An Interactive Data Visualization Website

Startup Funding Analytics

December 13, 2022
Final Presentation — CIS550
Joseph Poirier, Lisa Friedmann, Purvansh Jain

Purpose

Problem

Though securing external funding is a pivotal factor that determines startup success, there aren't many studies that show how young businesses should look to attract investors.

Goal

To help investors and startup executives navigate the early-stage investment process.

Concept

An interactive online database that allows users to examine trends in startup investment.

Tech Stack

Backend

MySQL®

React

node JS

Frontend

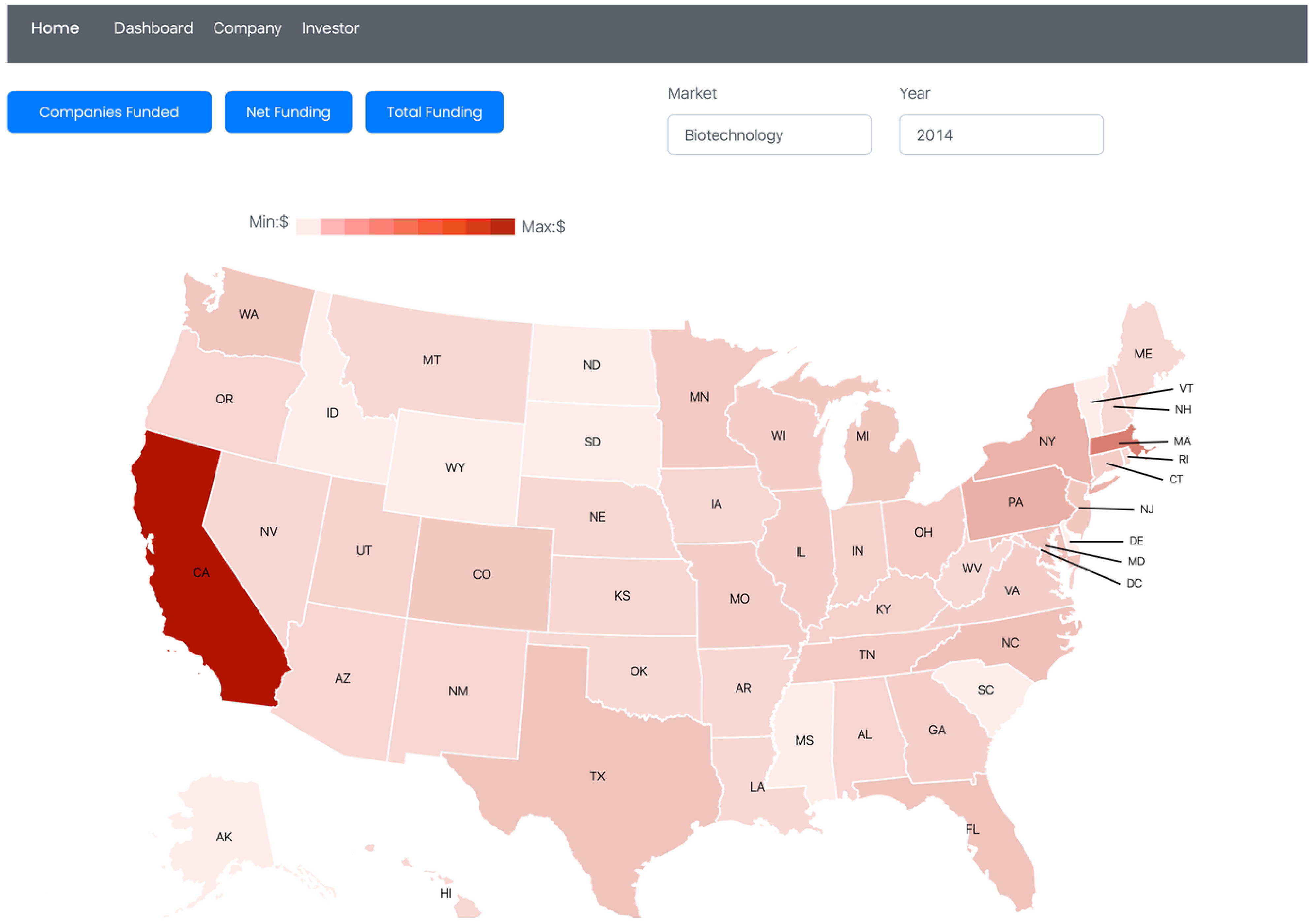
Preview

Sample User Questions:

Who invested in company XYZ?

Which funding rounds attracted the greatest proportion of international investment?

Which states attracted the mostnet funding for a given market?



Market: Biotechnology

Year: 2014

The "Dashboard" page, which allows for per-state KPI visualization.

The Dataset

2014 Startup Funding Dataset from Crunchbase

Data on investment in and acquisition of startups between 1960 and 2014

Companies.csv

18 columns x 49,439 rows

Contains company-specific information (city, country, founding_date) along with aggregated investment totals and calculated columns

Rounds.csv

16 columns x 83,871 rows

For each investment round, defined by company, investor, round, contains funding total and full information for both company and investor.

Investments.csv

22 columns x 13,701 rows

For each investment, contains investment details along with full company and investor information.

Acquisitions.csv

24 columns x 114,507 rows

For each acquisition, provide full details for acquired company and acquirer, along with details about the acquisition.

Technical Challenge #1

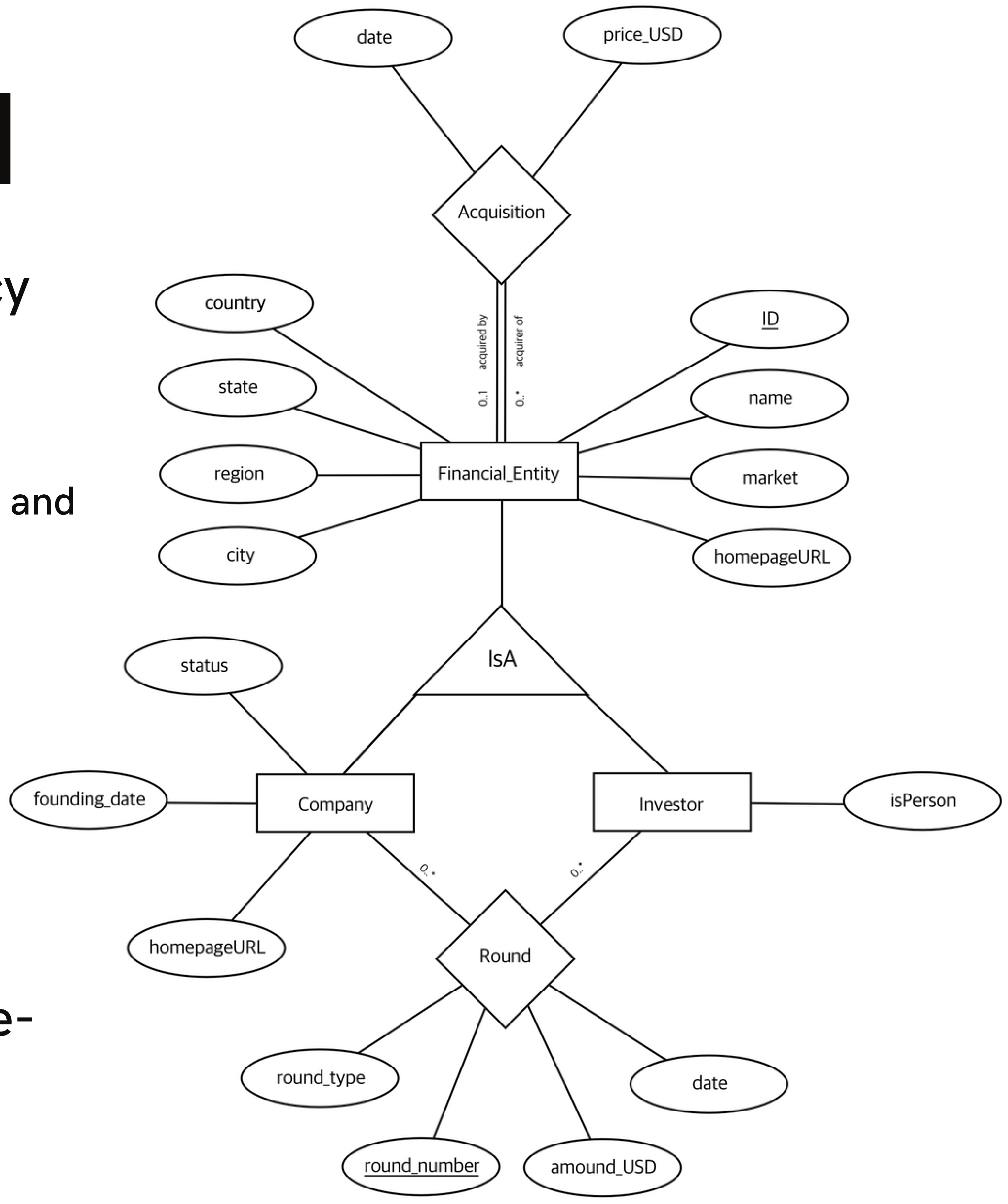
Efficiently reconfigure the dataset to eliminate redundancy and minimize storage requirements.

Key insight 1: there are three key "roles" in this dataset: company, investor, and acquirer. These share the vast majority (8) of their attributes.

Key insight 2: entities can be companies, investors, and acquirers, and can participate in each activity multiple times.

Key insight 3: acquisitions and rounds have only two and four of their own attributes, respectively

Solution: add all entities to a superclass and maintain type-specific information in subclass sets.



The New Dataset

2014 Startup Funding Dataset from Crunchbase

4.98M→ 2.25M

A 54.7 % reduction in database size.
New dataset is in Boyce-Codd Normal Form

Financial_Entity

9 columns x 117,205 rows

Generalized financial entity set, which removes calculated columns and allows for efficient relationship sets.

Company

4 columns x 83,871 rows

Reduced to a subset of financial_entity.

Investor

2 columns x 55,932 rows

Reduced to a subset of financial_entity

Acquisition

5 columns x 13,069 rows

Represents an acquisition. Relationship between two financial entities.

Round

6 columns x 114,507 rows

Represents a round of investment. Relationship between two financial entities.

Technical Challenge #2

Add interactive charts and onclick boxes to the website, which interface with the complex mySQL queries.

Since all of us are novices at web development, the first challenge in building the website was learning about node.js and figuring out how code blocks in the client/server side interfaced with one another.

Beyond the basics of node.js, which we were introduced to in homework 2, the graphics in this project required additional modules such as react-simple-maps

Handling invalid inputs was also a challenge, as many of the queries need to accommodate filtering via text searches. This added a degree of complexity to the query development process, as null values in complex aggregated queries can trigger errors.

Complex Query #1

Goal: find the investment round categories with the greatest proportion of international investment.

```
SELECT round_type,
       SUM(IF(c1.country != c2.country,1,0)) / COUNT(*) *
100 AS percentage_international
FROM Round r
    JOIN Financial_Entity c1 ON r.company_id = c1.id
    JOIN Financial_Entity c2 ON r.investor_id = c2.id
WHERE YEAR(date) = ${year} AND c1.market LIKE "%{market}%" AND c1.country IS NOT NULL
  AND c2.country IS NOT NULL AND c1.is_a = "company" AND c2.is_a = "investor"
GROUP BY round_type
ORDER BY SUM(IF(c1.country != c2.country,1,0)) / COUNT(*)
DESC;
```

Initial approach: merge round with financial_entity twice, matching each round with a company and an investor. Then, group by round type to get proportion of international investment.

2.11 seconds

Refinements

Moved selection, projection up (before joins).

Used CTEs to simplify tables before joining.

Projected 3/6 columns from Round

Projected 2/ 9 columns from Investor, and selected for market, year etc. before projection, reducing intermediate table sizes by a factor of 10-100 (depends on user-specified market, year).

```
WITH CleanedRound AS (
    SELECT company_ID, investor_ID, round_type, date
    FROM Round
),  CleanedCompany AS (
    SELECT id, country AS company_country
    FROM Financial_Entity
    WHERE is_a = "company" AND market LIKE "%" AND country IS NOT NULL
), CleanedInvestor AS (
    SELECT id, country AS investor_country
    FROM Financial_Entity
    WHERE is_a = "investor" AND country IS NOT NULL
)
SELECT round_type,
       SUM(IF(company_country != investor_country,1,0)) / COUNT(*) *
100 AS percentage_international
FROM CleanedRound r
    JOIN CleanedCompany c ON r.company_id = c.id
    JOIN CleanedInvestor i ON r.investor_id = i.id
WHERE YEAR(date) = ${year}
GROUP BY round_type
ORDER BY percentage_international
DESC;
```

888 ms (-58%)

Complex Query #2

Goal: find the total amount of investment by industry filtered on year, based on investment rounds. Return top 20 results, and aggregate additional funding into an "Other" category.

```
(SELECT market, SUM(amount_USD) AS total_funding
FROM Financial_Entity f JOIN (SELECT DISTINCT company_ID, round_number, date, amount_USD FROM Round) r ON f.ID = r.company_ID
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
GROUP BY market
ORDER BY total_funding DESC LIMIT 20)

UNION

(SELECT "Other" AS market, SUM(amount_USD) AS total_funding
FROM (SELECT * FROM
    Financial_Entity f JOIN (SELECT DISTINCT company_ID, round_number, date, amount_USD FROM Round) r ON f.ID = r.company_ID
    LIMIT 5000000 OFFSET 20) a
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
ORDER BY total_funding);
```

Initial approach: get distinct rounds, merge them with financial entities, and aggregate grouped by market. Do this twice, getting the top 20 results and the bottom len-20 results in other.

1.96 seconds

Refinement #1

Moved selection, projection up (before joins) (again).

Used CTEs to simplify tables before joining (again).

Replaced SELECT DISTINCT with GROUP BY without aggregation function for FUNDING_USD, since they're all the same within a group.

Moved duplicate filtering into CTE, so it's computed once instead of twice.

Moved Null filtering into CTE, also.

```sql
WITH cleaned_FE AS (
    SELECT market, ID
    FROM Financial_Entity
), cleaned_Round AS (
    SELECT company_ID, amount_USD, date
    FROM (
        SELECT company_ID, round_number, date, amount_USD
        FROM Round
        WHERE amount_USD IS NOT NULL
        GROUP BY company_ID, round_number, date) a
)

(SELECT market, SUM(amount_USD) AS total_funding
FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
GROUP BY market
ORDER BY total_funding DESC LIMIT 20)

UNION

(SELECT "Other" AS market, SUM(amount_USD) AS total_funding
FROM (SELECT *
    FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
    LIMIT 5000000 OFFSET 20) a
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
ORDER BY total_funding);
```

1.54 s (-21%)

Refinement #2: View

Kept refinements from last query, but performed selection, projection, and aggregation in a view.

While this code is much cleaner than the last query, it doesn't produce savings because non-materialized views have to be computed with each reference. A materialized view could have produced far better results here, but those aren't supported by MySQL.

```
CREATE VIEW Funding_by_Market AS
SELECT market, SUM(amount_USD) AS total_funding
FROM Financial_Entity f JOIN (SELECT company_ID, round_number, date, amount_USD
                               FROM Round
                               GROUP BY company_ID, round_number, date) r ON f.ID = r.company_ID
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
ORDER BY total_funding;

(SELECT *
FROM Funding_by_Market
ORDER BY total_funding DESC
LIMIT 20)
UNION
(SELECT "Other" AS market, SUM(total_funding) AS total_funding
FROM Funding_by_Market
LIMIT 5000000 OFFSET 20);
```

0.70 s View + 1.33 s Query = 2.03 s Query

Complex Query #3

Goal: for each distinct investor, return their base information (country, state, market, etc.) and the number of acquisitions and investments they've made. Return paginated result.

```sql
SELECT f.ID, f.name, IFNULL(f.market, "") AS market, IFNULL(f.country, "") AS country, IFNULL(f.state, "") AS state,
       IFNULL(f.city, "") AS city, IFNULL(i.is_person, "False") AS is_person, IFNULL(num_investments, 0) AS num_investments, IFNULL(num_acquisitions, 0) AS num_acquisitions
FROM Financial_Entity f LEFT JOIN Investor i ON f.ID = i.investor_ID LEFT JOIN (SELECT investor_ID, COUNT(*) AS num_investments FROM Round GROUP BY investor_ID) r
    ON i.investor_ID = r.investor_ID
LEFT JOIN (SELECT acquirer_ID, COUNT(*) AS num_acquisitions FROM Acquisition GROUP BY acquirer_ID) a ON f.ID = a.acquirer_ID
WHERE NOT ISNULL(num_investments) OR NOT ISNULL(num_acquisitions) AND
    name LIKE "%${name}%" AND
    market LIKE "%${market}%" AND
    country LIKE "%${country}%" AND
    state LIKE "%${state}%" AND
    city LIKE "%${city}%" AND
    is_person LIKE "%${is_person}%" AND
    num_investments >= ${num_investmentsLow} AND num_investments <= ${num_investmentsHigh} AND
    num_acquisitions >= ${num_acquisitionsLow} AND num_acquisitions <= ${num_acquisitionsHigh}
ORDER BY name
LIMIT ${start}, ${pagesize}
```

Initial approach: left join financial_entity, investor, round, and acquisition tables and group by investor ID to calculate columns.

1.42 seconds (filtering on "market")

Refinements

Moved selection, projection up (before joins).

Reordered joins to reduce intermediate table sizes.

Projected 3/6 columns from Round

Projected 2/ 9 columns from Investor, and selected for market, year etc. before projection, reducing intermediate table sizes by a factor of 10-100 (depends on user-specified market, year).

```
WITH cleaned_FE AS (
    SELECT market, ID
    FROM Financial_Entity
), cleaned_Round AS (
    SELECT company_ID, amount_USD, date
    FROM (
        SELECT company_ID, round_number, date, amount_USD
        FROM Round
        WHERE amount_USD IS NOT NULL
        GROUP BY company_ID, round_number, date) a
)

(SELECT market, SUM(amount_USD) AS total_funding
FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
GROUP BY market
ORDER BY total_funding DESC LIMIT 20)

UNION

(SELECT "Other" AS market, SUM(amount_USD) AS total_funding
FROM (SELECT *
    FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
    LIMIT 5000000 OFFSET 20) a
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
ORDER BY total_funding);
```

818 ms (-42%, filtered on "market")

Early Reduction of Financial_Entity

Field	Reduction Factor (RF)
State	0.016
Country	0.0081
Market	0.00076
City	0.00020
Name	0.000012

Application of filters to financial_entity before joining improves runtime efficiency by sharply reducing the number of tuples.

While performance for the revised query is similar to to the original for unfiltered searches, applying just one filter will reduce the computational cost of all the joins by a factor of 100 – 10,000.

As this is a search function, this is hugely important

Complex Query #4

Goal: Get the product categories that are acquired, as well as company trading information and the place of origin of each product. You can also see the organization's leadership structure, such as whether it is run by a single individual or a group.

Initial approach: merge round with financial_entity twice, matching each round with a company and an investor. Then, group by round type to get proportion of international investment.

```sql
SELECT categories,
    name AS Company_Name, country,
    CASE
        WHEN is_person = TRUE THEN 'Single_Investor'
        WHEN is_person = FALSE THEN 'Group_of_Investors'
        ELSE 'UNDEFINED'
    END AS Organisation_Type
FROM
    (SELECT
        TA.acquired_ID,
        MAX(price) AS max_price
    FROM Acquisition TA
    GROUP BY TA.acquired_ID) AS TA
JOIN Round RS ON RS.investor_id = TA.acquired_ID
JOIN Financial_Entity FES ON RS.investor_id = FES.id
JOIN Investor I ON RS.investor_id = I.investor_id;
```

1.69 seconds

Refinement #1

Moved projection of acquirer up using a CTE

Reorganized JOIN sequence to reduce intermediate table size.

```sql
WITH TopAquirer AS (
 SELECT acquirer_ID, acquired_ID, price
 FROM Acquisition
 ORDER BY price DESC
 )

SELECT categories,
 name AS Company_Name, country,
case
        when is_person = TRUE then 'Single_Investor'
        when is_person = FALSE then 'Group_of_Investors'
        else 'UNDEFINED'
    end AS Organisation_Type
FROM Round RS
JOIN TopAquirer TA ON RS.investor_id = TA.acquired_ID
JOIN Financial_Entity FES ON RS.investor_id = FES.id
JOIN Investor I ON RS.investor_id = I.investor_id;
```

952 ms (-43.56%)

Refinement #2

Added acquired id and max_price column and even though making it more optimized.

```
SELECT TA.acquired_ID, MAX(price) AS max_price, categories, name AS Company_Name,
    country,
    CASE
        WHEN is_person = TRUE THEN 'Single_Investor'
        WHEN is_person = FALSE THEN 'Group_of_Investors'
        ELSE 'UNDEFINED'
    END AS Organisation_Type
FROM Acquisition TA
JOIN Round RS ON RS.investor_id = TA.acquired_ID
JOIN Financial_Entity FES ON RS.investor_id = FES.id
JOIN Investor I ON RS.investor_id = I.investor_id
GROUP BY TA.acquired_ID;
```

597 ms (-64.61%)

Complex Query #5

Goal: for each distinct company, return basic information (name, market, country, city) and total_funding, aggregated from investment round data.

```sql
SELECT f.ID, f.name, IFNULL(f.market, "") AS market, IFNULL(f.country, "") AS country, IFNULL(f.state, "") AS state,
       IFNULL(f.city, "") AS city, IFNULL(i.is_person, "False") AS is_person, IFNULL(num_investments, 0) AS num_investments, IFNULL(num_acquisitions, 0) AS num_acquisitions
FROM Financial_Entity f LEFT JOIN Investor i ON f.ID = i.investor_ID LEFT JOIN (SELECT investor_ID, COUNT(*) AS num_investments FROM Round GROUP BY investor_ID) r
    ON i.investor_ID = r.investor_ID
LEFT JOIN (SELECT acquirer_ID, COUNT(*) AS num_acquisitions FROM Acquisition GROUP BY acquirer_ID) a ON f.ID = a.acquirer_ID
WHERE NOT ISNULL(num_investments) OR NOT ISNULL(num_acquisitions) AND
    name LIKE "%${name}%" AND
    market LIKE "%${market}%" AND
    country LIKE "%${country}%" AND
    state LIKE "%${state}%" AND
    city LIKE "%${city}%" AND
    is_person LIKE "%${is_person}%" AND
    num_investments >= ${num_investmentsLow} AND num_investments <= ${num_investmentsHigh} AND
    num_acquisitions >= ${num_acquisitionsLow} AND num_acquisitions <= ${num_acquisitionsHigh}
ORDER BY name
LIMIT ${start}, ${pagesize}
```

Initial approach: to get all of the relevant information, we need to join Financial_Entity, Company, and Round. In this approach, a CTE is used to calculate total funding of companies.

1.42 seconds (filtering on "market")

Refinements

Moved all selection on Financial_Entity up into a CTE (as with query #3).

Replaced the SELECT DISTINCT clause on Round with a GROUP BY, slightly improving efficiency.

Join reordering wasn't possible, since the initial implementation already did will to reduce intermediate size.

```
WITH cleaned_FE AS (
    SELECT market, ID
    FROM Financial_Entity
), cleaned_Round AS (
    SELECT company_ID, amount_USD, date
    FROM (
        SELECT company_ID, round_number, date, amount_USD
        FROM Round
        WHERE amount_USD IS NOT NULL
        GROUP BY company_ID, round_number, date) a
)

(SELECT market, SUM(amount_USD) AS total_funding
FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
GROUP BY market
ORDER BY total_funding DESC LIMIT 20)

UNION

(SELECT "Other" AS market, SUM(amount_USD) AS total_funding
FROM (SELECT *
    FROM cleaned_FE f JOIN cleaned_Round r ON f.ID = r.company_ID
    LIMIT 5000000 OFFSET 20) a
WHERE amount_USD IS NOT NULL AND YEAR(date) = ${year}
ORDER BY total_funding);
```

964 ms (-42%, filtered on "market")

THANK YOU!