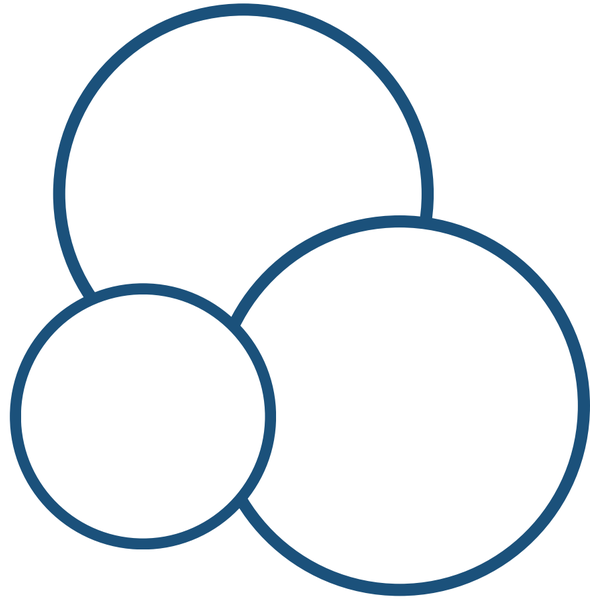
DATATECH LABS.

Data Engineering Course.

Week 5: Data Automation, Orchestration and Visualization

Outline: Week 5

- Introduction to Data Pipeline Automation and Orchestration
- Benefits and Challenges of Pipeline Automation
- Apache Airflow as Automation and Orchestration Tool
- Data Visualization and Reporting
- Data Visualization Principles and Reporting Tools

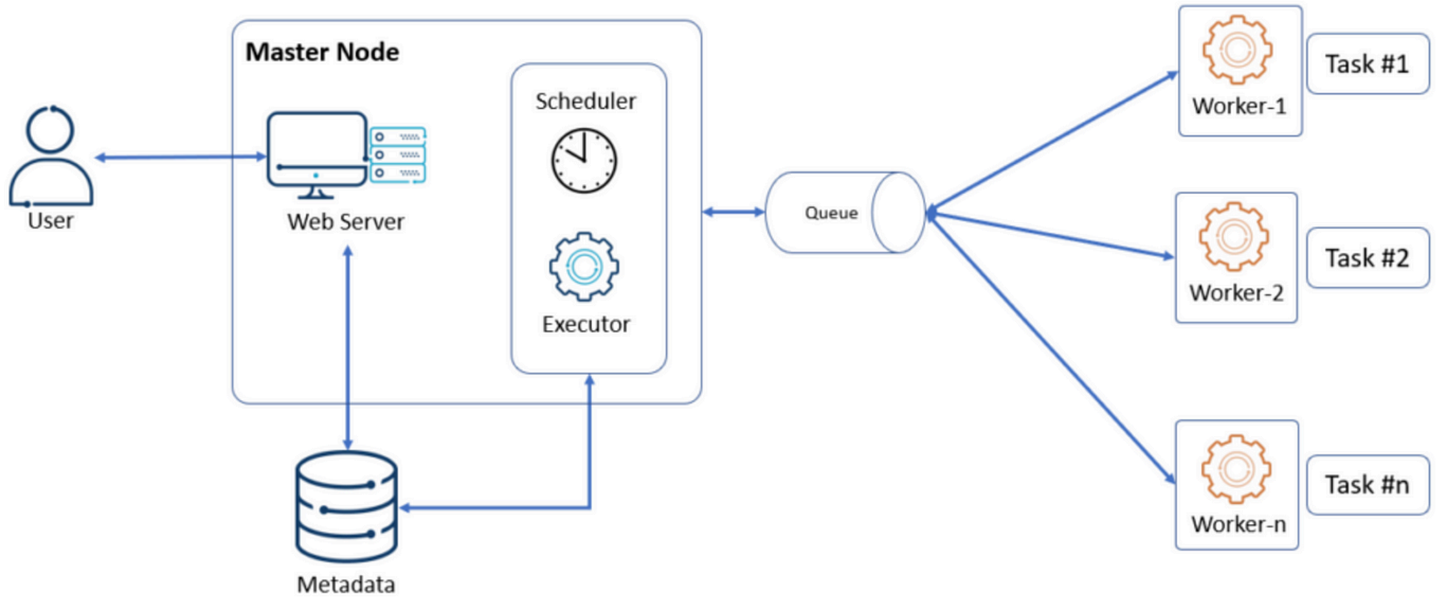


Data Pipelines Automation and Orchestration

Introduction to Data Pipeline Automation and Orchestration

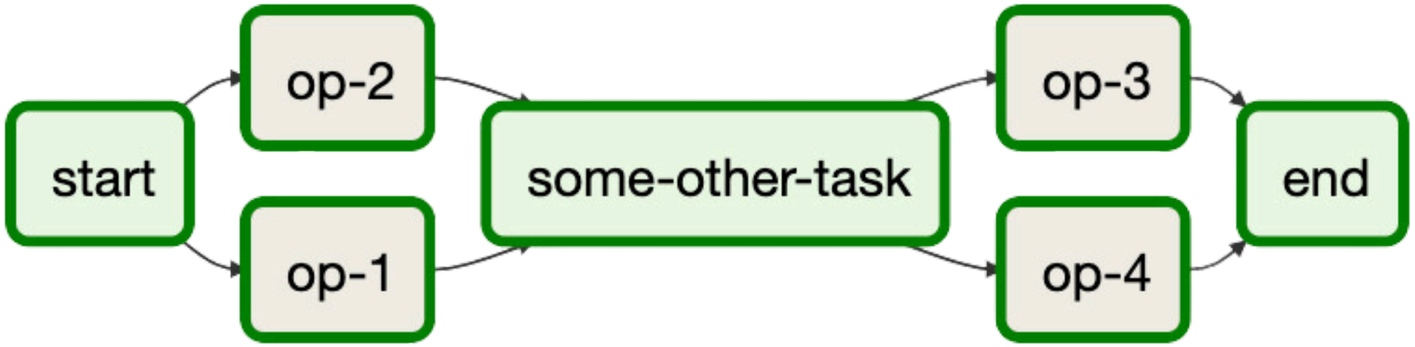
Introduction to Automation and Orchestration

- **Definition**: The process of automating and managing complex data workflows
- **Automation** is programming a task to be executed without the need for human intervention.
- **Orchestration** is the configuration of multiple tasks (some may be automated) into one complete end-to-end process or job.
- Faster and More Effective Insights for modern data architectures



Key Concepts in Pipeline Automation

1. **Workflow**: A series of tasks that process data
2. **Task**: An individual unit of work
3. **Dependency**: The relationship between tasks
4. **Scheduling**: Timing and frequency of workflow execution
5. **Monitoring**: Tracking the status and performance of workflows

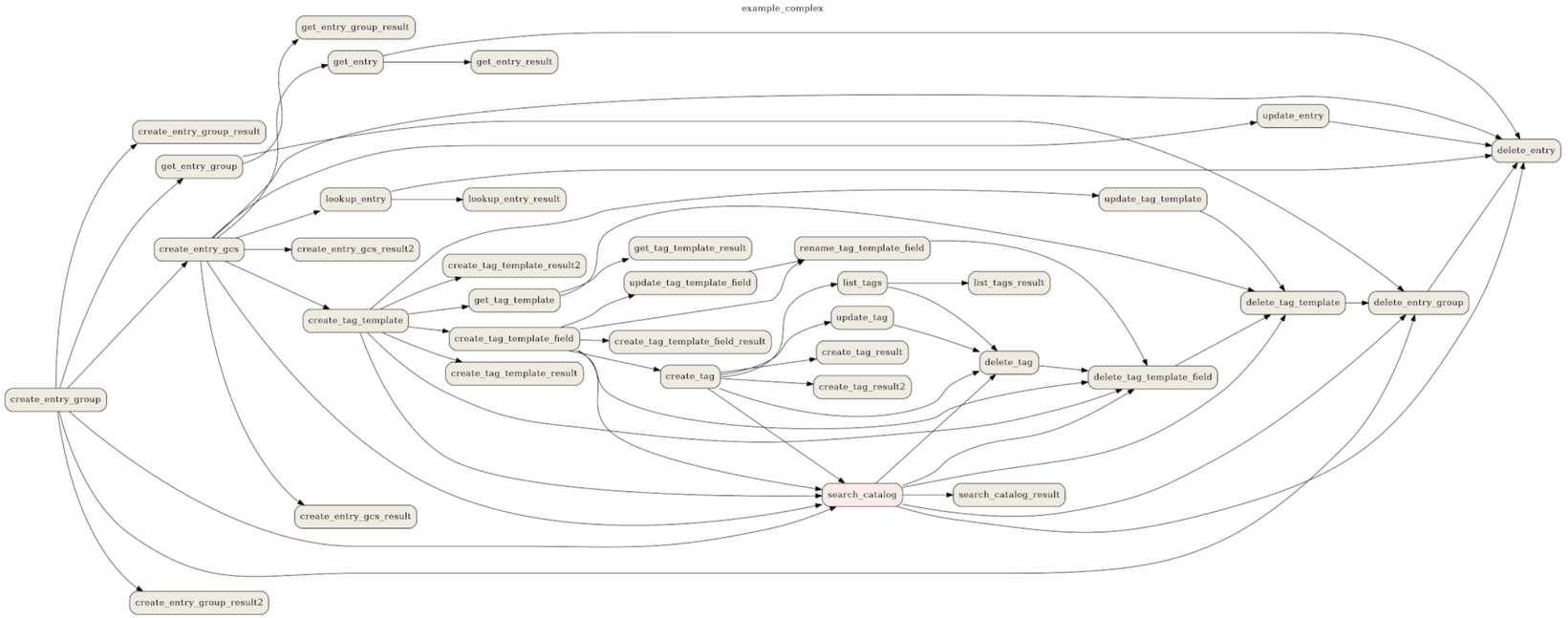


Benefits of Pipeline Automation

- Improved efficiency and reliability
- Reduced manual errors
- Better resource utilization
- Enhanced scalability
- Easier maintenance and troubleshooting

Common Challenges in Data Pipeline Management

- Complex dependencies between tasks
- Error handling and recovery
- Data quality assurance
- Performance optimization
- Versioning and reproducibility



Pipeline Orchestration Tools Landscape

- Apache Airflow
- Luigi
- Airbyte
- Dagster
- Argo Workflows

Apache Airflow

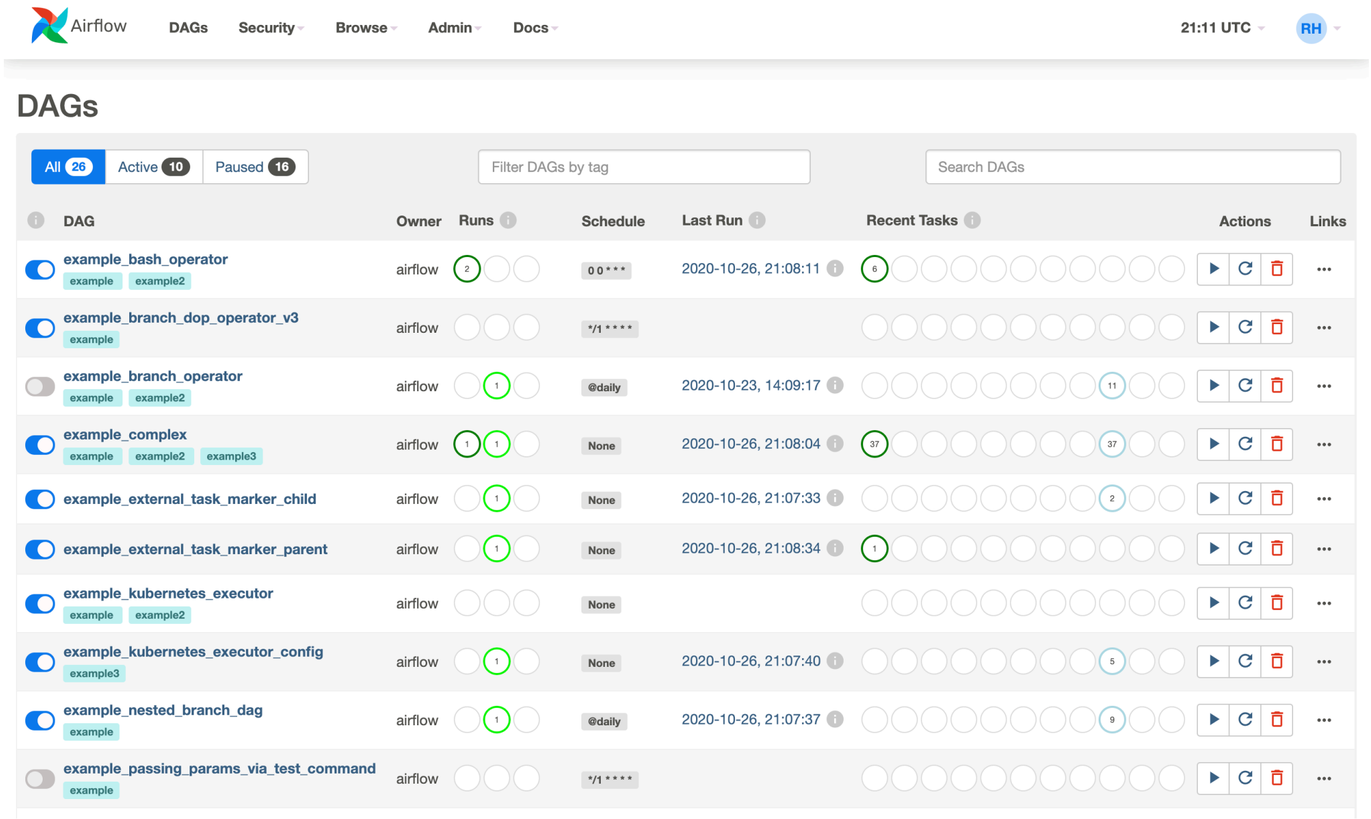
Airbyte

dagster

Apache Airflow as Automation Tool

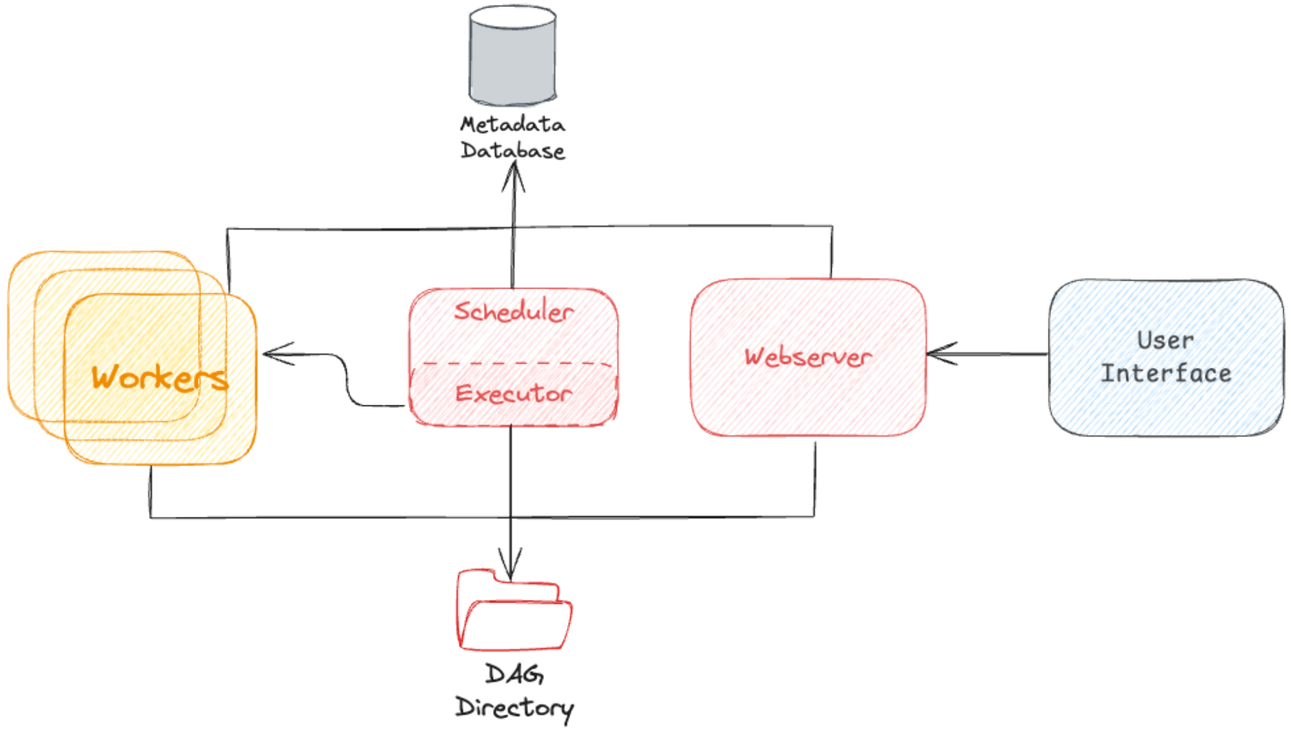
Introduction to Apache Airflow

- **Definition**: An open-source platform to programmatically author, schedule, and monitor workflows
- Created by Airbnb in 2014, now an Apache Software Foundation project



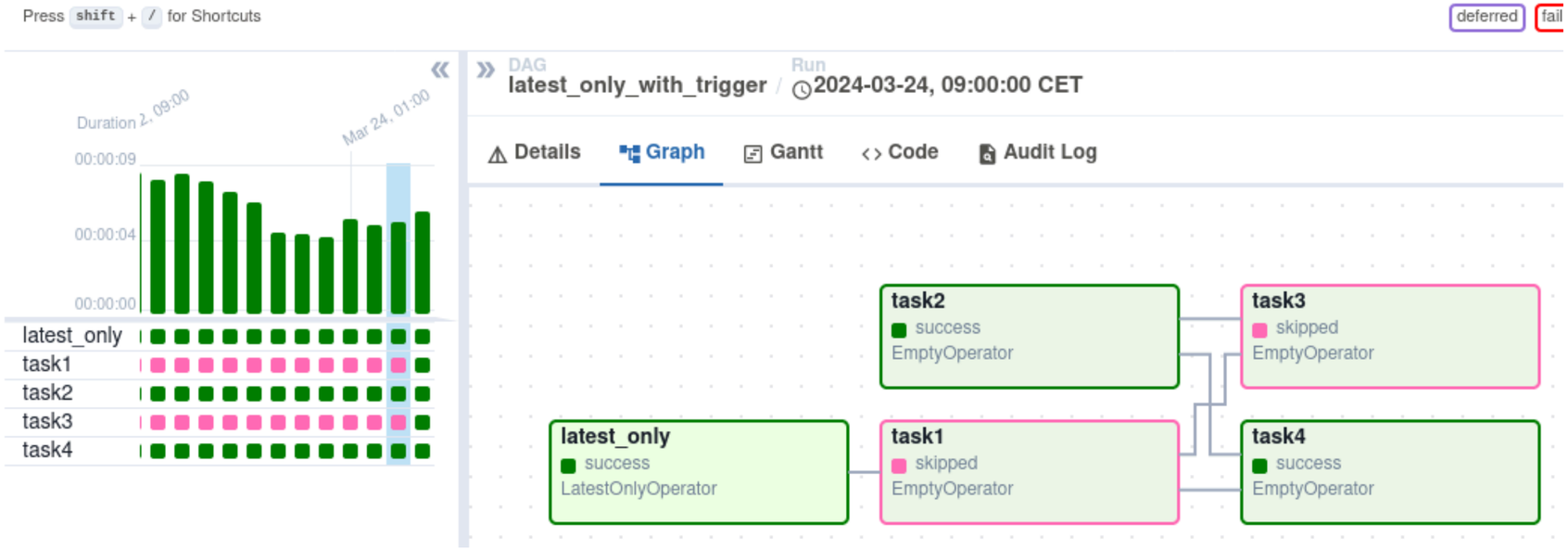
Airflow Architecture

1. **Webserver**: Provides the user interface
2. **Scheduler**: Monitors and triggers tasks
3. **Workers**: Execute the tasks
4. **Metadata Database**: Stores DAGs, task instances, and configurations
5. **DAG Files**: Python files defining the workflows



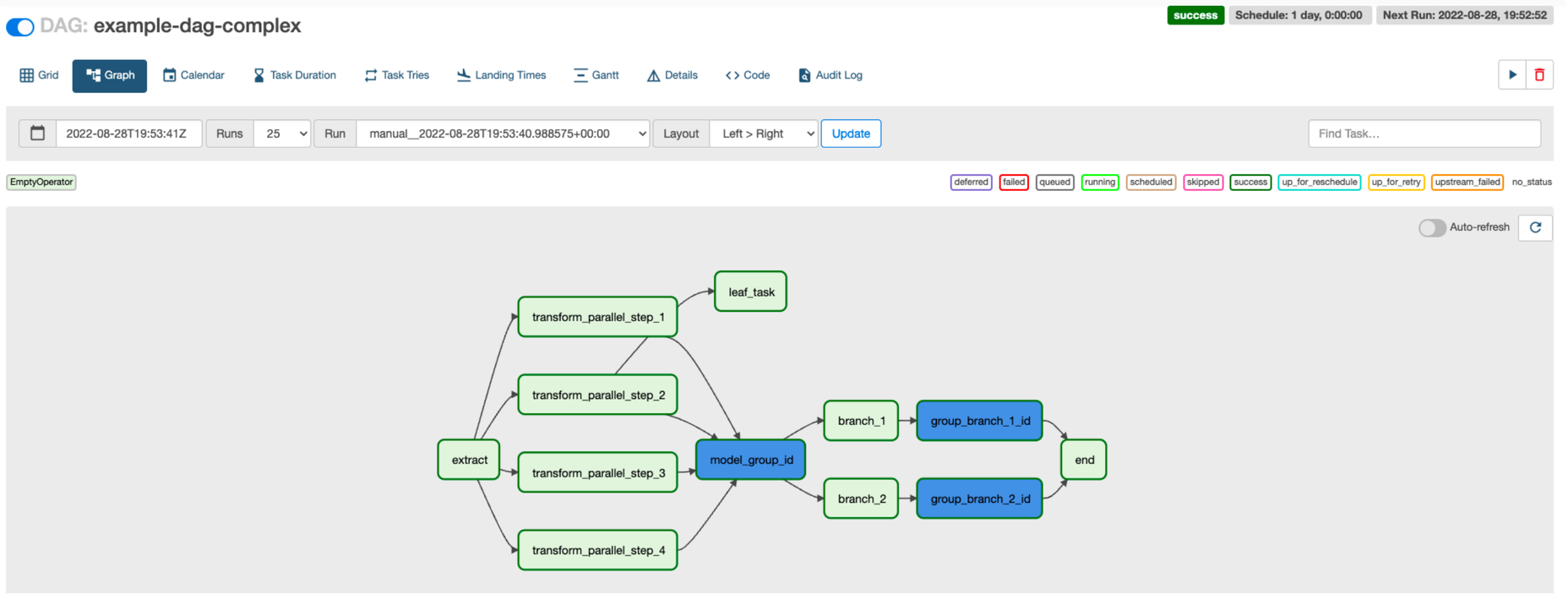
Key Concepts in Airflow

- DAG (Directed Acyclic Graph): Represents a workflow
- Operator: Defines a single task in the workflow
- Task: An instance of an operator
- Task Instance: A specific run of a task
- Execution Date: The logical date and time of a DAG run



Airflow DAGs

- Definition: A collection of tasks with dependencies
- Properties:
  - DAG ID
  - Description
  - Start date
  - Schedule interval
  - Default arguments



Airflow Operators

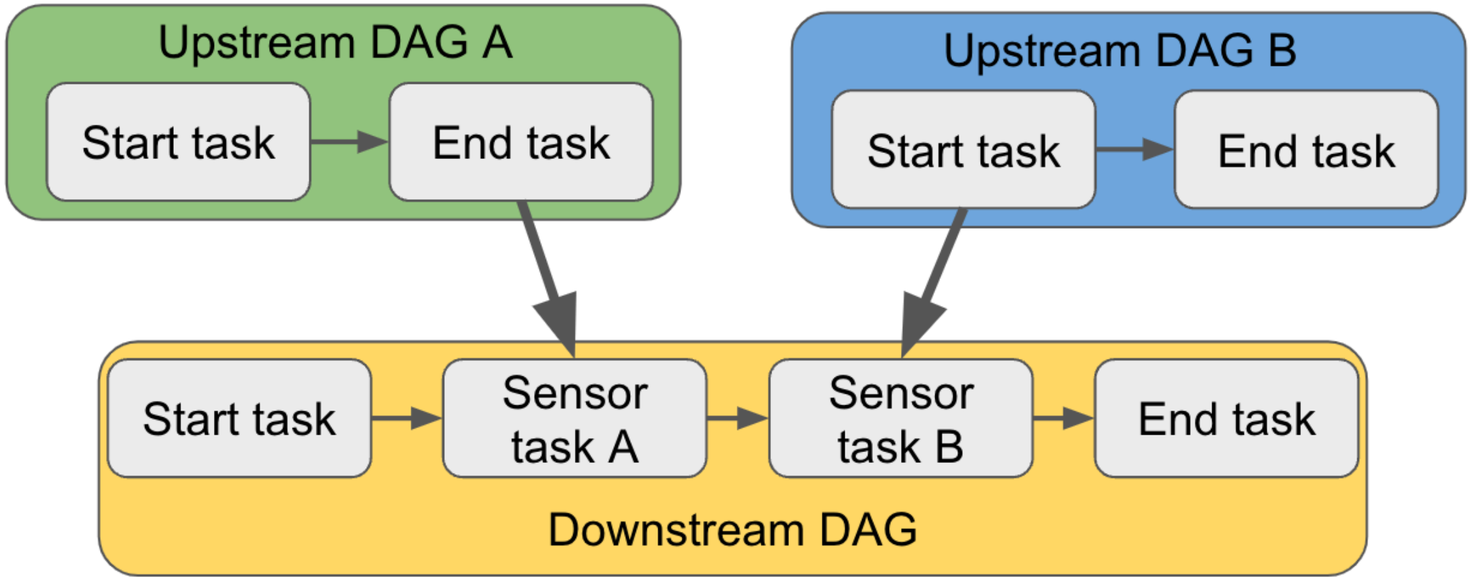
- BashOperator: Executes bash commands
- PythonOperator: Calls Python functions
- SQLOperator: Executes SQL queries
- EmailOperator: Sends emails
- Custom operators: Extend functionality for specific use cases

```python
from airflow.operators.bash_operator import BashOperator

generate_data = BashOperator(
    task_id='generate_data',
    bash_command='dd if=/dev/random of=/path/to/newfile.txt bs=1 count=10',
    dag=dag
)

```

```python
from datetime import datetime
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator

with DAG('my_dag',
         start_date=datetime(2016, 1, 1)
) as dag:
    op = DummyOperator('op')

op.dag is dag # True

```

Task Dependencies in Airflow

- Methods to set dependencies:
  - **set_upstream()** and **set_downstream()**
  - and >> operators
- Branching: Conditional execution of tasks
- Trigger rules: Specify when a task should be executed



Airflow Scheduling

- Cron expressions
- Timedelta objects
- External triggers
- Backfilling: Running DAGs for past dates

```
# ┌─────────────── minute (0 - 59)
# │ ┌─────────────── hour (0 - 23)
# │ │ ┌─────────────── day of the month (1 - 31)
# │ │ │ ┌─────────────── month (1 - 12)
# │ │ │ │ ┌─────────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                       7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * command to execute
```

```python
from datetime import datetime
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator

with DAG('my_dag',
         start_date=datetime(2016, 1, 1)
) as dag:
    op = DummyOperator('op')

op.dag is dag # True
```

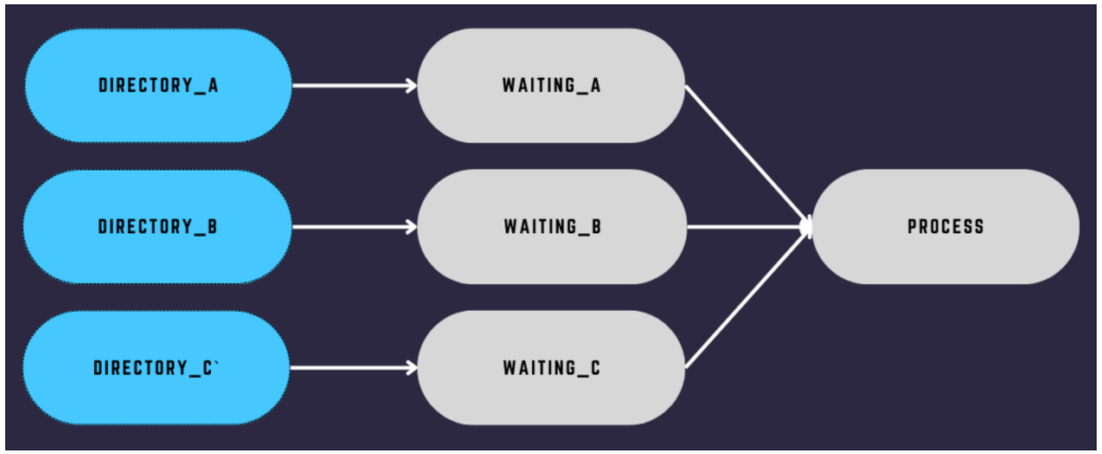
Airflow Executors

- Executor is a mechanism that gets tasks executed.
- it acts as a middle man to handle resource utilization and how to distribute work best.
- Executors types:
  - Sequential Executor: Default, runs one task at a time
  - Local Executor: Runs tasks in parallel on a single machine
  - Celery Executor: Distributed execution across multiple workers
  - Kubernetes Executor: Dynamically launches tasks as Kubernetes pods

LocalExecutor
SequentialExecutor
DebugExecutor

Airflow Sensors

- Definition: Special operators that wait for a certain condition to be true
- Types:
  - FileSensor: Wait for a file to land
  - ExternalTaskSensor: Wait for another DAG to complete
  - SqlSensor: Wait for a SQL query to return results
  - Custom sensors




```python
waiting_for_file = FileSensor(
  task_id='waiting_for_file',
  poke_interval=30
)
```

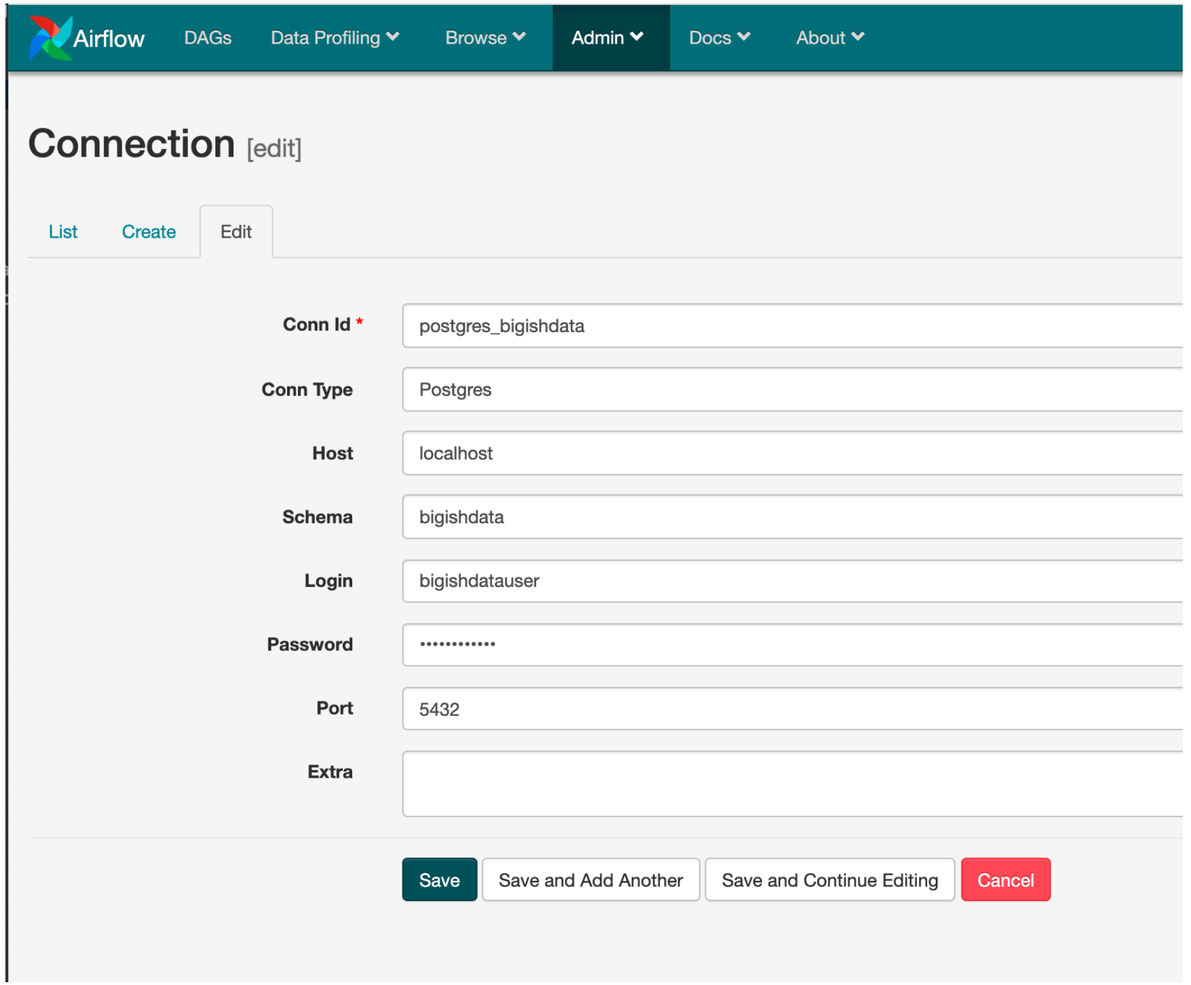
Airflow Hooks

- Purpose: Interface to external systems and databases
- Common hooks:
  - PostgresHook
  - S3Hook
  - HttpHook
  - Custom hooks

```python
# Postgres connection
hook = PostgresHook(postgres_conn_id = "postgres_localhost")
conn = hook.get_conn()
cursor = conn.cursor()


# S3 connection
 s3_hook = S3Hook(aws_conn_id = "s3_connection")
 s3_hook.load_file(
     filename = f.name,
     key = f"orders/{ds_nodash}.txt",
     bucket_name = "from-postgres-data",
     replace = True
 )
```

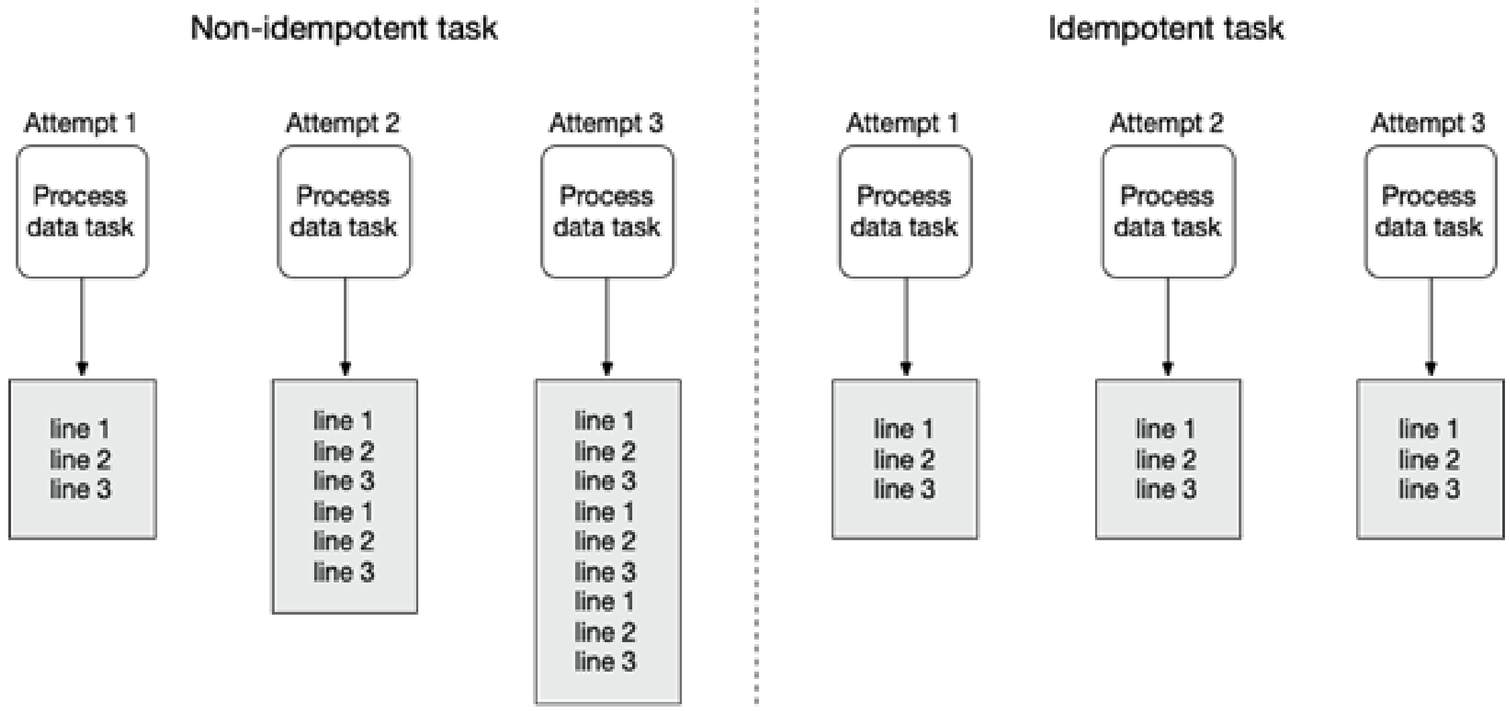
Airflow Variables and Connections

- Variables: Key-value store for configuration
- Connections: Store credentials and connection information
- Accessing variables and connections in DAGs



Airflow Best Practices

- Keep tasks idempotent
- Use templates for reusability
- Implement proper error handling
- Use pools for resource management
- Leverage tags for organization



Airflow Monitoring and Logging

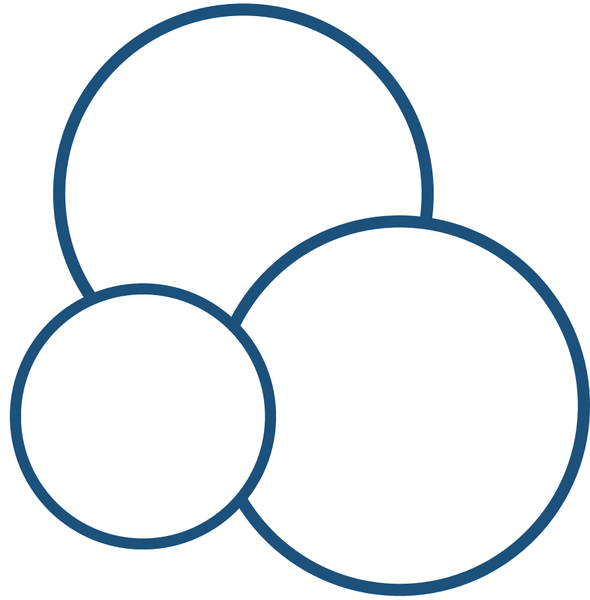
- Web UI for monitoring DAGs and tasks
- Logging configuration
- Alerting and notifications
- Integration with monitoring tools (e.g., Prometheus, Grafana)

Hands-on:

Apache Airflow for Automation and Orchestration

Q&A and Discussion

Data Visualization and Reporting



Introduction to Data Visualization and Reporting

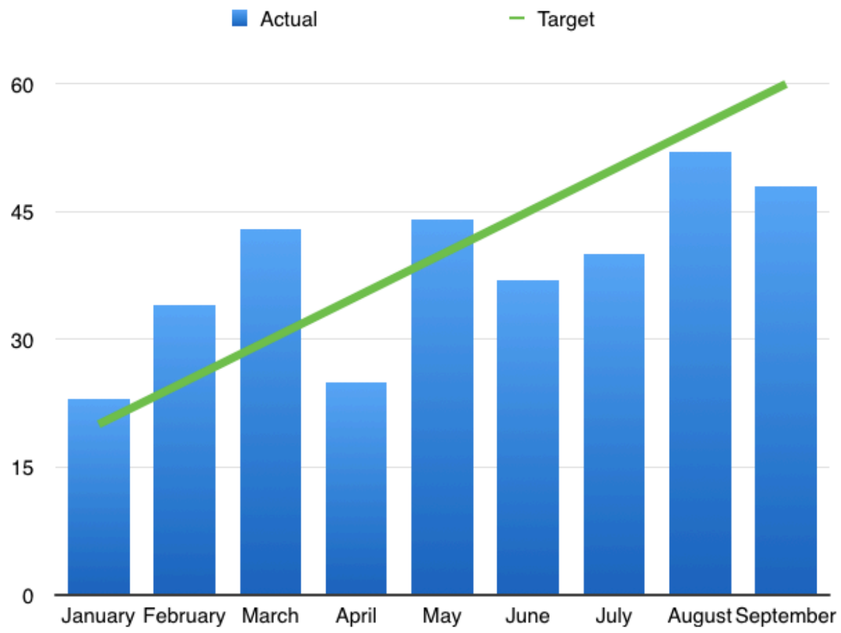
- **Definition**: The graphical representation of data and information
- **Importance** in data analysis and decision-making:
    a. Allows business users to gain insight into their vast amounts of data.
    b. Analyzing the Data in a Better Way
    c. Faster Decision Making
    d. Making Sense of Complicated Data

The Power of Data Visualization

- Simplifies complex information
- Identifies patterns and trends
- Facilitates decision-making
- Enhances data storytelling

Sales

Month	Actual	Target
January	23	20
February	34	25
March	43	30
April	25	35
May	44	40
June	37	45
July	40	50
August	52	55
September	48	60

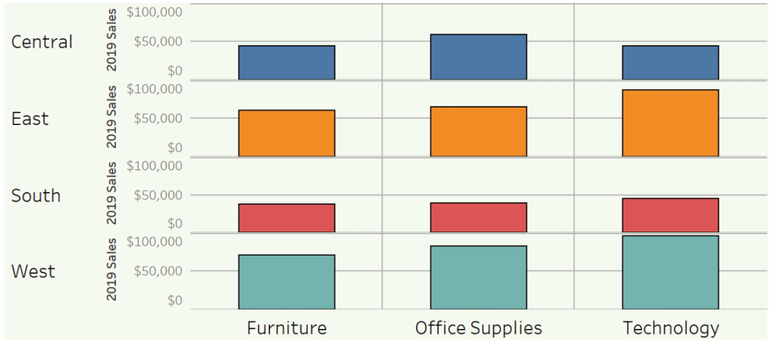


Simplification of complex information

Data Visualization Principles

1. **Clarity**: Make the data easy to understand
2. **Efficiency**: Maximize data-ink ratio
3. **Integrity**: Represent data accurately
4. **Aesthetics**: Create visually appealing graphics

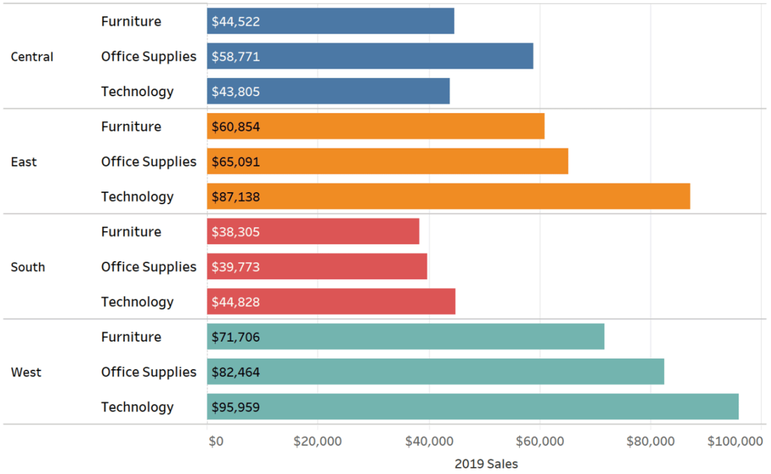
sales by region and category



	Central	East	South	West
Furniture	$44,522	$60,854	$38,305	$71,706
Office Supplies	$58,771	$65,091	$39,773	$82,464
Technology	$43,805	$87,138	$44,828	$95,959

ineffective

sales by region and category



effective

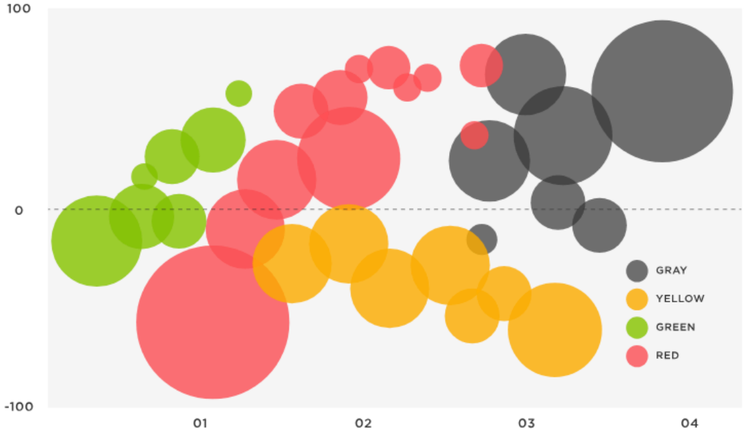
Common Chart Types

1. **Bar charts**: Comparing categories
2. **Line charts**: Showing trends over time
3. **Scatter plots**: Showing relationships between variables
4. **Pie charts**: Showing composition (use sparingly)
5. **Histograms**: Showing distribution of data

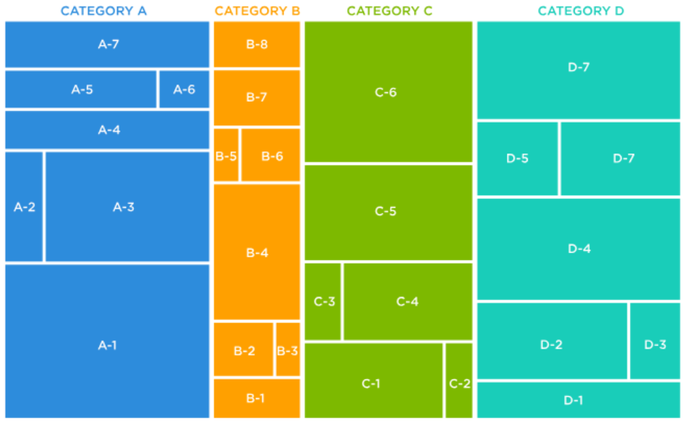
Advanced Chart Types

1. **Heatmaps**: Showing patterns in a matrix
2. **Tree maps**: Hierarchical data
3. **Bubble charts**: Comparing three variables
4. **Sankey diagrams**: Showing flow between categories
5. **Chord diagrams**: Showing inter-relationships

Heatmap



Bubble chart

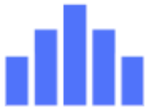


Tree maps

Choosing the Right Chart Type

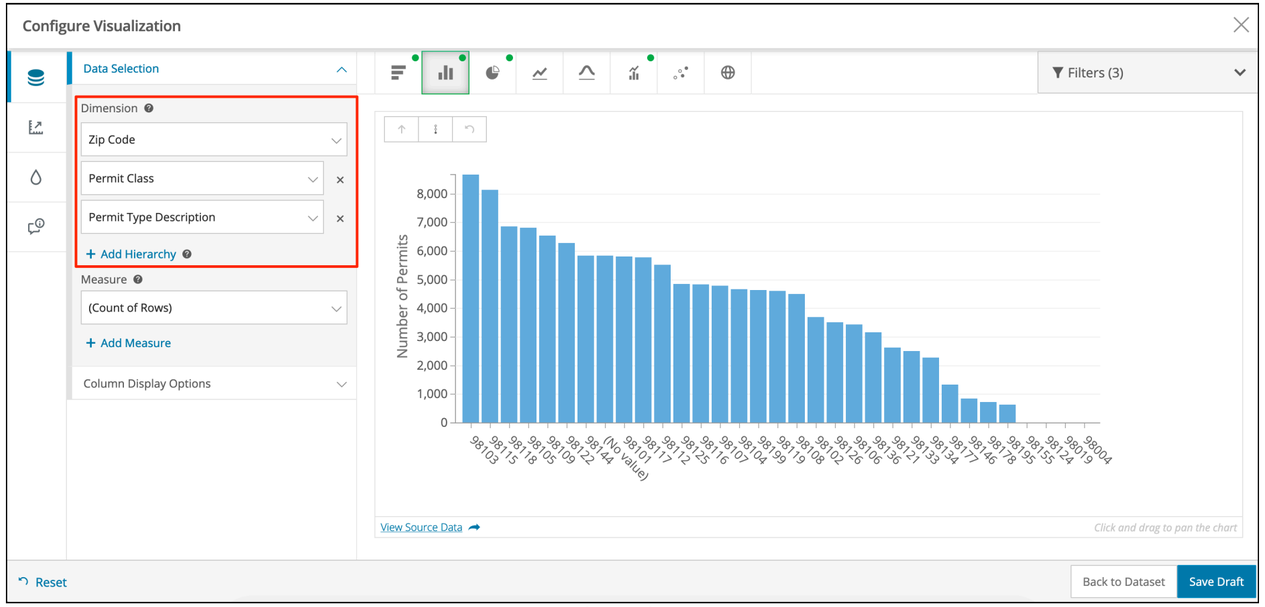
Based on:

1. Type of data (categorical, numerical, time-series)
2. Number of variables
3. Relationships to be shown

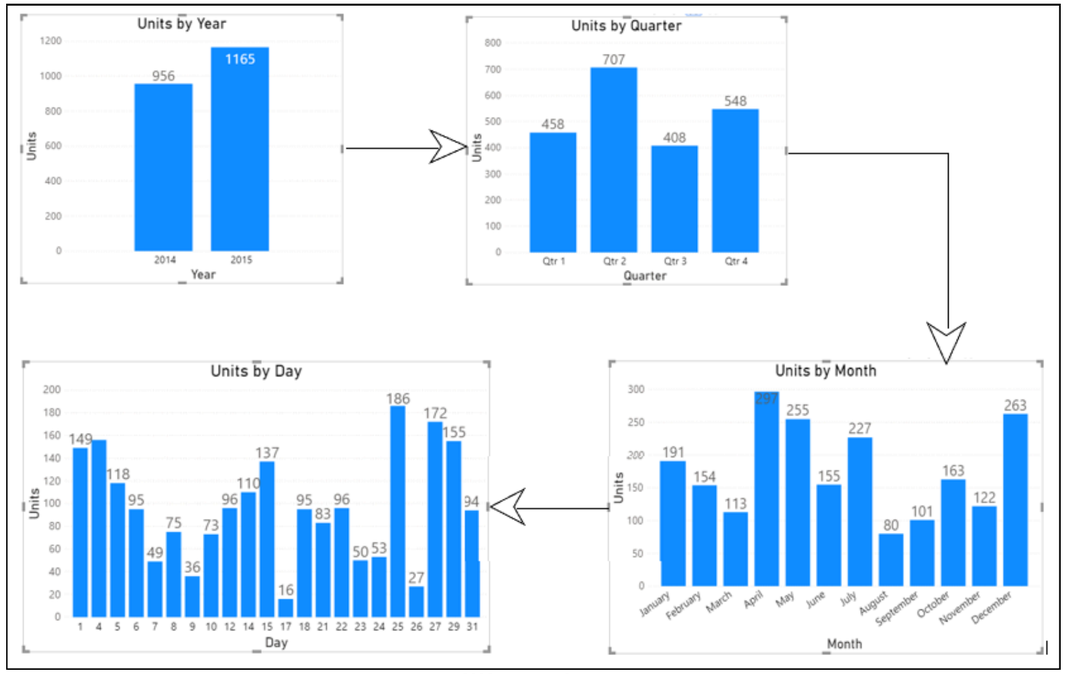
CHART TYPE	DATA TYPE
Line Chart 	Continuous or discrete
Bar Chart 	Discrete
Scatter Chart 	Continuous
Bubble Chart 	Relationship between three variables identifying trends
Histogram 	Distribution Analysis
Heatmap 	Density or distribution of data

Interactivity in Data Visualization

- Filtering
- Drilling down
- Tooltips
- Zooming and panning
- Linked views



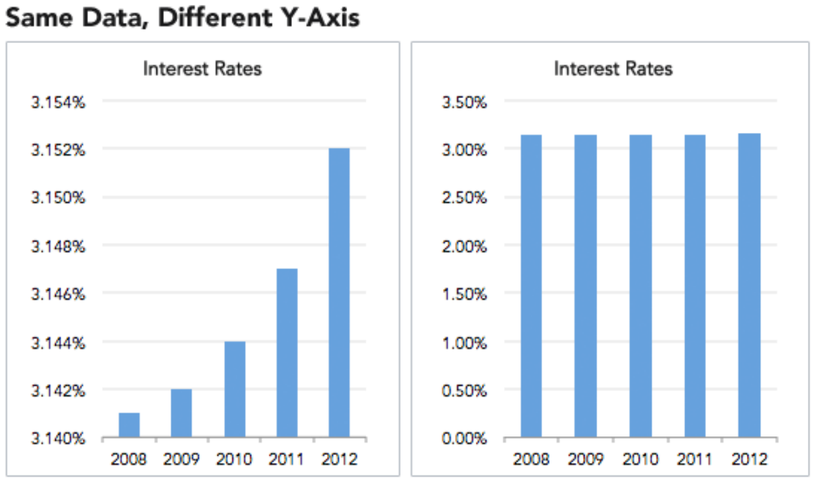
Filtering



Drilling down

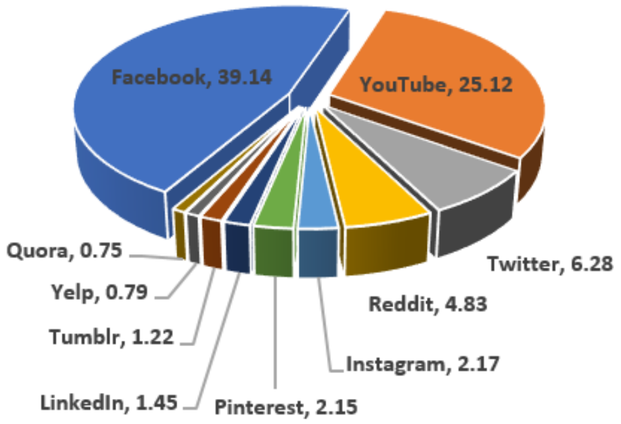
Common Pitfalls in Data Visualization

- Misleading scales
- 3D charts
- Overuse of pie charts
- Cluttered designs
- Ignoring context



Misleading scales

Cluttered designs



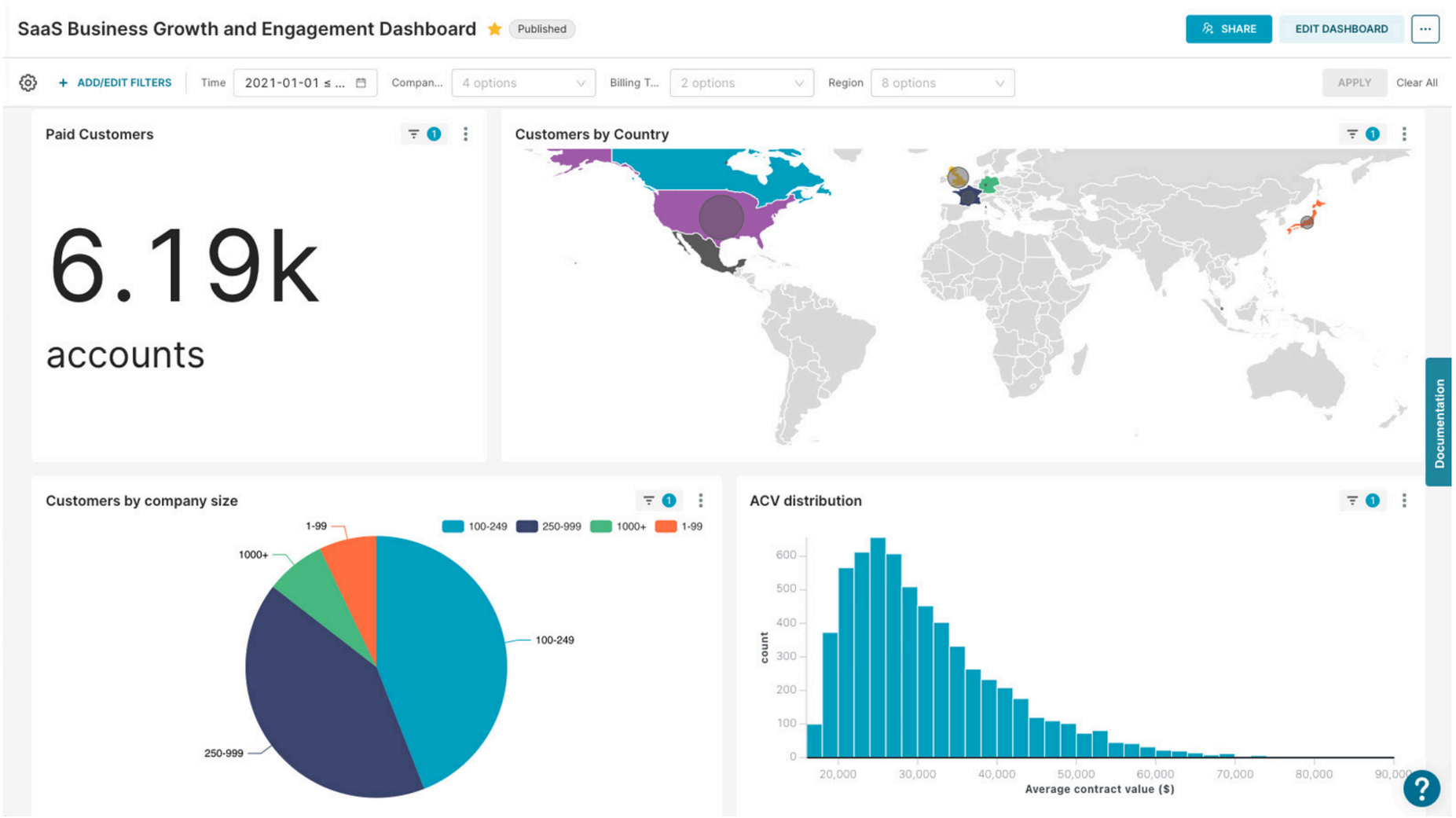
Overuse of pie charts
3D Charts

Data Visualization Tools: PowerBI & Superset

Introduction to Apache Superset

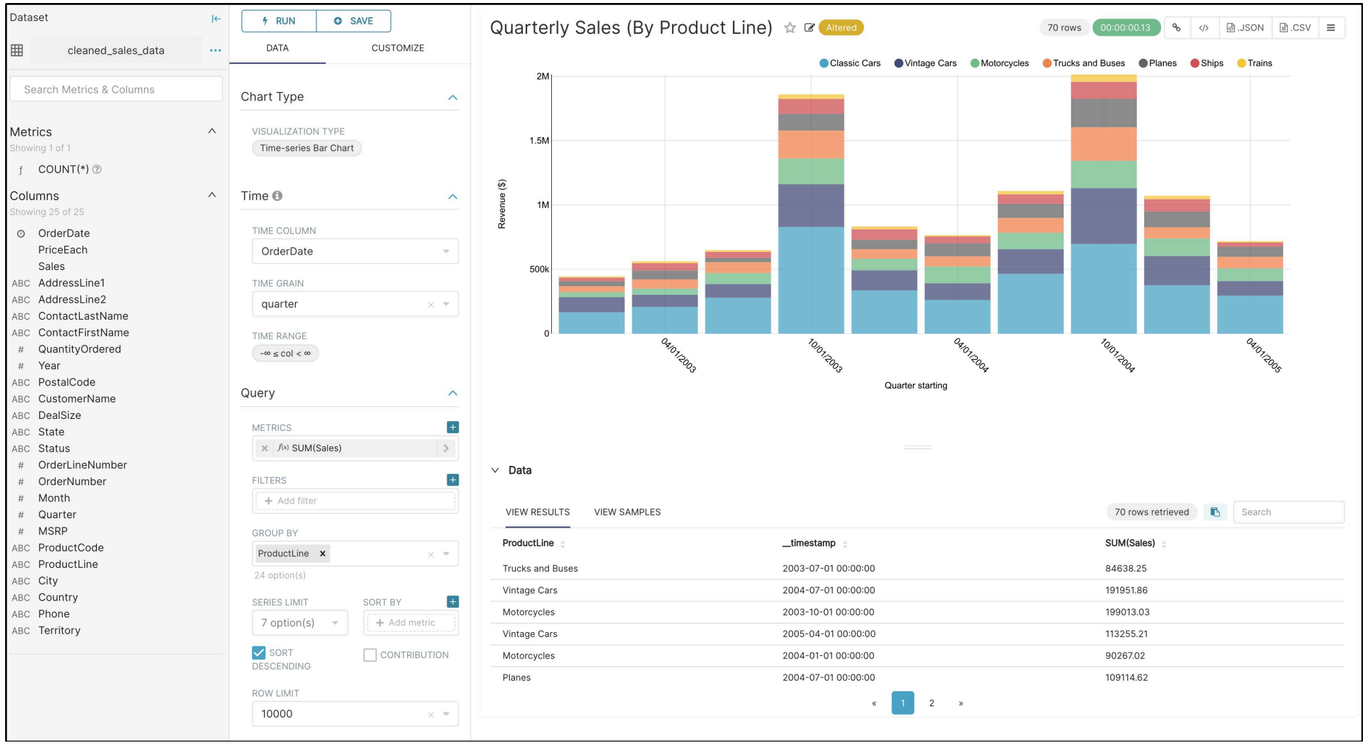
- Overview:
  - Apache Superset is an open-source data visualization and exploration platform.
  - It enables users to create and share interactive dashboards and reports directly from various data sources.
- Key Features:
  - **Data Exploration**: Query and visualize data with a user-friendly interface.
  - **Interactive Dashboards**: Build and share dashboards with various types of visualizations.
  - **Open-Source and Extensible**: Easily customizable and extensible to fit various needs.

Introduction to Apache Superset



Key Components of Apache Superset

- **Data Connectivity**: Supports a wide range of data sources such as SQL databases, cloud data warehouses, and more.
- **SQL Lab**: A built-in SQL editor for querying data directly from connected data sources.
- **Chart Builder**: An intuitive interface to create different types of charts like bar, line, pie, scatter, etc.
- **Dashboard**: A space to combine multiple charts and data elements into a cohesive, interactive dashboard.

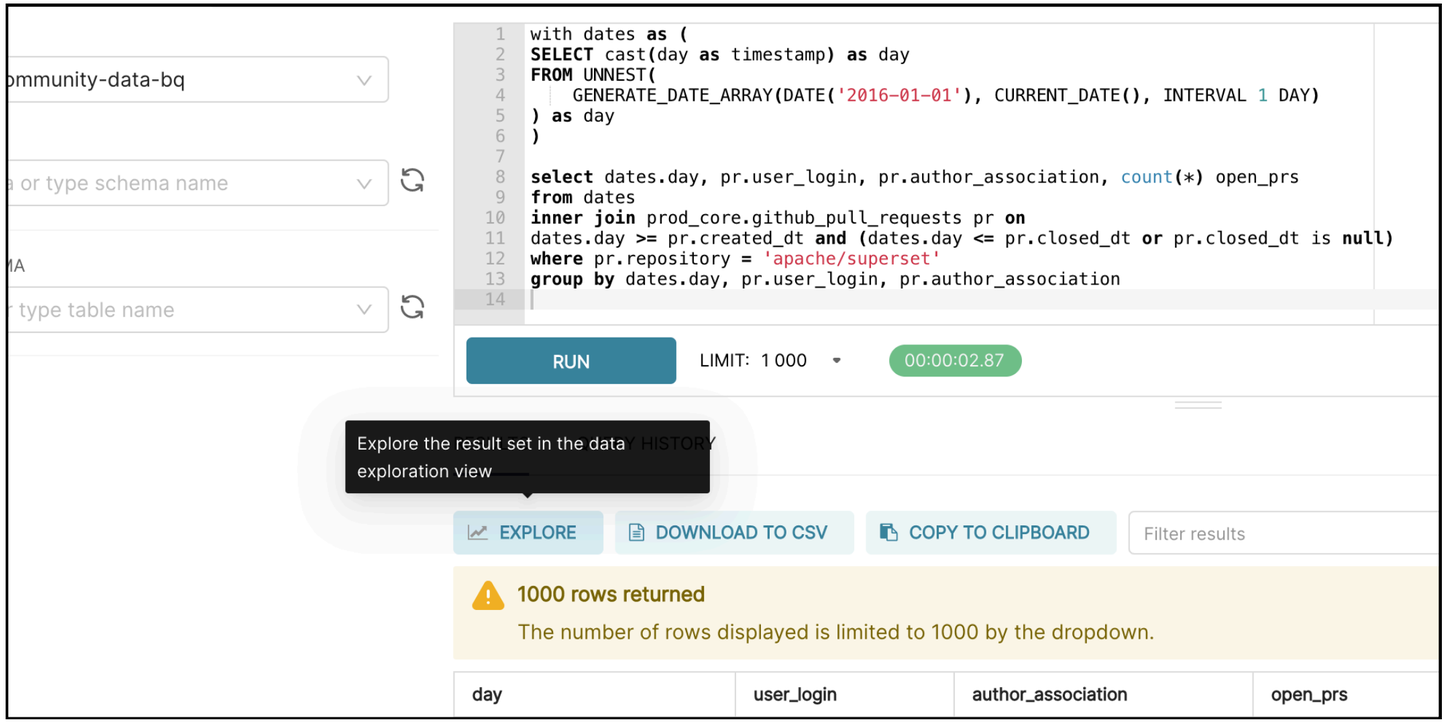


Benefits of Using Apache Superset

- **Easy Integration**: Connects with multiple data sources, including SQL databases, NoSQL stores, and cloud data warehouses.
- **User-Friendly**: No need for advanced programming skills; drag-and-drop interfaces and pre-built visualization templates make it accessible to a wide range of users.
- **Real-Time Analytics**: Provides the ability to visualize data in real-time, supporting decision-making processes.
- **Scalability and Performance**: Optimized for handling large datasets, making it suitable for big data environments.

Superset Example Use Cases

- **Business Intelligence (BI)**: Create interactive dashboards for monitoring KPIs and business metrics.
- **Data Exploration**: Allow data scientists and analysts to explore data directly from connected sources.
- **Reporting**: Automate and distribute reports to stakeholders on a scheduled basis.



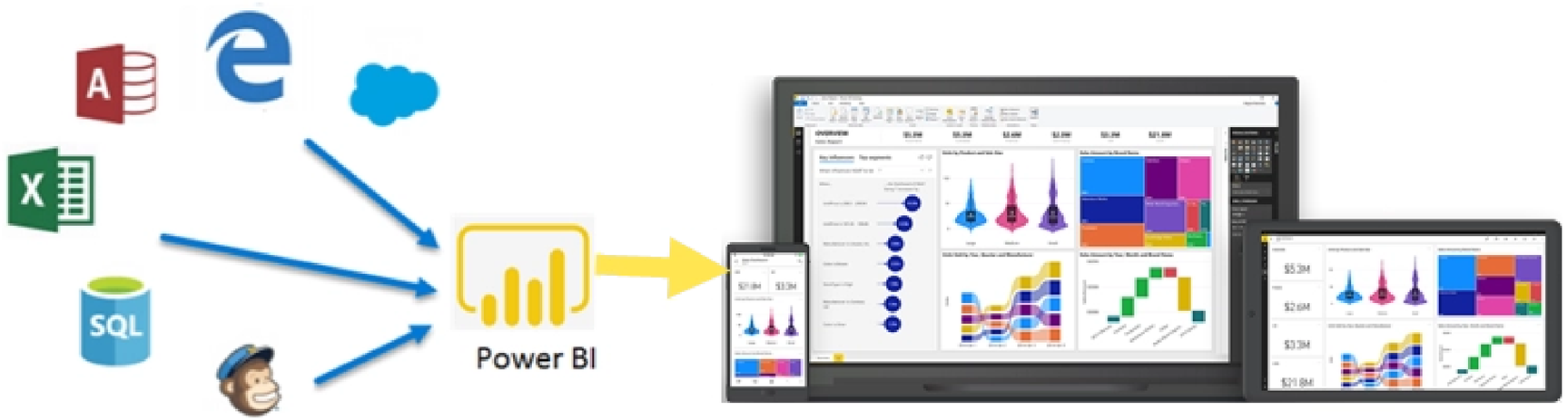
Data Exploration

Hands-on:

Exploring BI Tools:
Apache Superset

Introduction to Power BI

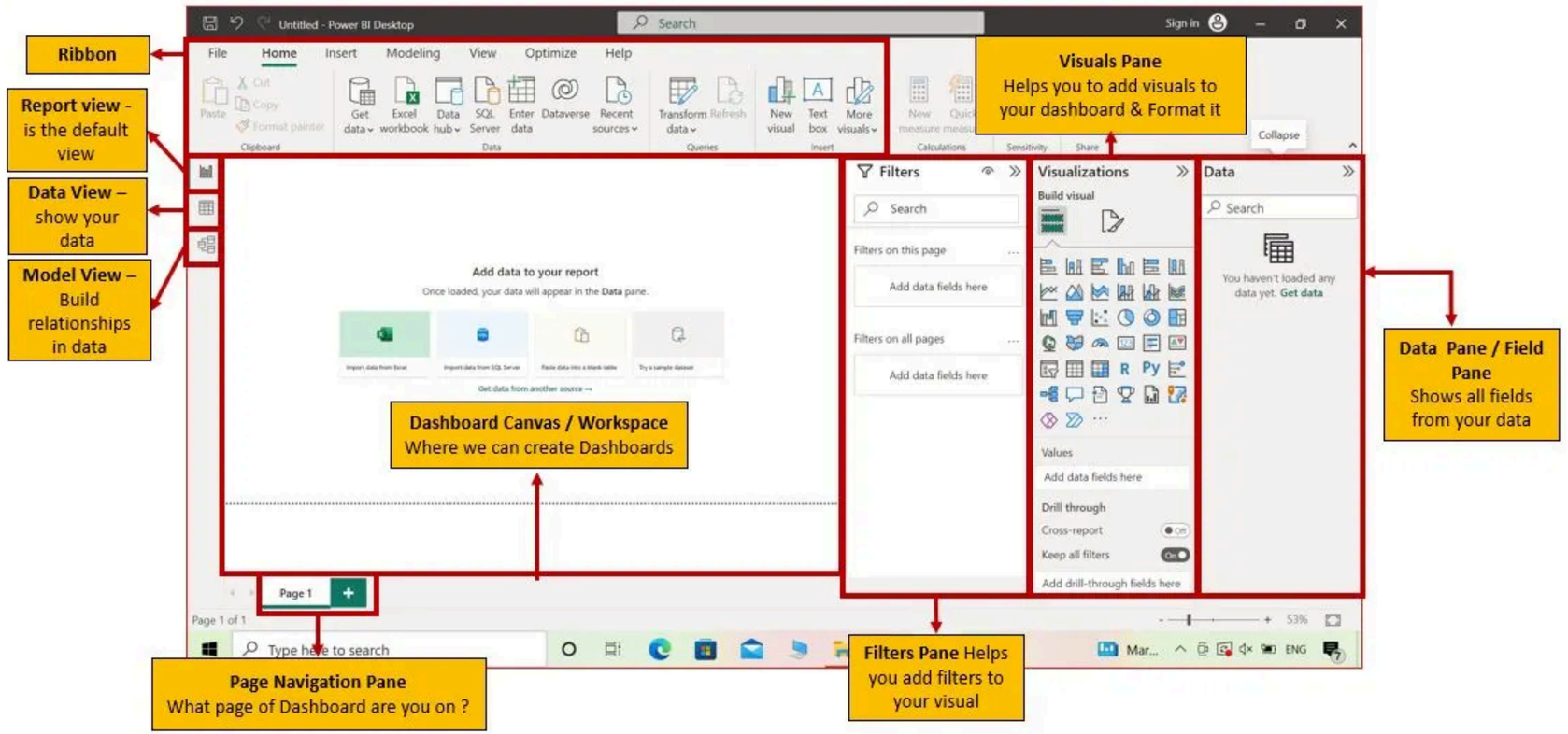
- **Definition**: Microsoft's business analytics tool for interactive visualizations
- Key features:
  - Integration with Microsoft ecosystem
  - Power Query for data transformation
  - DAX (Data Analysis Expressions) for calculations
  - AI-powered insights



Power BI Interface

Power BI

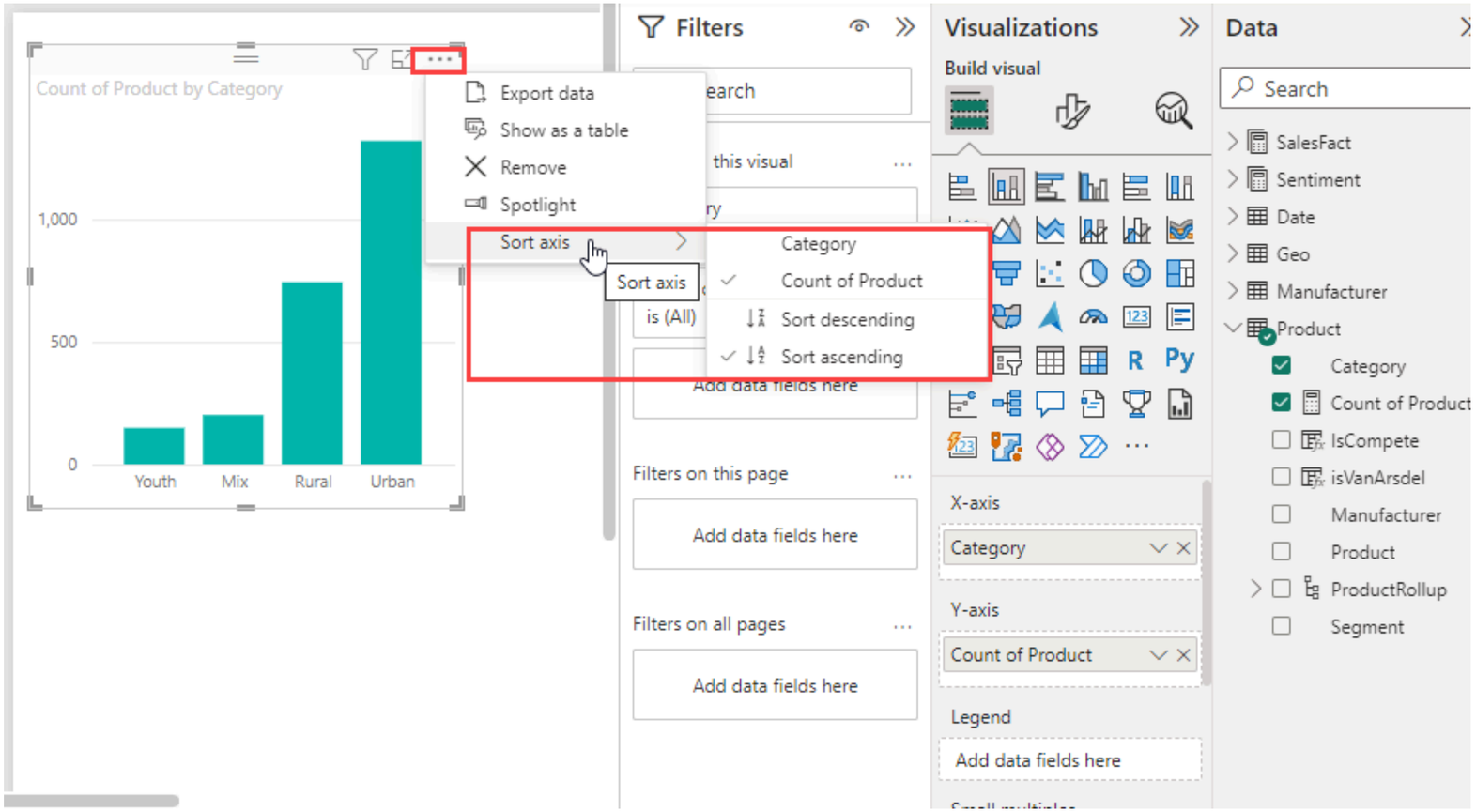
- Report view (dashboards)
- Data view
- Model view
- Fields pane
- Visualizations pane



Creating Visualizations in Power BI

Power BI

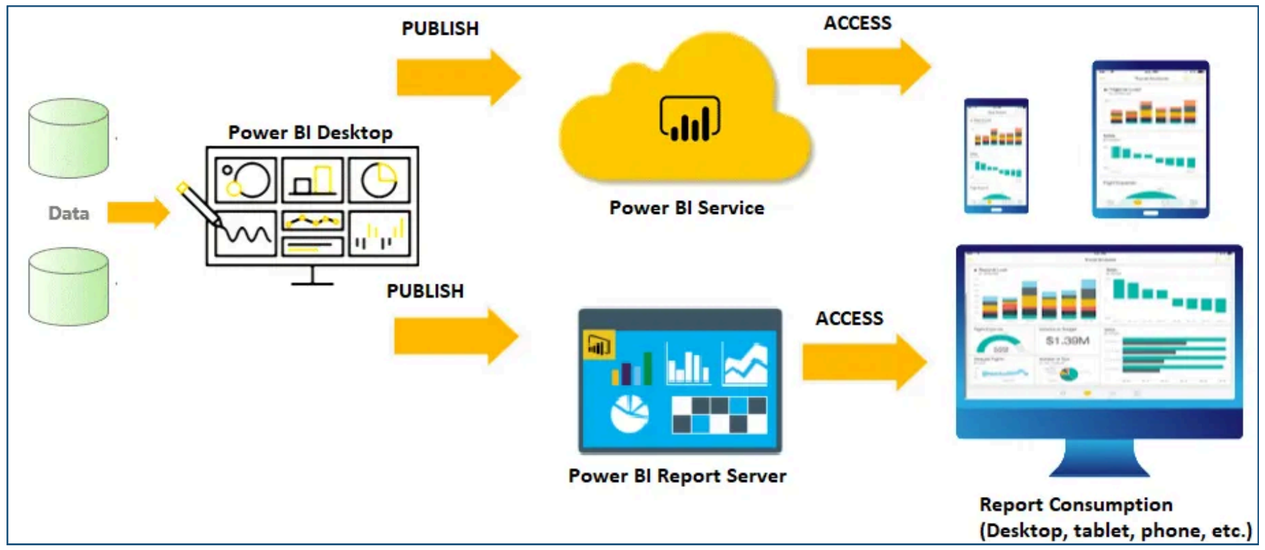
- Importing and connecting to data
- Creating measures and calculated columns
- Building basic charts
- Applying filters and slicers
- Formatting and customizing visualizations

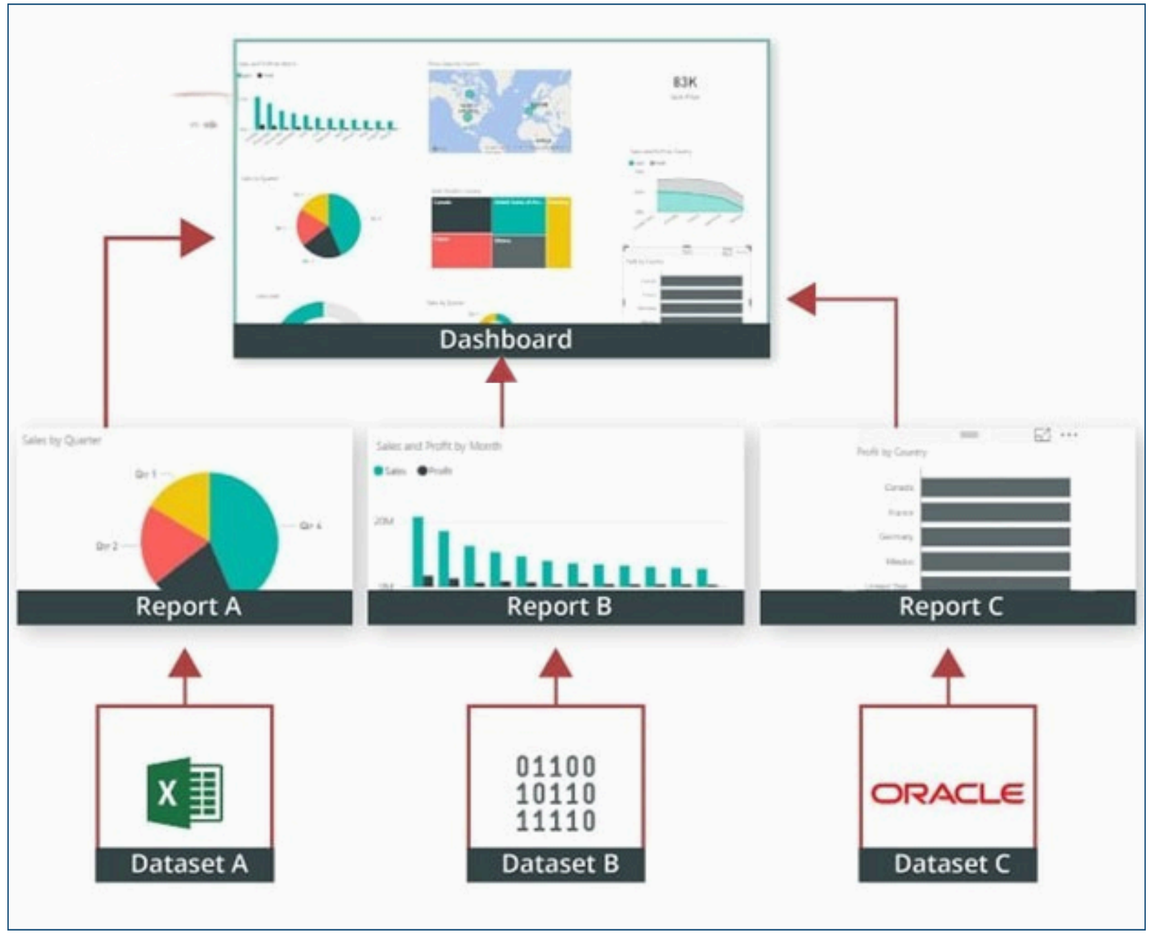


Power BI Dashboards and Reports

Power BI

- Difference between dashboards and reports
- Creating and customizing dashboards
- Sharing and collaboration features
- Power BI Desktop vs. service (Power BI Online)





Hands-on:

Exploring BI Tools:
Power BI

Q&A and Discussion

MEET OUR TEAM

Omar AlSaghier

Sr. Data Engineer

DATATECH LABS.

THANK YOU

OUR CONTACT

DataTechLabs

datatechlabs.ai@gmail.com

datatechlabs.ai

Amman, Jordan