IDS 706: Data Engineering Systems

Dr. Zhongyuan (Annie) Yu

Duke

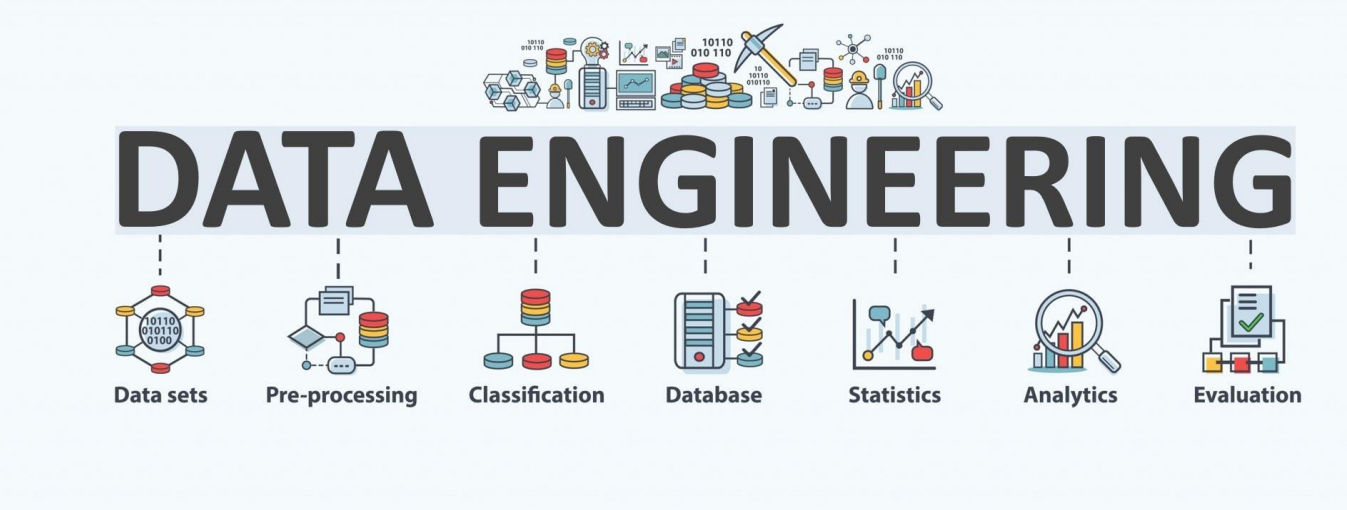
Course Basics

Instructor: Dr. Zhongyuan Yu

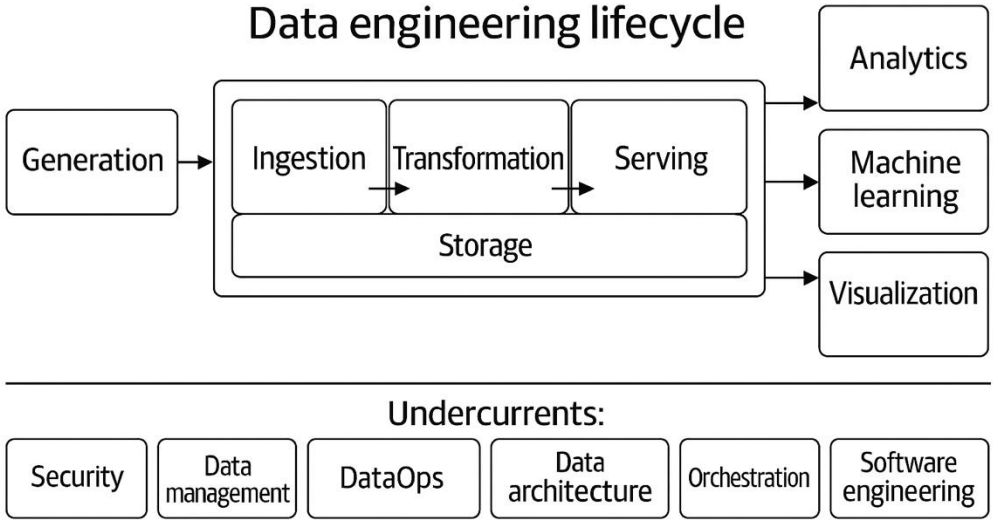
Contact Info: Zhongyuan.yu@duke.edu

Office Hours: Friday 11:45 am -12:45 pm Gross Hall 227 or Zoom

What is Data Engineering?



https://futurense.com/uni-blog/what-is-data-engineering



- Designing and managing the infrastructure that powers data generation, collection, and distribution.
- Building scalable pipelines to transform and move data across systems.
- Making data accessible, reliable, and analysis-ready for analysts and data scientists.

Core Engineering Mindset

- **Problem-solving focus**: Engineers approach challenges as opportunities to create effective solutions and break complex challenges into smaller, manageable pieces.
- **Systems thinking**: They understand how individual parts connect and influence the whole system, and understand dependencies and trade-offs when making decisions.
- **Iterative and continuous improvement**: Solutions evolve through prototyping, testing, and refining over time.
- **Creative but constraint-aware**: Innovation happens within practical limits like cost, time, and regulations.
- **Data-driven decision making**: Decisions are guided by metrics, evidence, and analysis, not assumptions.

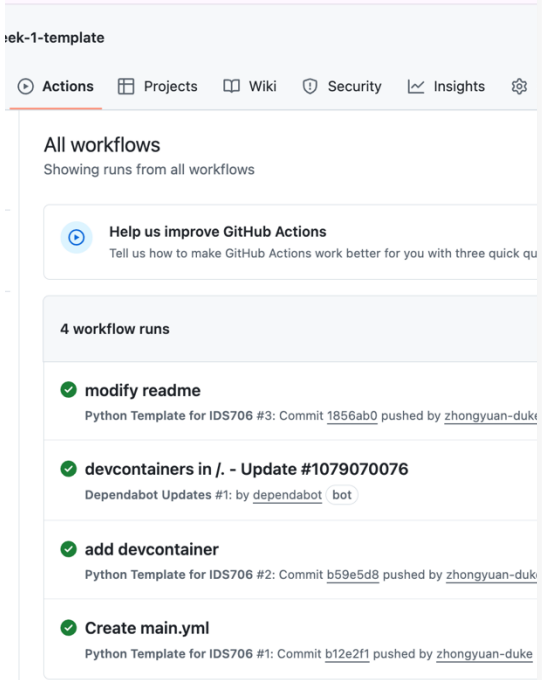
What this course is about?

- Understand the Role of a Data Engineer
- Apply software engineering practices like version control, testing, and CI/CD
- Work with data using Python, SQL, and cloud platforms
- Build and manage scalable data pipelines using modern tools
- Understand big data processing and real-time streaming

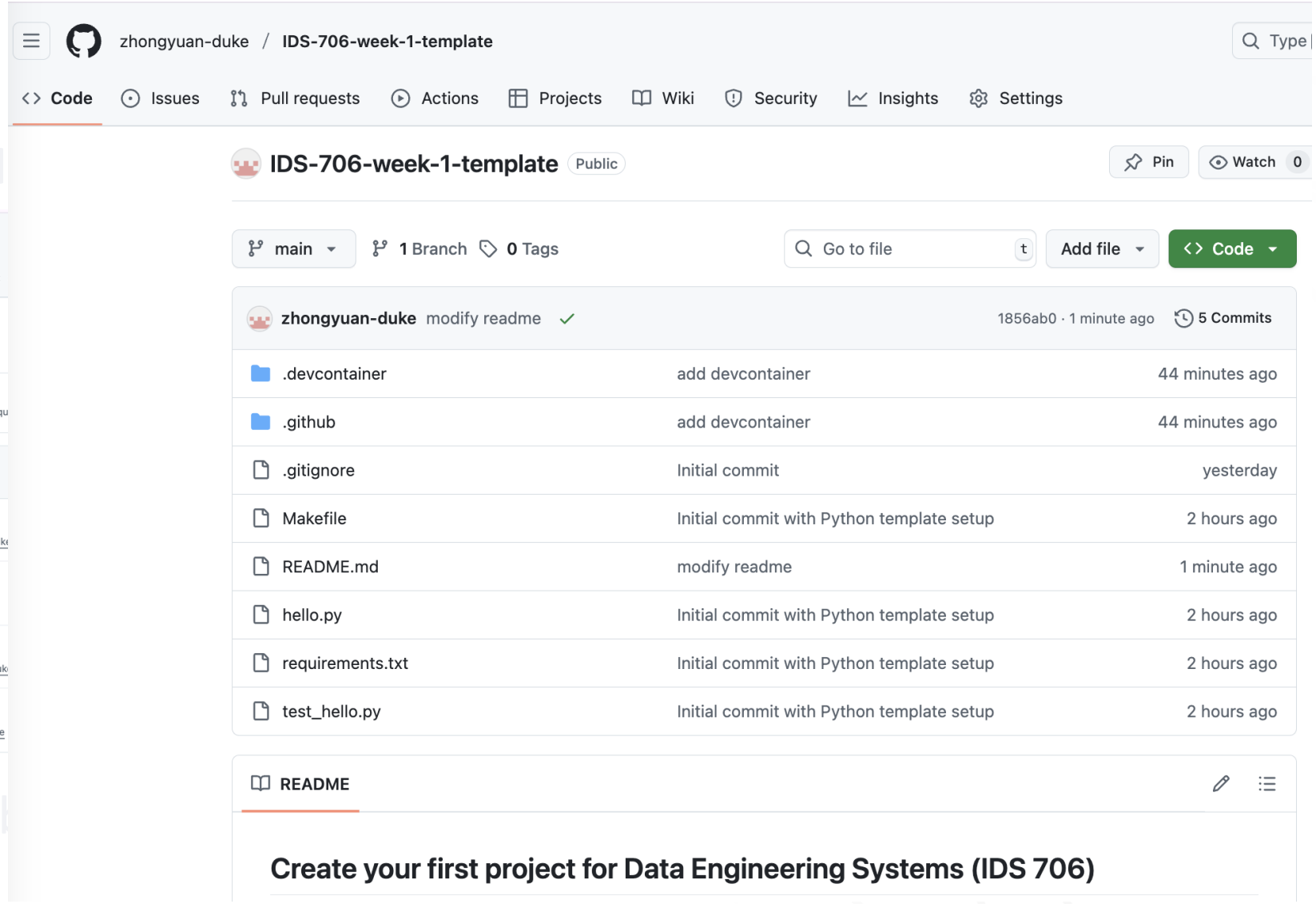
Prerequisite(s): None. But programming experience is helpful.

Cloud-based Containerized Python Template

- Create a GitH

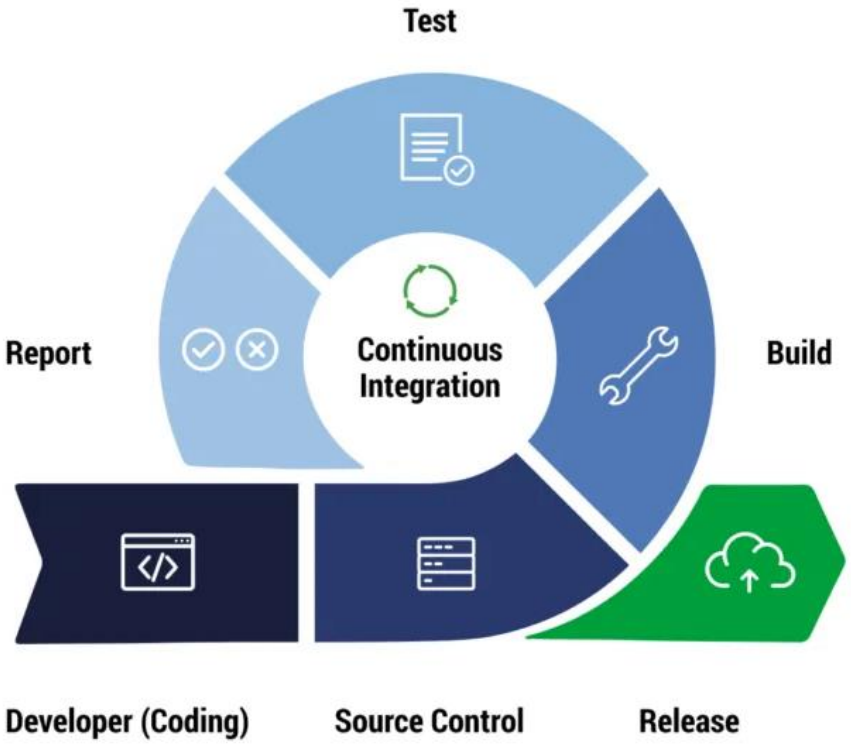


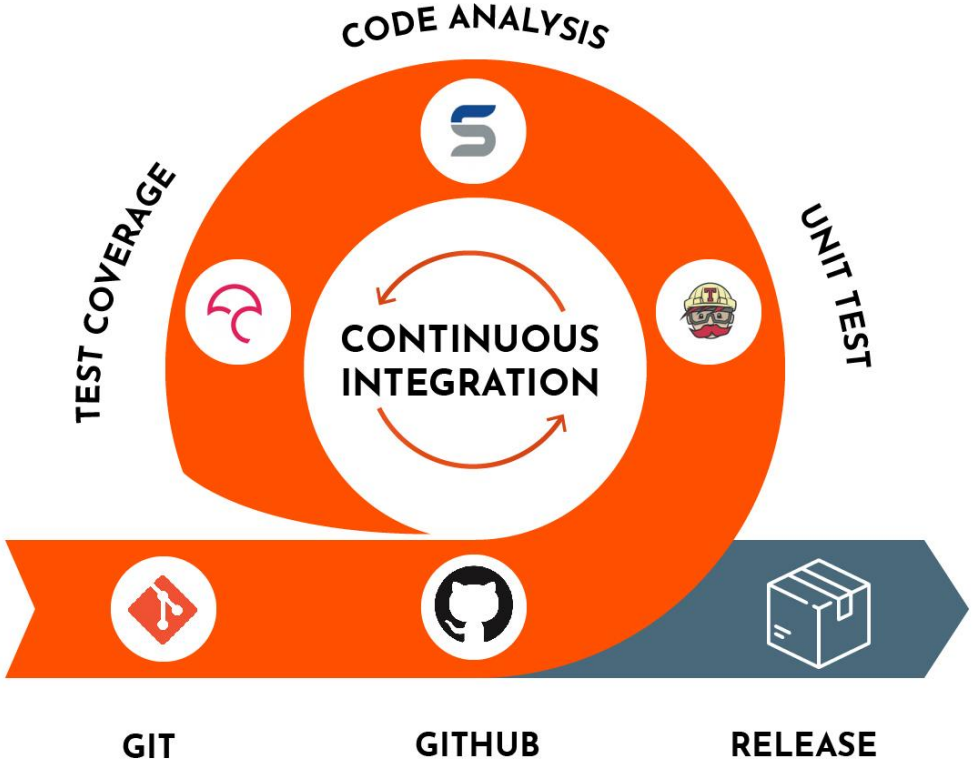
- Enable GitHub



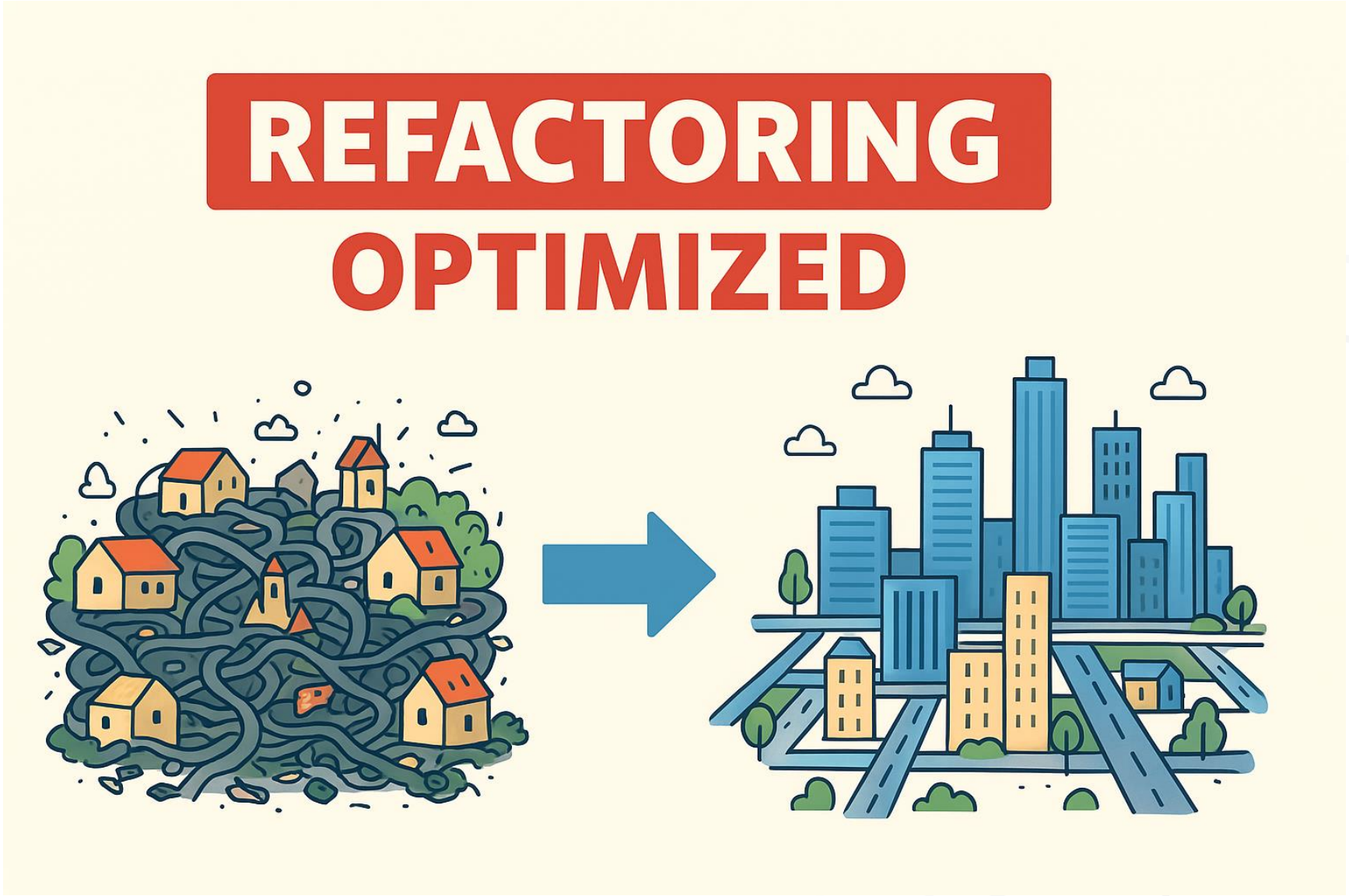
AI Generated

Continuous Integration





https://www.tierpoint.com/blog/benefits-of-continuous-integration/
https://devonblog.com/continuous-delivery/continuous-integration-best-practices/



AI Generated

SQL Cheat Sheet

Created by @JessRamosData | Big Data Energy⚡

The Basics

select	group by	=	<=	not
from	having	!=	>=	in
where	order by	<	True	between
	limit	>	False	

Aggregate Fun

min()	count()	
max()	median()	
avg()	mode()	
	stddev()	

String Manipulation

concat()	upper()	wildcards (%)
replace()	lower()	like / ilike
reverse()	len()	
trim()	str()	

Date Manipulat

day()	date_a
month()	datedi
year()	date_t
getdate()	date_f

Joins

inner join	anti join
left join	joining on multiple keys
outer join	joining on a condition
self join	

Cleaning & Trar

cast()	iif()
coalesce()	listagg()
case when	except
ifnull()	

Window Functions

sum()	row_number()	
count()	rank()	over(
avg()	denserank()	partition by ...
max()	lead()	order by ...)
min()	lag()	

Advanced

- CTEs
- Subqueries (correlated vs
- UDFs
- Data Modeling

Follow Jess Ramos for Data, SQL, & Tech Career tips

@JessRamosData

Big Data Energy

SQL Basics Cheat Sheet

LearnSQL.com

SQL

SQL, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

COUNTRY

id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY

id	name	country_id	population	rating
1	Paris	1	22430[illegible]	[illegible]
2	Berlin	2	34600	
...	...	...	...	

QUERYING SINGLE TABLE

Fetch all columns from the country table:

```
SELECT *
FROM country;
```

Fetch id and name columns from the city t

```
SELECT id, name
FROM city;
```

Fetch city names sorted by the rating colun in the default ASCending order:

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

Fetch city names sorted by the rating colun in the DESCending order:

```
SELECT name
FROM city
ORDER BY rating DESC;
```

ALIASES

COLUMNS

```
SELECT name AS city_name
FROM city;
```

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

Try out the interactive S

FILTERING THE OUTPUT

COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
  AND name != 'Madrid';
```

QUERYING MULTIPLE TABLES

INNER JOIN

`JOIN` (or explicitly `INNER JOIN`) returns rows that have matching values in both tables.

```
SELECT city.name, country.name
FROM city
[INNER] JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2
...	...	...

COUNTRY

id	name
1	France
2	Germany
...	...

FULL JOIN

`FULL JOIN` (or explicitly `FULL OUTER JOIN`) returns all rows from both tables – if there's no matching row in the second table, `NULL`s are returned.

```
SELECT city.name, country.name
FROM city
FULL [OUTER] JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2
3	Warsaw	4

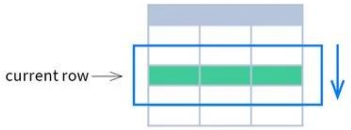
COUNTRY

id	name
1	France
2	Germany
NULL	NULL

SQL Window Functions Cheat Sheet

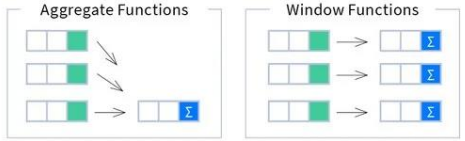
WINDOW FUNCTIONS

compute their result based on a sliding window frame, a set of rows that are somehow related to the current row.



AGGREGATE FUNCTIONS VS. WINDOW FUNCTIONS

unlike aggregate functions, window functions do not collapse rows.



PARTITION BY

divides rows into multiple groups, called **partitions**, to which the window function is applied.

month	city	sold
1	Rome	200
2	Paris	500
1	London	100
1	Paris	300
2	Rome	300
2	London	400
3	Rome	400

PARTITION BY city

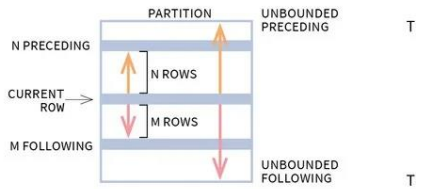
month	city	sold	sum
1	Paris	300	800
2	Paris	500	800
1	Rome	200	900
2	Rome	300	900
3	Rome	400	900
1	London	100	500
2	London	400	500

Default Partition: with no `PARTITION BY` clause, the entire result set is the partition.

WINDOW FRAME

is a set of rows that are somehow related to the current row. Th

ROWS | RANGE | GROUPS BETWEEN



ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING

city	sold	month
Paris	300	1
Rome	200	1

RANGE BETW AND 1

city
Paris
Rome

SYNTAX

```
SELECT city, month,
    sum(sold) OVER (
        PARTITION BY city
        ORDER BY month
        RANGE UNBOUNDED PRECEDING) total
FROM sales;
```

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Named Window Definition

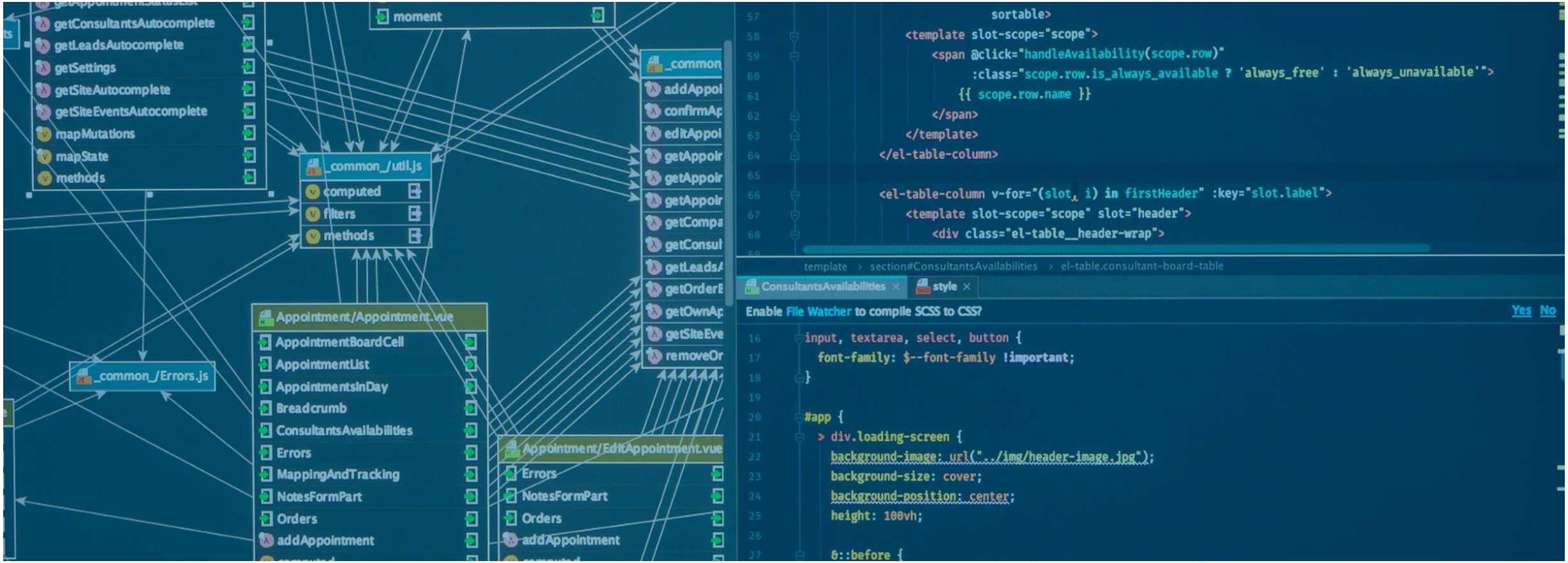
```
SELECT country, city,
    rank() OVER country_sold_avg
FROM sales
WHERE month BETWEEN 1 AND 6
GROUP BY country, city
HAVING sum(sold) > 10000
WINDOW country_sold_avg AS (
    PARTITION BY country
    ORDER BY avg(sold) DESC)
```

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER <window_name>
FROM <table_name>
WHERE <...>
GROUP BY <...>
HAVING <...>
WINDOW <window_name> AS (
    PARTITION BY <...>
    ORDER BY <...>
```

https://learnsql.com/blog/sql-basics-cheat-sheet/
https://jessramosdata.substack.com/p/sql-cheat-sheet-from-basic-to-advanced
https://www.reddit.com/r/SQL/comments/1d705u5/window_functions_cheat_sheet_from_learnsqlcom/

Duke

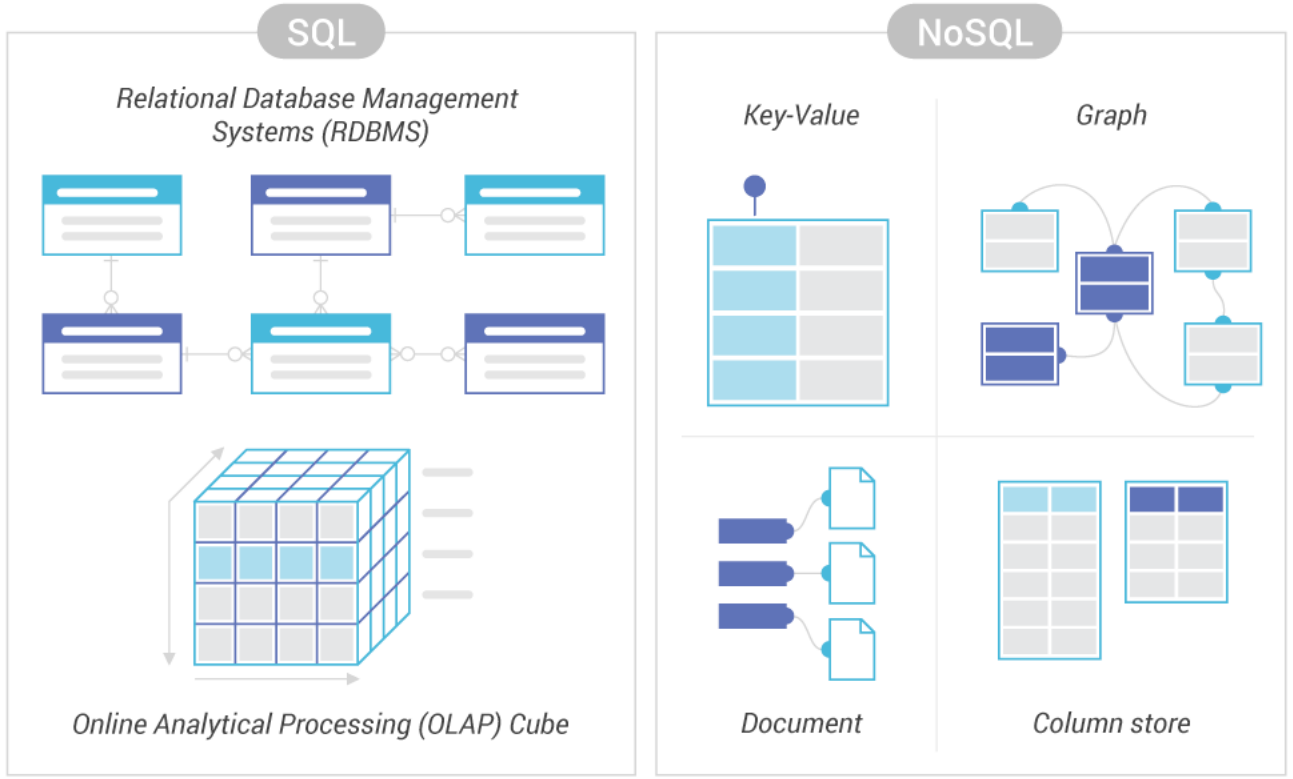
Relational Database Management System Design

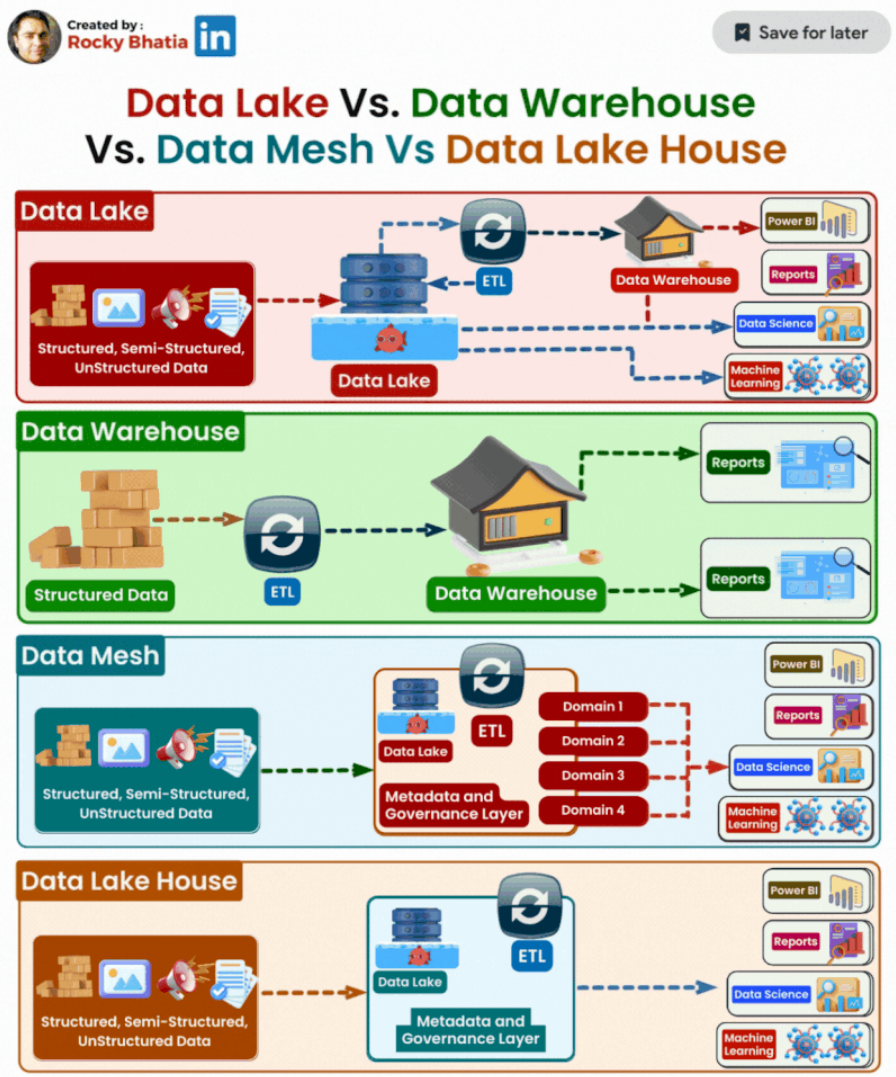


https://www.ucsc-extension.edu/courses/relational-database-design-and-sql-programming/

Duke

SQL vs NoSQL, Data Lakes & Warehouse

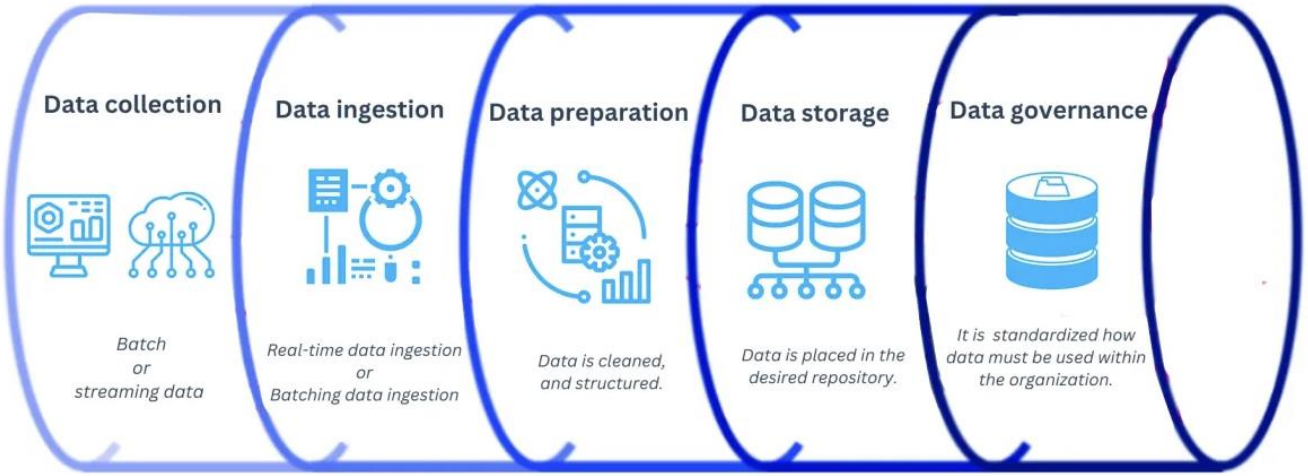


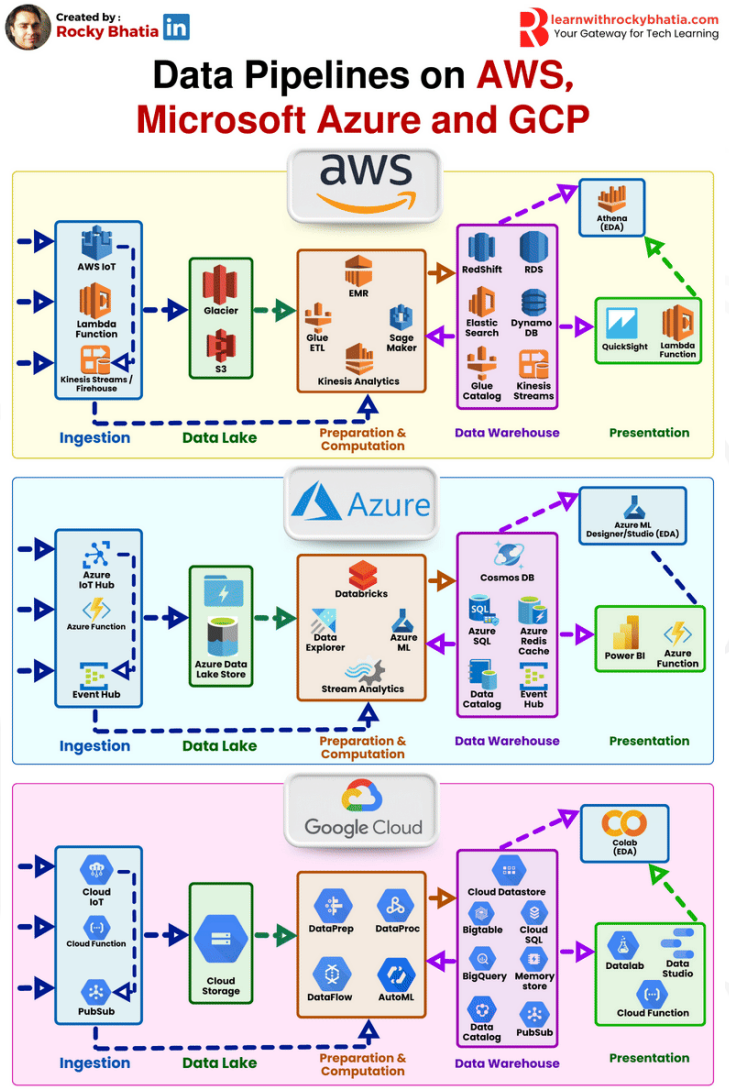


https://www.scylladb.com/learn/nosql/nosql-vs-sql/
https://www.linkedin.com/posts/rocky-bhatia-a4801010_%F0%9D%90%83%F0%9D%90%9A%F0%9D%90%AD%F0%9D%90%9A-%F0%9D%90%8B%F0%9D%90%9A%F0%9D%90%A4%F0%9D%90%9E-%F0%9D%90%AF%F0%9D%90%AC-%F0%9D%90%83%F0%9D%90%9A%F0%9D%90%AD%F0%9D%90%9A-%F0%9D%90%96%F0%9D%90%9A%F0%9D%90%AB%F0%9D%90%9E-activity-7274385298258575360-m_p6/

Duke

Data Pipelines and Cloud

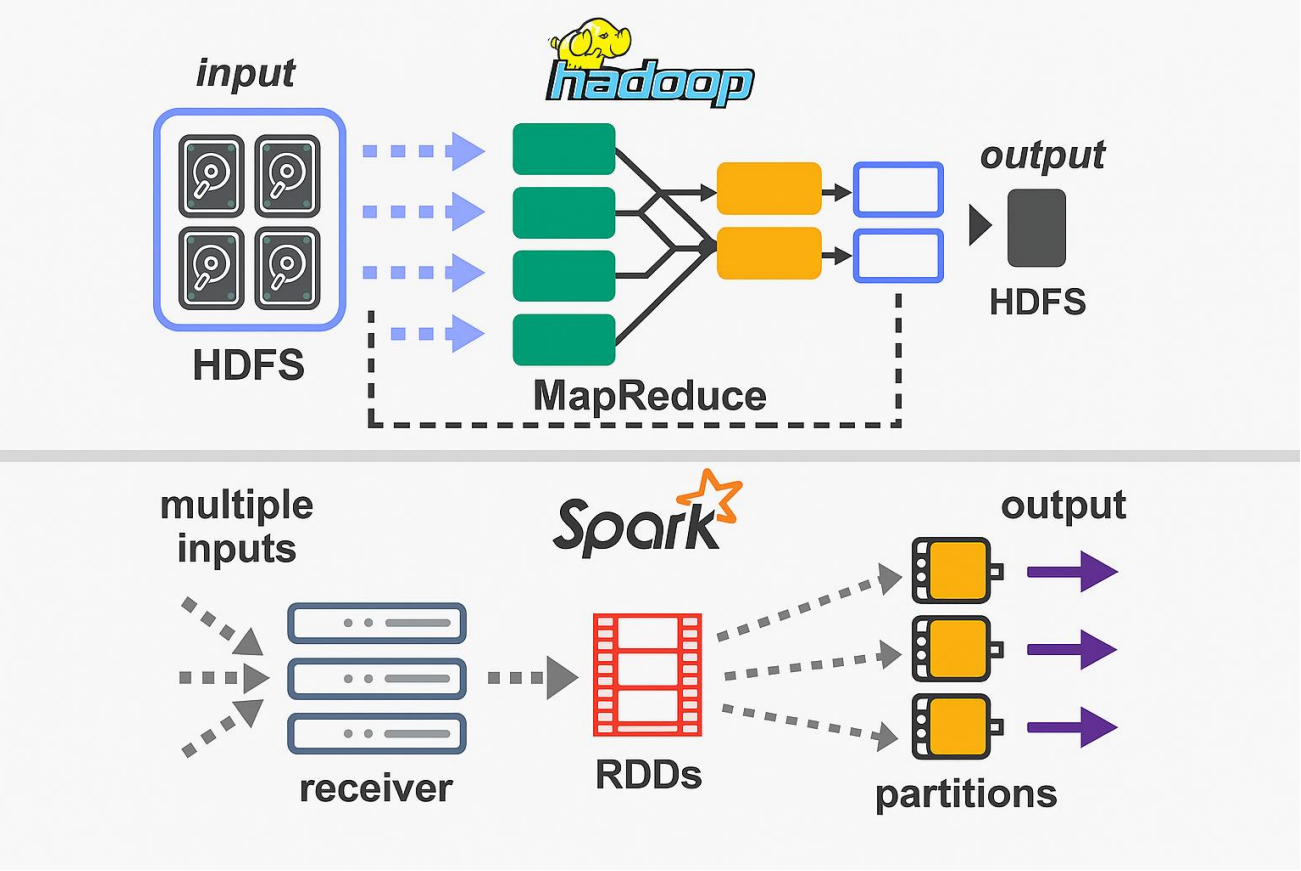




https://www.secoda.co/blog/10-best-practices-to-build-data-pipelines
https://rockybhatia.substack.com/p/data-pipelines-in-the-cloud-azure

Duke

Big Data Processing: Spark



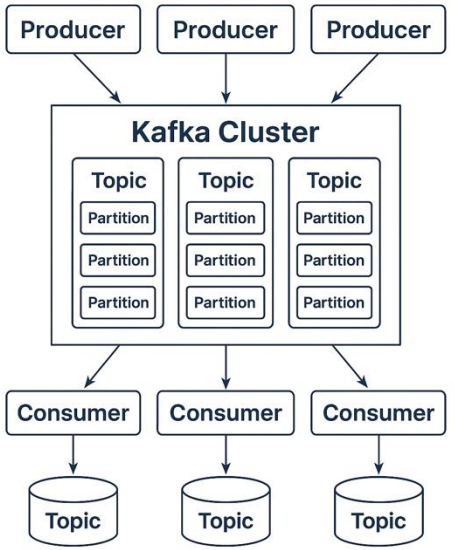
https://phoenixnap.com/kb/hadoop-vs-spark

comparison_BMw.py > ...

```
92
93  # Function to benchmark PySpark
94  def benchmark_pyspark(file_path):
95      results = {}
96      spark = SparkSession.builder.appName("BMWAnalysis").getOrCreate()
97
98      start = time.time()
99      df = spark.read.csv(file_path, header=True, inferSchema=True)
100     results['load'] = time.time() - start
101
102     start = time.time()
103     filtered = df.filter(col('Sales_Classification') == 'High')
104     filtered.count()
105     results['filter'] = time.time() - start
106
107     start = time.time()
108     agg = df.select(mean('Price_USD')).collect()
109     results['agg'] = time.time() - start
110
111     start = time.time()
112     group = df.groupBy('Region').agg(_sum('Sales_Volume')).collect()
113     results['group'] = time.time() - start
114
115     start = time.time()
116     df.write.mode("overwrite").parquet("spark_output.parquet")
117     results['write'] = time.time() - start
118
119     spark.stop()
```

Duke

Real-time Streaming



BARCLAYS

Barclays utilizes Kafka for streaming and analytical information.

airbnb

Used in our event pipeline, exception tracking & more to come.

APACHE KAFKA

*More than **80% of all Fortune 100 companies** trust, and use Kafka.*

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

box

At Box, Kafka is used for the production analytics pipeline & real time monitoring infrastructure. We are planning to use Kafka f
produc

NETFLIX

Real-time monitoring and event-processing pipeline.

10 OUT OF 10 *MANUFACTURING*

7 OUT OF 10 *BANKS*

10 OUT OF 10 *INSURANCE*

8 OUT OF 10 *TELECOM*

LinkedIn

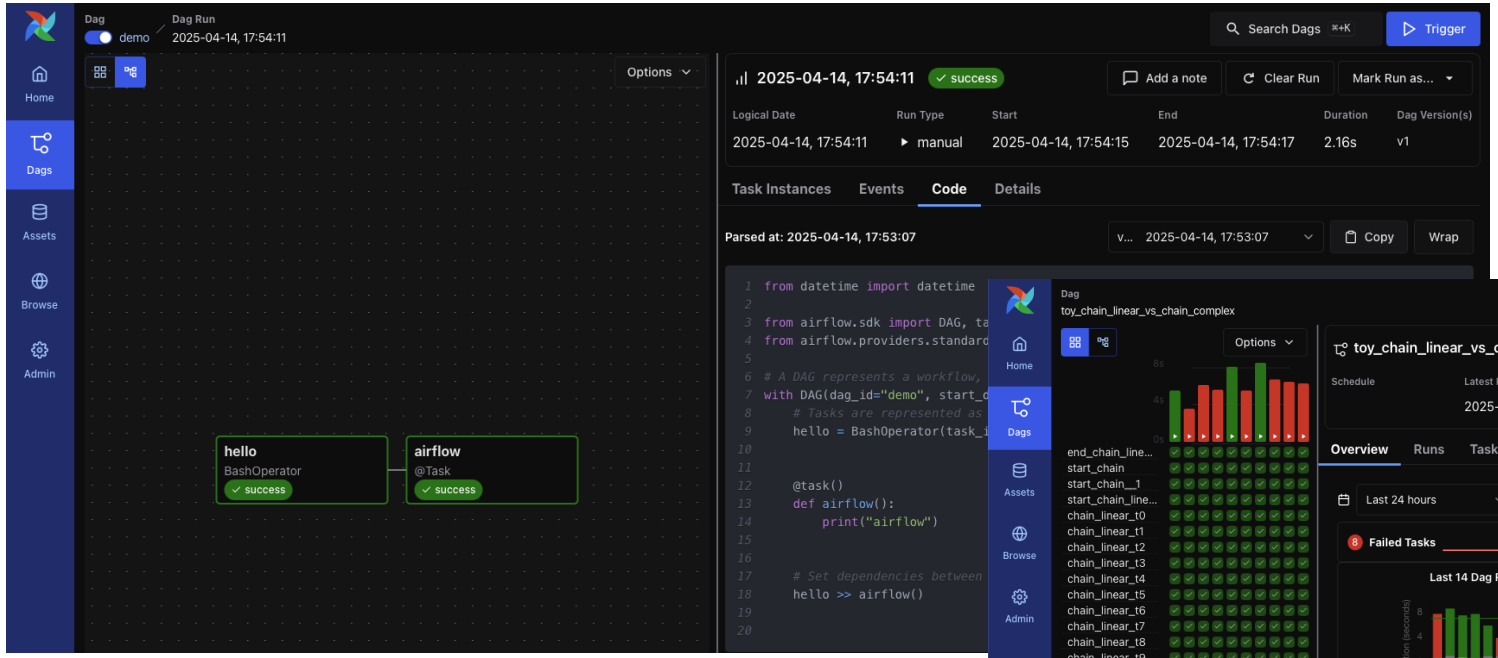
Apache Kafka is used at LinkedIn for activity stream data and operational metrics. This powers various products like LinkedIn Newsfeed, LinkedIn Today in addition to our offline analytics systems like Hadoop.

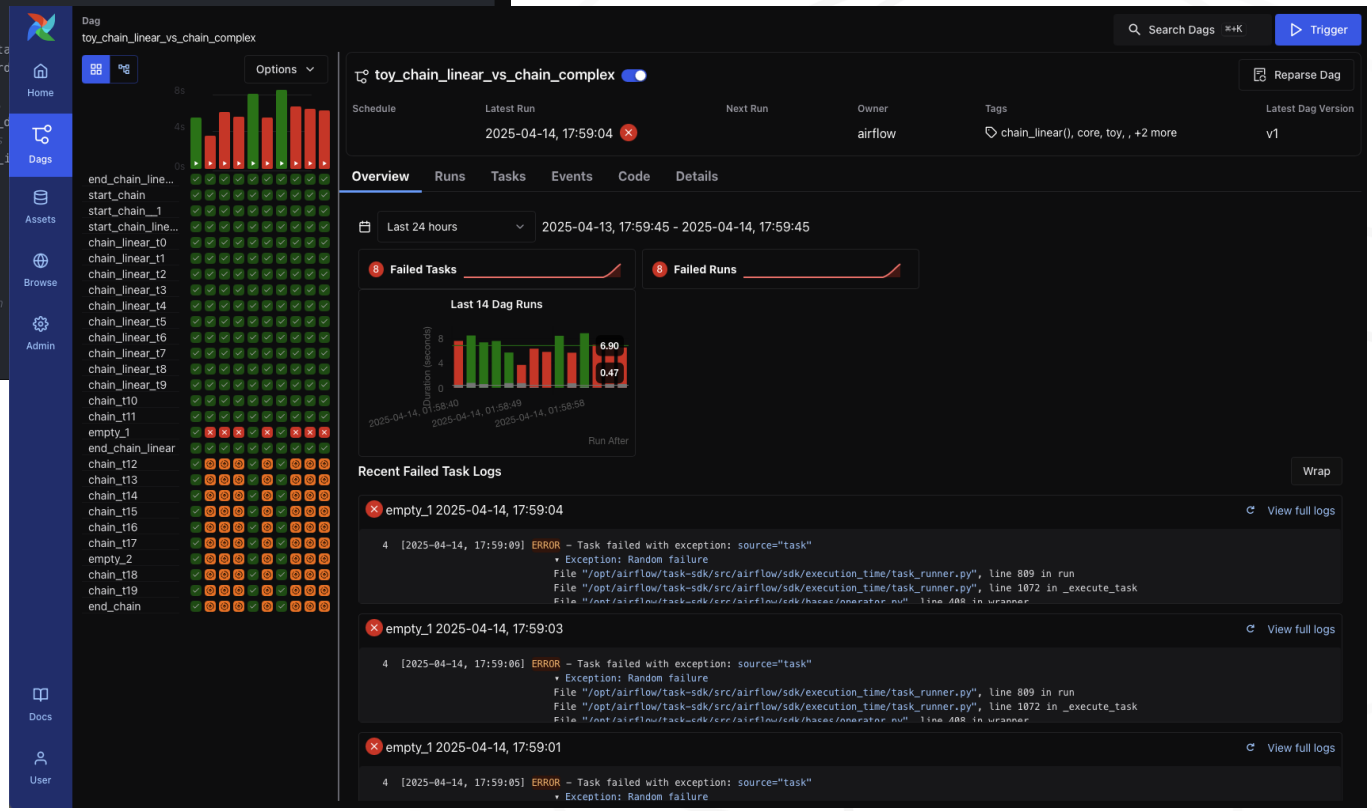
ByteDance 字节跳动

ByteDance uses Kafka as a data hub to collect events and logs for a variety of services such as online model training, stream data processing and real-time data analytics.

Duke

Data Orchestration with Apache Airflow





https://airflow.apache.org/docs/apache-airflow/stable/index.html

Apache Airflow

Grading

- Grades will be based on:
  - **Mini Assignments (45%)**
  - **Major Assignments (30%)**
  - **Final Project (25%)**
  - **Participation and Bonus points (+)**

- Grades will be posted on Canvas

Late Policy and Assignment Policy

- All mini-assignments are due at the same time each week if any, so that you can plan your schedule

- Late assignments
  - **Less than 24 hours: 5% deduction**
  - **1-3 days: 10% deduction**
  - **4-7 days, 30% deduction**
  - **Greater than a week, you will receive 0 for the corresponding assignment if without a written explanation in advance regarding a situation of unavoidable emergency**

- The **lowest mini-assignment grade will be dropped** for the final homework calculation.

Data Engineering vs Data Science vs Data Analytics

Data Engineers: Build the highways for data to travel—pipelines, storage systems, and infrastructure.

Data Scientists: Use advanced analytics and ML to predict trends and answer complex questions.

Data Analysts: Dig into the datasets provided by engineers to extract actionable insights.

Some Trends in Modern Data Engineering

- **Pay-as-you-go cloud infrastructure**
  Cloud services like Snowflake and BigQuery enable scalable, low-cost data analysis without upfront investment.
- **Shift from ETL to ELT?**
  Modern warehouses handle transformations post-load, leveraging SQL and cloud compute power.
- **Templated SQL & YAML pipelines**
  Modular, reusable SQL with YAML configs for scalable pipeline development.
- **Data observability (DataOps)**
  Monitoring + alerting tools for data similar to CI/CD in software.
- **Decentralized data ownership**
  Teams manage their own data products, reducing reliance on centralized data teams.

Questions?

Thank you