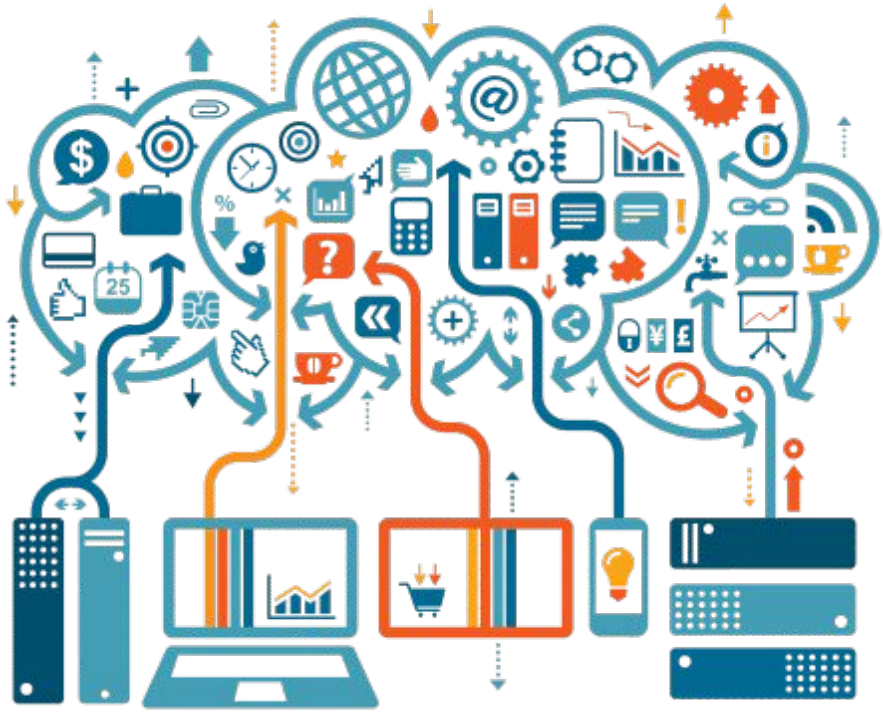
Module 2: Data Engineering



MySQL Guide

Hey Mr. DB, can you show me a list of our quarterly sales by department and region? K. Thanks.

??????

Database - Organized collection of structured data
SQL - Query language used to communicate with the database

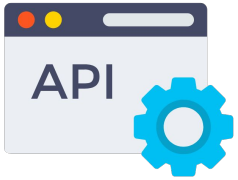
Structured Query Language

Database Server Hosting - *Where does the database (and the data) reside?*

Local

On-Premises

Cloud

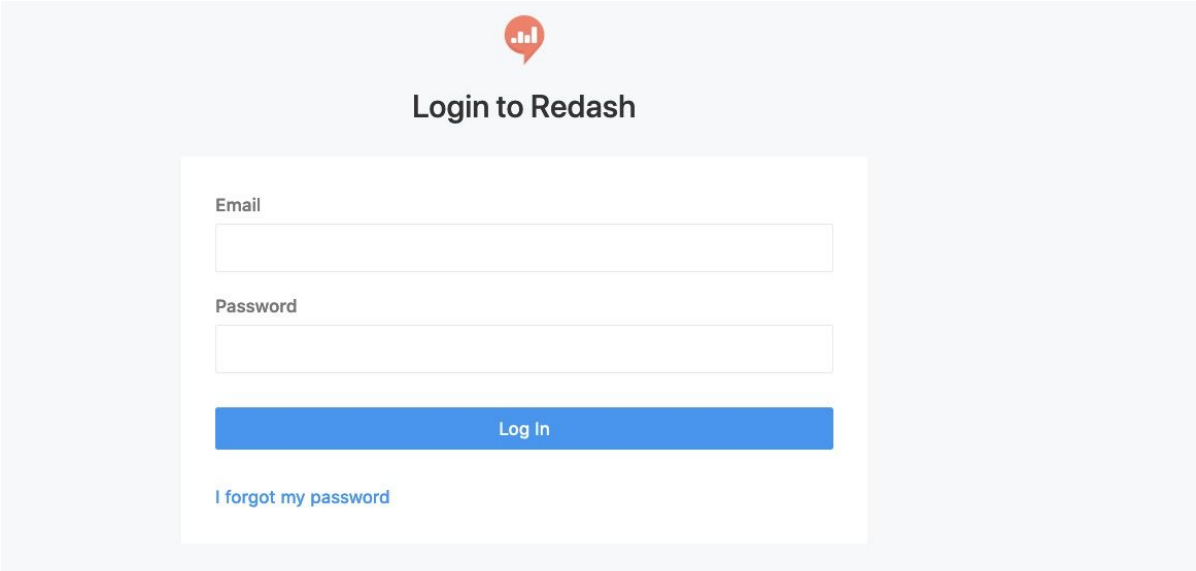


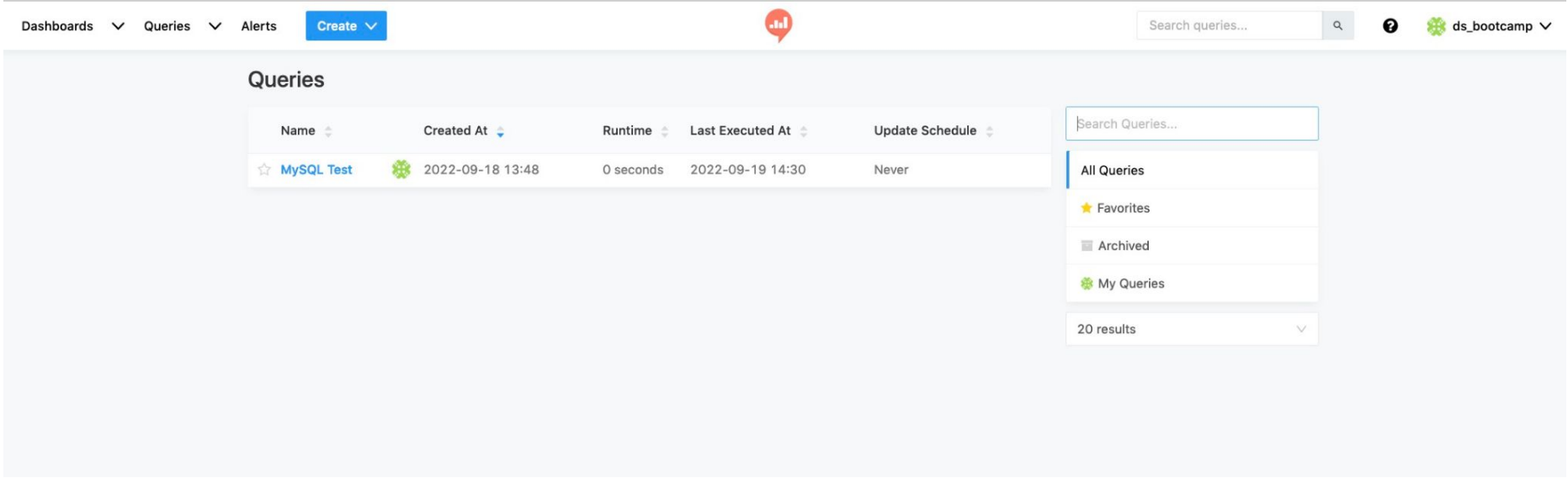
Web Service

Structured Query Language

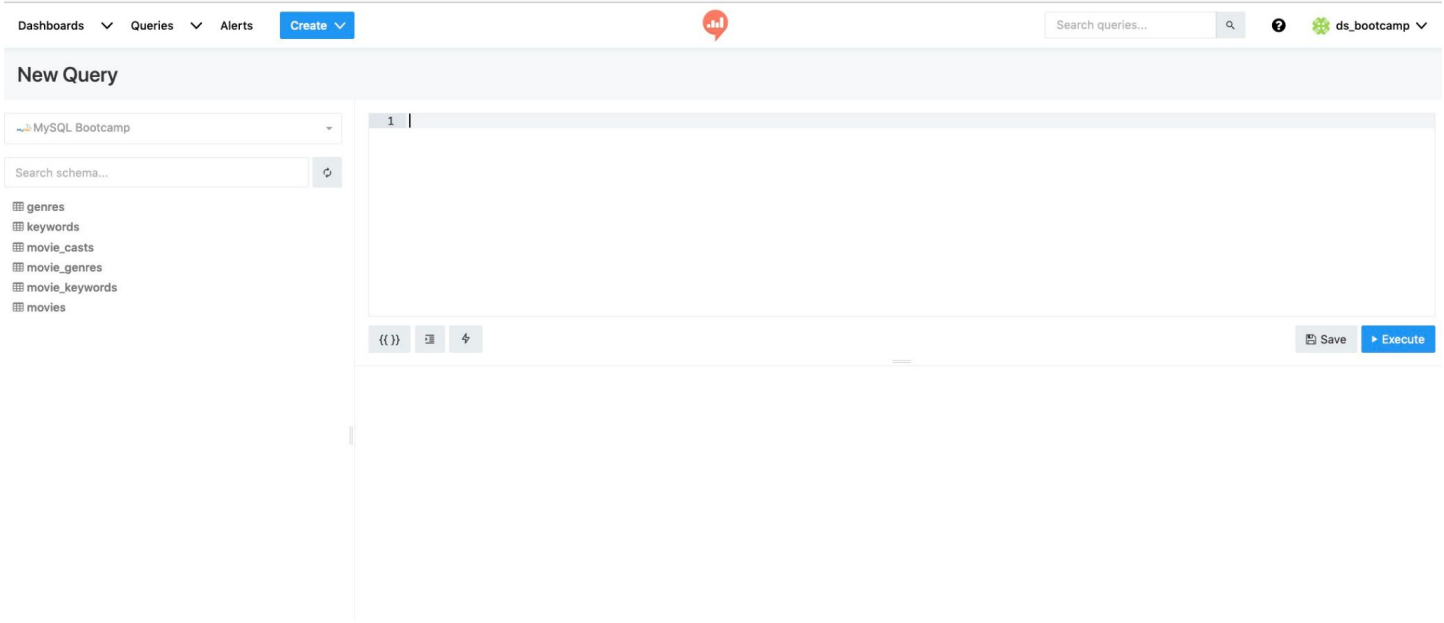
Hands-On Activities

- Login to: https://tinyurl.com/sql-bootcamp
  Email/Password: ds@bootcamp.com





Click on **Create** > **Query**


Dashboards
Queries
Alerts
Create
Search queries...
ds_bootcamp
New Query
MySQL Bootcamp
Search schema...
genres
keywords
movie_casts
movie_genres
movie_keywords
movies
1
{{}}
Save
Execute

Exploration

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
1	SHOW TABLES;	Show the tables in the database
2	DESCRIBE movies;	Show column names and types in movies table
3	SELECT * FROM movies;	Select ALL columns in movies (note: (*) means every column)
4	SELECT title, release_date FROM movies;	Select specific columns in movies
5	SELECT DISTINCT original_language FROM movies	Select unique film original language values

Exercise #1

A. List the name, tagline, release date and runtime of movies

B. What are the column types in the **movie_casts** table?

Limiting and sorting

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
5	SELECT title, release_date FROM movies LIMIT 10;	Get 10 rows only
6	SELECT title, release_date FROM movies ORDER BY release_date;	Sort films by release date (ascending)
7	SELECT title, release_date FROM movies ORDER BY release_date DESC;	Sort films by release date (descending)
8	SELECT title, release_date from movies ORDER BY release_date DESC LIMIT 10;	Get top latest 10 films

Exercise #2

A. List the name, tagline, release date and popularity of movies with the highest popularity score. Show the top 25 movies.

B. List the movie name, average votes and # of votes with highest vote average. Show the top 25 movies
Bonus: Add secondary sort by # of votes

Built-in functions

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
9	SELECT MIN(release_date), MAX(release_date), AVG(budget) FROM movies;	Get min, max and average of column values
10	SELECT COUNT(*) FROM movies;	Get number of movies in movies table. **(*) means every row**
11	SELECT title, release_date, YEAR(release_date) FROM movies;	Show the year of release_date
12	SELECT title, release_date, MONTH(release_date) from movies;	Show the month of release_date

Arithmetic

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
13	SELECT title, runtime, runtime/60 FROM movies;	Divide runtime by 60 to get runtime **hours**
14	SELECT title, revenue, budget, revenue-budget from movies;	Subtract budget from revenue
15	SELECT title, revenue, budget, revenue/budget from movies;	Get ratio of revenue to budget (revenue divided by budget)

Exercise #3

A. List the top 25 movies (title, revenue, budget) with the highest revenue to budget ratio.

B. List the top 25 movies (title, release date, runtime in hours) with the longest runtime.

Filtering part 1

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
16	SELECT * FROM movies WHERE vote_average > 7;	Get movies with vote_average greater than 7
17	SELECT * FROM movies WHERE vote_average >=7 AND vote_average <= 8;	Get movies with vote_average between 7 and 8
18	SELECT * FROM movies WHERE release_date >= '2015-01-01';	Get movies with release date 2015 onwards

Filtering part 2

MySQL Commands

Note: Make sure we run the queries one at a time only. Empty the big text box first before pasting the query.

Semicolon (**;**) is important to mark the end of your query. (don't forget to place at the end)

	MySQL Query	**Description**
19	SELECT * FROM movies WHERE title='spectre';	Get movie with title = spectre (**not case sensitive**)
20	SELECT * FROM movies WHERE title LIKE '%pirate%';	Get movies containing "pirate"
21	SELECT * FROM movies WHERE title LIKE '%pirate%' OR title LIKE '%shrek%';	Get pirate OR shrek movies
22	SELECT title, release_date FROM movies WHERE title LIKE '%pirate%' AND year(release_date) >= 2010;	Get 2010s pirate movies

Exercise #4

A. Compare the average runtime of movies of the 2010's, 2000's and the 90's

B. List down the movie title, release date, language and original title from Japanese movies

Exercise

Provide me a TOP 10 list of movies released from 2010-2015 with the highest popularity score. List their title, release date, release year, release month and runtime in **hours**

Exercise

Provide me a TOP 10 list of movies released from 2010-2015 with the highest popularity score. List their title, release date, release year, release month and runtime in **hours**

```
SELECT
        Title,
        Release_date,
        YEAR(release_date),
        MONTH(release_date),
        runtime/60
FROM
        movies
WHERE
        YEAR(release_date) >= 2010 AND YEAR(release_date)  <= 2015
ORDER BY
        popularity DESC
LIMIT 10;
```

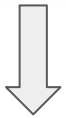
Exercise

Provide me a TOP 10 list of movies released from 2010-2015 with the highest popularity score. List their title, release date, release year, release month and runtime in **hours**

Title	Release_date	YEAR(release_date)	MONTH(release_date)	runtime/60
Minions	2015-06-17	2,015	6	1.52
Interstellar	2014-11-05	2,014	11	2.82
Guardians of the Galaxy	2014-07-30	2,014	7	2.02
Mad Max: Fury Road	2015-05-13	2,015	5	2.00
Jurassic World	2015-06-09	2,015	6	2.07
Dawn of the Planet of the Apes	2014-06-26	2,014	6	2.17
The Hunger Games: Mockingjay - Part 1	2014-11-18	2,014	11	2.05
Big Hero 6	2014-10-24	2,014	10	1.70
Terminator Genisys	2015-06-23	2,015	6	2.10
Whiplash	2014-10-10	2,014	10	1.75

Group by functions

title	year
Four Rooms	1,995
Star Wars	1,977
Finding Nemo	2,003
Forrest Gump	1,994



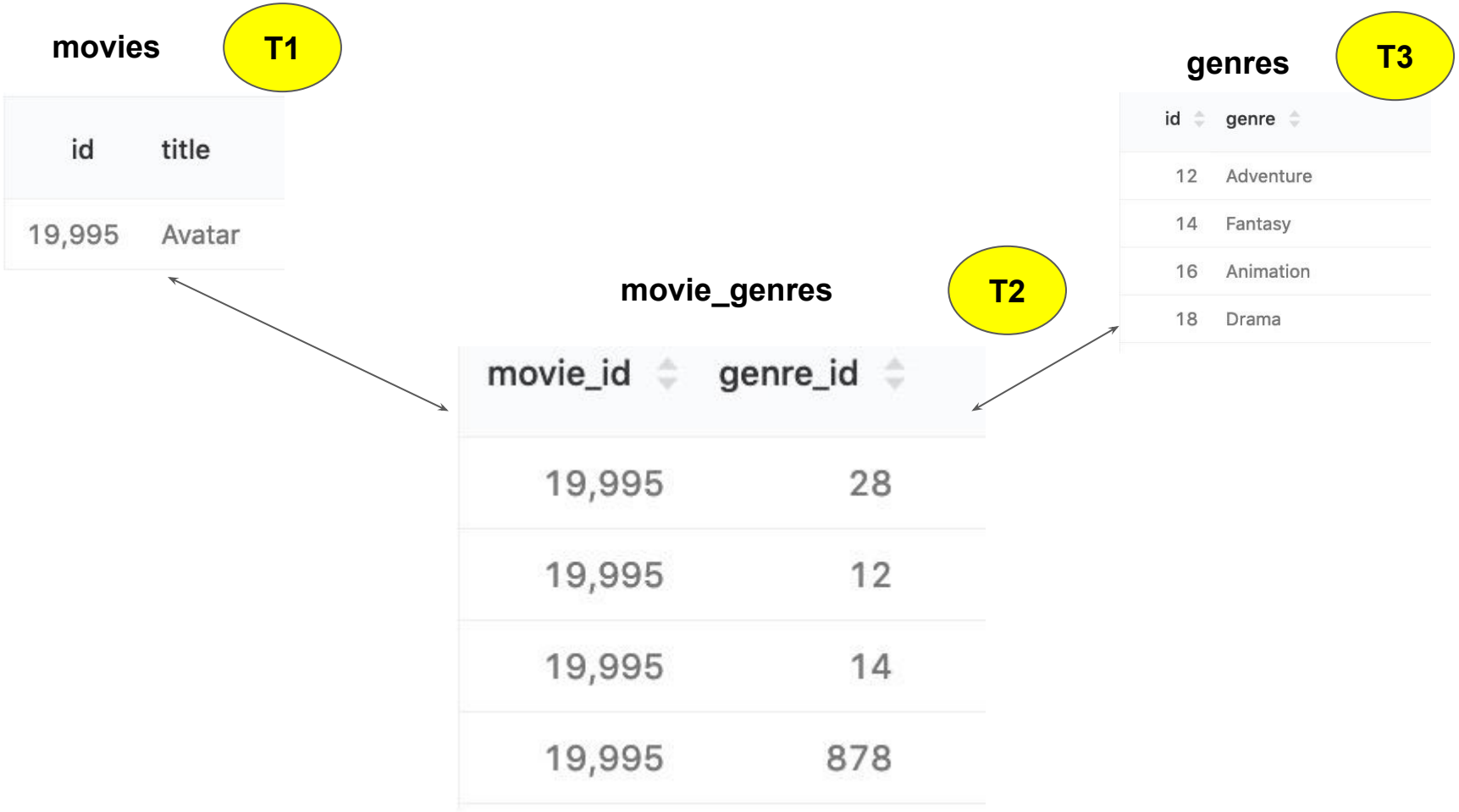
year(release_date)	year
1,995	70
1,977	16
2,003	169
1,994	55

Group by functions

23	SELECT year(release_date), count(*) as year_count FROM movies GROUP BY year(release_date) LIMIT 25;
24	SELECT year(release_date), count(*) as count, avg(budget) as avg_budget FROM movies GROUP BY year(release_date) ORDER BY count DESC LIMIT 25;

Group by functions

25	SELECT year(release_date), count(*) as count, avg(budget) as avg_budget FROM movies WHERE overview LIKE '%zombie%' GROUP BY year(release_date) ORDER BY count DESC LIMIT 25;


movies
T1
id
title
19,995
Avatar
genres
T3
id
genre
12
Adventure
14
Fantasy
16
Animation
18
Drama
movie_genres
T2
movie_id
genre_id
19,995
28
19,995
12
19,995
14
19,995
878

Join functions

SELECT id, title FROM movies;

id	title
5	Four Rooms
11	Star Wars
12	Finding Nemo
13	Forrest Gump

Join functions

```
SELECT
        movies.id,
        movies.title,
        movie_genres.genre_id
FROM movies JOIN movie_genres
ON movies.id = movie_genres.movie_id;
```

id	title	genre_id
5	Four Rooms	80
5	Four Rooms	35
11	Star Wars	12
11	Star Wars	28
11	Star Wars	878
12	Finding Nemo	16
12	Finding Nemo	10,751

Join functions

```
SELECT
    movies.id,
    movies.title,
    movie_genres.genre_id,
    genres.genre
    FROM
        movies
            JOIN
        movie_genres ON movies.id = movie_genres.movie_id
            JOIN
        genres ON movie_genres.genre_id = genres.id;
```

id	title	genre_id	genre
5	Four Rooms	80	Crime
5	Four Rooms	35	Comedy
11	Star Wars	12	Adventure
11	Star Wars	28	Action
11	Star Wars	878	Science Fiction
12	Finding Nemo	16	Animation
12	Finding Nemo	10,751	Family
13	Forrest Gump	35	Comedy

Datasets Search

https://www.kaggle.com/datasets

Datasets

Explore, analyze, and share quality data. Learn more about data types, creating, and collaborating.

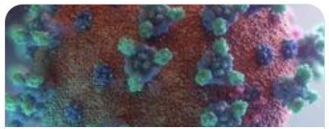
+ New Dataset | Your Work

Search datasets | Filters

All datasets | Computer Science | Education | Classification | Computer Vision | NLP | Data Visualization | Pre-Trained Model

Trending Datasets

See All



Covid-19 Cases & Deaths

Nikhil Anand · Updated a day ago
Usability **10.0** · 8 kB
1 File (CSV)

INDIA Tourism 2014-2020

RAJ KACHHADIYA · Updated 2 days ago
Usability **10.0** · 27 kB
12 Files (CSV)

BikeSharing

Shruti Pandit · Updated a day ago
Usability 8.8 · 22 kB
1 File (CSV)

Help us improve Kaggle

No Thanks | Take our Survey

Practice Datasets

Car Insurance Data *Insurance Claims over Cars*	LINK	car_insurance
Travel Insurance *Travel Insurance with the target attribute of: Claim Status (Yes or No)*	LINK	travel_insurance