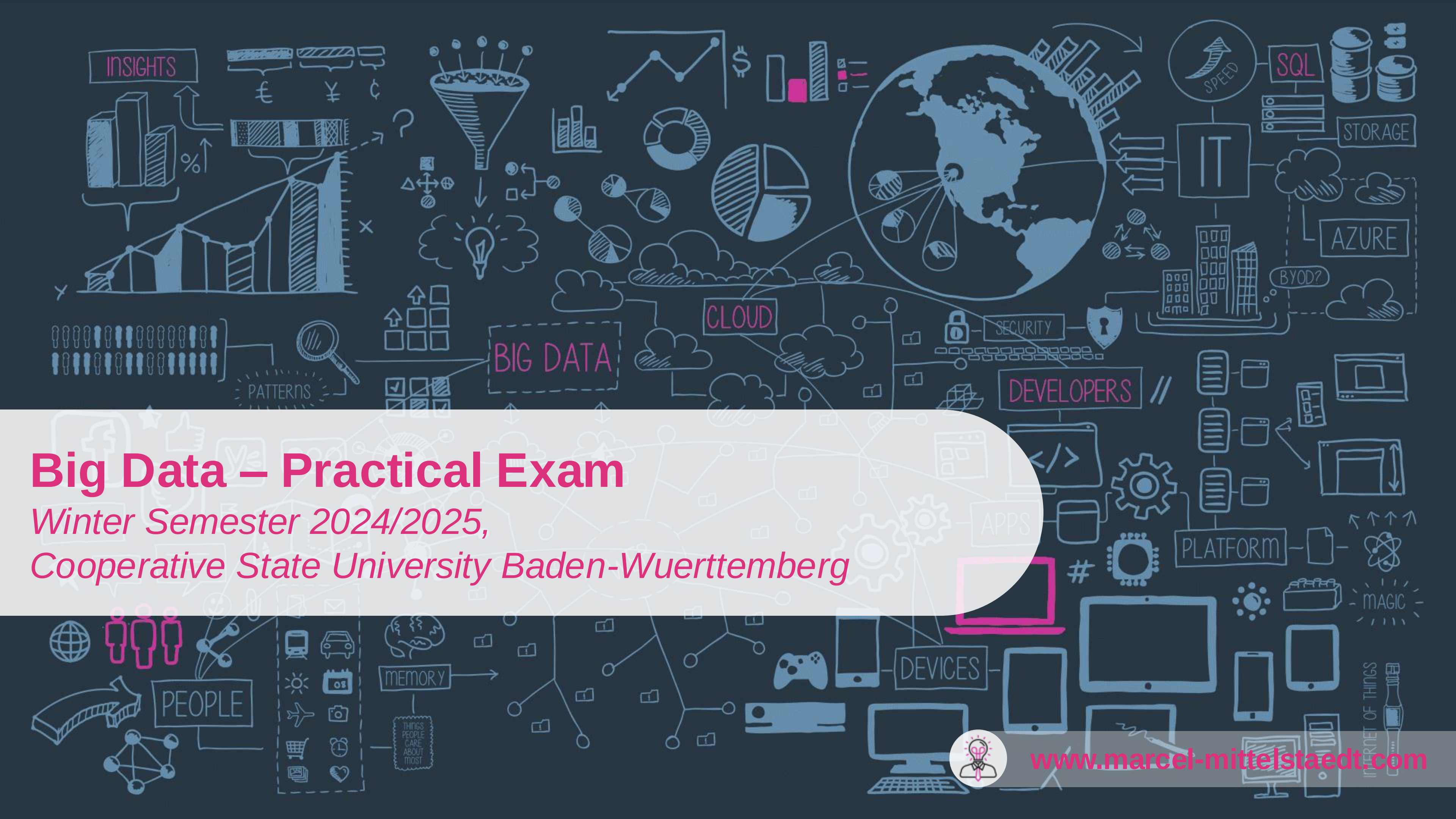
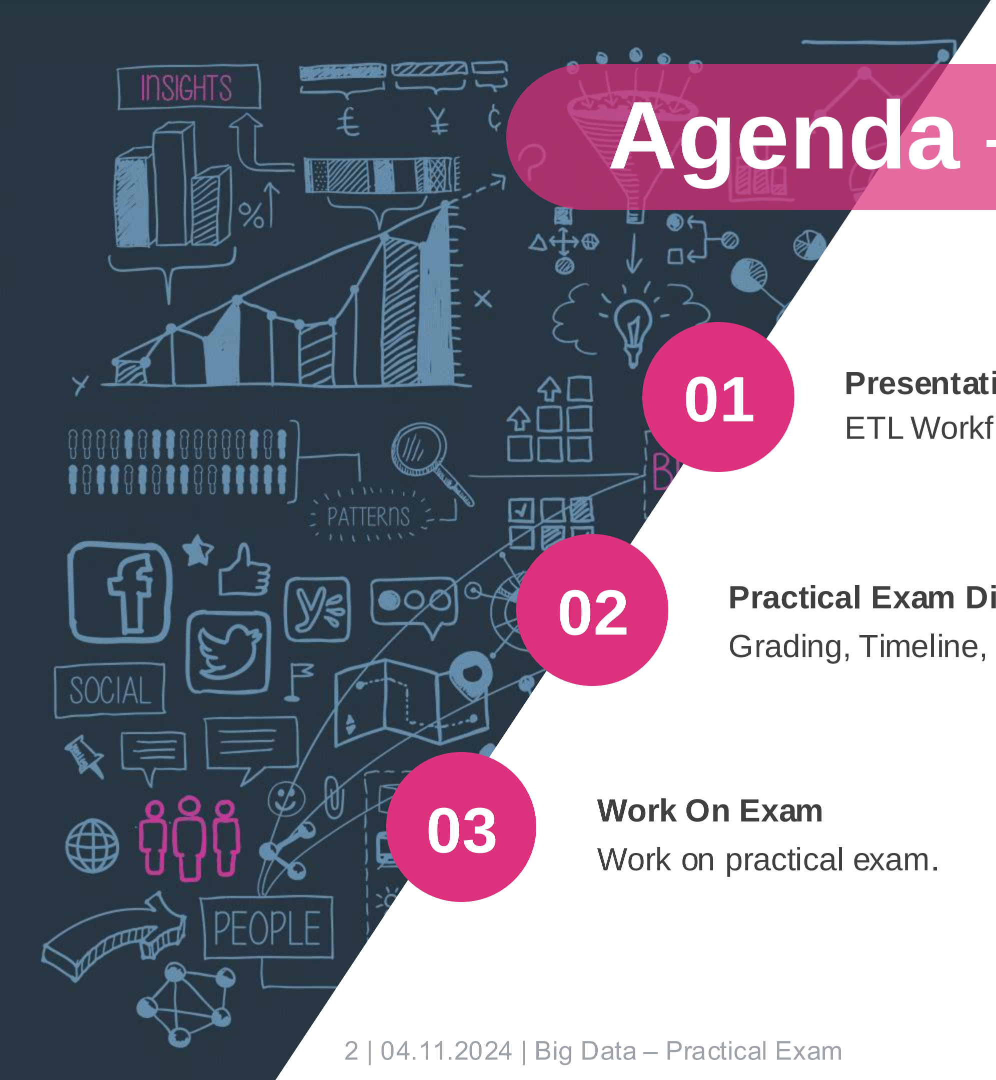


Big Data – Practical Exam

Winter Semester 2024/2025,
Cooperative State University Baden-Wuerttemberg

www.marcel-mittelstaedt.com

Agenda – 04.11.2024



01

Presentation and Discussion: Exercise Of Last Lecture

ETL Workflow: Airflow

02

Practical Exam Discussion

Grading, Timeline, Deliverables, Presentation Procedure, Exam Topics

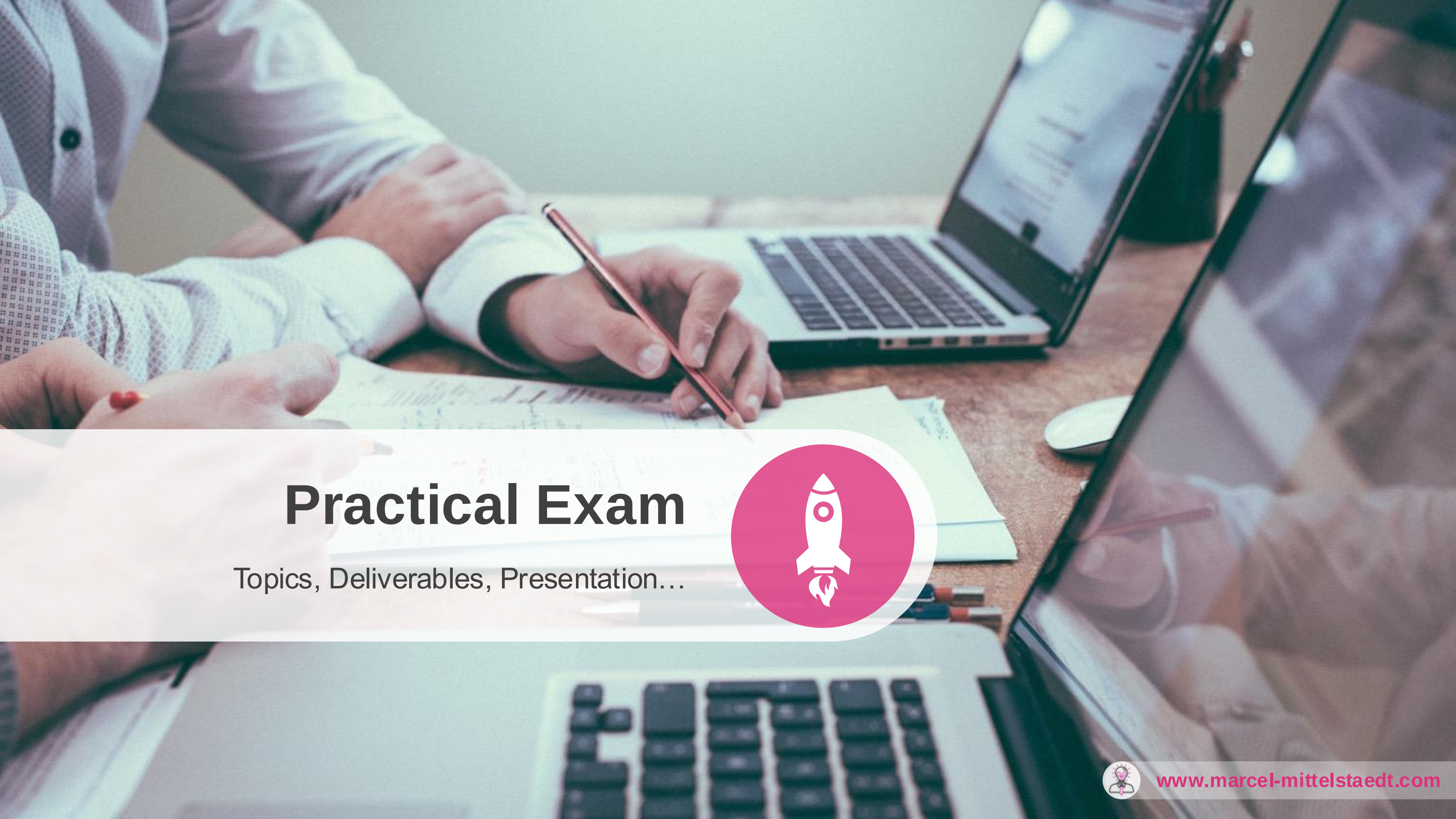
03

Work On Exam

Work on practical exam.

Schedule

			Lecture Topic	*HandsOn*
30.09.2024	13:00-15:45	Ro. N/A	About This Lecture, Introduction to Big Data	Setup Google Cloud, Create Own Hadoop Cluster and Run MapReduce
07.10.2024	13:00-15:45	Ro. N/A	(Non-)Functional Requirements Of Distributed Data-Systems, Data Models and Access	Hive and HiveQL
14.10.2024	13:00-15:45	Ro. N/A	Challenges Of Distributed Data Systems: Partitioning	HiveQL via JDBC, Data Partitioning (with HDFS and Hive)
21.10.2024	13:00-15:45	Ro. N/A	Challenges Of Distributed Data Systems: Replication	Spark, Scala, PySpark and Jupyter Notebooks
28.10.2024	13:15-15:45	Ro. N/A	ETL Workflow and Automation & Batch and Stream Processing	Airflow
04.11.2024	**13:00-15:45**	**Ro. N/A**	**Practical Exam**	Work On Practical Exam
11.11.2024	13:00-15:45	Ro. N/A	Practical Exam	Work On Practical Exam
18.11.2024	13:00-15:45	Ro. N/A	Practical Exam	Work On Practical Exam
25.11.2024	**13:00-15:45**	**Ro. N/A**	**Practical Exam Presentation**	
02.12.2024	**13:00-15:45**	**Ro. N/A**	**Practical Exam Presentation**	



Practical Exam

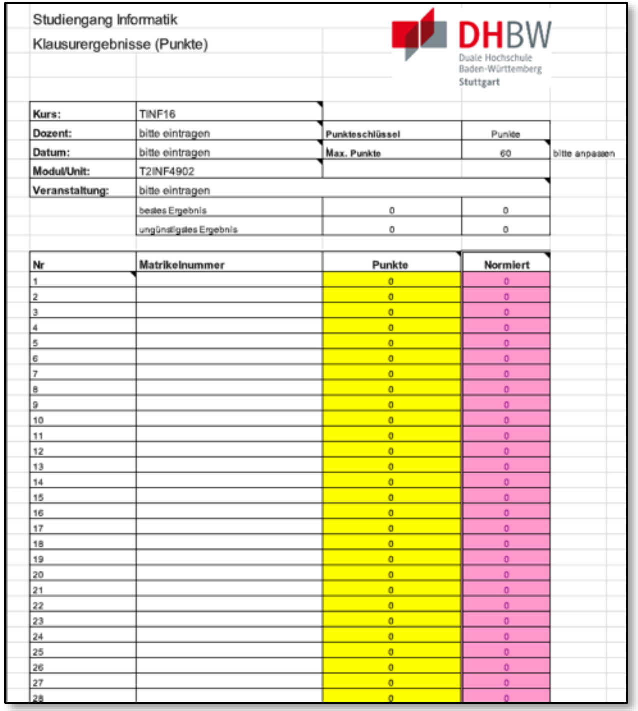
Topics, Deliverables, Presentation…

Practical Exam - **Grading**

Grade/Percentage:

- grade will be mixed with grade of another lecture
- therefore, the rating won‘t be a grade (1-6) but a percentage:

Percentage	Grade
100%	1.0
…	…
50%	4.0
…	…



Studiengang Informatik

Klausurergebnisse (Punkte)

Kurs:	TINF16			
Dozent:	bitte eintragen	**Punkteschlüssel**	Punkte	
Datum:	bitte eintragen	**Max. Punkte**	60	bitte anpassen
Modul/Unit:	T2INF4902			
Veranstaltung:	bitte eintragen			
	bestes Ergebnis	0	0	
	ungünstigstes Ergebnis	0	0	

Nr	Matrikelnummer	Punkte	Normiert
1		0	0
2		0	0
3		0	0
4		0	0
5		0	0
6		0	0
7		0	0
8		0	0
9		0	0
10		0	0
11		0	0
12		0	0
13		0	0
14		0	0
15		0	0
16		0	0
17		0	0
18		0	0
19		0	0
20		0	0
21		0	0
22		0	0
23		0	0
24		0	0
25		0	0
26		0	0
27		0	0
28		0	0

Timeline:

22.11.2024 23:59 All deliverables (next slide) will be delivered to following email address (DropBox, Google-Drive…):

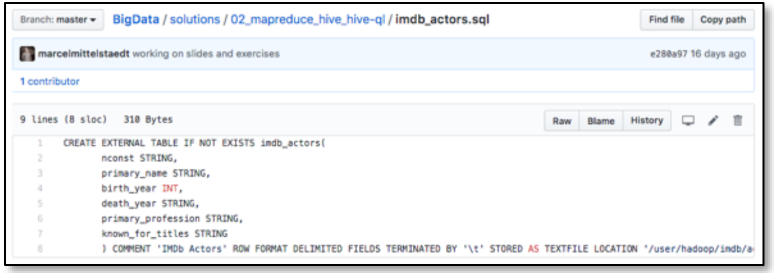
contact@marcel-mittelstaedt.com

25.11./02.12.2023 13:15-15:45 Presentation of Practical Exam

Practical Exam - **Deliverables**

Deliverables:

- A simple Documentation:
    - Explanation of whole ETL Workflow
    - List of Jobs/Transformations in Case of PDI or DAGs and Steps in Case of Airflow
    - Short description of the purpose of each job/transformation or task and applied business rules
    - all PDI Jobs, Transformations and related files (ktr, kjb, kettle.properties, shared.xml… files)
    - All Airflow DAGs and tasks
- All Scripts (e.g. Download) or other external applications called within PDI
- All Airflow DAGs, Python Files etc.
- All DDLs (`CREATE Table…`):
    - One file for each table
    - Table name = File Name, e.g.:


Branch: master ▾
BigData / solutions / 02_mapreduce_hive_hive-ql / imdb_actors.sql
Find file
Copy path
marcelmittelstaedt working on slides and exercises
e280a97 16 days ago
1 contributor
9 lines (8 sloc)   310 Bytes
Raw
Blame
History

```
CREATE EXTERNAL TABLE IF NOT EXISTS imdb_actors(
        nconst STRING,
        primary_name STRING,
        birth_year INT,
        death_year STRING,
        primary_profession STRING,
        known_for_titles STRING
        ) COMMENT 'IMDb Actors' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/a
```

- Depending on Exam Type:
    - Code of Frontend Application and related Database (DDLs) or
    - Calculated KPIs

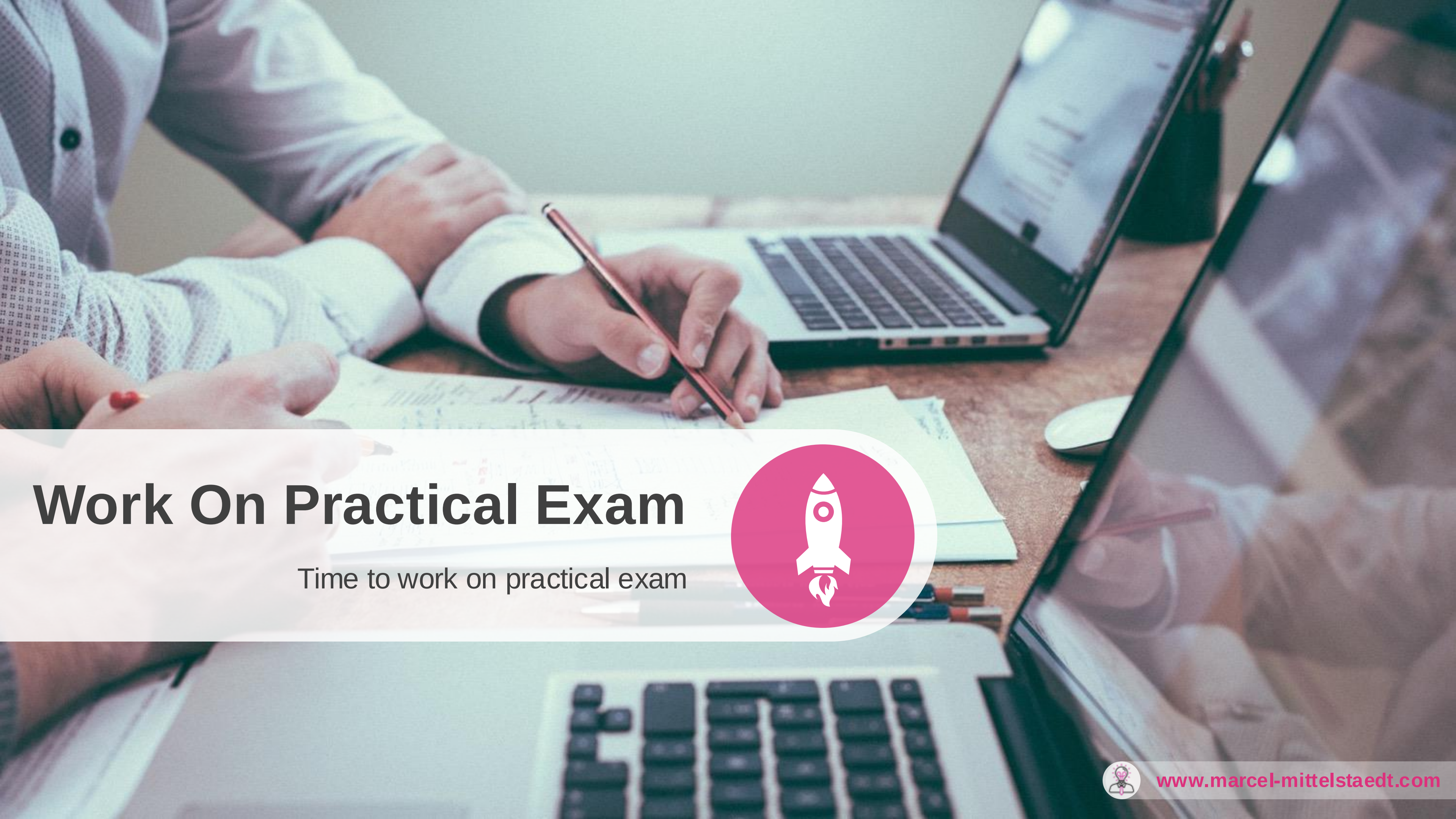
Practical Exam - **Presentation**

Procedure:

1. Start ETL Workflow

2. During execution:
   - Quickly explain data source
     - API
     - Data Structure
     - Approach for gathering data
   - Quickly Explain whole ETL Workflow
     - Explain Idea and purpose of each Job/Transformation
     - External ressources/scripts (e.g. download)
     - Explain Data Model (Raw Layer, Final Layer, simple Frontend)

3. After execution:
   - Depending on Exam:
     - Demo of simple Frontend application or
     - Explanation of calculated KPIs



Work On Practical Exam

Time to work on practical exam

Use Spotify Audio Features To Categorize Music

Practical Exam

Goal

Spotify provides an API for basic track information:

- **ID**, e.g. *2lEcSduKEXEK5KJ9hJzlCz*
- **name**, e.g. *Gloana Bauer (Teenage Dirtbag)*
- **artist**, e.g. *D‘ Hundskrippln*
- …

as well as related audio features:

- **energy**, e.g. *0.454* — *Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.* Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.
- **speechiness**, e.g. *0.0388* — *Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.*
- **loudness**, e.g. *-8.758* — *The overall loudness of a track in decibels (dB). Loudness values are averaged.*
- **tempo**, e.g. *97.532* — *The overall estimated tempo of a track in beats per minute (BPM)*
- **acousticness**, *e.g. 0.268* — *A confidence measure from 0.0 to 1.0 of whether the track is acoustic.*
- …

https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

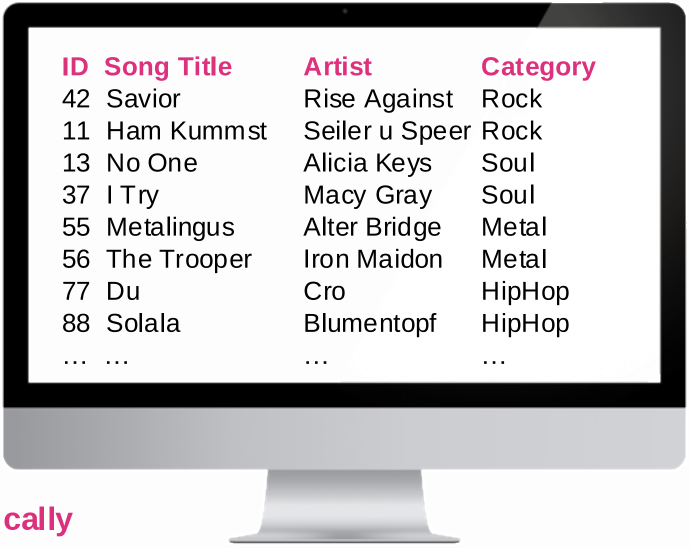
Goal

We want to make use of those audio features to automatically assign each track to a certain category:

Metal
Electro
Classic
Podcast
Rock
Soul
Vocal
HipHop

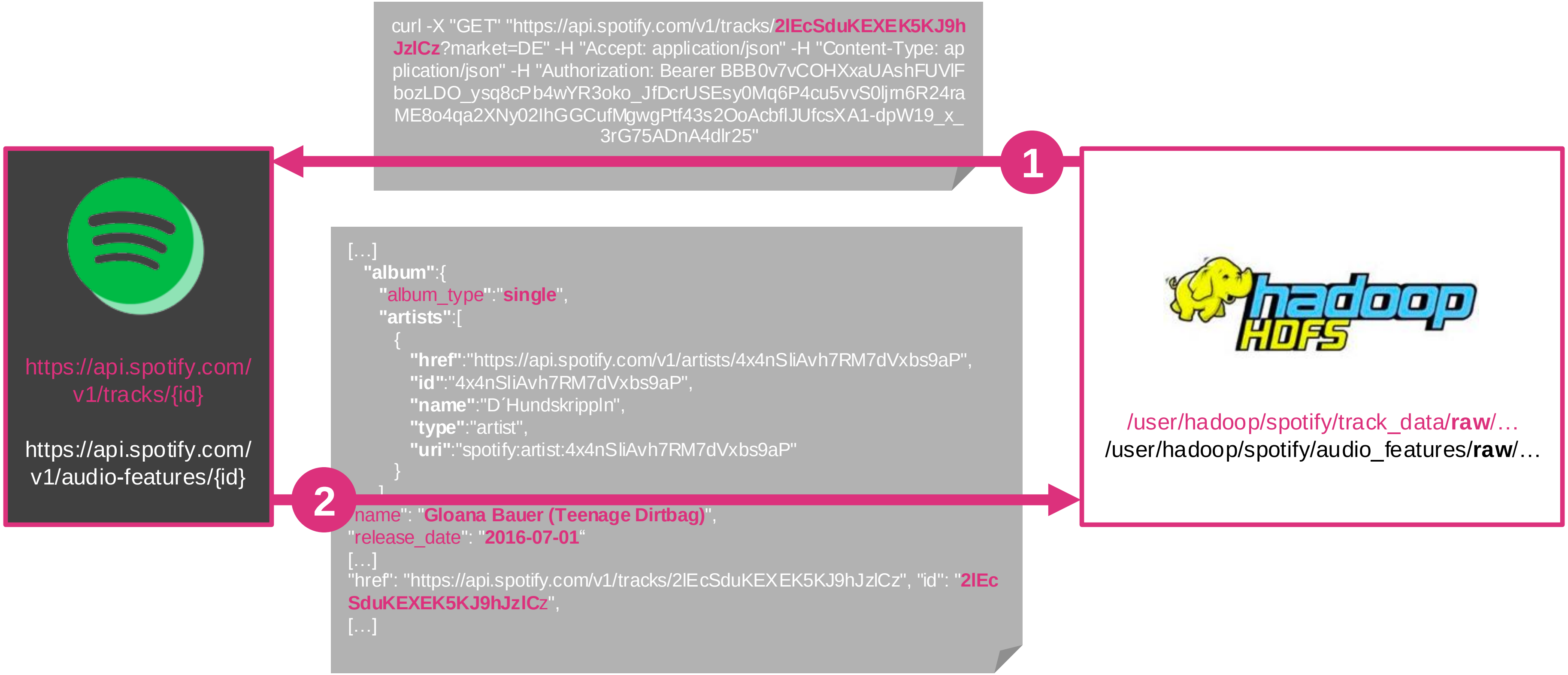
Workflow:

- **Query data** from Spotify API
- **Save raw data** (*JSON* files) to HDFS
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Calculate categories** (*Metal, Classic, Rock, …*)
- **Join track information** and **audio festures** and **save** everything **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple HTML Frontend which reads from end-user database and displays result
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



ID	Song Title	Artist	Category
42	Savior	Rise Against	Rock
11	Ham Kummst	Seiler u Speer	Rock
13	No One	Alicia Keys	Soul
37	I Try	Macy Gray	Soul
55	Metalingus	Alter Bridge	Metal
56	The Trooper	Iron Maidon	Metal
77	Du	Cro	HipHop
88	Solala	Blumentopf	HipHop
…	…	…	…

Dataflow: **1. Get Track Information**



```
curl -X "GET" "https://api.spotify.com/v1/tracks/2lEcSduKEXEK5KJ9hJzlCz?market=DE" -H "Accept: application/json" -H "Content-Type: application/json" -H "Authorization: Bearer BBB0v7vCOHXxaUAshFUVlFbozLDO_ysq8cPb4wYR3oko_JfDcrUSEsy0Mq6P4cu5vvS0ljrn6R24raME8o4qa2XNy02IhGGCufMgwgPtf43s2OoAcbflJUfcsXA1-dpW19_x_3rG75ADnA4dlr25"
```

```
[…]
  "album":{
    "album_type":"single",
    "artists":[
      {
        "href":"https://api.spotify.com/v1/artists/4x4nSliAvh7RM7dVxbs9aP",
        "id":"4x4nSliAvh7RM7dVxbs9aP",
        "name":"D´Hundskrippln",
        "type":"artist",
        "uri":"spotify:artist:4x4nSliAvh7RM7dVxbs9aP"
      }
    ]
"name": "Gloana Bauer (Teenage Dirtbag)",
"release_date": "2016-07-01“
[…]
"href": "https://api.spotify.com/v1/tracks/2lEcSduKEXEK5KJ9hJzlCz", "id": "2lEcSduKEXEK5KJ9hJzlCz",
[…]
```

https://api.spotify.com/v1/tracks/{id}

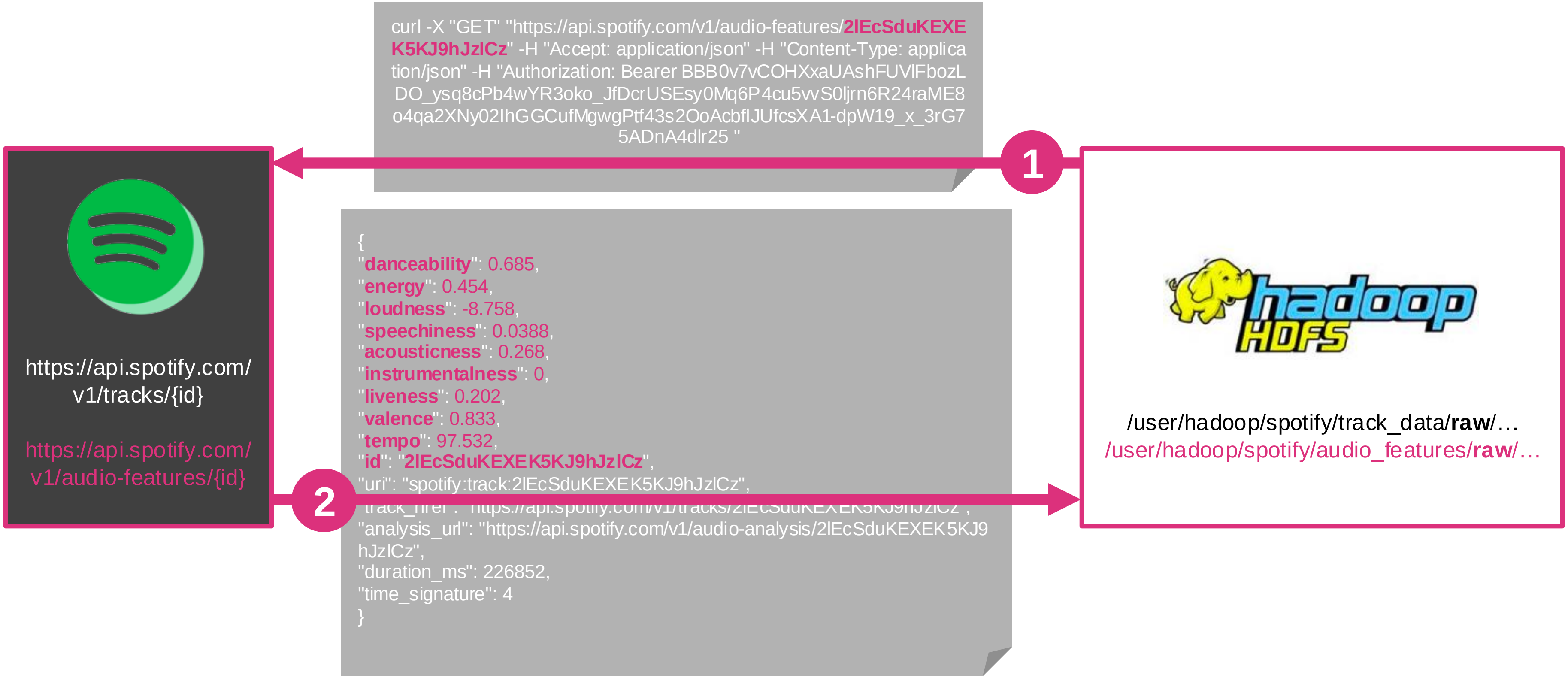
https://api.spotify.com/v1/audio-features/{id}

/user/hadoop/spotify/track_data/**raw**/…

/user/hadoop/spotify/audio_features/**raw**/…

https://developer.spotify.com/documentation/web-api/reference/#/operations/get-track

Dataflow: **2. Get Track Audio Features**



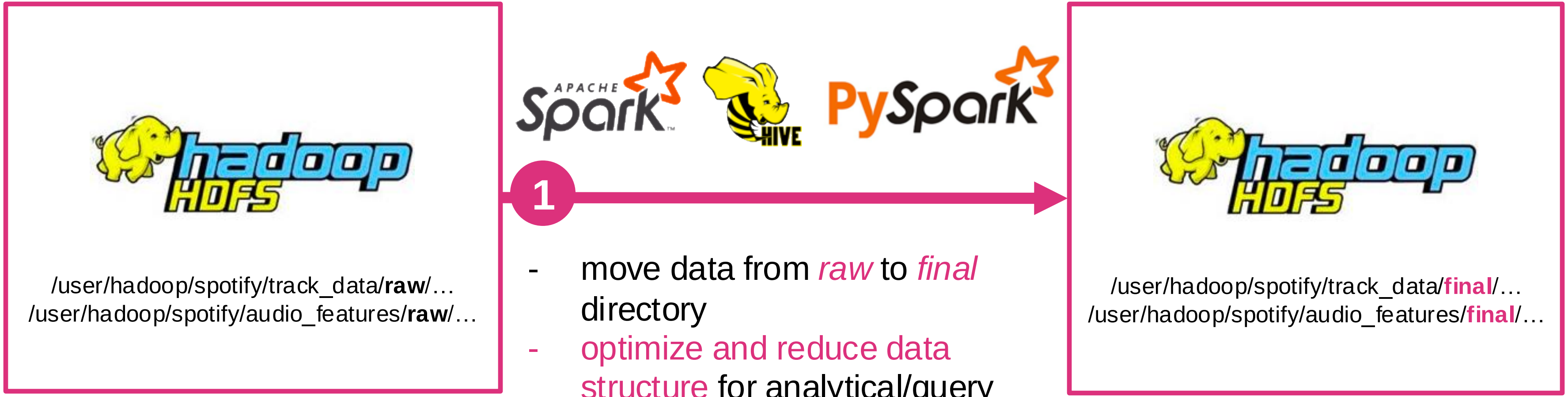
1
2
https://api.spotify.com/v1/tracks/{id}
https://api.spotify.com/v1/audio-features/{id}
/user/hadoop/spotify/track_data/**raw**/…
/user/hadoop/spotify/audio_features/**raw**/…

```
curl -X "GET" "https://api.spotify.com/v1/audio-features/2lEcSduKEXEK5KJ9hJzlCz" -H "Accept: application/json" -H "Content-Type: application/json" -H "Authorization: Bearer BBB0v7vCOHXxaUAshFUVlFbozLDO_ysq8cPb4wYR3oko_JfDcrUSEsy0Mq6P4cu5vvS0ljrn6R24raME8o4qa2XNy02IhGGCufMgwgPtf43s2OoAcbflJUfcsXA1-dpW19_x_3rG75ADnA4dlr25 "
```

```
{
"danceability": 0.685,
"energy": 0.454,
"loudness": -8.758,
"speechiness": 0.0388,
"acousticness": 0.268,
"instrumentalness": 0,
"liveness": 0.202,
"valence": 0.833,
"tempo": 97.532,
"id": "2lEcSduKEXEK5KJ9hJzlCz",
"uri": "spotify:track:2lEcSduKEXEK5KJ9hJzlCz",
"track_href": "https://api.spotify.com/v1/tracks/2lEcSduKEXEK5KJ9hJzlCz",
"analysis_url": "https://api.spotify.com/v1/audio-analysis/2lEcSduKEXEK5KJ9hJzlCz",
"duration_ms": 226852,
"time_signature": 4
}
```

https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

Dataflow: **3. Raw To Final Transfer**

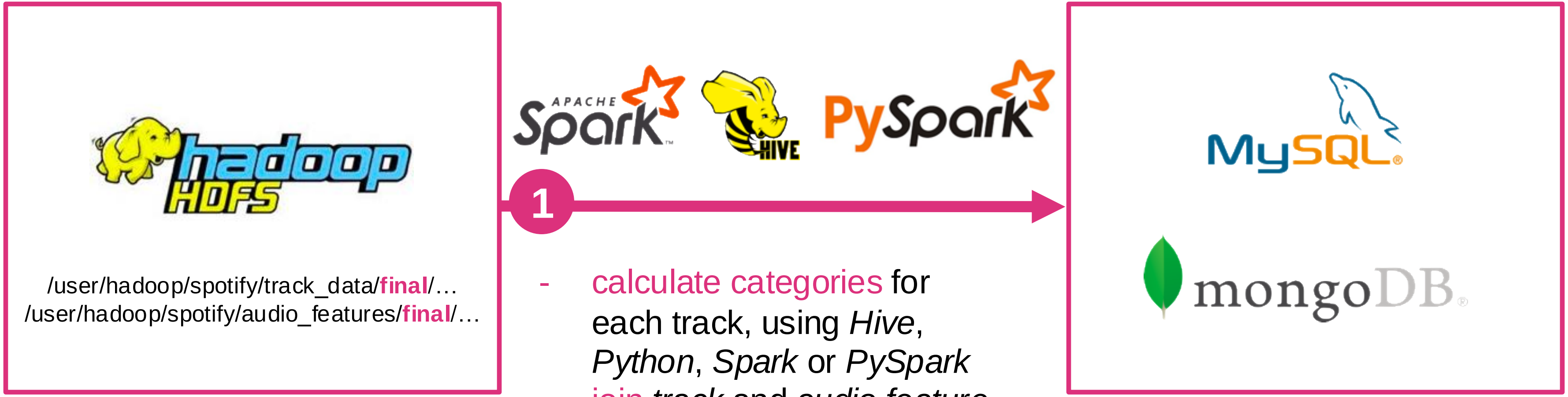


/user/hadoop/spotify/track_data/**raw**/…
/user/hadoop/spotify/audio_features/**raw**/…

- move data from *raw* to *final* directory
- optimize and reduce data structure for analytical/query purposes (JSON to tabular, only needed attributes etc.)
- remove duplicates if necessary

/user/hadoop/spotify/track_data/**final**/…
/user/hadoop/spotify/audio_features/**final**/…

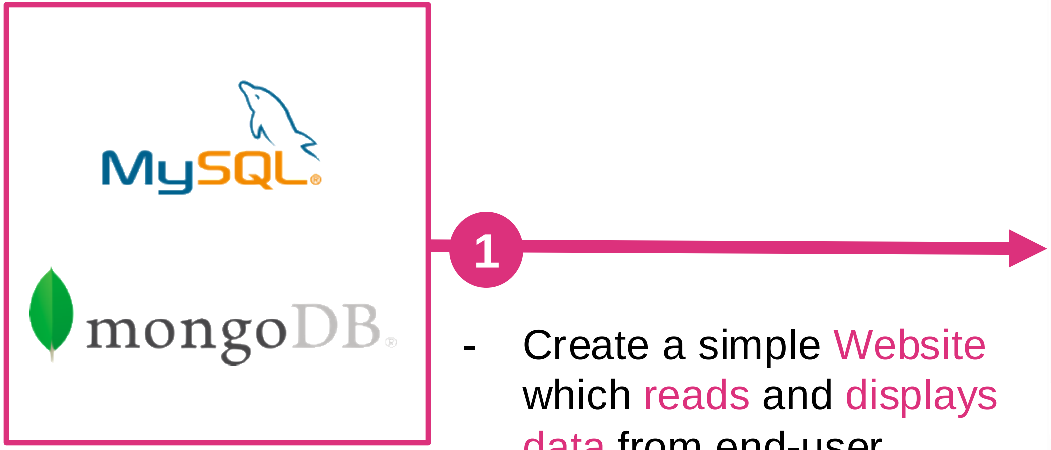
Dataflow: **4. Run Analysis and Save Results**



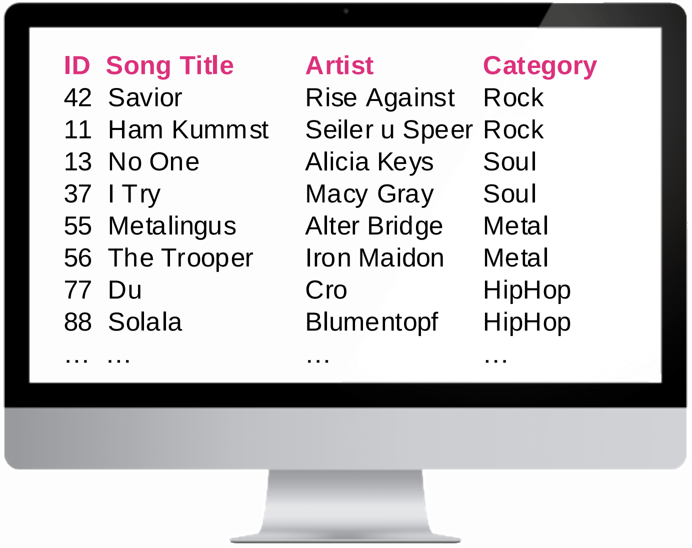
/user/hadoop/spotify/track_data/**final**/…
/user/hadoop/spotify/audio_features/**final**/…

- calculate categories for each track, using *Hive*, *Python*, *Spark* or *PySpark*
- join *track* and *audio feature* data
- save everything to a end-user database (e.g. *MySQL*, *M ongoDB*)

Dataflow: **5. Provide Simple Web Interface**



- Create a simple Website which reads and displays data from end-user database



ID	Song Title	Artist	Category
42	Savior	Rise Against	Rock
11	Ham Kummst	Seiler u Speer	Rock
13	No One	Alicia Keys	Soul
37	I Try	Macy Gray	Soul
55	Metalingus	Alter Bridge	Metal
56	The Trooper	Iron Maidon	Metal
77	Du	Cro	HipHop
88	Solala	Blumentopf	HipHop
…	…	…	…

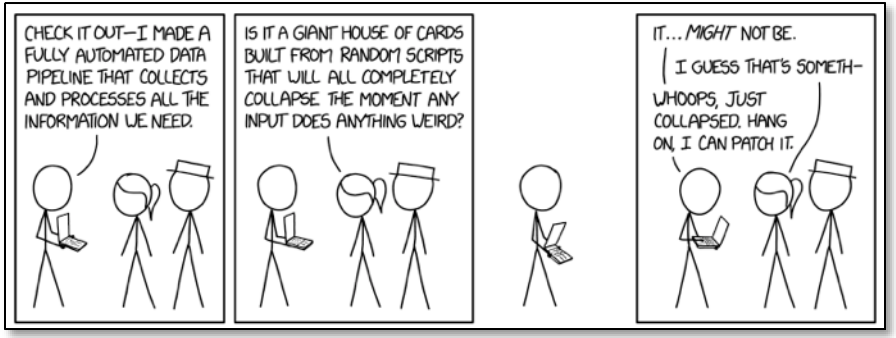
Use XKCD API To Build A Searchable Database of XKCD Comics

Practical Exam

Goal

XKCD provides regularly comics:

- https://xkcd.com/
- JSON API: https://xkcd.com/2054/info.0.json


CHECK IT OUT—I MADE A FULLY AUTOMATED DATA PIPELINE THAT COLLECTS AND PROCESSES ALL THE INFORMATION WE NEED.
IS IT A GIANT HOUSE OF CARDS BUILT FROM RANDOM SCRIPTS THAT WILL ALL COMPLETELY COLLAPSE THE MOMENT ANY INPUT DOES ANYTHING WEIRD?
IT... MIGHT NOT BE.
I GUESS THAT'S SOMETH-
WHOOPS, JUST COLLAPSED. HANG ON, I CAN PATCH IT.

https://xkcd.com/2054/

```
{
        "month": "10",
        "num": 2054,
        "link": "",
        "year": "2018",
        "news": "",
        "safe_title": "Data Pipeline",
        "transcript": "",
        "alt": "\"Is the pipeline literally running from your laptop?\" \"Don't be silly, my laptop disconnects far too often to host a service we rely on. It's running on my phone.\"", "img": "https://imgs.xkcd.com/comics/data_pipeline.png",
        "title": "Data Pipeline",
        "day": "3"
}
```

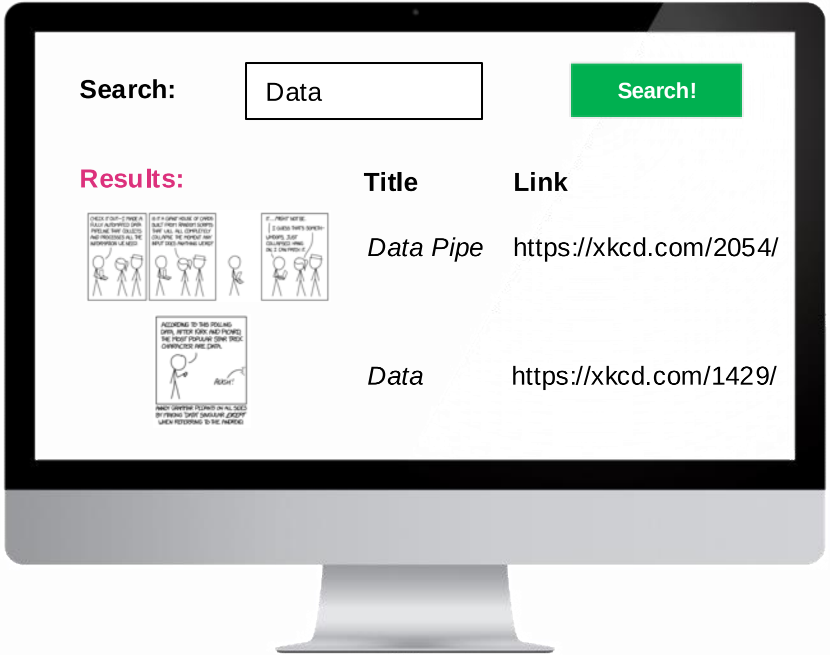
2054.json

Goal

We want to make use of this data to build a searchable database for XKCD comics.

Workflow:

- **Gather data** from xkcd.com
- **Save raw data** (*JSON files*) to HDFS (partitioned by year, e.g. *2018*, *2017*, *2016*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** xkcd data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (search phrase…)
  - checks against xkcd data in end-user database
  - Display result (comics containing search phrase)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

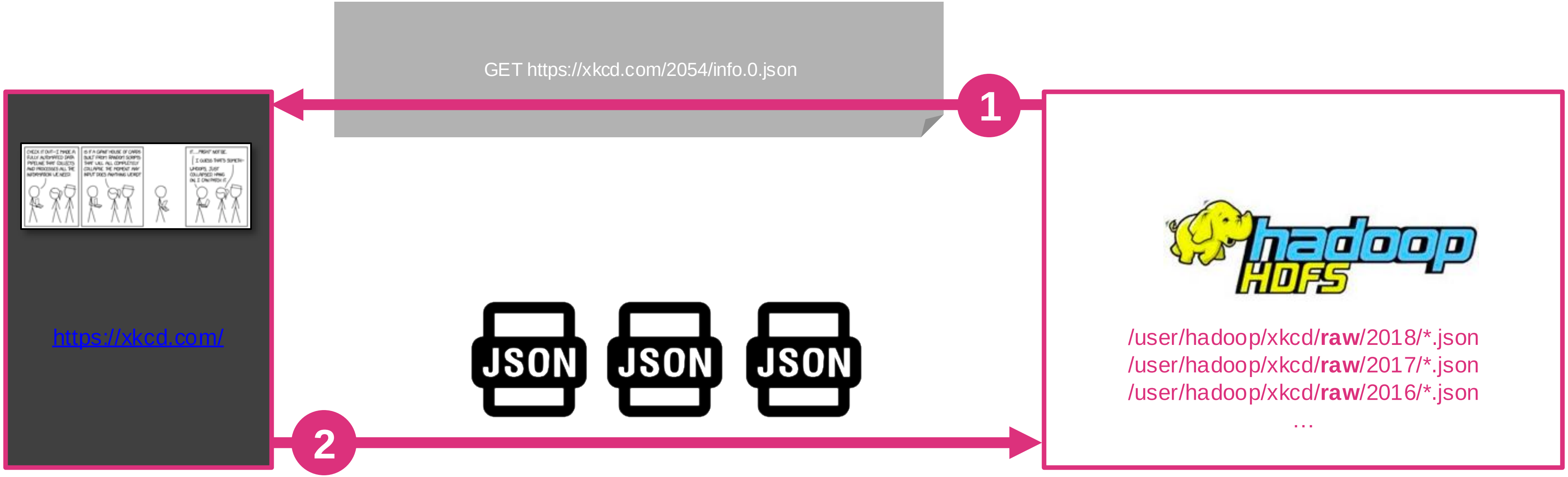


Search: Data Search!

Results:

	Title	Link
	Data Pipe	https://xkcd.com/2054/
	Data	https://xkcd.com/1429/

Dataflow: **1. Get XKCD Data**

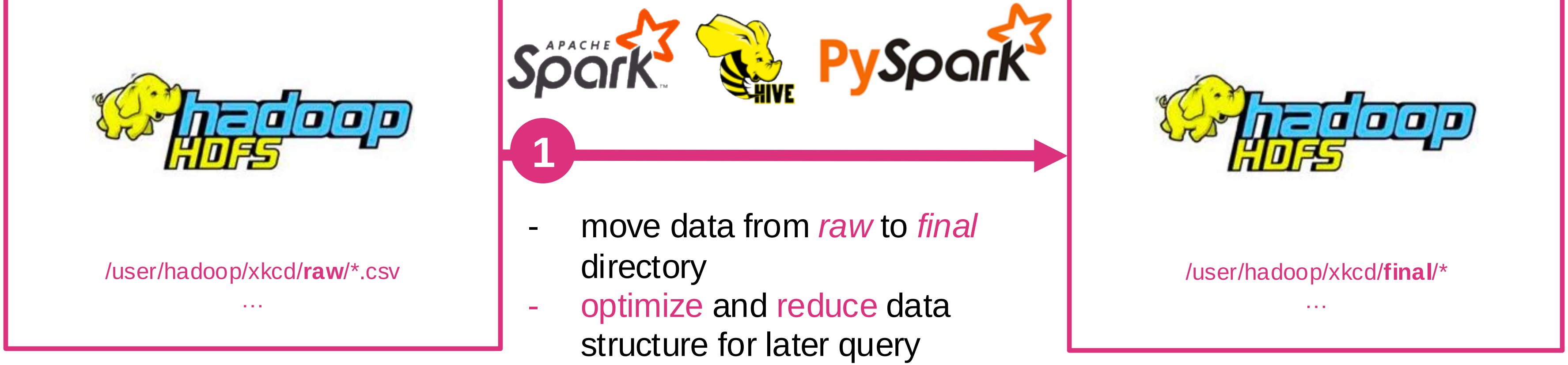


GET https://xkcd.com/2054/info.0.json

https://xkcd.com/

/user/hadoop/xkcd/**raw**/2018/*.json
/user/hadoop/xkcd/**raw**/2017/*.json
/user/hadoop/xkcd/**raw**/2016/*.json
…

Dataflow: **2. Raw To Final Transfer**

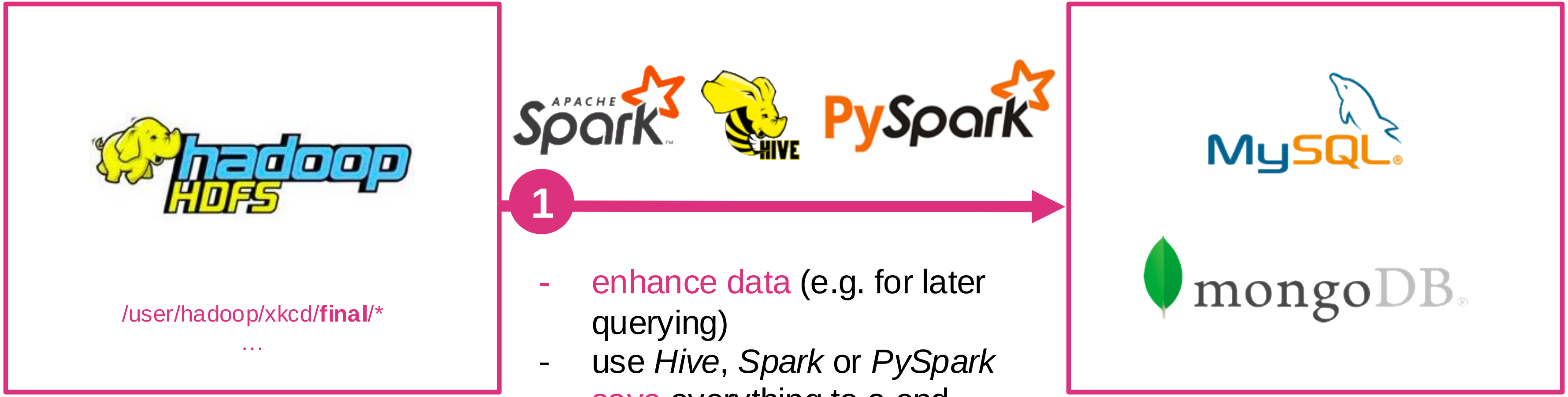


/user/hadoop/xkcd/**raw**/*.csv
…

/user/hadoop/xkcd/**final**/*
…

- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

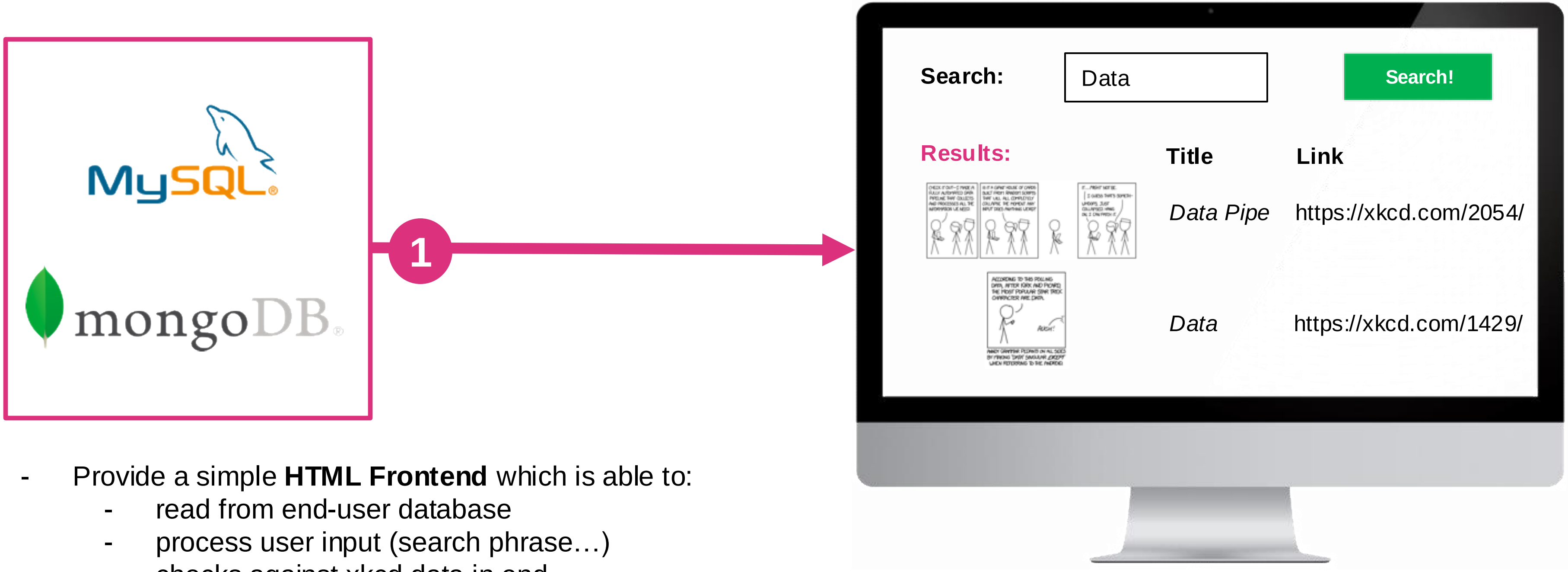
Dataflow: **3. Enhance Data And Save Results**



/user/hadoop/xkcd/**final**/*
…

- enhance data (e.g. for later querying)
- use *Hive*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



Search: Data Search!

Results:

	Title	Link
	Data Pipe	https://xkcd.com/2054/
	Data	https://xkcd.com/1429/

- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (search phrase…)
  - checks against xkcd data in end-user database
  - Display result (comics containing search phrase)

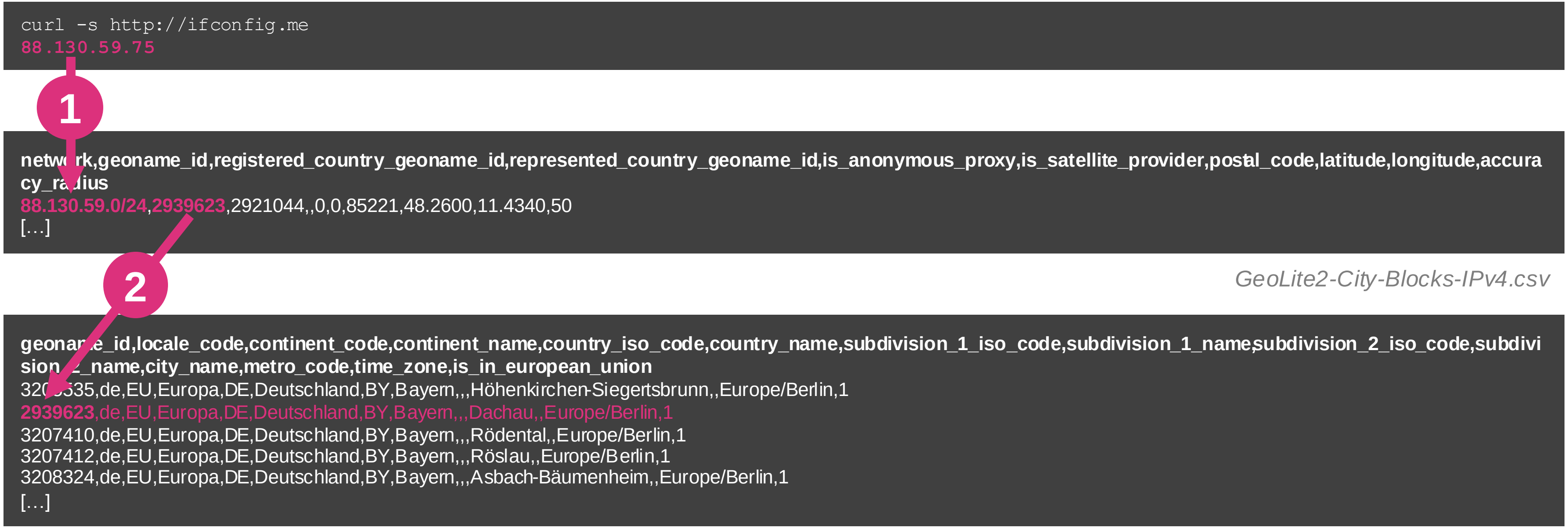
Use GeoLite2 To Create A Searchable IP and GeoLocation Database

Practical Exam

Goal

Maxmind.com provides regulary exports of worldwide IP and Geolocation data:

- https://dev.maxmind.com/geoip/geolite2-free-geolocation-data



```
curl -s http://ifconfig.me
88.130.59.75
```

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname_id,is_anonymous_proxy,is_satellite_provider,postal_code,latitude,longitude,accuracy_radius
88.130.59.0/24,2939623,2921044,,0,0,85221,48.2600,11.4340,50
[…]
```

GeoLite2-City-Blocks-IPv4.csv

```
geoname_id,locale_code,continent_code,continent_name,country_iso_code,country_name,subdivision_1_iso_code,subdivision_1_name,subdivision_2_iso_code,subdivision_2_name,city_name,metro_code,time_zone,is_in_european_union
3205535,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Höhenkirchen-Siegertsbrunn,,Europe/Berlin,1
2939623,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Dachau,,Europe/Berlin,1
3207410,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Rödental,,Europe/Berlin,1
3207412,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Röslau,,Europe/Berlin,1
3208324,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Asbach-Bäumenheim,,Europe/Berlin,1
[…]
```

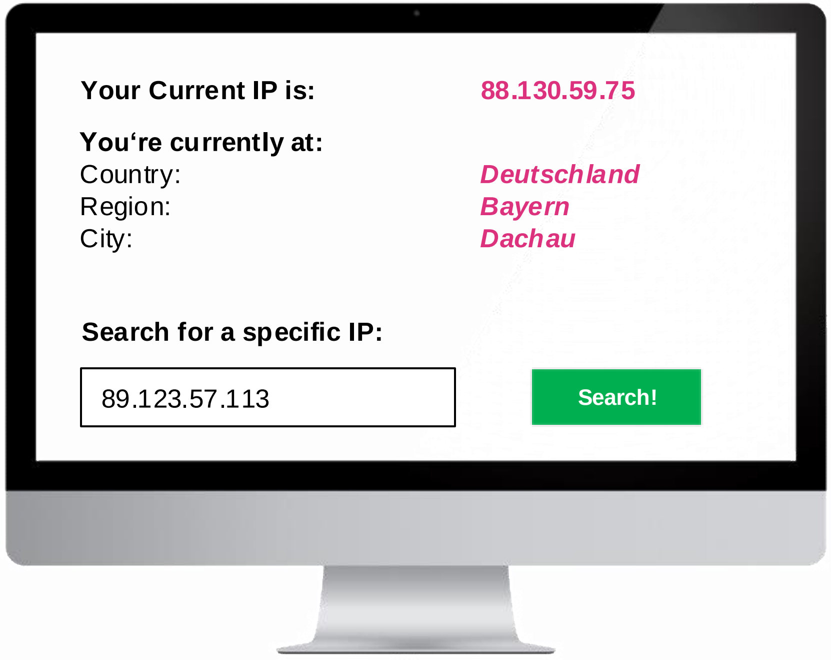
GeoLite2-City-Locations-[XX].csv

Goal

We want to make use of this data to build a real time IP-Geolocation resolution as well as a searchable database for Ips and related Geolocations.

Workflow:

- **Gather data** from maxmind.com
- **Save raw data** (*CSV files*) to HDFS (partitioned by country code, e.g. *de*, *es*, *en*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** Geolite2 data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user's IP address, **lookup** and **show Geolocation**
  - process user input (IP…) and check against end-user database
  - Display result Geolocation
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



Your Current IP is: 88.130.59.75

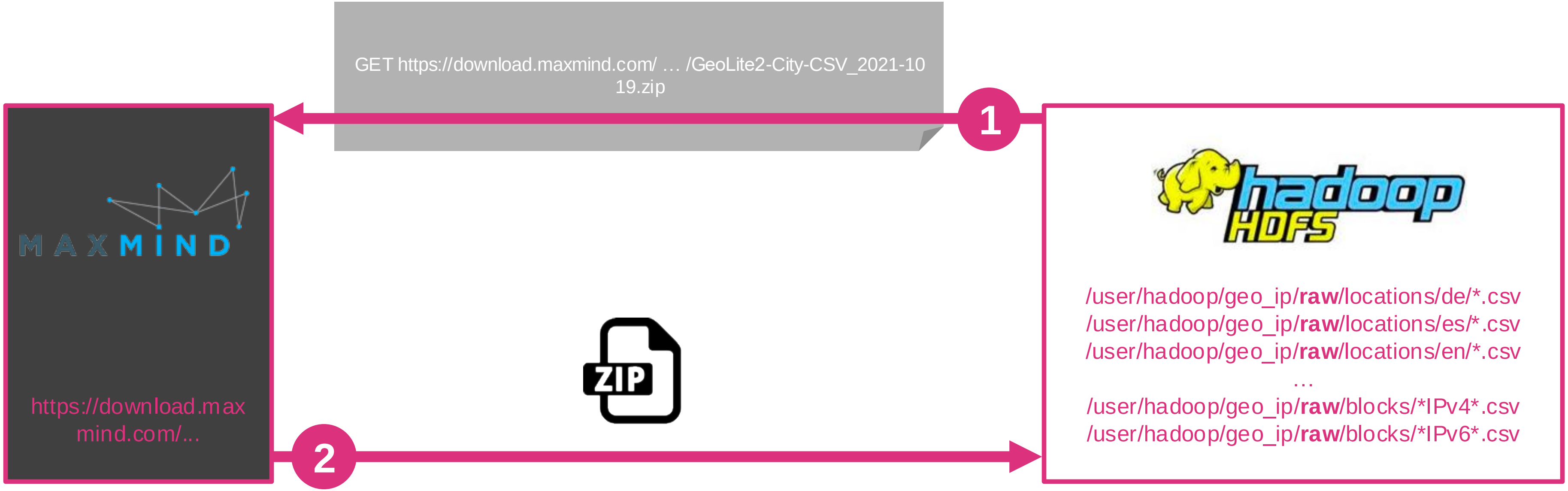
You're currently at:

Country: *Deutschland*
Region: *Bayern*
City: *Dachau*

Search for a specific IP:

89.123.57.113

Dataflow: **1. Get Geolite2 Data**

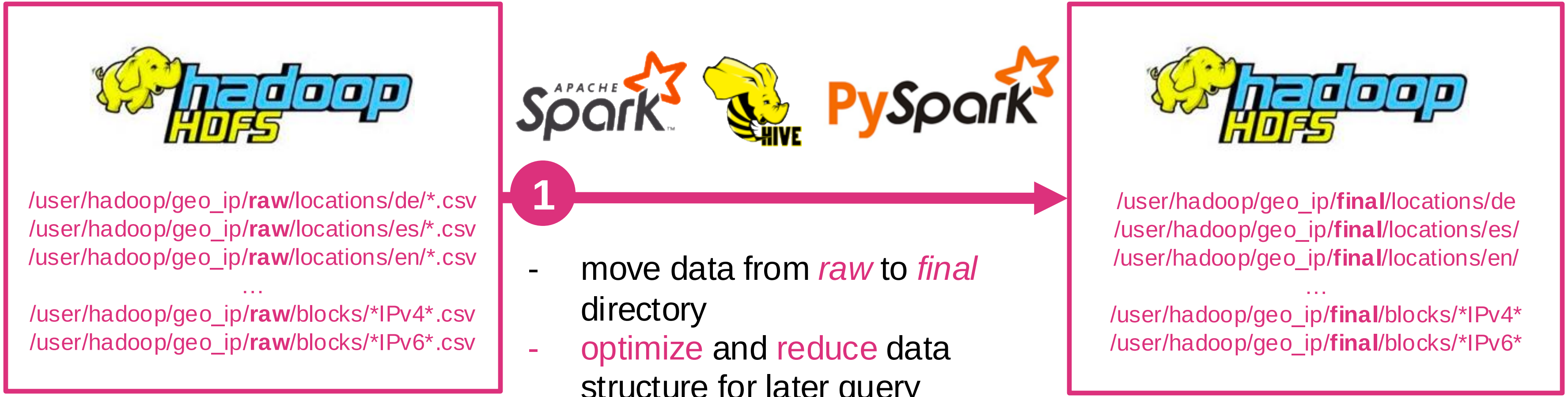


GET https://download.maxmind.com/ … /GeoLite2-City-CSV_2021-1019.zip

https://download.maxmind.com/...

/user/hadoop/geo_ip/**raw**/locations/de/*.csv
/user/hadoop/geo_ip/**raw**/locations/es/*.csv
/user/hadoop/geo_ip/**raw**/locations/en/*.csv
…
/user/hadoop/geo_ip/**raw**/blocks/*IPv4*.csv
/user/hadoop/geo_ip/**raw**/blocks/*IPv6*.csv

Dataflow: **2. Raw To Final Transfer**



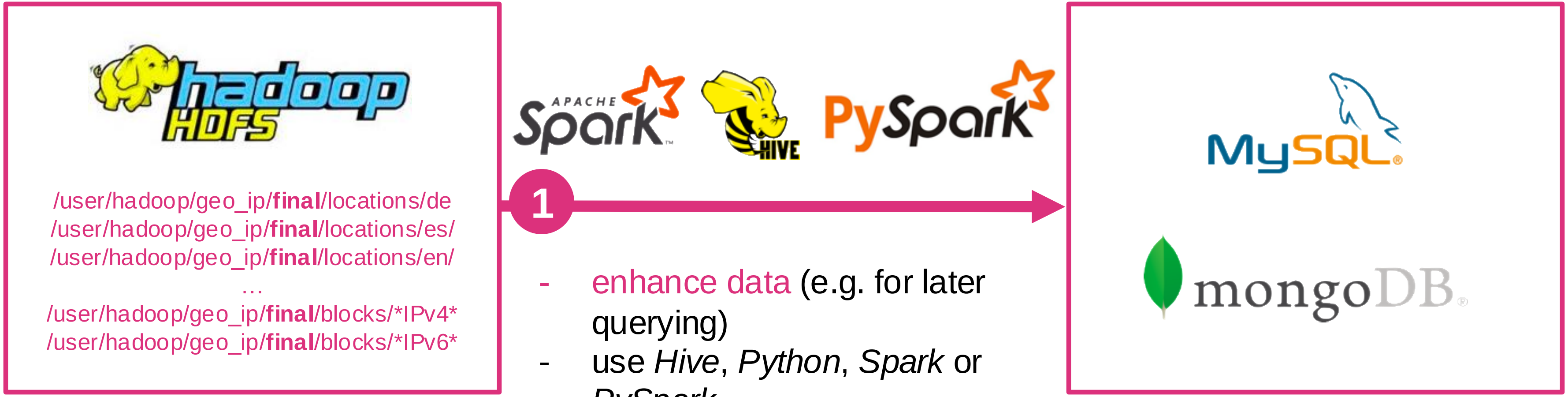
/user/hadoop/geo_ip/**raw**/locations/de/*.csv
/user/hadoop/geo_ip/**raw**/locations/es/*.csv
/user/hadoop/geo_ip/**raw**/locations/en/*.csv
…
/user/hadoop/geo_ip/**raw**/blocks/*IPv4*.csv
/user/hadoop/geo_ip/**raw**/blocks/*IPv6*.csv

1

/user/hadoop/geo_ip/**final**/locations/de
/user/hadoop/geo_ip/**final**/locations/es/
/user/hadoop/geo_ip/**final**/locations/en/
…
/user/hadoop/geo_ip/**final**/blocks/*IPv4*
/user/hadoop/geo_ip/**final**/blocks/*IPv6*

- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

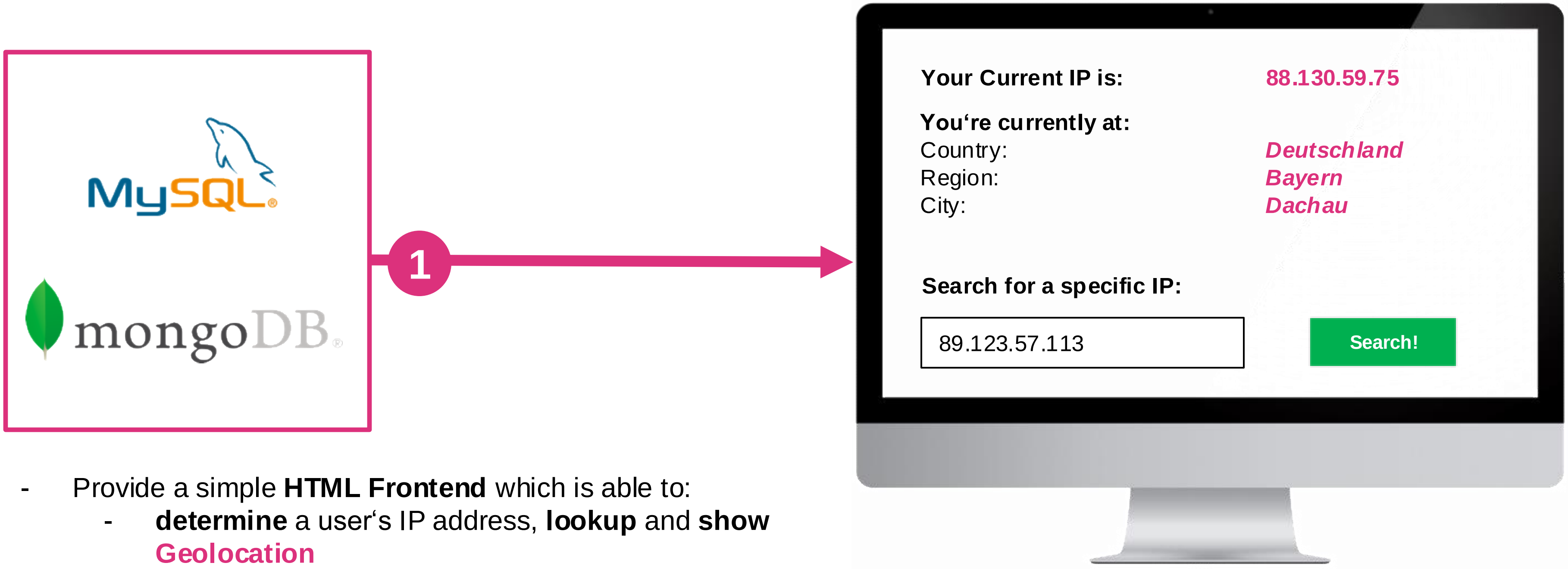
Dataflow: **3. Enhance Data And Save Results**



/user/hadoop/geo_ip/**final**/locations/de
/user/hadoop/geo_ip/**final**/locations/es/
/user/hadoop/geo_ip/**final**/locations/en/
…
/user/hadoop/geo_ip/**final**/blocks/*IPv4*
/user/hadoop/geo_ip/**final**/blocks/*IPv6*

- enhance data (e.g. for later querying)
- use *Hive*, *Python*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



Your Current IP is: 88.130.59.75

You‘re currently at:
Country: *Deutschland*
Region: *Bayern*
City: *Dachau*

Search for a specific IP:

89.123.57.113

- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user‘s IP address, **lookup** and **show Geolocation**
  - process user input (IP…) and check against end-user database
  - Display result Geolocation

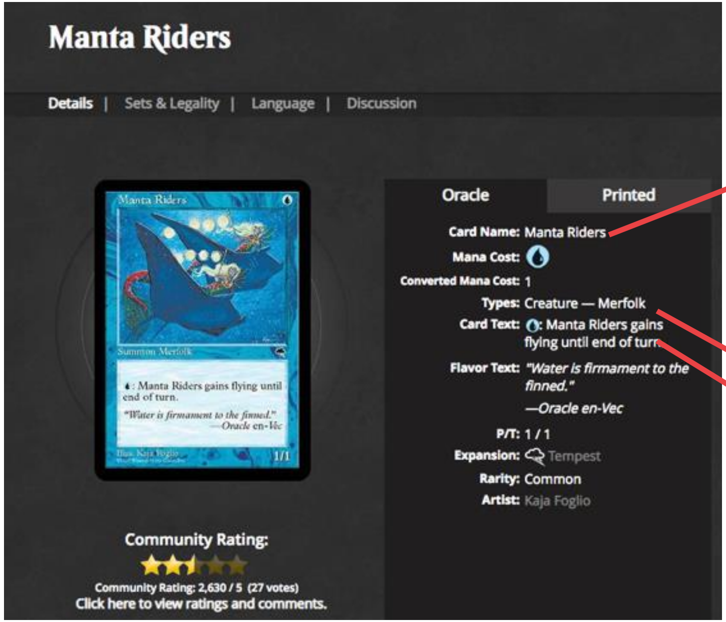
Create MTG Trading Card Database By Crawler

Practical Exam

Goal

magicthegathering.io provides up-to-date information regarding all MTG trading cards available:

- https://gatherer.wizards.com/Pages/



Manta Riders

Details | Sets & Legality | Language | Discussion

Oracle | Printed

Card Name: Manta Riders

Mana Cost:

Converted Mana Cost: 1

Types: Creature — Merfolk

Card Text: : Manta Riders gains flying until end of turn.

Flavor Text: "Water is firmament to the finned."
—Oracle en-Vec

P/T: 1 / 1

Expansion: Tempest

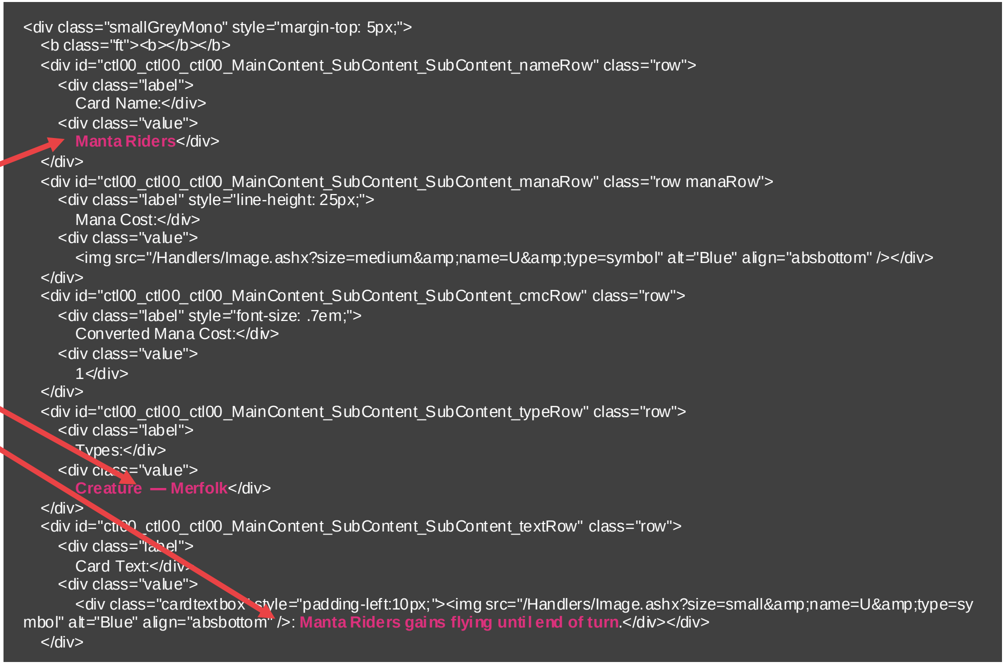
Rarity: Common

Artist: Kaja Foglio

Community Rating:

Community Rating: 2,630 / 5 (27 votes)
Click here to view ratings and comments.

http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=4711



```
<div class="smallGreyMono" style="margin-top: 5px;">
   <b class="ft"><b></b></b>
   <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_nameRow" class="row">
      <div class="label">
         Card Name:</div>
      <div class="value">
         Manta Riders</div>
   </div>
   <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_manaRow" class="row manaRow">
      <div class="label" style="line-height: 25px;">
         Mana Cost:</div>
      <div class="value">
         <img src="/Handlers/Image.ashx?size=medium&amp;name=U&amp;type=symbol" alt="Blue" align="absbottom" /></div>
   </div>
   <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_cmcRow" class="row">
      <div class="label" style="font-size: .7em;">
         Converted Mana Cost:</div>
      <div class="value">
         1</div>
   </div>
   <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_typeRow" class="row">
      <div class="label">
         Types:</div>
      <div class="value">
         Creature  — Merfolk</div>
   </div>
   <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_textRow" class="row">
      <div class="label">
         Card Text:</div>
      <div class="value">
         <div class="cardtextbox" style="padding-left:10px;"><img src="/Handlers/Image.ashx?size=small&amp;name=U&amp;type=sy
mbol" alt="Blue" align="absbottom" />: Manta Riders gains flying until end of turn.</div></div>
   </div>
```

www.marcel-mittelstaedt.com

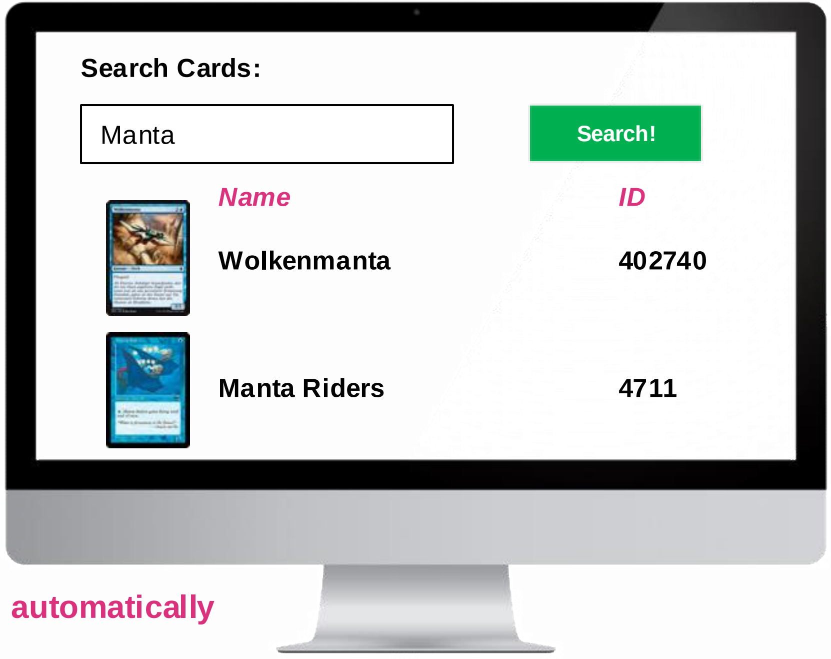
Goal

We want to make use of this data to build a searchable database of all MTG trading cards.

Workflow:

- **Crawl data** from gatherer.wizards.com
- **Parse** required **information** and save them to **HDFS** (queryable through Hive)
- **Export** MTG data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

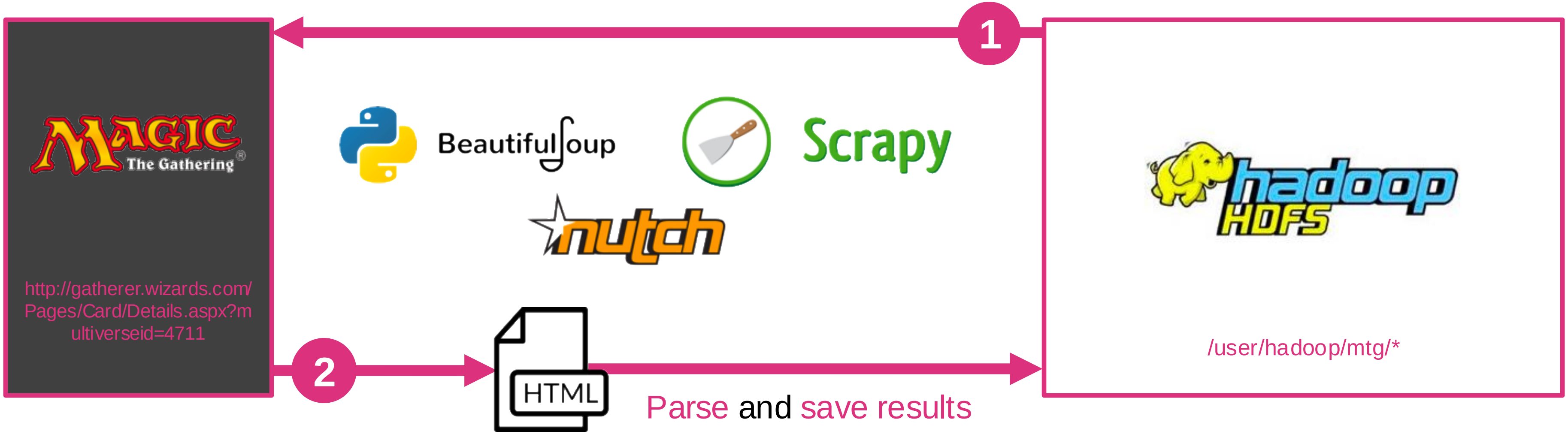


Search Cards:

Manta

	Name	*ID*
	Wolkenmanta	**402740**
	Manta Riders	**4711**

Dataflow: **1. Get MTG Data**



Crawl HTML pages

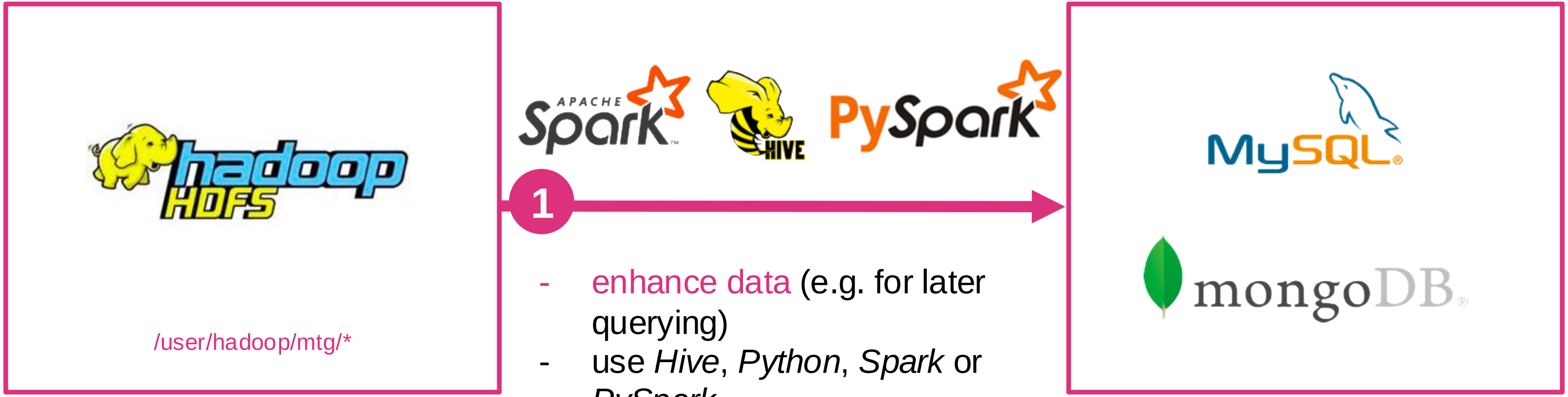
http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=4711

/user/hadoop/mtg/*

Parse and save results using, e.g.:

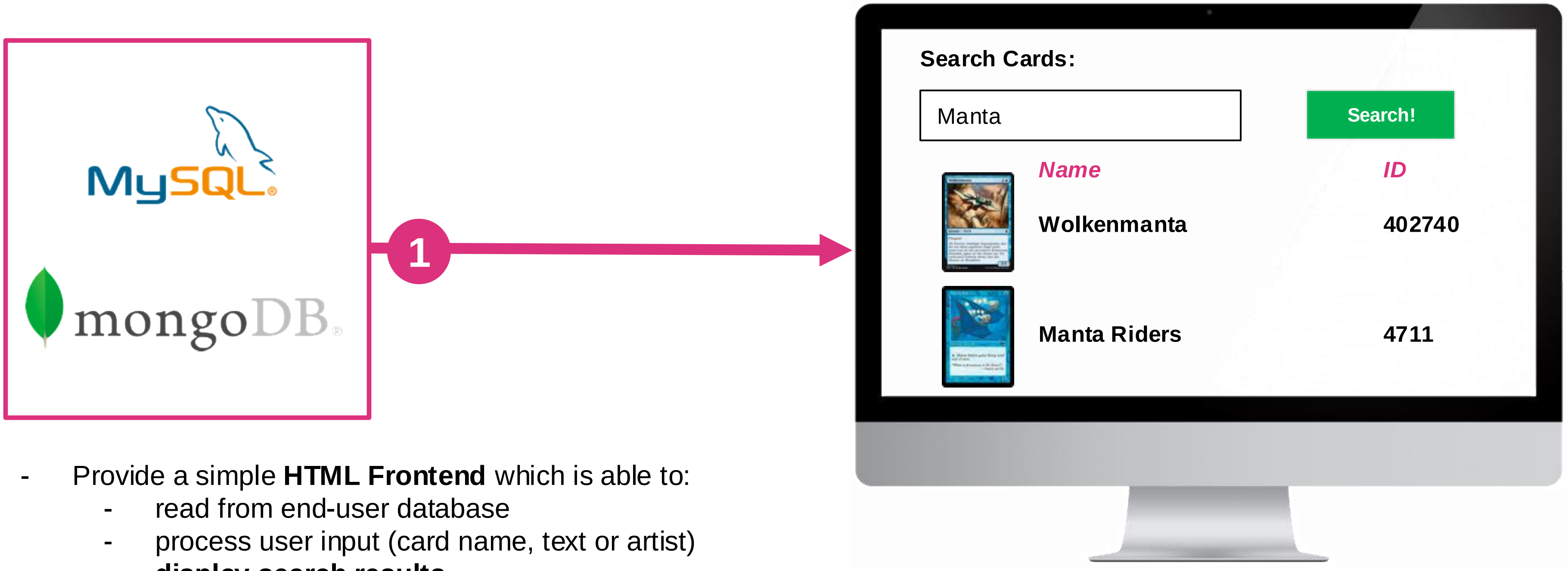
- Python BeautifulSoup
- Scrapy
- Apache nutch

Dataflow: **3. Enhance Data And Save Results**



- enhance data (e.g. for later querying)
- use *Hive*, *Python*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



Search Cards:

Manta

	Name	*ID*
	Wolkenmanta	**402740**
	Manta Riders	**4711**

- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

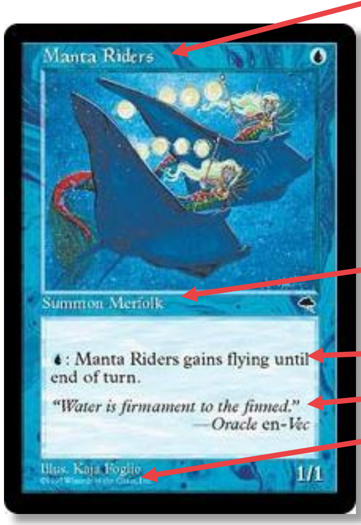
Create MTG Trading Card Database By API

Practical Exam

Goal

magicthegathering.io provides up-to-date information regarding all MTG trading cards available:

- https://docs.magicthegathering.io/
- E.g. https://api.magicthegathering.io/v1/cards/4711


Manta Riders
Summon Merfolk
: Manta Riders gains flying until end of turn.
"Water is firmament to the finned."
—Oracle en-Vec
Illus. Kaja Foglio
1/1

```
{
  "card":{
    "name":"Manta Riders",
    "manaCost":"{U}",
    "cmc":1,
    "colors":[
      "Blue"
    ],
    "colorIdentity":[
      "U"
    ],
    "type":"Creature — Merfolk",
    "types":[
      "Creature"
    ],
    "subtypes":[
      "Merfolk"
    ],
    "rarity":"Common",
    "set":"TMP",
    "setName":"Tempest",
    "text":"{U}: Manta Riders gains flying until end of turn.",
    "flavor":"\"Water is firmament to the finned.\"\n—Oracle en-Vec",
    "artist":"Kaja Foglio",
    "power":"1",
    "toughness":"1",
    "layout":"normal",
    "multiverseid":4711,
    "imageUrl":"http://gatherer.wizards.com/Handlers/Image.ashx?multiverseid=4711&type=
card",
 […]
```

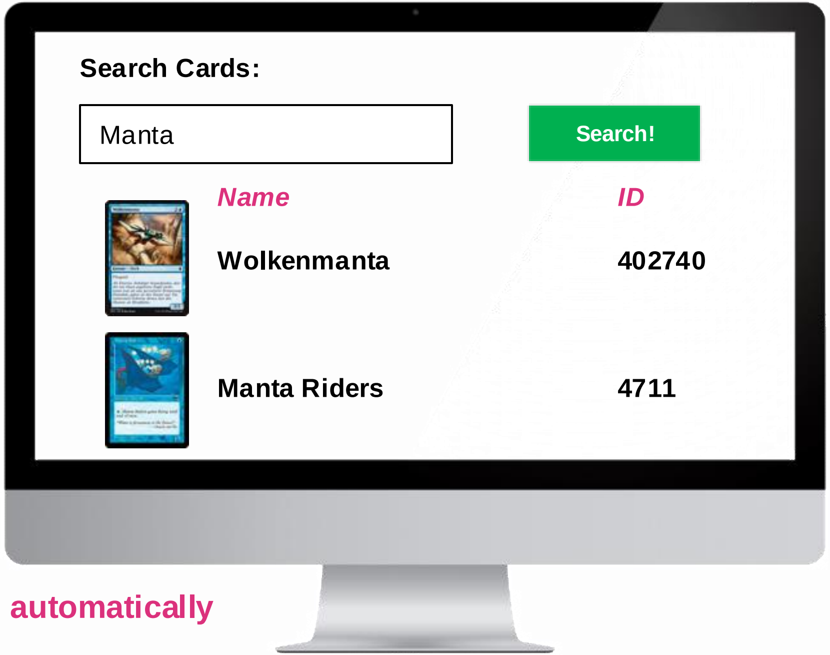
Goal

We want to make use of this data to build a searchable database of all MTG trading cards.

Workflow:

- **Gather data** from api.magicthegathering.io
- **Save raw data** (*JSON files*) to HDFS
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** MTG data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

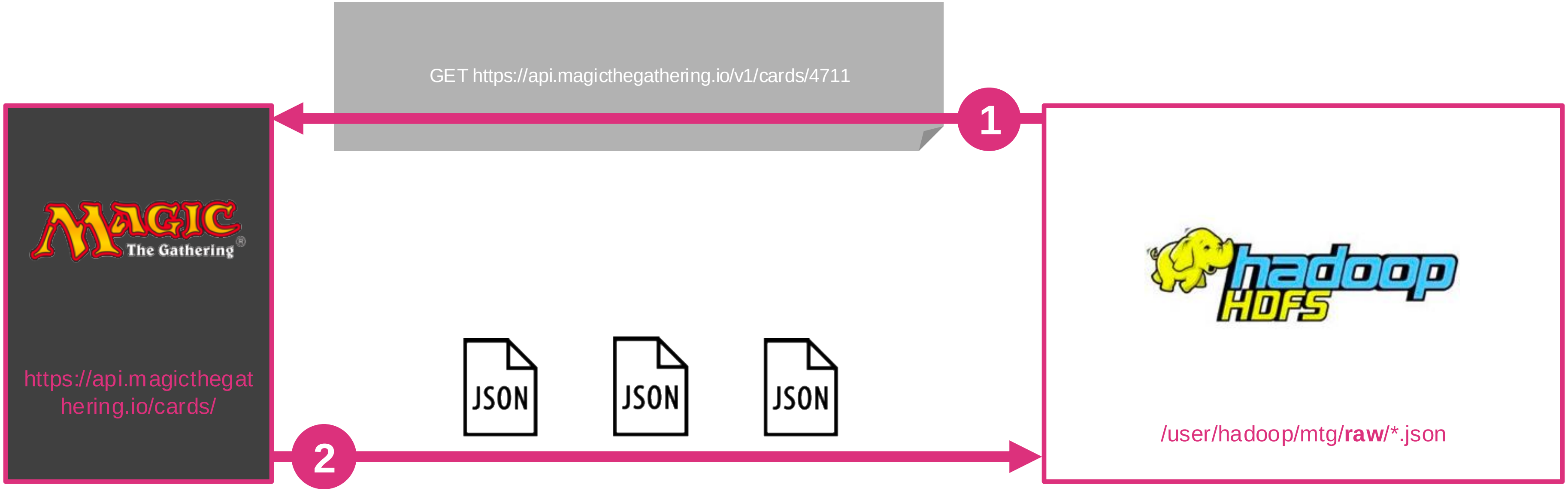


Search Cards:

Manta

	Name	*ID*
	Wolkenmanta	**402740**
	Manta Riders	**4711**

Dataflow: **1. Get MTG Data**

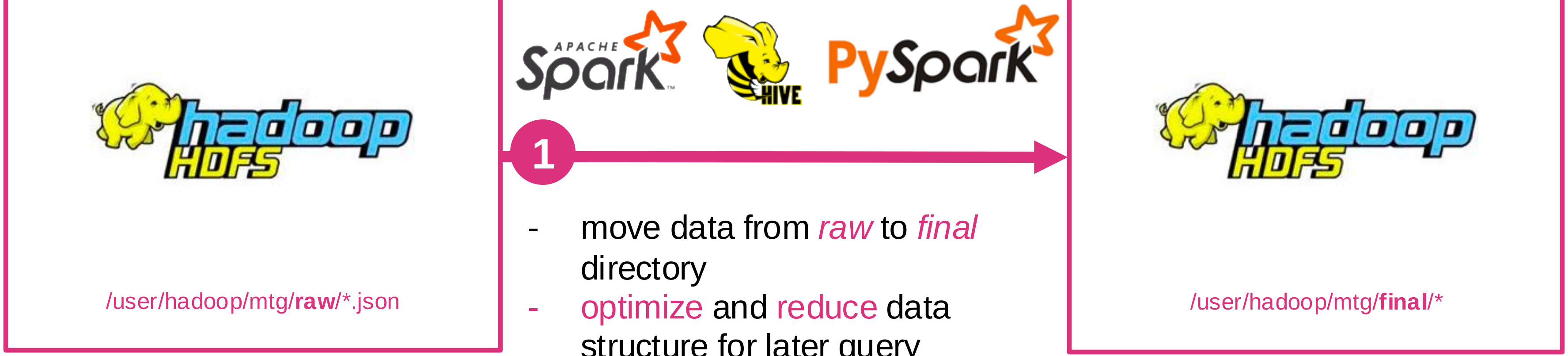


GET https://api.magicthegathering.io/v1/cards/4711

https://api.magicthegathering.io/cards/

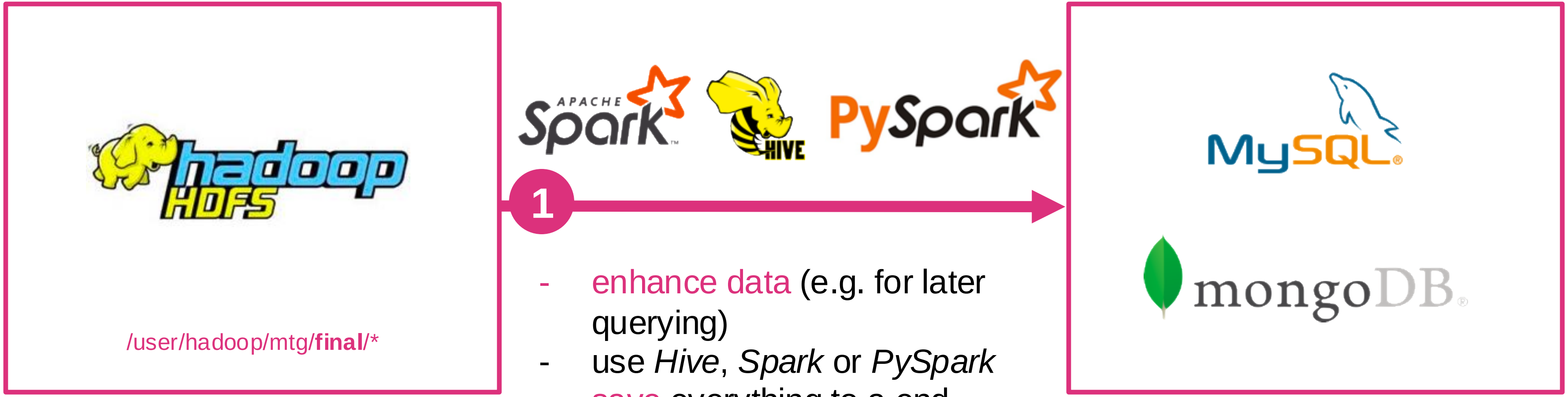
/user/hadoop/mtg/**raw**/*.json

Dataflow: **2. Raw To Final Transfer**



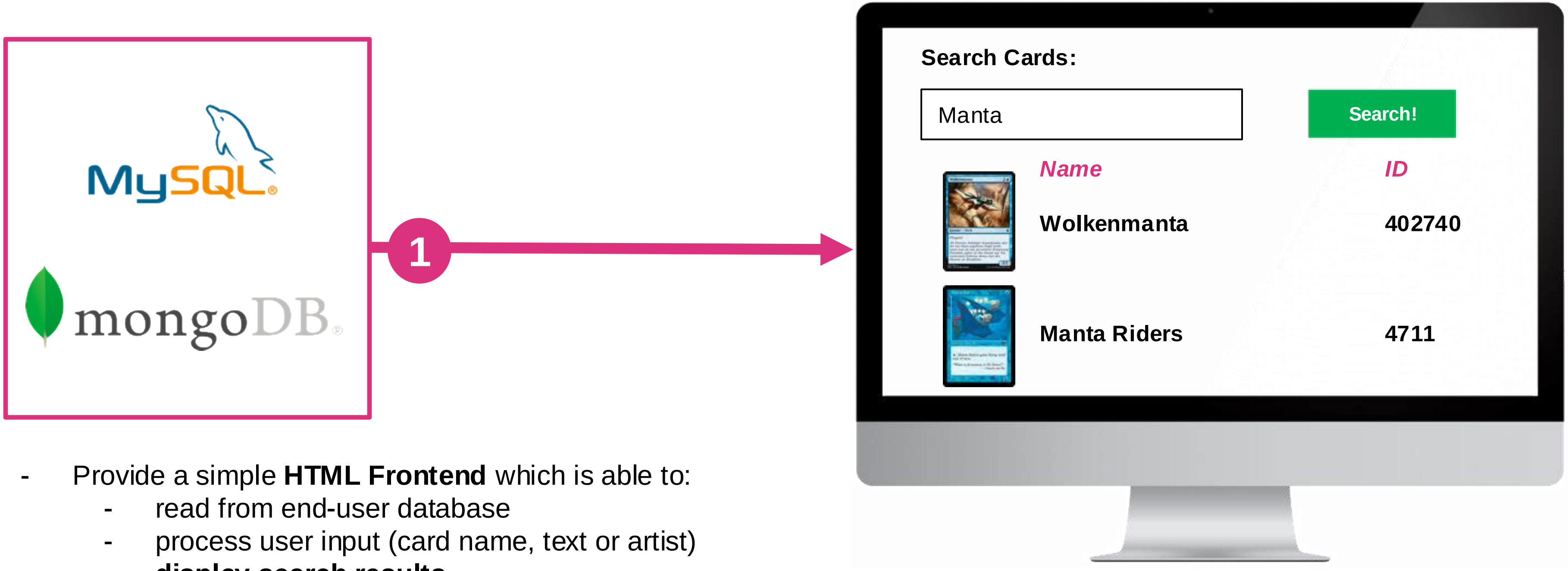
- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

Dataflow: **3. Enhance Data And Save Results**



- enhance data (e.g. for later querying)
- use *Hive*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

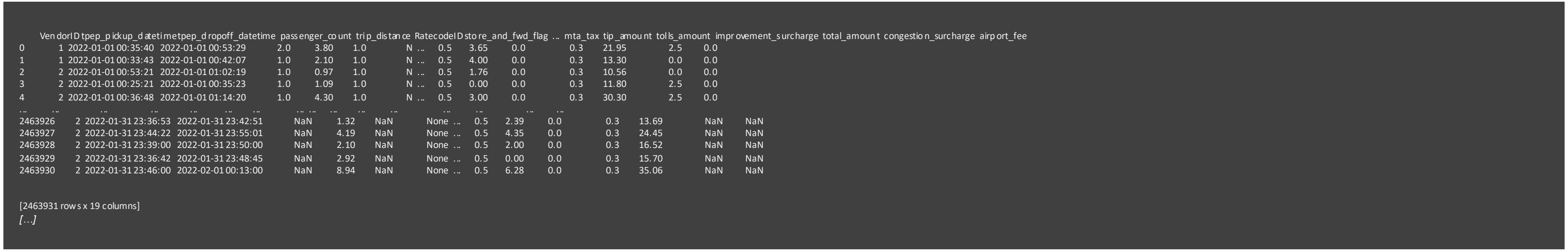
Use NYC Taxi Trip Record Data To Calculate Performance KPIs

Practical Exam

Goal

NYC.gov provides monthly exports of NYC yellow taxi trip records:

- https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- Latest Full Dumps:
  - *https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-01.parquet*
  - *https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-02.parquet*
  - *https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-03.parquet*
  - …



```
         VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  trip_distance  RatecodeID store_and_fwd_flag  ...  mta_tax  tip_amount  tolls_amount  improvement_surcharge  total_amount  congestion_surcharge  airport_fee
0               1  2022-01-01 00:35:40  2022-01-01 00:53:29              2.0           3.80         1.0                  N  ...      0.5        3.65           0.0                    0.3         21.95                   2.5          0.0
1               1  2022-01-01 00:33:43  2022-01-01 00:42:07              1.0           2.10         1.0                  N  ...      0.5        4.00           0.0                    0.3         13.30                   0.0          0.0
2               2  2022-01-01 00:53:21  2022-01-01 01:02:19              1.0           0.97         1.0                  N  ...      0.5        1.76           0.0                    0.3         10.56                   0.0          0.0
3               2  2022-01-01 00:25:21  2022-01-01 00:35:23              1.0           1.09         1.0                  N  ...      0.5        0.00           0.0                    0.3         11.80                   2.5          0.0
4               2  2022-01-01 00:36:48  2022-01-01 01:14:20              1.0           4.30         1.0                  N  ...      0.5        3.00           0.0                    0.3         30.30                   2.5          0.0
...           ...                  ...                  ...              ...            ...         ...                ...  ...      ...         ...           ...                    ...           ...                   ...          ...
2463926         2  2022-01-31 23:36:53  2022-01-31 23:42:51              NaN           1.32         NaN               None  ...      0.5        2.39           0.0                    0.3         13.69                   NaN          NaN
2463927         2  2022-01-31 23:44:22  2022-01-31 23:55:01              NaN           4.19         NaN               None  ...      0.5        4.35           0.0                    0.3         24.45                   NaN          NaN
2463928         2  2022-01-31 23:39:00  2022-01-31 23:50:00              NaN           2.10         NaN               None  ...      0.5        2.00           0.0                    0.3         16.52                   NaN          NaN
2463929         2  2022-01-31 23:36:42  2022-01-31 23:48:45              NaN           2.92         NaN               None  ...      0.5        0.00           0.0                    0.3         15.70                   NaN          NaN
2463930         2  2022-01-31 23:46:00  2022-02-01 00:13:00              NaN           8.94         NaN               None  ...      0.5        6.28           0.0                    0.3         35.06                   NaN          NaN


[2463931 rows x 19 columns]
[…]
```

yellow_tripdata_2022-01.parquet

Goal

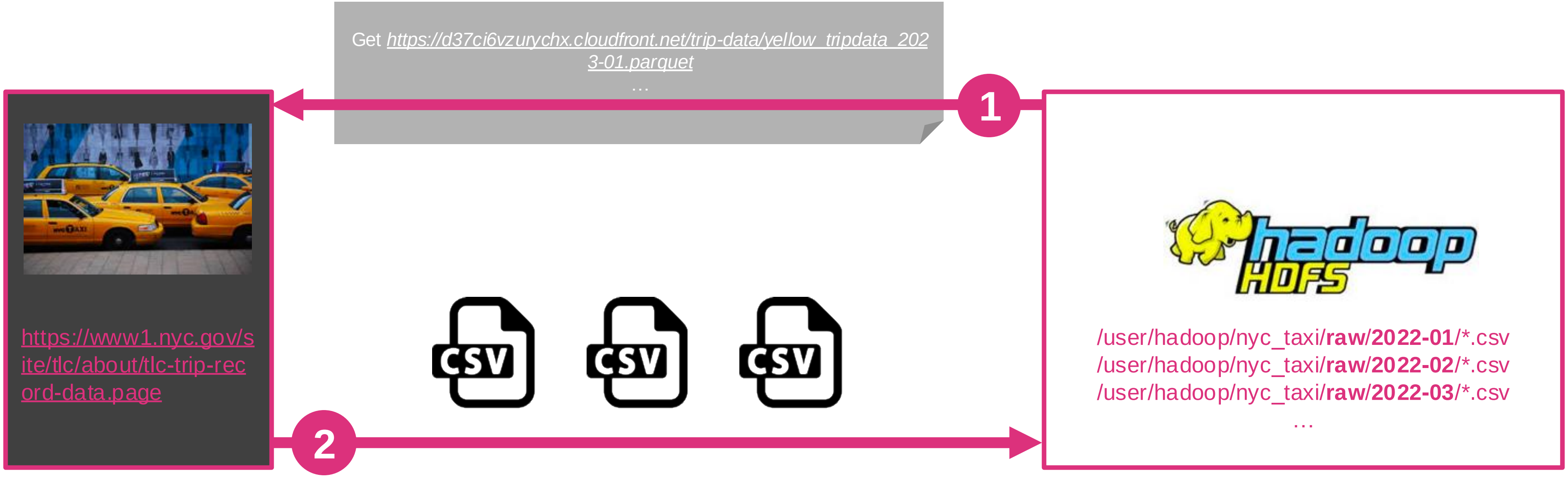
We want to make use of this data to calculate some KPIs

Workflow:

- **Gather** **data** from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- **Save** **raw data** (*CSV files*) to HDFS (partitioned by *YYYY-MM*)
- **Optimize**, **reduce** and **clean** **raw** **data** and save it to **final** directory on HDFS
- **Calculate KPIs** and **Export** them to an **Excel File**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

Dataflow: **1. Get TLC NYC Taxi Data**

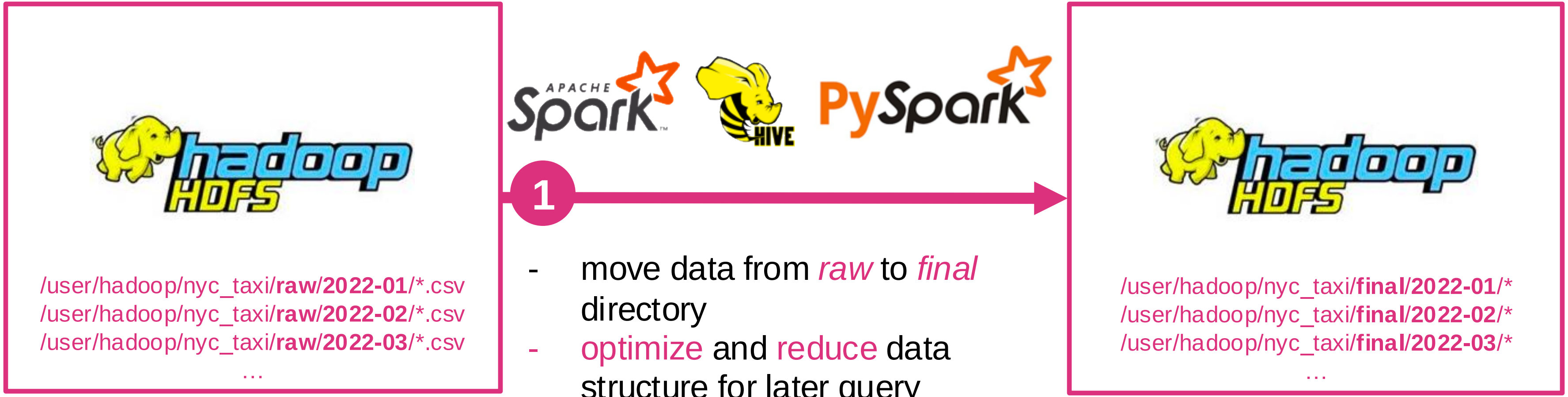


Get https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-01.parquet
…

https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

/user/hadoop/nyc_taxi/**raw**/**2022-01**/*.csv
/user/hadoop/nyc_taxi/**raw**/**2022-02**/*.csv
/user/hadoop/nyc_taxi/**raw**/**2022-03**/*.csv
…

Dataflow: **2. Raw To Final Transfer**

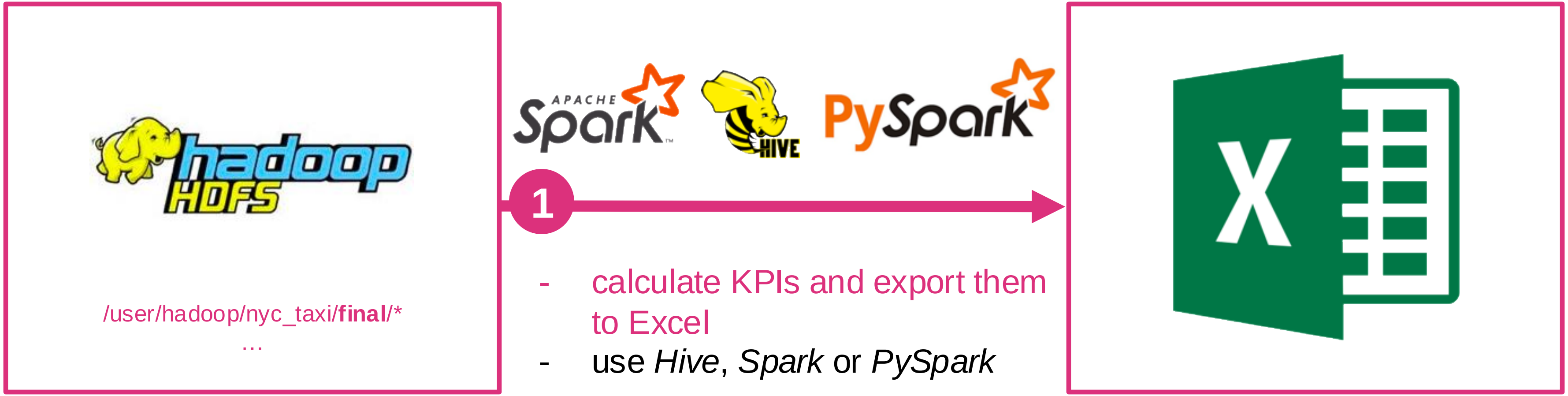


/user/hadoop/nyc_taxi/**raw/2022-01**/*.csv
/user/hadoop/nyc_taxi/**raw/2022-02**/*.csv
/user/hadoop/nyc_taxi/**raw/2022-03**/*.csv
…

- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

/user/hadoop/nyc_taxi/**final/2022-01**/*
/user/hadoop/nyc_taxi/**final/2022-02**/*
/user/hadoop/nyc_taxi/**final/2022-03**/*
…

Dataflow: **3. Calculate And Export KPIs**



/user/hadoop/nyc_taxi/**final**/*
…

- calculate KPIs and export them to Excel
- use *Hive*, *Spark* or *PySpark*

Dataflow: **4. KPIs To Calculate**

Calculate per Month:

- Average Trip Duration (in minutes)
- Average Trip Distance (in miles)
- Average total amount (in USD)
- Average tip amount (in USD)
- Average passenger count (as Number)
- Usage Share by payment type (credit card, cash… in percent)
- Usage share per timeslot (in percent):
  - 00:00-06:00
  - 06:00-12:00
  - 12:00-18:00
  - 18:00-24:00

Use *kaggle.com* Hubway Data To Calculate Bike Sharing Usage KPIs

Practical Exam

Goal

kaggle.com provides monthly exports of Hubway bike sharing trip records:

- https://www.bluebikes.com/
- Latest Full Dumps: https://www.kaggle.com/acmeyer/hubway-data

```
"tripduration","starttime","stoptime","start station id","start station name","start station latitude","start station longitude","end station id","end station name","end station latitude","end station longitude","bikeid","usertype","birth year","gender"
"133","2015-12-01 00:01:52","2015-12-01 00:04:06","9","Agganis Arena - 925 Comm Ave.","42.351246","-71.115639","41","Packard's Corner - Comm. Ave. at Brighton Ave.","42.352261","-71.123831","199","Customer","1995","1"
"1522","2015-12-01 00:05:30","2015-12-01 00:30:53","41","Packard's Corner - Comm. Ave. at Brighton Ave.","42.352261","-71.123831","54","Tremont St / West St","42.354979","-71.063348","876","Customer","1983","1"
"153","2015-12-01 00:07:46","2015-12-01 00:10:20","75","Lafayette Square at Mass Ave / Main St / Columbia St","42.36346469304347","-71.10057324171066","67","MIT at Mass Ave / Amherst St","42.3581","-71.093198","757","Subscriber","1995","1"
"435","2015-12-01 00:07:48","2015-12-01 00:15:04","68","Central Square at Mass Ave / Essex St","42.36507","-71.1031","29","Innovation Lab - 125 Western Ave. at Batten Way","42.363732","-71.124565","853","Subscriber","1988","1"
"1208","2015-12-01 00:12:15","2015-12-01 00:32:23","36","Boston Public Library - 700 Boylston St.","42.349673","-71.077303","110","Harvard University Gund Hall at Quincy St / Kirkland S","42.376369","-71.114025","437","Customer","1982","1"
"1117","2015-12-01 00:16:31","2015-12-01 00:35:09","31","Seaport Hotel","42.348833","-71.041747","67","MIT at Mass Ave / Amherst St","42.3581","-71.093198","1161","Subscriber","1988","1"
"1287","2015-12-01 00:16:50","2015-12-01 00:38:18","10","B.U. Central - 725 Comm. Ave.","42.350406","-71.108279","23","Mayor Martin J Walsh - 28 State St","42.35892","-71.057629","565","Subscriber","1966","1"
```

Goal

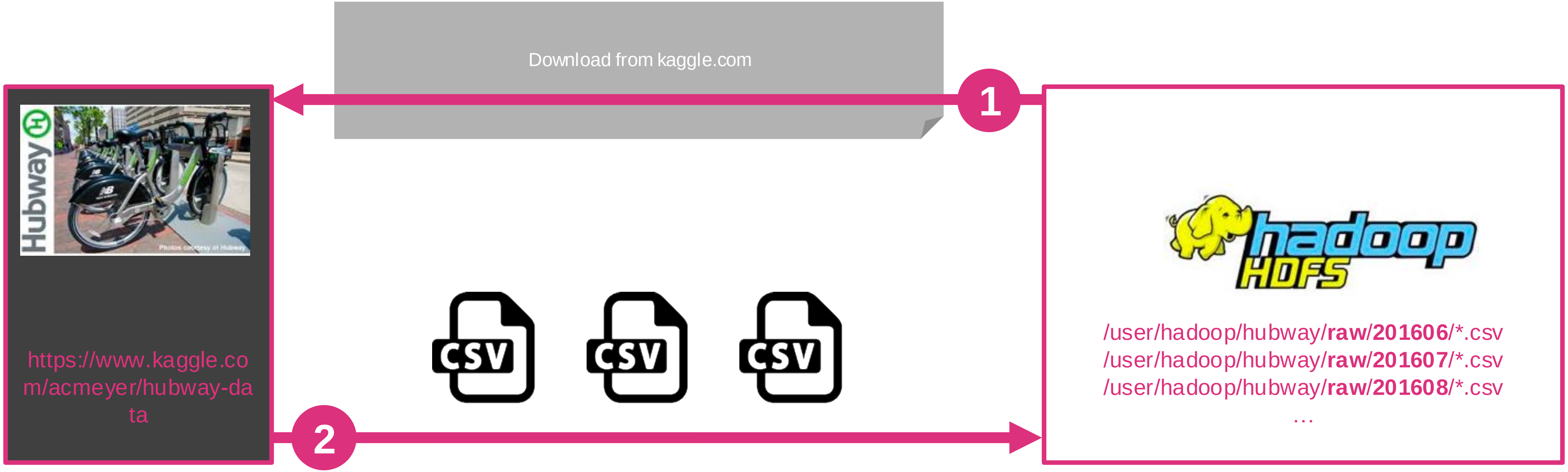
We want to make use of this data to calculate some Usage KPIs.

Workflow:

- **Gather data** from https://www.kaggle.com/acmeyer/hubway-data
- **Save raw data** (*CSV files*) to HDFS (partitioned by *YYYYMM*)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Calculate KPIs** and **Export** them to an **Excel File**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

Dataflow: **1. Get Hubway Bike Sharing Data**

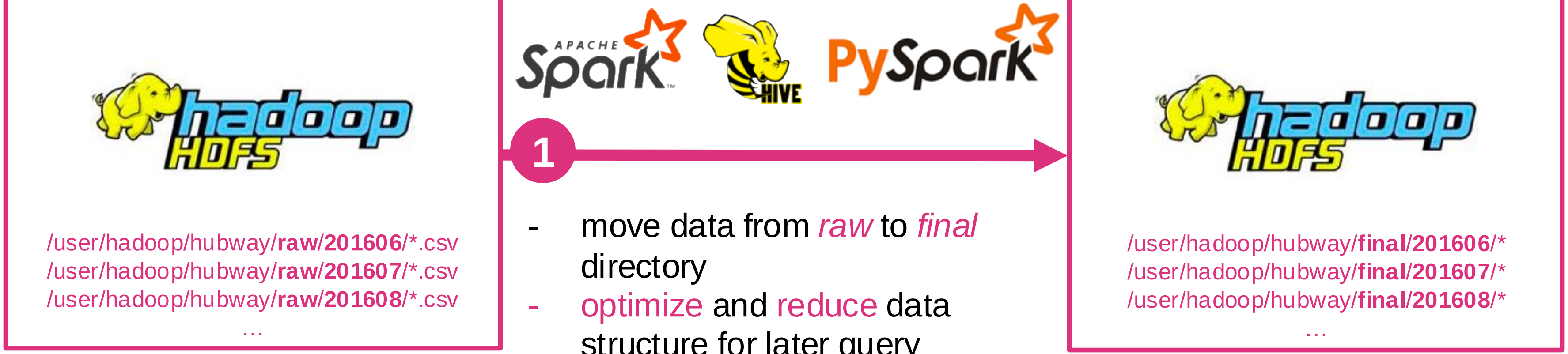


Download from kaggle.com

https://www.kaggle.com/acmeyer/hubway-data

/user/hadoop/hubway/**raw/201606**/*.csv
/user/hadoop/hubway/**raw/201607**/*.csv
/user/hadoop/hubway/**raw/201608**/*.csv
…

Dataflow: **2. Raw To Final Transfer**

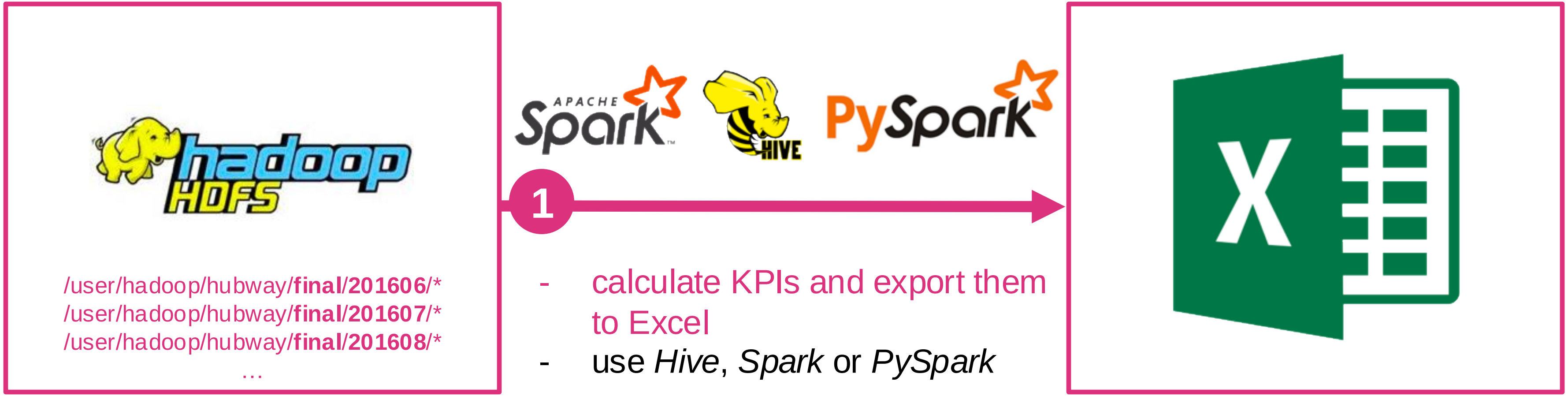


/user/hadoop/hubway/**raw/201606**/*.csv
/user/hadoop/hubway/**raw/201607**/*.csv
/user/hadoop/hubway/**raw/201608**/*.csv
…

- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

/user/hadoop/hubway/**final/201606**/*
/user/hadoop/hubway/**final/201607**/*
/user/hadoop/hubway/**final/201608**/*
…

Dataflow: **3. Calculate And Export KPIs**



/user/hadoop/hubway/**final/201606**/*
/user/hadoop/hubway/**final/201607**/*
/user/hadoop/hubway/**final/201608**/*
…

- calculate KPIs and export them to Excel
- use *Hive*, *Spark* or *PySpark*

Dataflow: **4. KPIs To Calculate**

Calculate per Month:

- Average Trip Duration (in minutes)
- Average Trip Distance (in km)
- Usage Share by gender (in percent)
- Usage Share by age (in percent)
- Top 10 most used bikes
- Top 10 most start stations
- Top 10 most end stations
- Usage share per timeslot (in percent):
  - 00:00-06:00
  - 06:00-12:00
  - 12:00-18:00
  - 18:00-24:00

Use OpenCellID Data To Estimate GSM/UMTS/LTE Coverage

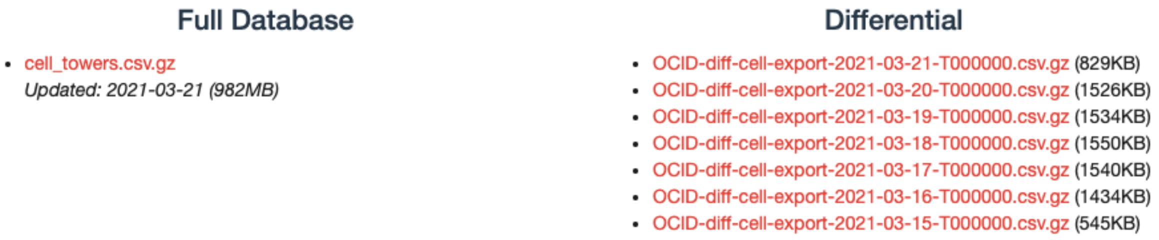
Practical Exam

www.marcel-mittelstaedt.com

Goal

OpenCellID.com provides regulary exports of worldwide cell data:

- https://www.opencellid.org
- Latest Full Dump and Diffs: https://www.opencellid.org/downloads.php

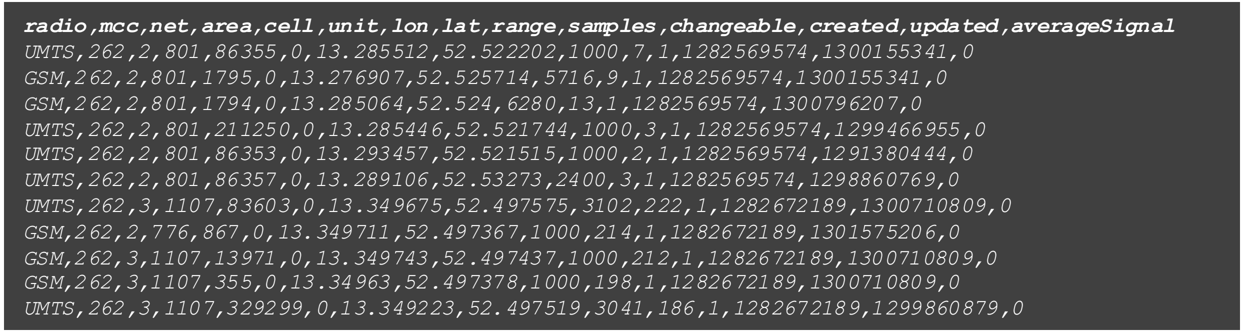


Full Database

- cell_towers.csv.gz
  Updated: 2021-03-21 (982MB)

Differential

- OCID-diff-cell-export-2021-03-21-T000000.csv.gz (829KB)
- OCID-diff-cell-export-2021-03-20-T000000.csv.gz (1526KB)
- OCID-diff-cell-export-2021-03-19-T000000.csv.gz (1534KB)
- OCID-diff-cell-export-2021-03-18-T000000.csv.gz (1550KB)
- OCID-diff-cell-export-2021-03-17-T000000.csv.gz (1540KB)
- OCID-diff-cell-export-2021-03-16-T000000.csv.gz (1434KB)
- OCID-diff-cell-export-2021-03-15-T000000.csv.gz (545KB)



```
radio,mcc,net,area,cell,unit,lon,lat,range,samples,changeable,created,updated,averageSignal
UMTS,262,2,801,86355,0,13.285512,52.522202,1000,7,1,1282569574,1300155341,0
GSM,262,2,801,1795,0,13.276907,52.525714,5716,9,1,1282569574,1300155341,0
GSM,262,2,801,1794,0,13.285064,52.524,6280,13,1,1282569574,1300796207,0
UMTS,262,2,801,211250,0,13.285446,52.521744,1000,3,1,1282569574,1299466955,0
UMTS,262,2,801,86353,0,13.293457,52.521515,1000,2,1,1282569574,1291380444,0
UMTS,262,2,801,86357,0,13.289106,52.53273,2400,3,1,1282569574,1298860769,0
UMTS,262,3,1107,83603,0,13.349675,52.497575,3102,222,1,1282672189,1300710809,0
GSM,262,2,776,867,0,13.349711,52.497367,1000,214,1,1282672189,1301575206,0
GSM,262,3,1107,13971,0,13.349743,52.497437,1000,212,1,1282672189,1300710809,0
GSM,262,3,1107,355,0,13.34963,52.497378,1000,198,1,1282672189,1300710809,0
UMTS,262,3,1107,329299,0,13.349223,52.497519,3041,186,1,1282672189,1299860879,0
```

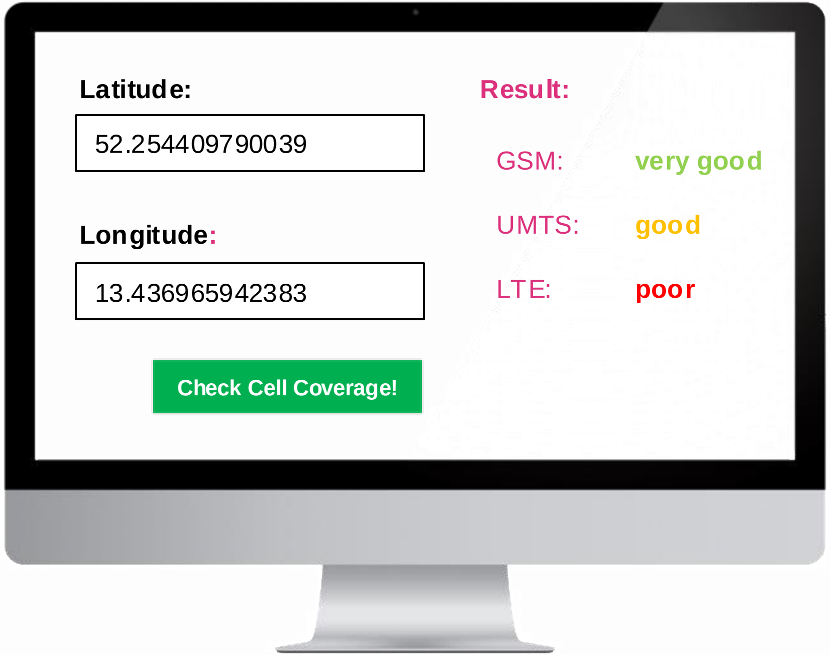
cell_towers.csv

Goal

We want to make use of this data to estimate the coverage of GSM, UMTS and LTE for a certain place (*latitude, longitude*).

Workflow:

- **Gather data** from OpenCellID.com
- **Save raw data** (*CSV files*) to HDFS (partitioned by radio, e.g. *GSM*, *UMTS*, *LTE*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** address data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Latitude, Longitude…)
  - checks against OpenCellID data in end-user database
  - Display result (GSM, LTE and UMTS coverage)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. **Pentaho Data Integration** or **Airflow**) and **run automatically**



Latitude: 52.254409790039

Longitude: 13.436965942383

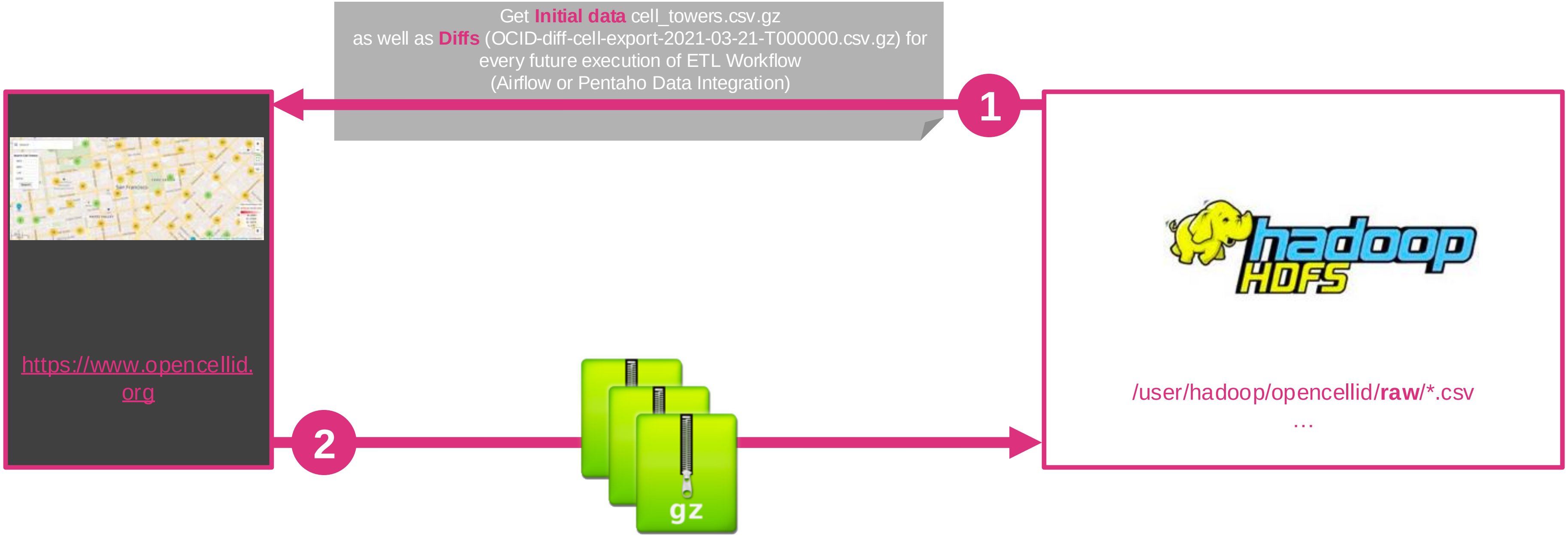
Result:

GSM: very good

UMTS: good

LTE: poor

Dataflow: **1. Get Cell Data**

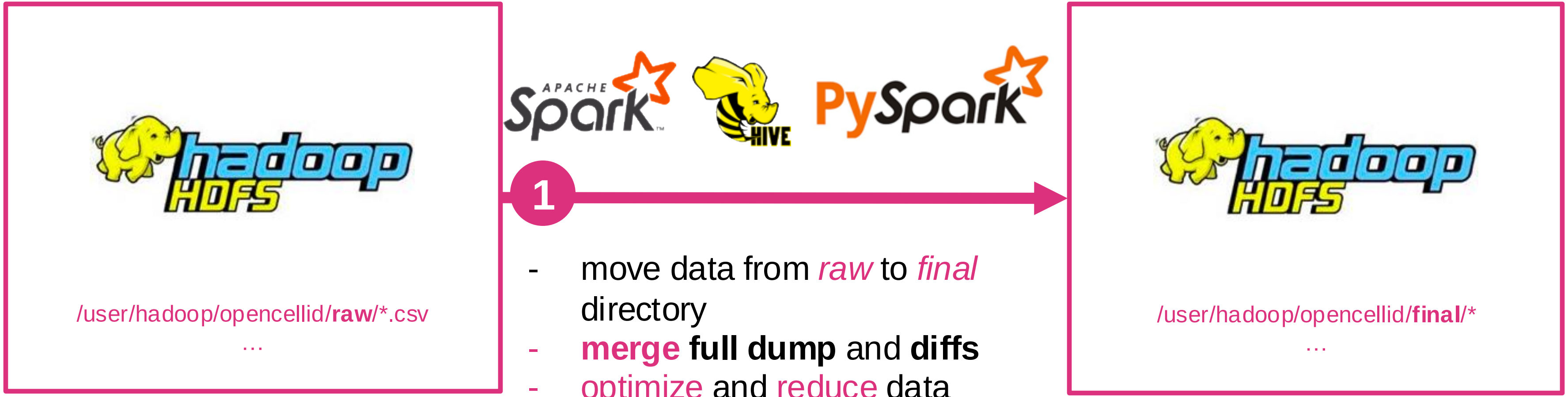


Get **Initial data** cell_towers.csv.gz
as well as **Diffs** (OCID-diff-cell-export-2021-03-21-T000000.csv.gz) for
every future execution of ETL Workflow
(Airflow or Pentaho Data Integration)

https://www.opencellid.org

/user/hadoop/opencellid/**raw**/*.csv
…

Dataflow: **2. Raw To Final Transfer**



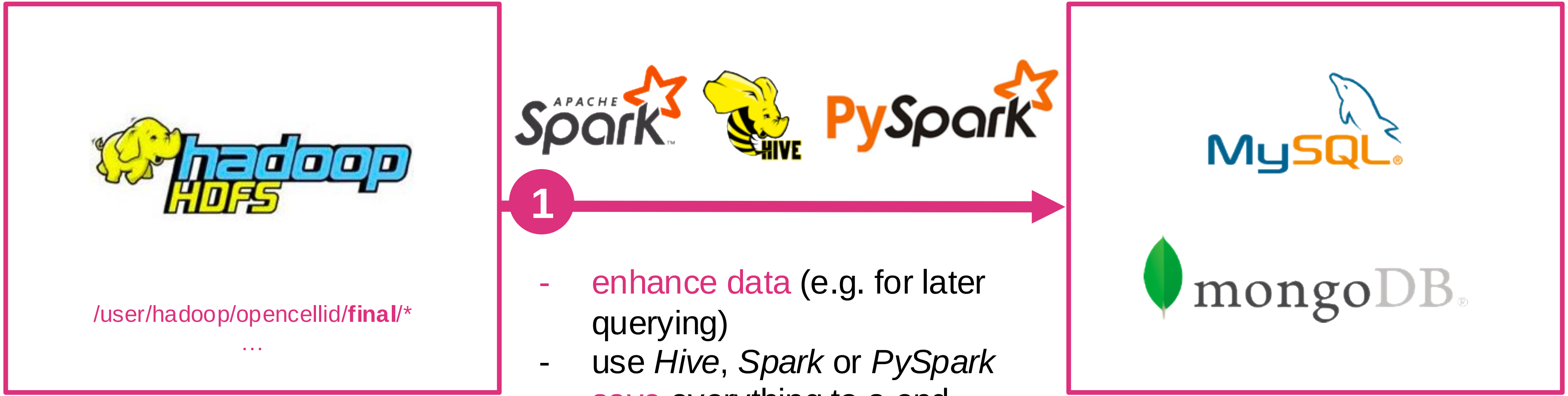
/user/hadoop/opencellid/**raw**/*.csv
…

1

/user/hadoop/opencellid/**final**/*
…

- move data from *raw* to *final* directory
- **merge full dump** and **diffs**
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

Dataflow: **3. Enhance Data And Save Results**

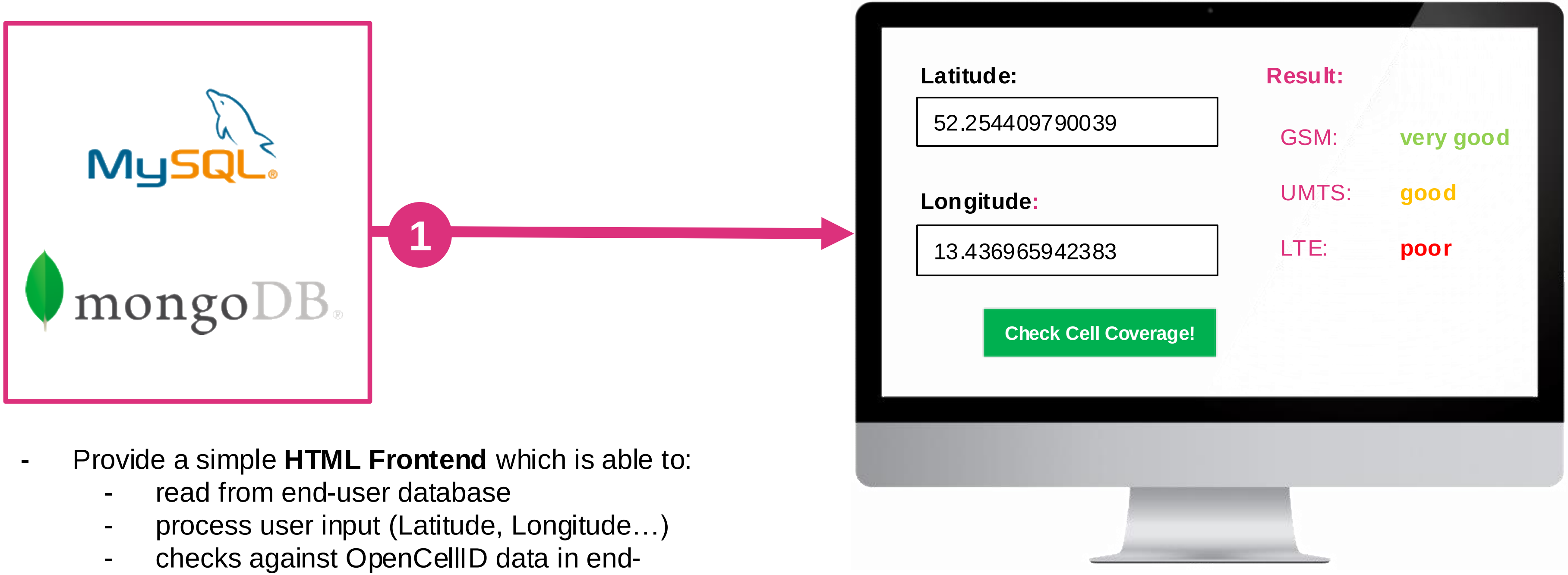


/user/hadoop/opencellid/**final**/*

…

- enhance data (e.g. for later querying)
- use *Hive*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



Latitude:

52.254409790039

Longitude:

13.436965942383

Check Cell Coverage!

Result:

GSM:	very good
UMTS:	good
LTE:	poor

- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Latitude, Longitude…)
  - checks against OpenCellID data in end-user database
  - Display result (GSM, LTE and UMTS coverage)

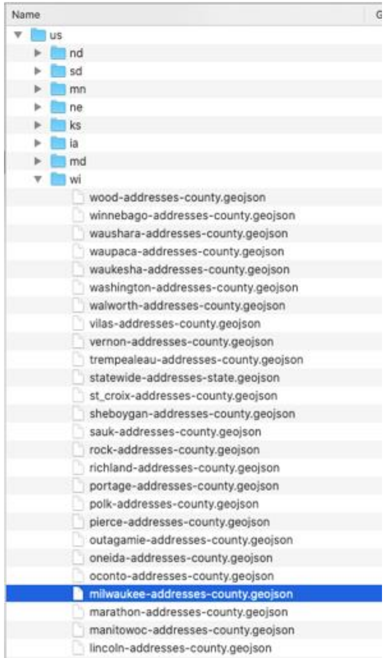
Use OpenAdresses Data To Validate Adresses

Practical Exam

Goal

OpenAddresses.io provides regulary exports of worldwide adresses (we will focus on US south/west/midwest/northeast for now):

- https://batch.openaddresses.io/data



```
Name
▼ us
  ▶ nd
  ▶ sd
  ▶ mn
  ▶ ne
  ▶ ks
  ▶ ia
  ▶ md
  ▼ wi
      wood-addresses-county.geojson
      winnebago-addresses-county.geojson
      waushara-addresses-county.geojson
      waupaca-addresses-county.geojson
      waukesha-addresses-county.geojson
      washington-addresses-county.geojson
      walworth-addresses-county.geojson
      vilas-addresses-county.geojson
      vernon-addresses-county.geojson
      trempealeau-addresses-county.geojson
      statewide-addresses-state.geojson
      st_croix-addresses-county.geojson
      sheboygan-addresses-county.geojson
      sauk-addresses-county.geojson
      rock-addresses-county.geojson
      richland-addresses-county.geojson
      portage-addresses-county.geojson
      polk-addresses-county.geojson
      pierce-addresses-county.geojson
      outagamie-addresses-county.geojson
      oneida-addresses-county.geojson
      oconto-addresses-county.geojson
      milwaukee-addresses-county.geojson
      marathon-addresses-county.geojson
      manitowoc-addresses-county.geojson
      lincoln-addresses-county.geojson
```

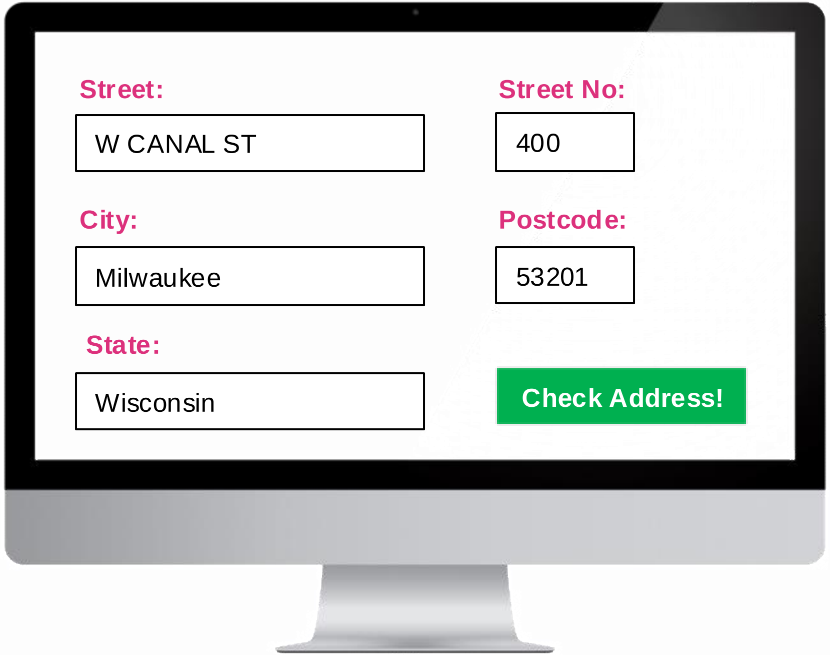
```
{"type":"Feature","properties":{"hash":"394d6a8e3e6cecbf","number":"7705","street":"W LINCOLN AVE","unit":"1","city":"West Allis","district":"","region":"","postcode":"53219","id":""},"geometry":{"type":"Point","coordinates":[-88.0088621,43.0025845]}}
{"type":"Feature","properties":{"hash":"6101cecbe71c7bbe","number":"7705","street":"W LINCOLN AVE","unit":"2","city":"West Allis","district":"","region":"","postcode":"53219","id":""},"geometry":{"type":"Point","coordinates":[-88.0088621,43.0025845]}}
{"type":"Feature","properties":{"hash":"81e3634e904916db","number":"1060","street":"N 115TH ST","unit":"106","city":"Wauwatosa","district":"","region":"","postcode":"53226","id":""},"geometry":{"type":"Point","coordinates":[-88.0551894,43.0441061]}}
{"type":"Feature","properties":{"hash":"fbf0248cdd1623ad","number":"12137","street":"W BURLEIGH ST","unit":"2","city":"Wauwatosa","district":"","region":"","postcode":"53222","id":""},"geometry":{"type":"Point","coordinates":[-88.0649444,43.0741544]}}
{"type":"Feature","properties":{"hash":"6c5867d0d98b7e9a","number":"11515","street":"W CLEVELAND AVE","unit":"B231","city":"West Allis","district":"","region":"","postcode":"53227","id":""},"geometry":{"type":"Point","coordinates":[-88.0560418,42.9946529]}}

[...]
```

Goal

We want to make use of this data to validate adresses entered on a website, to check whether they are real or not.

Workflow:

- **Gather data** from OpenAddresses.io
- **Save raw data** (*JSON files*) to HDFS (partitioned by state
- shortcut, e.g. *wi, nd, sd*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** address data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
    - read from end-user database
    - process user input (Street, City, Postcode…)
    - validate user input against OpenAddress data in end-user database
    - Display result (real or non real address)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. **Pentaho Data Integration** or **Airflow**) and **run automatically**



Street: W CANAL ST

Street No: 400

City: Milwaukee

Postcode: 53201

State: Wisconsin

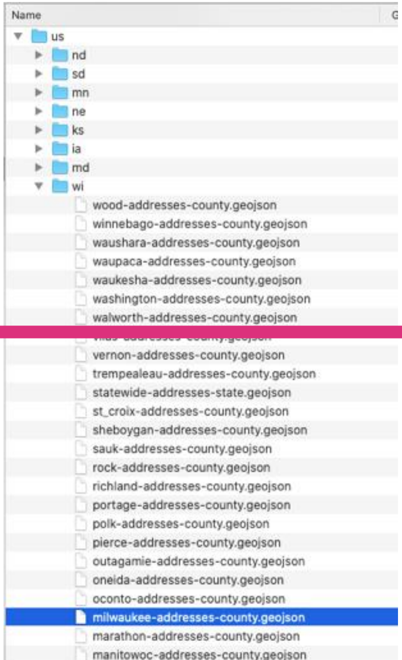
Dataflow: **1. Get Address Data**

http://results.openaddresses.io/

1

Get https://batch.openaddresses.io/data

2

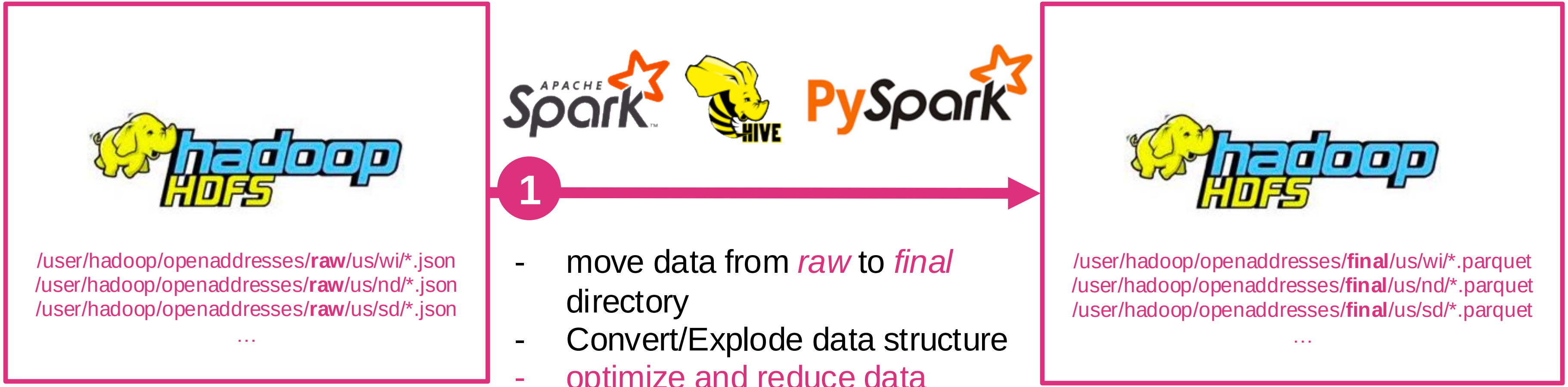


```
Name
▼ us
    ▶ nd
    ▶ sd
    ▶ mn
    ▶ ne
    ▶ ks
    ▶ ia
    ▶ md
    ▼ wi
        wood-addresses-county.geojson
        winnebago-addresses-county.geojson
        waushara-addresses-county.geojson
        waupaca-addresses-county.geojson
        waukesha-addresses-county.geojson
        washington-addresses-county.geojson
        walworth-addresses-county.geojson
        vilas-addresses-county.geojson
        vernon-addresses-county.geojson
        trempealeau-addresses-county.geojson
        statewide-addresses-state.geojson
        st_croix-addresses-county.geojson
        sheboygan-addresses-county.geojson
        sauk-addresses-county.geojson
        rock-addresses-county.geojson
        richland-addresses-county.geojson
        portage-addresses-county.geojson
        polk-addresses-county.geojson
        pierce-addresses-county.geojson
        outagamie-addresses-county.geojson
        oneida-addresses-county.geojson
        oconto-addresses-county.geojson
        milwaukee-addresses-county.geojson
        marathon-addresses-county.geojson
        manitowoc-addresses-county.geojson
```

/user/hadoop/openaddresses/**raw**/us/wi/*.json
/user/hadoop/openaddresses/**raw**/us/nd/*.json
/user/hadoop/openaddresses/**raw**/us/sd/*.json
…

www.marcel-mittelstaedt.com

Dataflow: **2. Raw To Final Transfer**

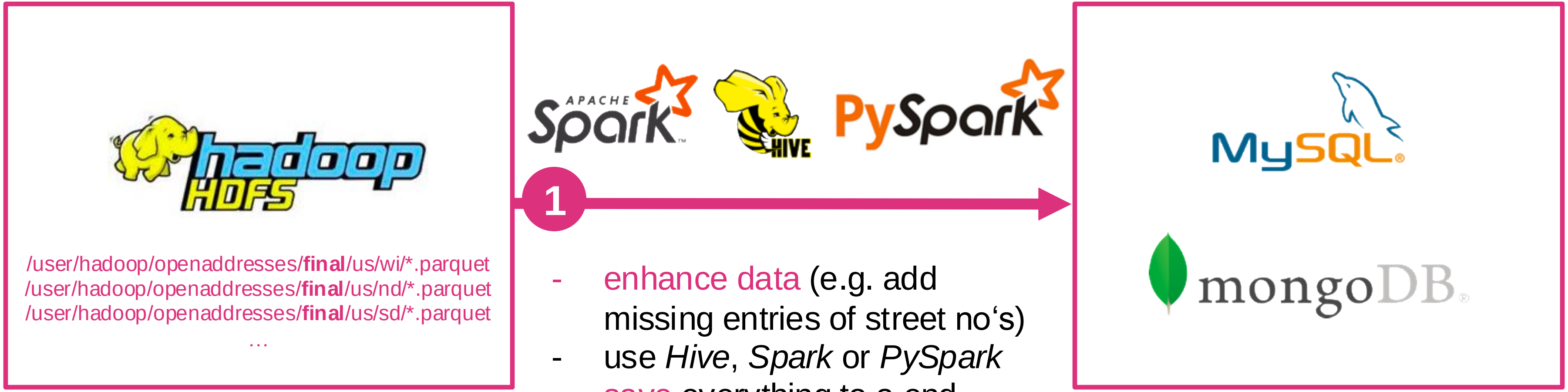


/user/hadoop/openaddresses/**raw**/us/wi/*.json
/user/hadoop/openaddresses/**raw**/us/nd/*.json
/user/hadoop/openaddresses/**raw**/us/sd/*.json
…

/user/hadoop/openaddresses/**final**/us/wi/*.parquet
/user/hadoop/openaddresses/**final**/us/nd/*.parquet
/user/hadoop/openaddresses/**final**/us/sd/*.parquet
…

- move data from *raw* to *final* directory
- Convert/Explode data structure
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

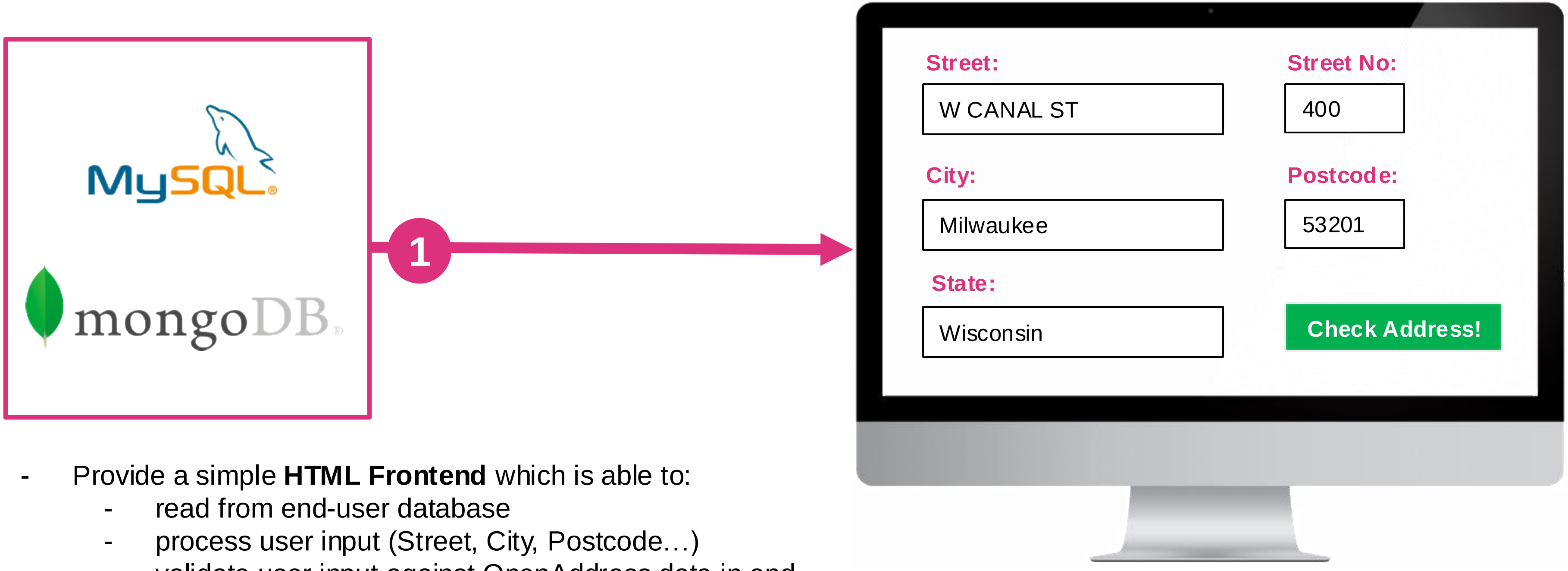
Dataflow: **3. Enhance Data And Save Results**



/user/hadoop/openaddresses/**final**/us/wi/*.parquet
/user/hadoop/openaddresses/**final**/us/nd/*.parquet
/user/hadoop/openaddresses/**final**/us/sd/*.parquet
…

- enhance data (e.g. add missing entries of street no‘s)
- use *Hive*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: **4. Provide Simple Web Interface**



Street: W CANAL ST

Street No: 400

City: Milwaukee

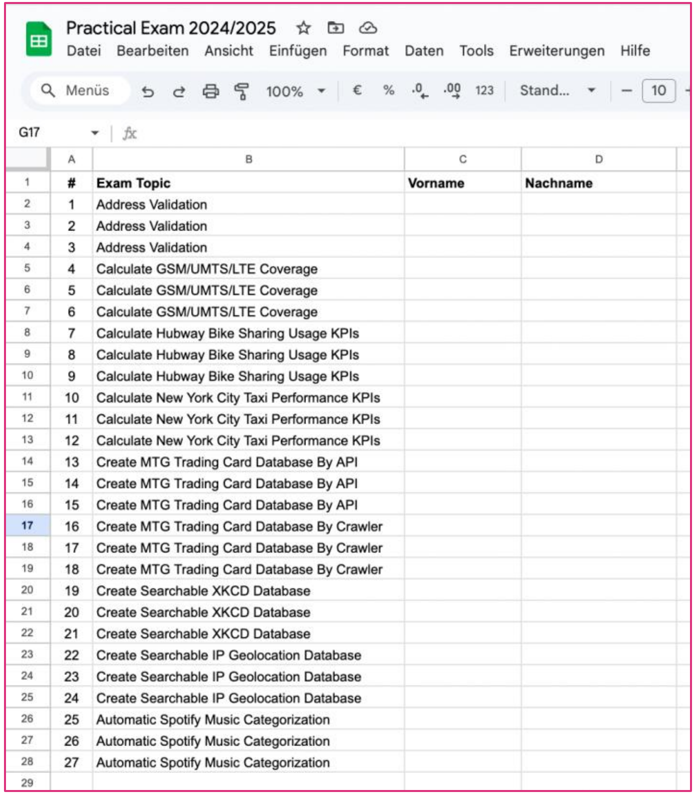
Postcode: 53201

State: Wisconsin

- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Street, City, Postcode…)
  - validate user input against OpenAddress data in end-user database
  - Display result (real or non real address)

Good Luck and have fun!

Please choose an exam topic!



#	Exam Topic	Vorname	Nachname
1	Address Validation		
2	Address Validation		
3	Address Validation		
4	Calculate GSM/UMTS/LTE Coverage		
5	Calculate GSM/UMTS/LTE Coverage		
6	Calculate GSM/UMTS/LTE Coverage		
7	Calculate Hubway Bike Sharing Usage KPIs		
8	Calculate Hubway Bike Sharing Usage KPIs		
9	Calculate Hubway Bike Sharing Usage KPIs		
10	Calculate New York City Taxi Performance KPIs		
11	Calculate New York City Taxi Performance KPIs		
12	Calculate New York City Taxi Performance KPIs		
13	Create MTG Trading Card Database By API		
14	Create MTG Trading Card Database By API		
15	Create MTG Trading Card Database By API		
16	Create MTG Trading Card Database By Crawler		
17	Create MTG Trading Card Database By Crawler		
18	Create MTG Trading Card Database By Crawler		
19	Create Searchable XKCD Database		
20	Create Searchable XKCD Database		
21	Create Searchable XKCD Database		
22	Create Searchable IP Geolocation Database		
23	Create Searchable IP Geolocation Database		
24	Create Searchable IP Geolocation Database		
25	Automatic Spotify Music Categorization		
26	Automatic Spotify Music Categorization		
27	Automatic Spotify Music Categorization		

https://tinyurl.com/yahbnfmm

Stop Your VM Instances

```
gcloud compute instances stop big-data
```