

## Unicorn Companies Analysis: Industry Trends & Investment Insights

Did you know that the average return from investing in stocks is [10% per year](#) (not accounting for inflation)? But who wants to be average?!

You have been asked to support an investment firm by analyzing trends in high-growth companies. They are interested in understanding which industries are producing the highest valuations and the rate at which new high-value companies are emerging. Providing them with this information gives them a competitive insight as to industry trends and how they should structure their portfolio looking forward.

You have been given access to their `unicorns` database, which contains the following tables:

### dates

Column	Description
<code>company_id</code>	A unique ID for the company.
<code>date_joined</code>	The date that the company became a unicorn.
<code>year_founded</code>	The year that the company was founded.

### funding

Column	Description
<code>company_id</code>	A unique ID for the company.
<code>valuation</code>	Company value in US dollars.
<code>funding</code>	The amount of funding raised in US dollars.
<code>select_investors</code>	A list of key investors in the company.

### industries

Column	Description
<code>company_id</code>	A unique ID for the company.
<code>industry</code>	The industry that the company operates in.

### companies

Column	Description
<code>company_id</code>	A unique ID for the company.
<code>company</code>	The name of the company.
<code>city</code>	The city where the company is headquartered.
<code>country</code>	The country where the company is headquartered.
<code>continent</code>	The continent where the company is headquartered.

## Data Validation and Initial Exploration

Projects Data DataFrame as `data_types`

-- review the structure of all required tables.

```
SELECT table_name, column_name, data_type
FROM information_schema.columns
WHERE table_name IN ('dates', 'funding', 'industries', 'companies')
ORDER BY table_name, ordinal_position;
```

index	...	↑↓	table_name	...	↑↓	column_name	...	↑↓	data_type	...	↑↓
			0			companies			company_id		integer
			1			companies			company		character varying
			2			companies			city		character varying
			3			companies			country		character varying
			4			companies			continent		character varying
			5			dates			company_id		integer
			6			dates			date_joined		date
			7			dates			year_founded		integer
			8			funding			company_id		integer
			9			funding			valuation		bigint
			10			funding			funding		bigint
			11			funding			select_investors		character varying
			12			industries			company_id		integer
			13			industries			industry		character varying

Rows: 14

Expand

Projects Data DataFrame as `d`

--- Overview of table dates

```
SELECT *
FROM dates
LIMIT 5;
```

...	↑↓	c...	...	↑↓	date_joined	...	↑↓	year...	...	↑↓
					0			189		2017-06-24T00:00:00.000
					1			848		2021-06-01T00:00:00.000
					2			556		2022-02-15T00:00:00.000
					3			999		2021-11-17T00:00:00.000
					4			396		2021-10-21T00:00:00.000

Rows: 5

Expand

Projects Data DataFrame as `f`

--- Overview of table funding

```
SELECT *
FROM funding
LIMIT 5;
```

...	↑↓	c...	...	↑↓	v...	...	↑↓	f...	...	↑↓	select_investors	...	↑↓
					0			189			4000000000		0
					1			848			1000000000		100000000
					2			556			2000000000		100000000
					3			999			1000000000		100000000
					4			396			2000000000		100000000

Rows: 5

Expand

Projects Data DataFrame as

--- Overview of table industries

```
SELECT *
FROM industries
LIMIT 5;
```

...	↑↓	c...	...	↑↓	industry	...	↑↓
0		189			Health		
1		848			Fintech		
2		556			Internet software & services		
3		999			Internet software & services		
4		396			Fintech		

Rows: 5

[Expand](#)

Projects Data DataFrame as

--- Overview of table companies

```
SELECT *
FROM companies
LIMIT 5;
```

...	↑↓	c...	...	↑↓	company	...	↑↓	city	...	↑↓	country	...	↑↓	contin...	...	↑↓
0		189			Otto Bock HealthCare			Duderstadt			Germany			Europe		
1		848			Matrixport						Singapore			Asia		
2		556			Cloudinary			Santa Clara			United States			North America		
3		999			PLACE			Bellingham			United States			North America		
4		396			candy.com			New York			United States			North America		

Rows: 5

[Expand](#)

Projects Data DataFrame as

--- Checks the minimum and maximum values for date\_joined and year\_founded.

```
SELECT
    MIN(date_joined) AS min_date_joined,
    MAX(date_joined) AS max_date_joined,
    MIN(year_founded) AS min_year_founded,
    MAX(year_founded) AS max_year_founded
FROM dates;
```

...	↑↓	min_date_joined	...	↑↓	max_date_joined	...	↑↓	min_year_f...	...	↑↓	max_year_f...	...	↑↓
0		2007-07-02T00:00:00.000			2022-04-05T00:00:00.000			1919			2021		

Rows: 1

[Expand](#)

Projects Data DataFrame as

--- Checks for missing values in key date-related fields.

```
SELECT date_joined, year_founded
FROM dates
WHERE date_joined IS NULL
    AND year_founded IS NULL;
```

Your query ran successfully but returned no results.

Projects Data DataFrame as

--- Checks the minimum, maximum, and average values for valuation and funding.

```
SELECT
    MIN(valuation) AS min_value,
    MAX(valuation) AS max_value,
    AVG(valuation) AS avg_value,
    MIN(funding) AS min_fund,
    MAX(funding) AS max_fund,
    AVG(funding) AS avg_fund
FROM funding;
```

...	↑↓	m...	...	↑↓	max...	...	↑↓	avg_value	...	↑↓	...	↑↓	ma...	...	↑↓	avg_fund	...	↑↓
0		1000000000			180000000000			3455307262.5698323			0		14000000000			551042830.5400373		

Rows: 1

↗ Expand

Projects Data DataFrame as

--- Checks whether any records are missing valuation, funding, or investor information.

```
SELECT valuation, funding
FROM funding
WHERE valuation IS NULL
    AND funding IS NULL
    AND select_investors IS NULL;
```

Your query ran successfully but returned no results.

Projects Data DataFrame as

--- Checks for missing industry classifications.

```
SELECT industry
FROM industries
WHERE industry IS NULL;
```

Your query ran successfully but returned no results.

Projects Data DataFrame as

--- Checks whether any records are missing the company name, city, country, or continents.

```
SELECT company, city, country, continent
FROM companies
WHERE company IS NULL
    AND city IS NULL
    AND country IS NULL
    AND continent IS NULL;
```

Your query ran successfully but returned no results.

## Industry Concentration and High-Growth Sectors

Projects Data DataFrame as

```
-- Identify the top industries with the highest number of unicorn companies formed between 2019-2021
```

```
SELECT
    i.industry,
    COUNT(*) AS num_unicorns
FROM industries AS i
INNER JOIN dates AS d USING(company_id)
WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
GROUP BY i.industry
ORDER BY num_unicorns DESC
LIMIT 5;
```

...	↑↓	industry	...	↑↓	num...	...	↑↓
0		Fintech			173		
1		Internet software & services			152		
2		E-commerce & direct-to-consumer			75		
3		Artificial intelligence			53		
4		Health			52		

Rows: 5

[↗ Expand](#)

CTE1\_top\_performing\_industries\_ ▼

Projects Data DataFrame as

```
-- Analyze how unicorn creation and average valuation vary by industry and year
```

```
SELECT
  i.industry,
  EXTRACT(YEAR FROM d.date_joined) AS year,
  COUNT(i.company_id) AS num_unicorns,
  ROUND(AVG(f.valuation), 2) AS avg_valuation
FROM industries AS i
INNER JOIN dates AS d USING(company_id)
INNER JOIN funding AS f USING(company_id)
WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
GROUP BY i.industry, year
ORDER BY year ASC;
```

...	↑↓	industry	...	↑↓	...	↑↓	num...	...	↑↓	avg_v...	...	↑↓
0		Artificial intelligence			2019		14			4500000000		
1		Auto & transportation			2019		6			4166666666.67		
2		Consumer & retail			2019		3			3666666666.67		
3		Cybersecurity			2019		4			2250000000		
4		Data management & analytics			2019		4			11500000000		
5		E-commerce & direct-to-consumer			2019		12			2583333333.33		
6		Edtech			2019		1			1000000000		
7		Fintech			2019		20			6800000000		
8		Health			2019		3			3333333333.33		
9		Internet software & services			2019		13			4230769230.77		
10		Mobile & telecommunications			2019		4			2000000000		
11		Other			2019		9			2888888888.89		
12		Supply chain, logistics, & delivery			2019		8			3000000000		
13		Travel			2019		3			4000000000		
14		Artificial intelligence			2020		3			4000000000		
15		Auto & transportation			2020		5			3000000000		

Rows: 43

Expand

## Top Performing Industries and Valuation Trends (CTE-Based Analysis)

Projects Data DataFrame as

```

--- CTE1 top_performing_industries

-- Select the top 3 industries based on total unicorn count

WITH top_performing_industries AS (
  SELECT
    i.industry,
    COUNT(*) AS number_of_companies
  FROM industries AS i
  INNER JOIN dates AS d USING(company_id)
  WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
  GROUP BY i.industry
  ORDER BY number_of_companies DESC
  LIMIT 3
),

-----

--- CTE2 top_valuation

-- Calculate yearly unicorn counts and average valuations for each industry

top_valuation AS (
  SELECT
    i.industry,
    EXTRACT(YEAR FROM d.date_joined) AS year,
    COUNT(i.company_id) AS num_unicorns,
    ROUND(AVG(f.valuation), 2) AS avg_valuation
  FROM industries AS i
  INNER JOIN dates AS d USING(company_id)
  INNER JOIN funding AS f USING(company_id)
  WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
  GROUP BY i.industry, year
)

-----

--- Final Query

-- Focus valuation analysis on top-performing industries only

SELECT
  industry,
  year,
  num_unicorns,
  ROUND(AVG(avg_valuation) / 1000000000, 2) AS average_valuation_billions
FROM top_valuation
INNER JOIN top_performing_industries USING(industry)
GROUP BY industry, year, num_unicorns
ORDER BY year DESC, num_unicorns DESC;

```

...	↑↓	industry	...	↑↓	...	↑↓	num...	...	↑↓	average_valuation_billions	...	↑↓
0		Fintech			2021			138		2.75		
1		Internet software & services			2021			119		2.15		
2		E-commerce & direct-to-consumer			2021			47		2.47		
3		Internet software & services			2020			20		4.35		
4		E-commerce & direct-to-consumer			2020			16		4		
5		Fintech			2020			15		4.33		
6		Fintech			2019			20		6.8		
7		Internet software & services			2019			13		4.23		
8		E-commerce & direct-to-consumer			2019			12		2.58		

Rows: 9

Expand

## Speed of Value Creation (Time to Unicorn)

Projects Data DataFrame as

```
-- Measure how quickly companies reach unicorn status after being founded, by industry
```

```
SELECT
  i.industry,
  COUNT(*) AS num_unicorns,
  ROUND(AVG(EXTRACT(YEAR FROM d.date_joined) - d.year_founded)::NUMERIC, 2) AS avg_years_to_unicorn,
  ROUND(
    PERCENTILE_CONT(0.5)
    WITHIN GROUP (ORDER BY EXTRACT(YEAR FROM d.date_joined) - d.year_founded)::NUMERIC,
    2
  ) AS median_years_to_unicorn
FROM industries i
JOIN dates d USING(company_id)
WHERE d.year_founded IS NOT NULL
GROUP BY i.industry
ORDER BY avg_years_to_unicorn ASC;
```

...	↑↓	industry	...	↑↓	num...	...	↑↓	avg_years_to_unic...	...	↑↓	median_years_to_unico...	...	↑↓	
	0	Auto & transportation			31			5.03			4			
	1	Artificial intelligence			84			5.89			5			
	2	Hardware			34			5.94			5			
	3	Mobile & telecommunications			38			6.32			5			
	4	E-commerce & direct-to-consumer			111			6.43			6			
	5	Fintech			224			6.54			6			
	6	Travel			14			6.57			5			
	7	Cybersecurity			50			6.8			6			
	8	Supply chain, logistics, & delivery			57			6.93			6			
	9	Edtech			28			7.71			6			
	10	Other			58			7.78			6.5			
	11	Internet software & services			205			7.87			7			
	12	Data management & analytics			41			8.07			8			
	13	Consumer & retail			25			8.08			5			
	14	Health			74			8.19			6			

Rows: 15

Expand

## Capital Efficiency Analysis

Projects Data DataFrame as

```
-- Evaluate capital efficiency by comparing valuation generated per dollar of funding
```

```
SELECT
  i.industry,
  COUNT(*) AS num_unicorns,
  ROUND(AVG(f.valuation / NULLIF(f.funding, 0))::NUMERIC, 2) AS avg_valuation_funding_ratio,
  ROUND(
    PERCENTILE_CONT(0.5)
    WITHIN GROUP (ORDER BY f.valuation / NULLIF(f.funding, 0))::NUMERIC,
    2
  ) AS median_valuation_funding_ratio
FROM funding f
JOIN industries i USING(company_id)
WHERE f.funding > 0
GROUP BY i.industry
ORDER BY avg_valuation_funding_ratio DESC;
```

...	↑↓	industry	...	↑↓	num...	...	↑↓	avg_valuation_funding_ratio	...	↑↓	median_valuation_funding_ratio	...	↑↓
0		Internet software & services			203			28.3					
1		Other			56			8.98					
2		Mobile & telecommunications			37			8.92					
3		Fintech			222			8.72					
4		Hardware			34			7.59					
5		Consumer & retail			24			7.5				3.	
6		Data management & analytics			40			6.25				5.	
7		Artificial intelligence			84			6				4.	
8		Health			73			5.82					
9		E-commerce & direct-to-consumer			109			5.63					
10		Cybersecurity			50			5.58					
11		Edtech			28			5.39					
12		Auto & transportation			31			3.97					
13		Supply chain, logistics, & delivery			56			3.32					
14		Travel			14			2.93				2.	

Rows: 15

Expand

## Year-over-Year Unicorn Growth Trends

Projects Data DataFrame as

```
-- Count unicorns created each year within each industry
```

```
WITH unicorns_by_year AS (
  SELECT
    i.industry,
    EXTRACT(YEAR FROM d.date_joined) AS year,
    COUNT(*) AS num_unicorns
  FROM industries i
  JOIN dates d USING(company_id)
  WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
  GROUP BY i.industry, year
)
SELECT *
FROM unicorns_by_year
ORDER BY industry, year;
```

...	↑↓	industry	...	↑↓	...	↑↓	num...	...	↑↓
0		Artificial intelligence			2019		14		
1		Artificial intelligence			2020		3		
2		Artificial intelligence			2021		36		
3		Auto & transportation			2019		6		
4		Auto & transportation			2020		5		
5		Auto & transportation			2021		4		
6		Consumer & retail			2019		3		
7		Consumer & retail			2020		1		
8		Consumer & retail			2021		7		
9		Cybersecurity			2019		4		
10		Cybersecurity			2020		7		
11		Cybersecurity			2021		27		
12		Data management & analytics			2019		4		
13		Data management & analytics			2020		6		
14		Data management & analytics			2021		21		
15		E-commerce & direct-to-consumer			2019		12		

Rows: 43

[↗ Expand](#)

Projects Data DataFrame as

```
-- Calculate year-over-year growth rate in unicorn creation for each industry

WITH unicorns_by_year AS (
  SELECT
    i.industry,
    EXTRACT(YEAR FROM d.date_joined) AS year,
    COUNT(*) AS num_unicorns
  FROM industries i
  JOIN dates d USING(company_id)
  WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
  GROUP BY i.industry, year
),

yoy_growth AS (
  SELECT
    industry,
    year,
    num_unicorns,
    LAG(num_unicorns) OVER (PARTITION BY industry ORDER BY year) AS prev_year_unicorns
  FROM unicorns_by_year
)

SELECT
  industry,
  year,
  num_unicorns,
  prev_year_unicorns,
  ROUND(
    (num_unicorns - prev_year_unicorns) * 100.0 / NULLIF(prev_year_unicorns, 0),
    2
  ) AS yoy_growth_percentage
FROM yoy_growth
ORDER BY yoy_growth_percentage DESC;
```

...	↑↓	industry	...	↑↓	...	↑↓	num...	...	↑↓	prev_year_unic...	...	↑↓	yoy_growth_percent...	...	⌵
15		Hardware			2021		14			1			1300		
16		Supply chain, logistics, & delivery			2021		25			2			1150		
17		Artificial intelligence			2021		36			3			1100		
18		Fintech			2021		138			15			820		
19		Consumer & retail			2021		7			1			600		
20		Internet software & services			2021		119			20			495		
21		Health			2021		40			9			344.44		
22		Edtech			2020		4			1			300		
23		Cybersecurity			2021		27			7			285.71		
24		Data management & analytics			2021		21			6			250		
25		Edtech			2021		12			4			200		
26		Health			2020		9			3			200		
27		E-commerce & direct-to-consumer			2021		47			16			193.75		
28		Mobile & telecommunications			2020		8			4			100		
29		Other			2021		21			11			90.91		
30		Cybersecurity			2020		7			4			75		

Rows: 43

Expand

Projects Data DataFrame as

```
-- Calculate year-over-year growth rate in unicorn creation for top 10 industry

WITH unicorns_by_year AS (
    SELECT
        i.industry,
        EXTRACT(YEAR FROM d.date_joined) AS year,
        COUNT(*) AS num_unicorns
    FROM industries i
    JOIN dates d USING(company_id)
    WHERE EXTRACT(YEAR FROM d.date_joined) IN (2019, 2020, 2021)
    GROUP BY i.industry, year
),

yoy_growth AS (
    SELECT
        industry,
        year,
        num_unicorns,
        LAG(num_unicorns) OVER (PARTITION BY industry ORDER BY year) AS prev_year_unicorns
    FROM unicorns_by_year
),

top_industries AS (
    SELECT
        industry
    FROM yoy_growth
    WHERE year = 2021
    ORDER BY
        (num_unicorns - prev_year_unicorns) * 1.0 / prev_year_unicorns DESC
    LIMIT 10
)

SELECT
    u.industry,
    u.year,
    u.num_unicorns
FROM unicorns_by_year u
JOIN top_industries t
    ON u.industry = t.industry
ORDER BY u.industry, u.year;
```

...	↑↓	industry	...	↑↓	...	↑↓	num...	...	↑↓
0		Artificial intelligence					2019		14
1		Artificial intelligence					2020		3
2		Artificial intelligence					2021		36
3		Consumer & retail					2019		3
4		Consumer & retail					2020		1
5		Consumer & retail					2021		7
6		Cybersecurity					2019		4
7		Cybersecurity					2020		7
8		Cybersecurity					2021		27
9		Data management & analytics					2019		4
10		Data management & analytics					2020		6
11		Data management & analytics					2021		21
12		Edtech					2019		1
13		Edtech					2020		4
14		Edtech					2021		12
15		Fintech					2019		20

Rows: 29

[Expand](#)

top\_10\_YoY\_growth ▾

## Valuation Concentration and Industry Risk

Projects Data DataFrame as

-- Compare average and median valuations to assess valuation concentration within industries

```
SELECT
  i.industry,
  COUNT(*) AS num_unicorns,
  ROUND(AVG(f.valuation / 1000000000)::NUMERIC, 2) AS avg_valuation,
  ROUND(
    PERCENTILE_CONT(0.5)
    WITHIN GROUP (ORDER BY f.valuation / 1000000000)::NUMERIC,
    2
  ) AS median_valuation
FROM funding f
JOIN industries i USING(company_id)
GROUP BY i.industry
ORDER BY avg_valuation;
```

...	↑↓	industry	...	↑↓	num...	...	↑↓	avg_v...	...	↑↓	median_val...	...	↑↓
0		Mobile & telecommunications			38			2.34			1.5		
1		Cybersecurity			50			2.58			2		
2		Health			74			2.68			2		
3		Internet software & services			205			2.9			2		
4		Hardware			34			2.91			2		
5		Supply chain, logistics, & delivery			57			3.11			1		
6		Auto & transportation			31			3.19			2		
7		Travel			14			3.29			2		
8		Data management & analytics			41			3.32			2		
9		Edtech			28			3.57			2		
10		E-commerce & direct-to-consumer			111			3.84			2		
11		Fintech			224			3.94			2		
12		Consumer & retail			25			4.24			2		
13		Other			58			4.34			2		
14		Artificial intelligence			84			4.49			2		

Rows: 15

Expand

Projects Data DataFrame as

```
-- Measure how much of an industry's total valuation is controlled by the top 10% of companies
```

```
WITH ranked_valuations AS (
  SELECT
    i.industry,
    f.valuation,
    NTILE(10) OVER (PARTITION BY i.industry ORDER BY f.valuation DESC) AS valuation_decile
  FROM industries i
  JOIN funding f USING(company_id)
),

industry_totals AS (
  SELECT
    industry,
    SUM(valuation) AS total_valuation
  FROM ranked_valuations
  GROUP BY industry
)
SELECT
  r.industry,
  ROUND(
    SUM(CASE WHEN valuation_decile = 1 THEN valuation ELSE 0 END)
    / t.total_valuation * 100,
    2
  ) AS top_10_percent_valuation_share
FROM ranked_valuations r
JOIN industry_totals t USING(industry)
GROUP BY r.industry, t.total_valuation
ORDER BY top_10_percent_valuation_share DESC;
```

...	↑↓	industry	...	↑↓	top_10_percent_valuation_share	...	↑↓
	0	Other			65.48		
	1	Artificial intelligence			64.99		
	2	Consumer & retail			60.38		
	3	E-commerce & direct-to-consumer			55.63		
	4	Supply chain, logistics, & delivery			53.11		
	5	Fintech			50		
	6	Edtech			49		
	7	Data management & analytics			47.79		
	8	Travel			41.3		
	9	Internet software & services			38.99		
	10	Auto & transportation			38.38		
	11	Hardware			36.36		
	12	Health			33.33		
	13	Cybersecurity			31.78		
	14	Mobile & telecommunications			31.46		

Rows: 15

Expand

## Valuation Stability and Market Volatility

Projects Data DataFrame as

```
-- Track changes in average valuation and dispersion to assess market stability over time  
-- Use coefficient of variation to normalize valuation volatility across years
```

SELECT

```
EXTRACT(YEAR FROM d.date_joined) AS year,  
COUNT(*) AS num_unicorns,  
ROUND(AVG(f.valuation) / 1000000000, 2) AS avg_valuation_billions,  
ROUND(STDDEV(f.valuation) / 1000000000, 2) AS stddev_valuation_billions,  
ROUND(STDDEV(f.valuation) / AVG(f.valuation), 2) AS valuation_volatility_ratio
```

FROM dates d

JOIN funding f USING(company\_id)

WHERE EXTRACT(YEAR FROM d.date\_joined) IN (2019, 2020, 2021)

GROUP BY year

ORDER BY year;

...	↑↓	...	↑↓	num...	...	↑↓	avg_valuation_billions	...	↑↓	stddev_valuation_billions	...	↑↓	valuation_volatility_ratio	...	↑↓
0		2019		104			4.39			6.12			1.39		
1		2020		108			3.72			2.99			0.8		
2		2021		520			2.29			2.44			1.07		

Rows: 3

[Expand](#)

avg\_stddev\_valuation\_ratio ▾

