color - for all sizes

```
select item_name, color, sum(number)
    from sales
    group by item_name, color;
```

Yet Another Cube Slice



color	skirt	dress	shirts	pant	all
dark	8	20	14	20	62
pastel	35	10	7	2	54
white	10	8	28	5	48
all	53	35	49	27	164

Sales by color and size - for all item_names

```
select color, size, sum(number)
    from sales
    group by color, size;
```

Slicing with SQL

- 2^n SQL queries needed to generate all summary representations for a cube (where n = number of dimensions)
  - For item_name, color, and size (3 dimensions), $2^3 = 8$ queries

```
select item_name, color, size, sum(number)
   from sales
   group by item_name, color, size;
```

```
select item_name, color, sum(number)
    from sales
   group by item_name, color;
```

```
select item_name, size, sum(number)
   from sales
   group by item_name, size;
```

```
select color, size, sum(number)
   from sales
   group by color, size;
```

```
select item_name, sum(number)
   from sales
   group by item_name;
```

```
select color, sum(number)
   from sales
   group by color;
```

```
select size, sum(number)
   from sales
   group by size;
```

```
select sum(number)
   from sales;
```

SQL Cube Function

- cube (*dimension*$_1$*, dimension*$_2$*, …, dimension*$_n$)
  - Used in the group by clause
  - Produces all summary representations in the cube

Cube Example

```
SELECT item_name,
  color,
  size,
  SUM(number)
FROM sales
GROUP BY
  CUBE(item_name, color, size);
```

```
 item_name | color  |  size  | sum
-----------+--------+--------+------
 dress     | dark   | large  |   52
 dress     | dark   | medium |   77
 dress     | dark   | small  |   69
 dress     | dark   |        |  198
 dress     | pastel | large  |   72
 dress     | pastel | medium |   53
 dress     | pastel | small  |   52
 dress     | pastel |        |  177
 dress     | white  | large  |   60
 dress     | white  | medium |   52
 dress     | white  | small  |  105
 dress     | white  |        |  217
 dress     |        | large  |  184
 dress     |        | medium |  182
 dress     |        | small  |  226
 dress     |        |        |  592
 pants     | dark   | large  |   90
 pants     | dark   | medium |   80
 pants     | dark   | small  |   58
--More--
```

SQL Rollup Function

- Summarize data based on the first listed dimension
  - Similar to cube (which yields 2^n groups) for n dimensions
    - Includes all possible combinations of various dimensions and “all”
  - Yields n+1 groups for n dimensions
    - All the dimensions
    - All dimensions except the last
    - All the dimensions except the last and second to last

- rollup($dimension_1$, $dimension_2$, … $dimension_n$)
  - Used in group by clause
  - “Rolling up” dimensions from right to left…

Cube vs Rollup Queries

```
SELECT item_name, color, size, SUM(number)
FROM sales
GROUP BY CUBE(item_name, color, size);
```

```
SELECT item_name, color, size, SUM(number)
FROM sales
GROUP BY ROLLUP(item_name, color, size);
```

Cube vs Rollup Results

Cube

```
 item_name | color  |  size  | sum
-----------+--------+--------+------
 dress     | dark   | large  |   52
 dress     | dark   | medium |   77
 dress     | dark   | small  |   69
 dress     | dark   |        |  198
 dress     | pastel | large  |   72
 dress     | pastel | medium |   53
 dress     | pastel | small  |   52
 dress     | pastel |        |  177
 dress     | white  | large  |   60
 dress     | white  | medium |   52
 dress     | white  | small  |  105
 dress     | white  |        |  217
 dress     |        | large  |  184
 dress     |        | medium |  182
 dress     |        | small  |  226
 dress     |        |        |  592
 pants     | dark   | large  |   90
--More--
```

Rollup

```
 item_name | color  |  size  | sum
-----------+--------+--------+------
 dress     | dark   | large  |   52
 dress     | dark   | medium |   77
 dress     | dark   | small  |   69
 dress     | dark   |        |  198
 dress     | pastel | large  |   72
 dress     | pastel | medium |   53
 dress     | pastel | small  |   52
 dress     | pastel |        |  177
 dress     | white  | large  |   60
 dress     | white  | medium |   52
 dress     | white  | small  |  105
 dress     | white  |        |  217
 dress     |        |        |  592
 pants     | dark   | large  |   90
 pants     | dark   | medium |   80
 pants     | dark   | small  |   58
 pants     | dark   |        |  228
--More--
```

SQL Window Functions

"A window function performs a calculation across a set of table rows that are somehow related to the current row. This is comparable to the type of calculation that can be done with an aggregate function. However, window functions do not cause rows to become grouped into a single output row like non-window aggregate calls would. Instead, the rows retain their separate identities. **Behind the scenes, the window function is able to access more than just the current row of the query result."**

https://www.postgresql.org/docs/13/tutorial-window.html

Window Function Example

- Compare each employee's salary with the average salary in their department:

```
SELECT depname, empno, salary,
  avg(salary) OVER (PARTITION BY depname)
FROM empsalary;

  depname  | empno | salary |          avg
-----------+-------+--------+-----------------------
 develop   |    11 |   5200 | 5020.0000000000000000
 develop   |     7 |   4200 | 5020.0000000000000000
 develop   |     9 |   4500 | 5020.0000000000000000
 develop   |     8 |   6000 | 5020.0000000000000000
 develop   |    10 |   5200 | 5020.0000000000000000
 personnel |     5 |   3500 | 3700.0000000000000000
 personnel |     2 |   3900 | 3700.0000000000000000
 sales     |     3 |   4800 | 4866.6666666666666667
 sales     |     1 |   5000 | 4866.6666666666666667
 sales     |     4 |   4800 | 4866.6666666666666667
(10 rows)
```

https://www.postgresql.org/docs/13/tutorial-window.html

Window Function Components

- A window function call always contains an OVER clause directly following the window function's name and argument(s).
  - The OVER clause determines exactly how the rows of the query are split up for processing by the window function.
  - The PARTITION BY clause within OVER divides the rows into groups, or partitions, that share the same values of the PARTITION BY expression(s).
- For each row, the window function is computed across the rows that fall into the same partition as the current row.

https://www.postgresql.org/docs/13/tutorial-window.html

Window Function Example

- See which employees are paid most/least/etc. using rank()

```
SELECT depname, empno, salary,
  rank() OVER (PARTITION BY depname ORDER BY salary DESC)
FROM empsalary;

 depname   | empno | salary | rank
-----------+-------+--------+------
 develop   |     8 |   6000 |    1
 develop   |    10 |   5200 |    2
 develop   |    11 |   5200 |    2
 develop   |     9 |   4500 |    4
 develop   |     7 |   4200 |    5
 personnel |     2 |   3900 |    1
 personnel |     5 |   3500 |    2
 sales     |     1 |   5000 |    1
 sales     |     4 |   4800 |    2
 sales     |     3 |   4800 |    2
(10 rows)
```

https://www.postgresql.org/docs/13/tutorial-window.html

Privacy Tools Study

Smaller scale data analysis

Where to Analyze Your Data?

- Read data directly from the database
  - Pros:
    - A single authoritative version of your data
    - Analyses can be continuously updated
    - Potentially higher-performance (if data can't fit in memory)
- Alternatively, analyze a .csv file or Pandas dataframe
  - Pros:
    - Flexible data preprocessing
    - You can use any type of analysis software (e.g., SPSS, MATLAB, R, scipy, etc.)
    - .csv files are portable and future-proof

Example: Privacy Tools Study

1. To what extent are people aware of these tools, and how frequently do they use them?
2. How interested are people in preventing specific privacy and security threats?
3. How accurately can people determine whether these tools afford protection from specific privacy and security threats?
4. What misconceptions, if any, do people have about these tools?

https://usableprivacy.org/static/files/story_popets_2021.pdf

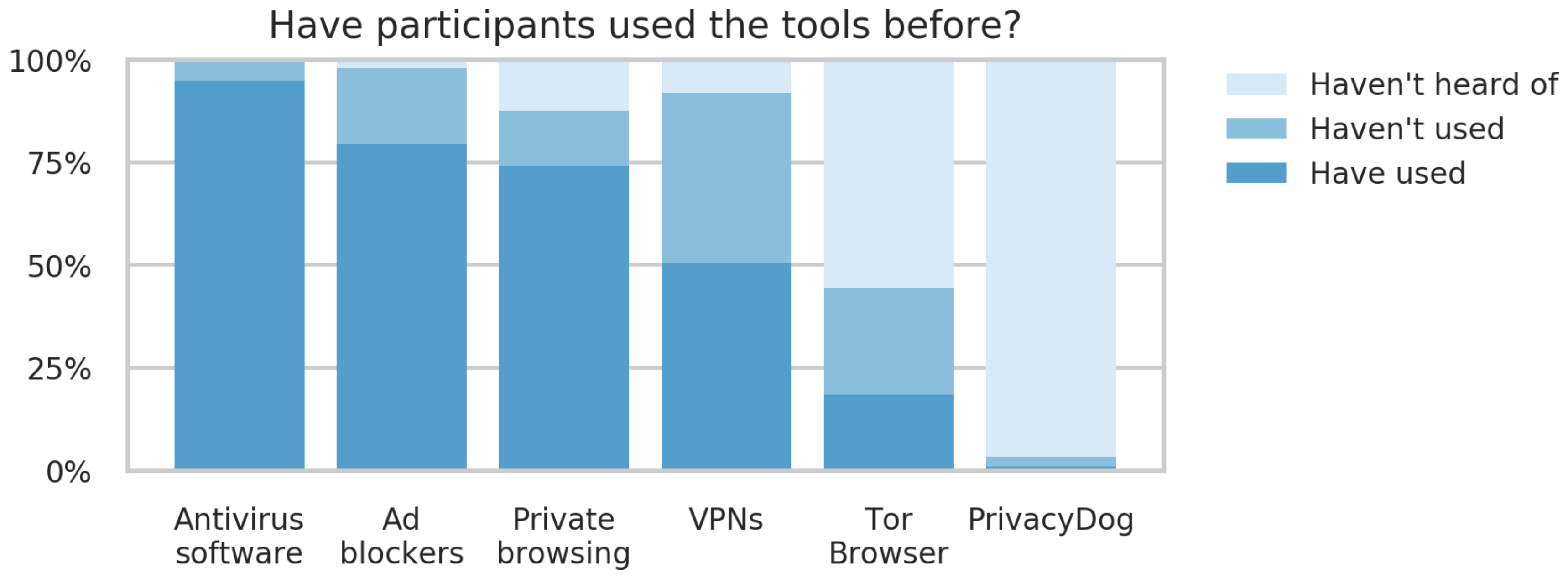
Protocol Overview

- We recruited ~500 participants from Prolific
- Participants took a Qualtrics survey
  - Django web app API randomized tools and assessment scenarios
- Survey data was automatically imported into the Django web app
- I reviewed written responses to ensure they were sufficiently high-quality
  - Only those who passed "attention checks" were paid

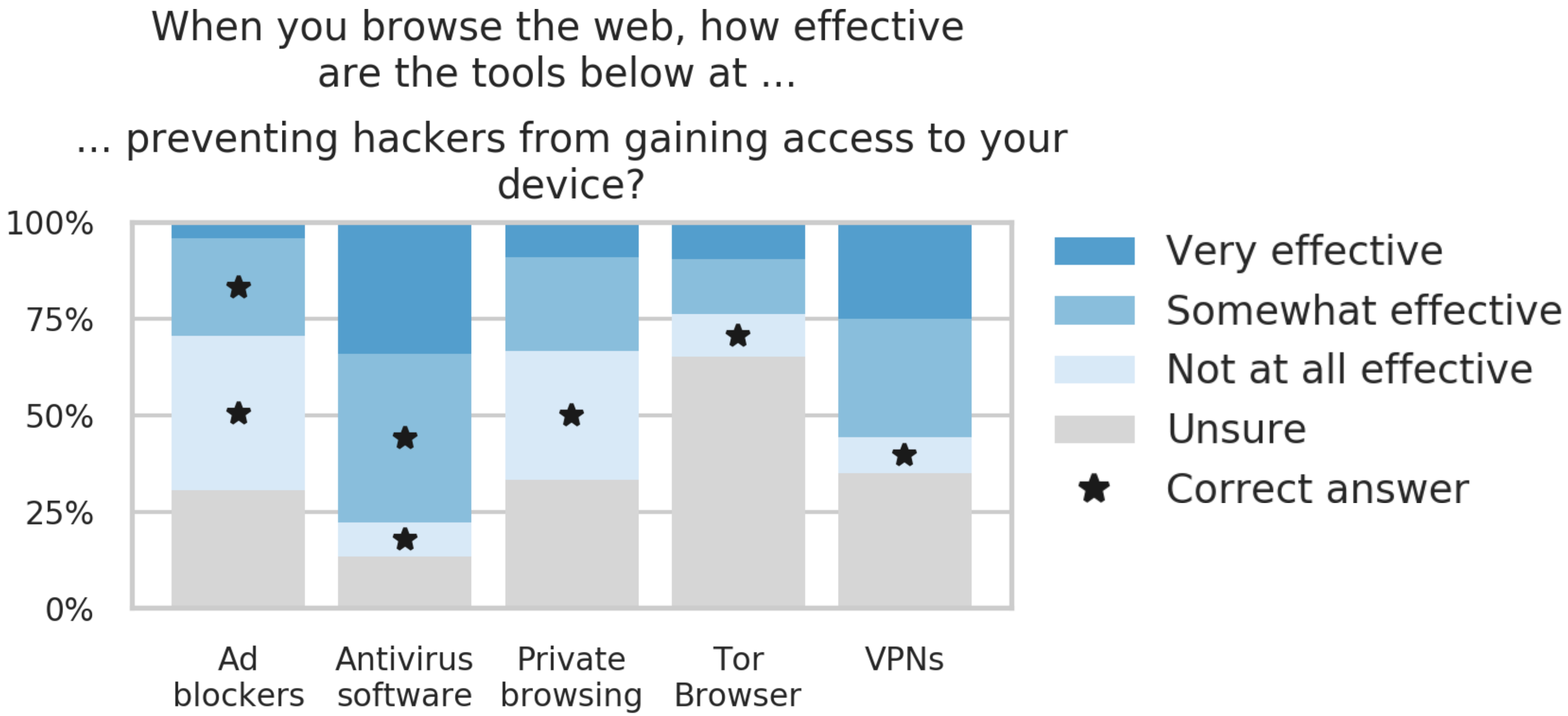
Data Analysis

- Data was exported from Django into an intermediate .csv file
  - Sensitive information was redacted (e.g., Prolific ID)
  - ~200 columns
- Additional preprocessing was performed, and a Pandas dataframe was created for analysis
  - For example, income ranges were converted to 0 to 6, the number of "correct" responses was calculated, etc.
  - ~500 columns
- Automation is essential: avoid errors and document preprocessing details!

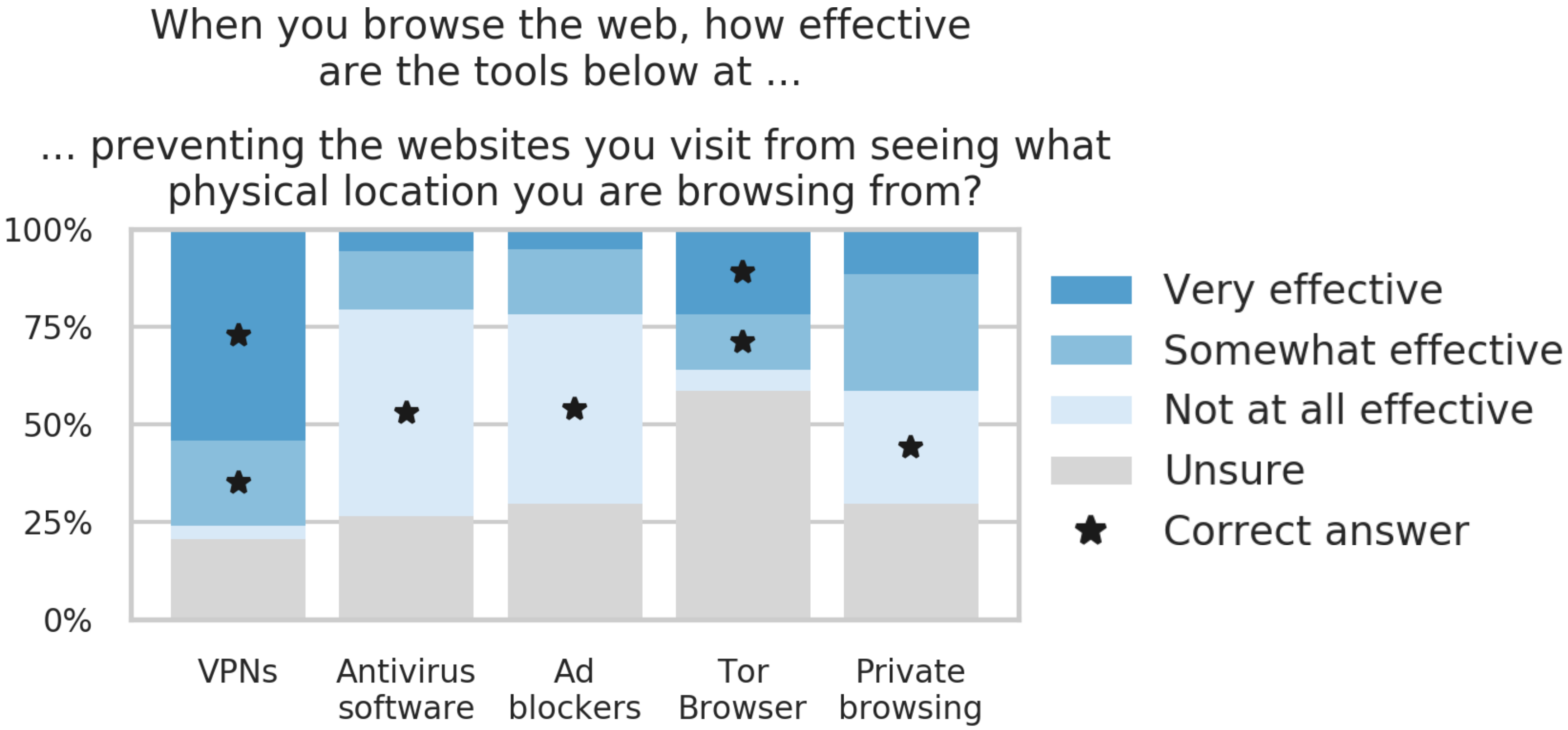
Tool Awareness

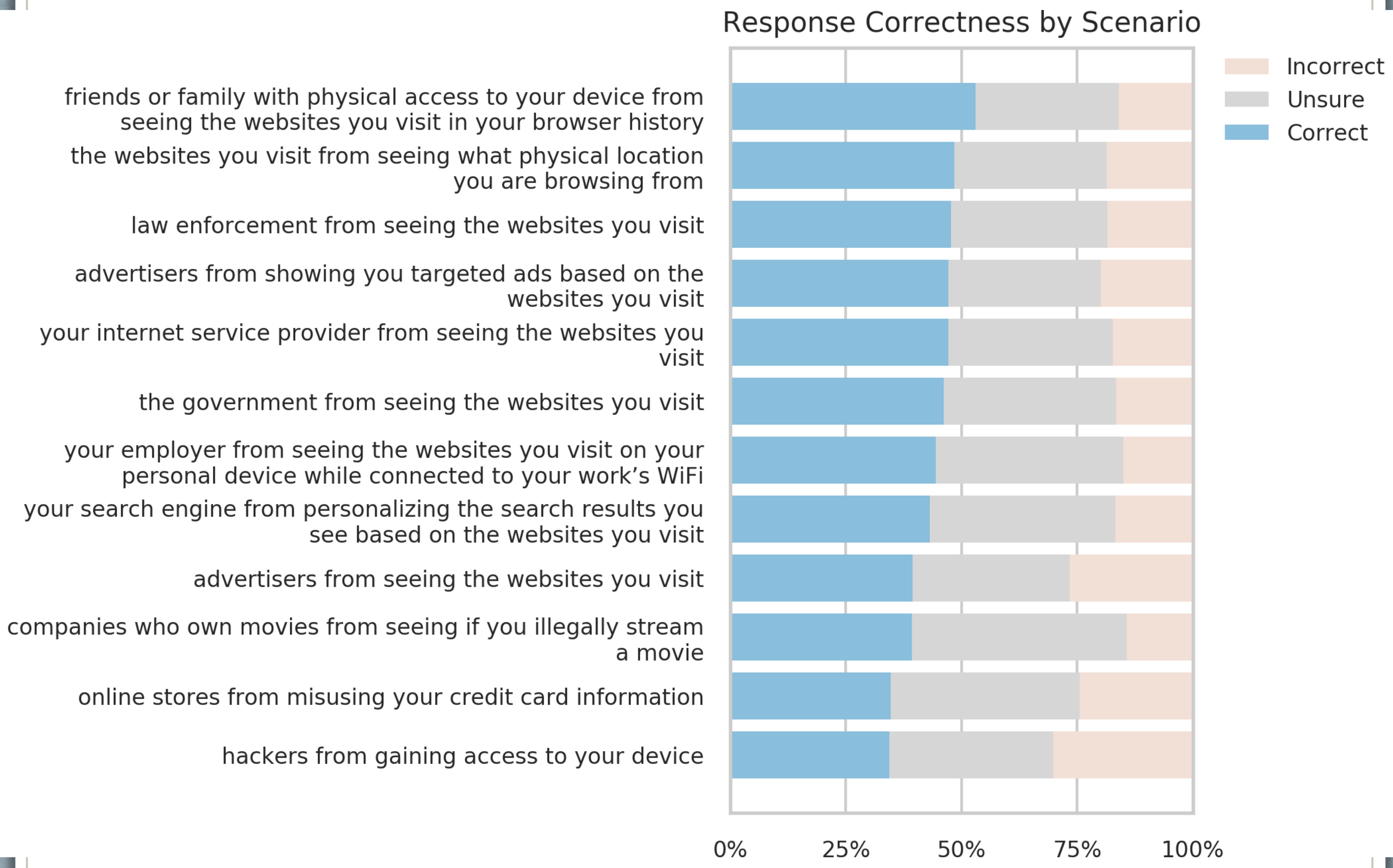


Tool Knowledge



Tool Knowledge




Response Correctness by Scenario
Incorrect
Unsure
Correct
friends or family with physical access to your device from seeing the websites you visit in your browser history
the websites you visit from seeing what physical location you are browsing from
law enforcement from seeing the websites you visit
advertisers from showing you targeted ads based on the websites you visit
your internet service provider from seeing the websites you visit
the government from seeing the websites you visit
your employer from seeing the websites you visit on your personal device while connected to your work’s WiFi
your search engine from personalizing the search results you see based on the websites you visit
advertisers from seeing the websites you visit
companies who own movies from seeing if you illegally stream a movie
online stores from misusing your credit card information
hackers from gaining access to your device
0%
25%
50%
75%
100%

Summary of Scripts

- load_survey.py
  - Automatically loads survey data from Qualtrics
  - Data is saved in a JSONField
- export_data.py
  - Exports intermediate .csv from database
- load.py
  - Uses intermediate .csv
  - Computes additional columns in a Pandas dataframe
  - Dataframe can be saved as a final .csv
- generate_graphs.py
  - Saves all the graphs needed for paper, based on the dataframe
- calculate_stats.py
  - Runs statistical tests and creates summary tables, based on the dataframe

Intermediate .csv

- id: 1627
- TOOL1: VPNs, TOOL2: Tor Browser, TOOL3: Ad blockers, TOOL4: Antivirus software, TOOL5: DuckDuckGo, TOOL6: Private browsing
- ASSESS1_1STPERSON: the websites I visit from seeing what physical location I am browsing from
- ASSESS1_2NDPERSON: the websites you visit from seeing what physical location you are browsing from
- ASSESS2_1STPERSON, …
- Q121_1: Very effective, Q121_2: Unsure, Q121_3: Unsure, Q121_4: Not at all effective, Q121_5: Unsure, Q121_6: Unsure
- …

Dataframe (final .csv)

- id: 1627
- websites_seeing_VPNS_efficacy: Very effective
- websites_seeing_TOR_BROWSER_efficacy: Unsure
- websites_seeing_AD_BLOCKERS_efficacy: Unsure
- websites_seeing_ANTIVIRUS_efficacy: Not at all effective
- websites_seeing_PRIVATE_BROWSING_efficacy: unsure
- …

Takeaways

- Think about all the ways you will use your data
- Where does the authoritative version of your data live?
    - On Qualtrics? In your database?
- If the authoritative version can't be used directly, automate data transformations
    - Otherwise, you might introduce errors into your data…