



DATATECH LABS.

Data Engineering Course.

**Week 5: Data Automation,
Orchestration and Visualization**

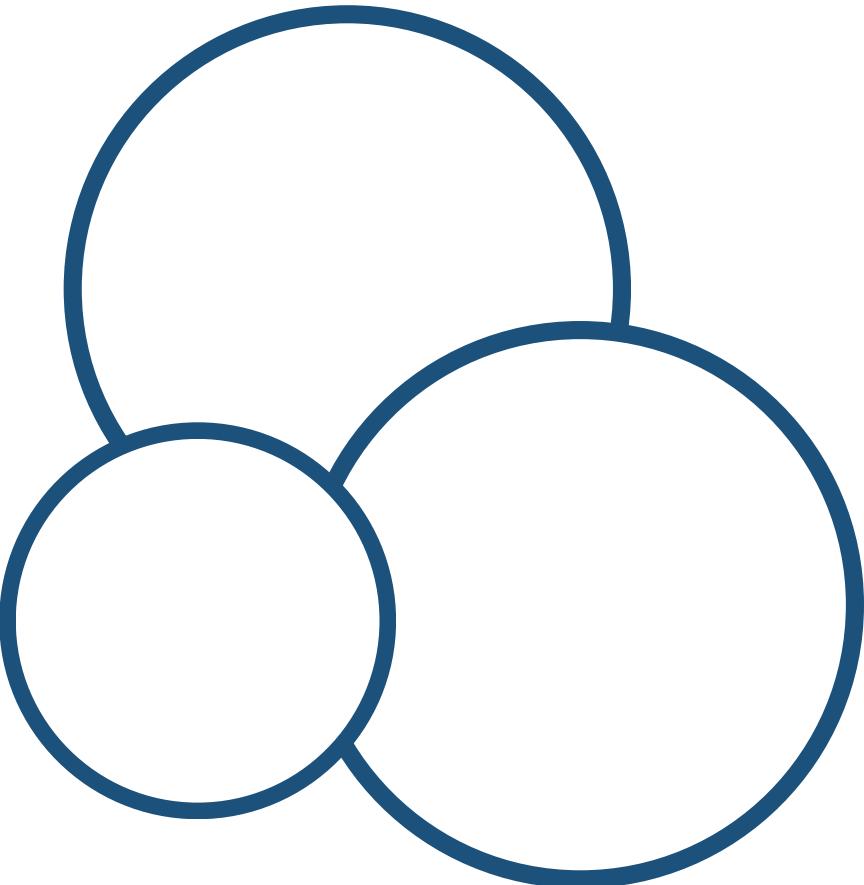
Outline: Week 5

- 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

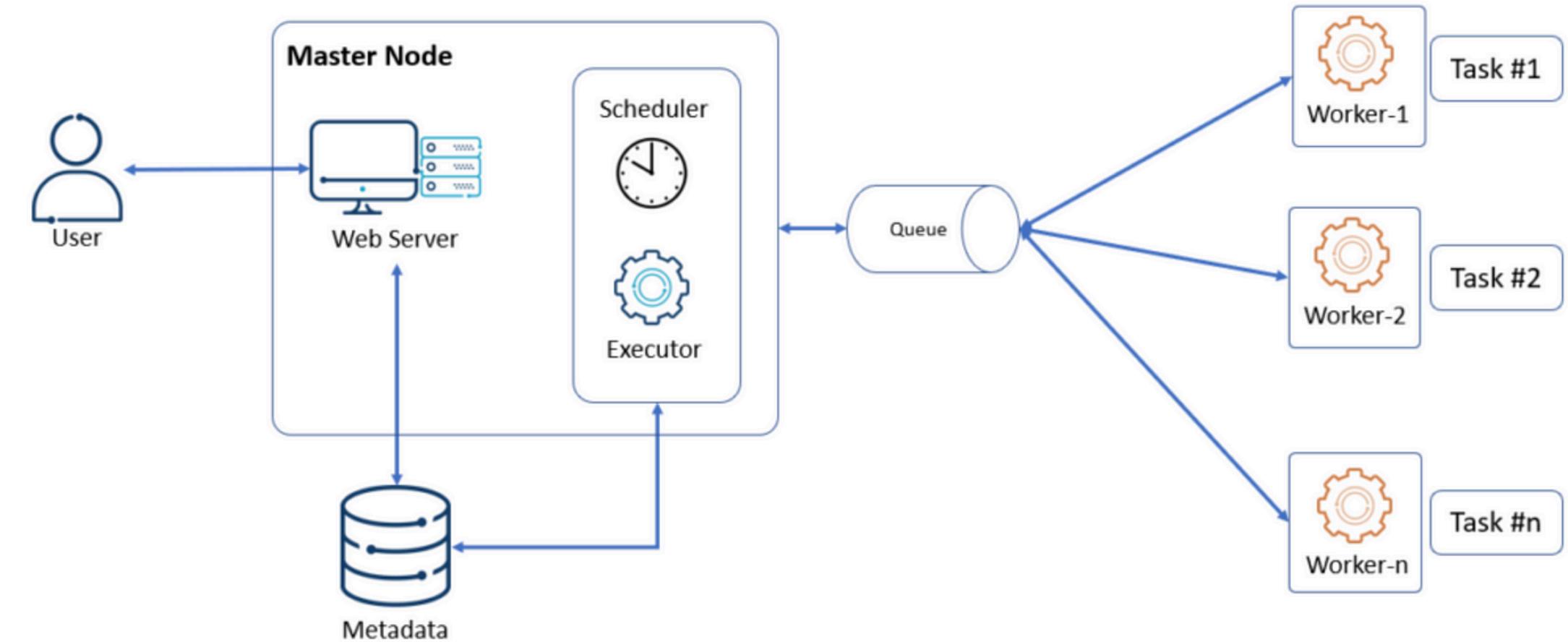
Orchestration



Introduction to Data Pipeline Automation and Orchestration

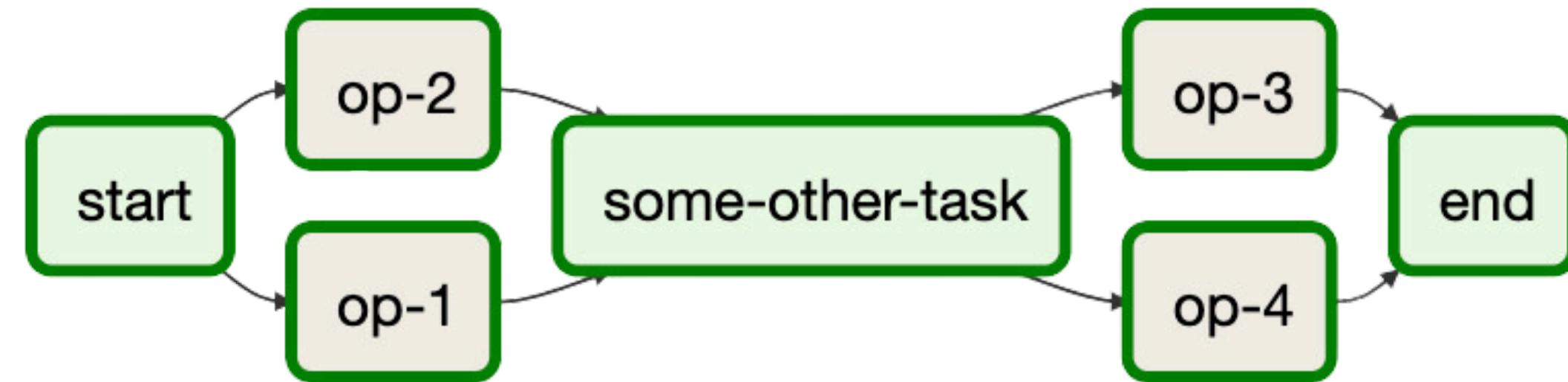
Introduction to Automation and Orchestration

- **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



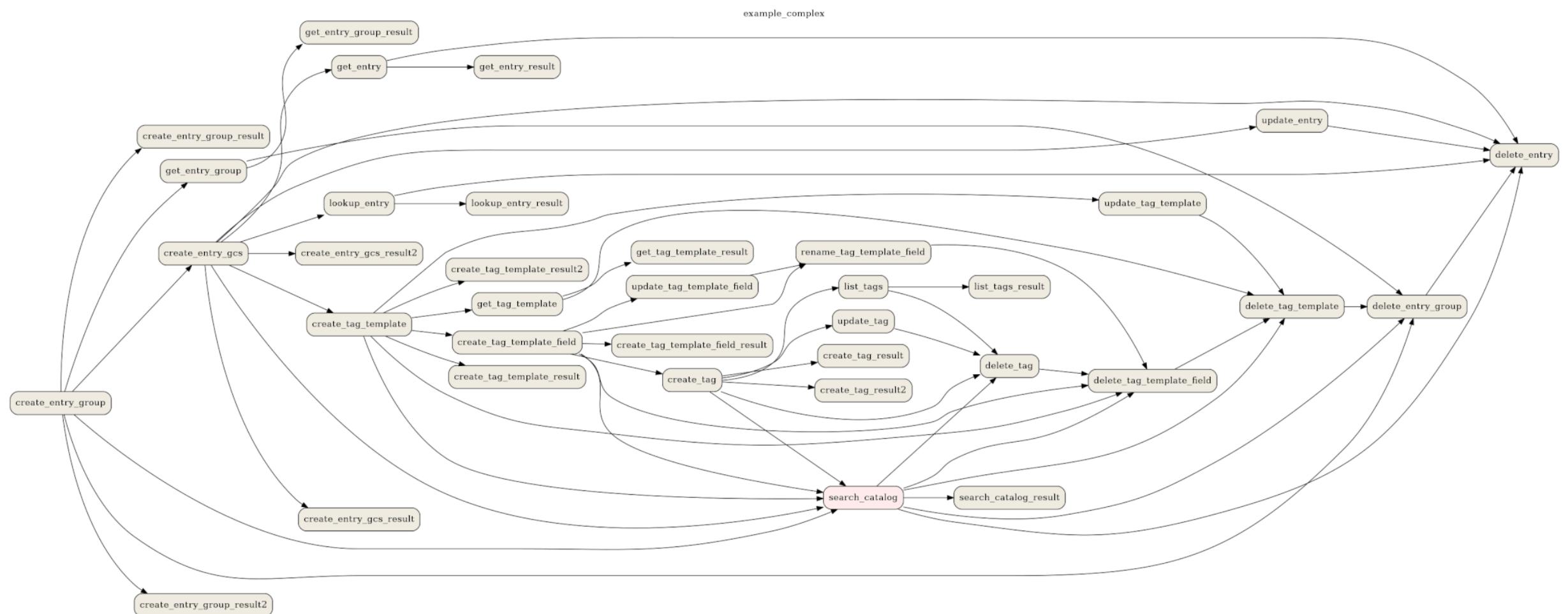
Benefits of Pipeline Automation

- 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

The Airbyte logo consists of a blue icon resembling a stylized '9' or a series of connected arcs, followed by the word "Airbyte" in a bold, dark blue sans-serif font.

dagster



Luigi

The Luigi logo is the word "Luigi" written in a green, blocky, 3D-style font that looks like it's made of interlocking pipes or tubes.

Apache Airflow as Automation Tool

Introduction to Apache Airflow



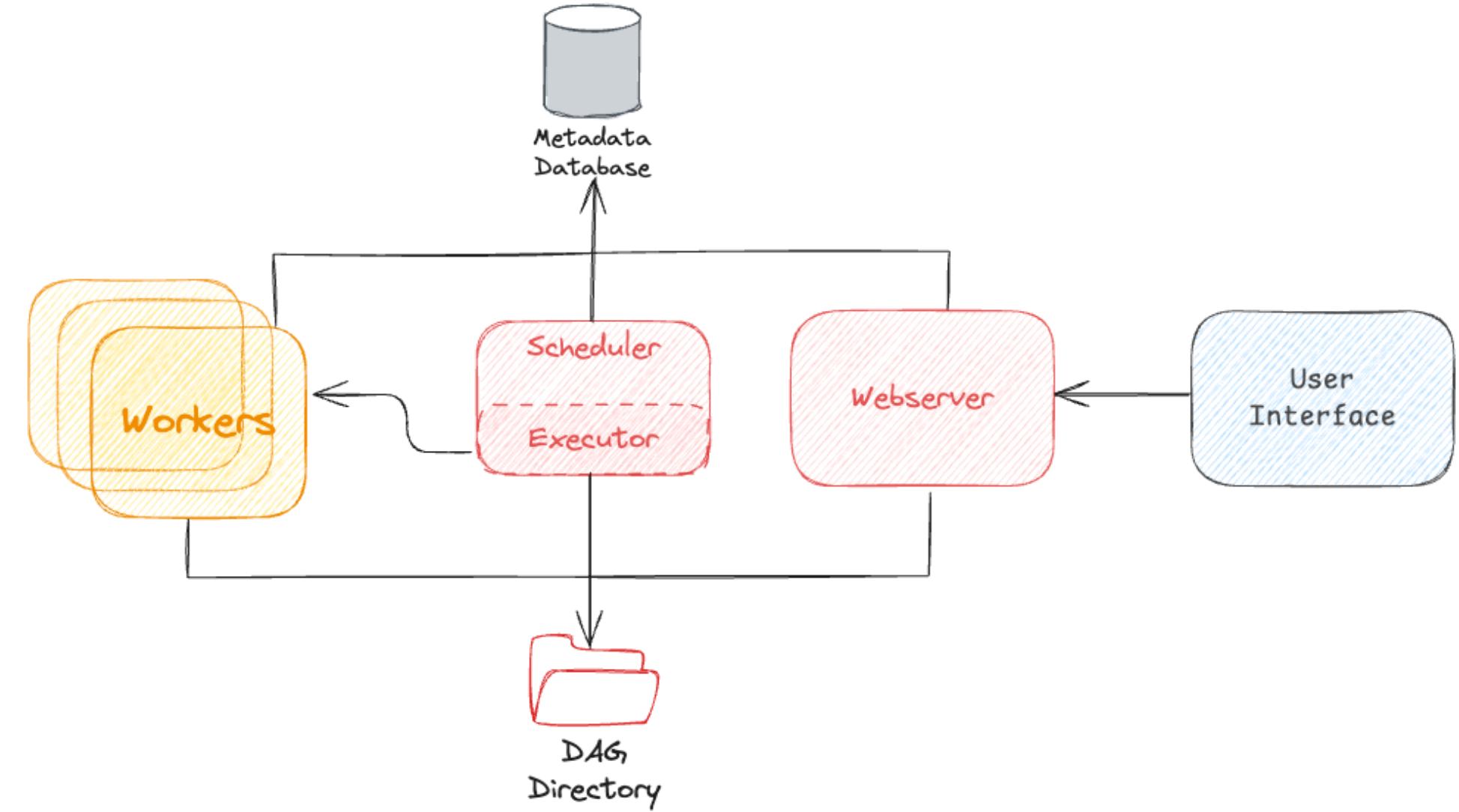
- **Definition:** An open-source platform to programmatically author, schedule, and monitor workflows
- Created by Airbnb in 2014, now an Apache Software Foundation project

A screenshot of the Apache Airflow web interface. At the top, there is a navigation bar with links for 'Airflow', 'DAGs', 'Security', 'Browse', 'Admin', and 'Docs'. On the far right, it shows the time '21:11 UTC' and a user icon 'RH'. Below the navigation bar, the page title is 'DAGs'. There are three tabs at the top left of the main content area: 'All 26' (highlighted in blue), 'Active 10', and 'Paused 16'. To the right of these tabs are two search bars: 'Filter DAGs by tag' and 'Search DAGs'. The main content is a table listing ten DAGs. Each row in the table contains the DAG name, owner, number of runs, schedule, last run date, recent tasks count, actions (represented by icons for view, edit, delete, and more), and a 'Links' column. The DAG names listed are: example_bash_operator, example_branch_dop_operator_v3, example_branch_operator, example_complex, example_external_task_marker_child, example_external_task_marker_parent, example_kubernetes_executor, example_kubernetes_executor_config, example_nested_branch_dag, and example_passing_params_via_test_command. Most DAGs are owned by 'airflow' and have a status of 'Active' except for 'example_branch_operator' which is 'Paused'. The 'example_complex' DAG has 37 recent tasks, while others have fewer.

Airflow Architecture



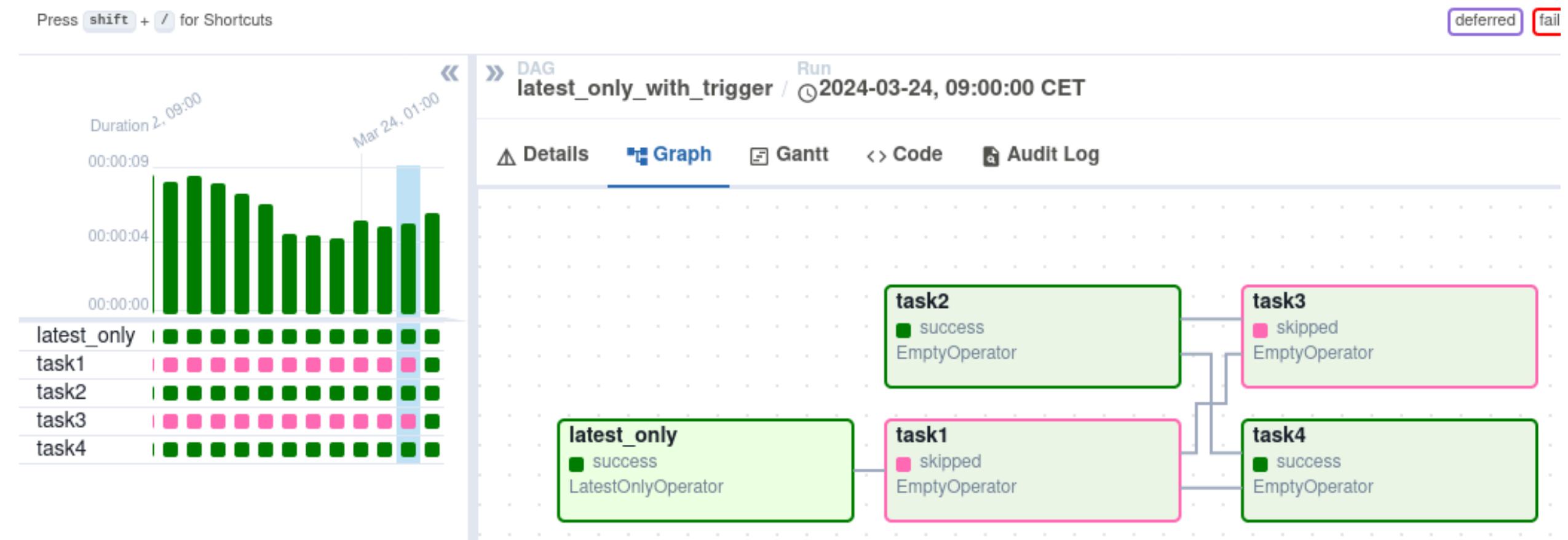
1. **Webserver**: Provides the user interface
2. **Scheduler**: Monitors and triggers tasks
3. **Workers**: Execute the tasks
4. **Metadata Database**: Stores DAGs, task instances, and configurations
5. **DAG Files**: Python files defining the workflows



Key Concepts in Airflow



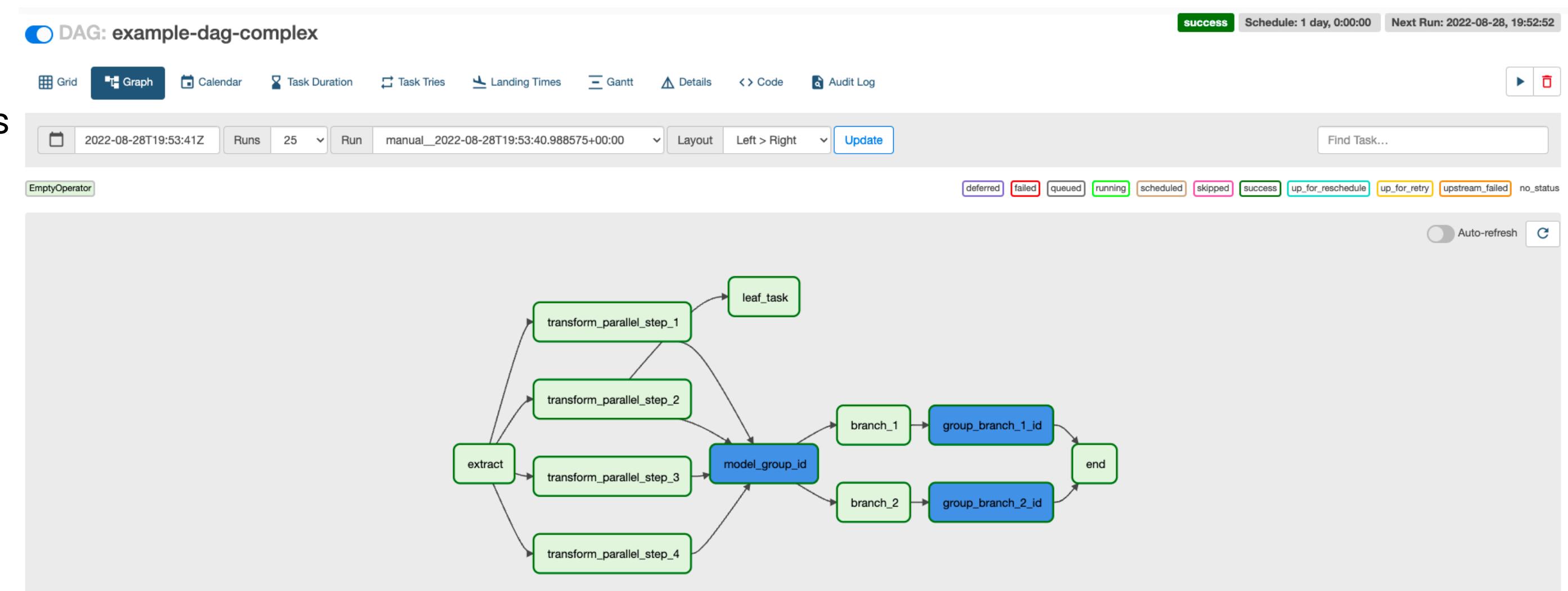
- DAG (Directed Acyclic Graph): Represents a workflow
- Operator: Defines a single task in the workflow
- Task: An instance of an operator
- Task Instance: A specific run of a task
- Execution Date: The logical date and time of a DAG run



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

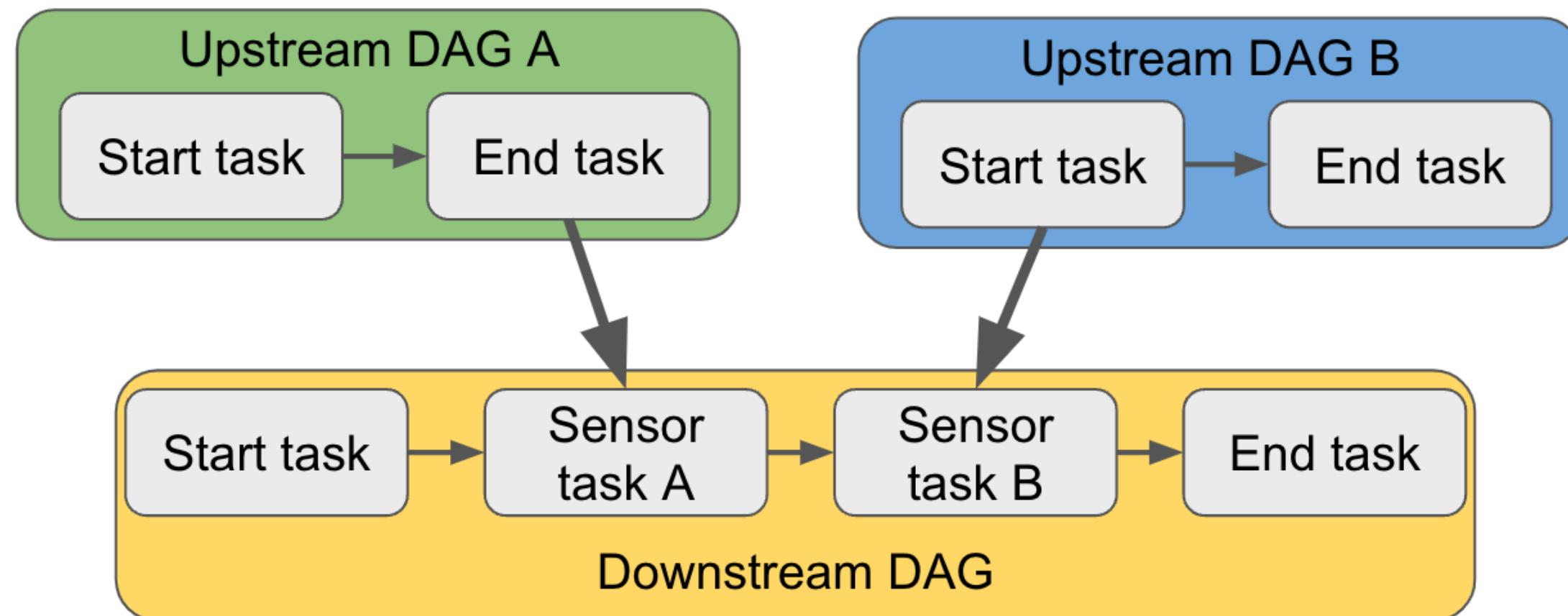
```
● ● ●  
1 from airflow.operators.bash_operator import BashOperator  
2  
3 generate_data = BashOperator(  
4     task_id='generate_data',  
5     bash_command='dd if=/dev/random of=/path/to/newfile.txt bs=1 count=10',  
6     dag=dag  
7 )  
8
```

```
● ● ●  
1 from datetime import datetime  
2 from airflow import DAG  
3 from airflow.operators.dummy_operator import DummyOperator  
4  
5 with DAG('my_dag',  
6           start_date=datetime(2016, 1, 1)  
7 ) as dag:  
8     op = DummyOperator('op')  
9  
10 op.dag is dag # True  
11
```

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
```

```
● ● ●
1 from datetime import datetime
2 from airflow import DAG
3 from airflow.operators.dummy_operator import DummyOperator
4
5 with DAG('my_dag',
6           start_date=datetime(2016, 1, 1) →
7 ) as dag:
8     op = DummyOperator('op')
9
10 op.dag = dag # True
11
```

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



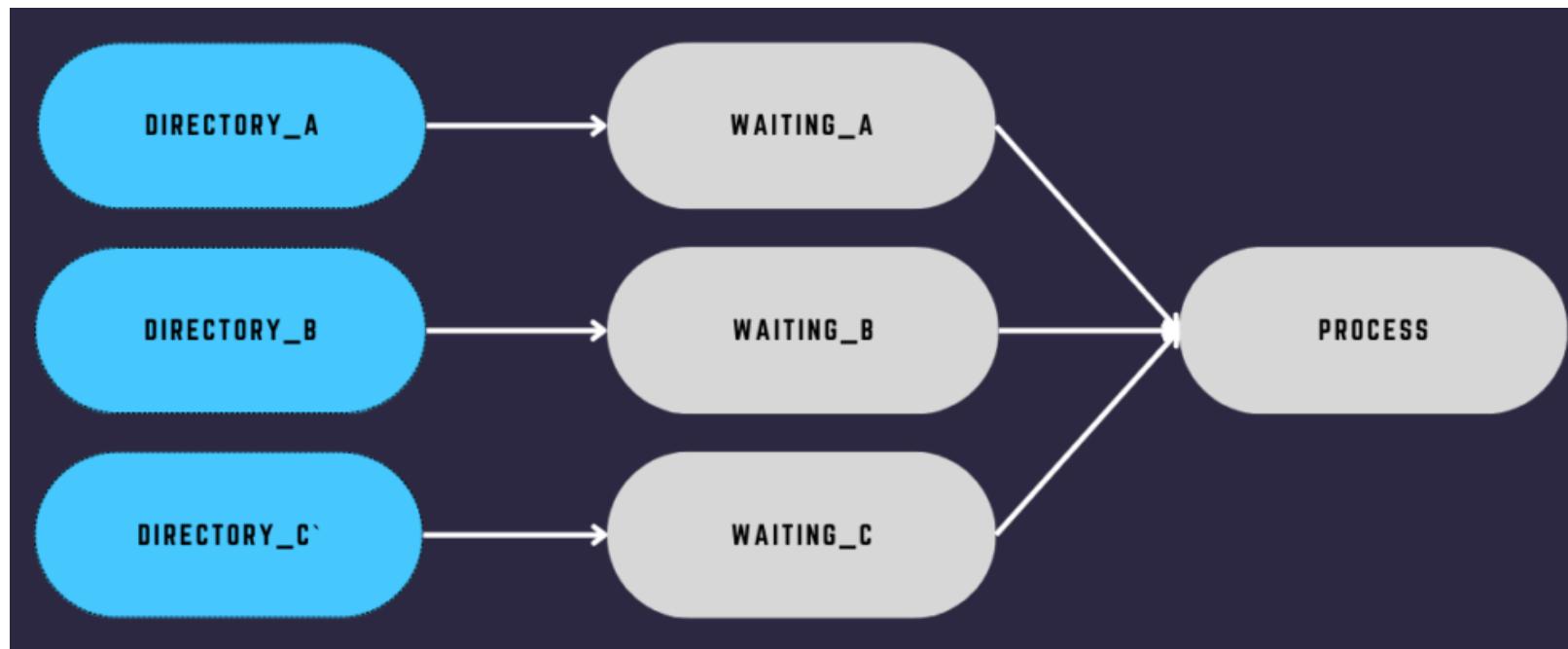
LocalExecutor
SequentialExecutor
DebugExecutor



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



```
1 waiting_for_file = FileSensor(  
2     task_id='waiting_for_file',  
3     poke_interval=30  
4 )
```

Airflow Hooks



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

```
● ● ●  
1 # Postgres connection  
2 hook = PostgresHook(postgres_conn_id = "postgres=localhost")  
3 conn = hook.get_conn()  
4 cursor = conn.cursor()  
5  
6  
7 # S3 connection  
8 s3_hook = S3Hook(aws_conn_id = "s3_connection")  
9 s3_hook.load_file(  
10     filename = f.name,  
11     key = f"orders/{ds_nodash}.txt",  
12     bucket_name = "from-postgres-data",  
13     replace = True  
14 )
```

Airflow Variables and Connections



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

Screenshot of the Apache Airflow web interface showing the "Connection [edit]" page. The page displays a form for creating a new connection to a Postgres database. The fields and their values are:

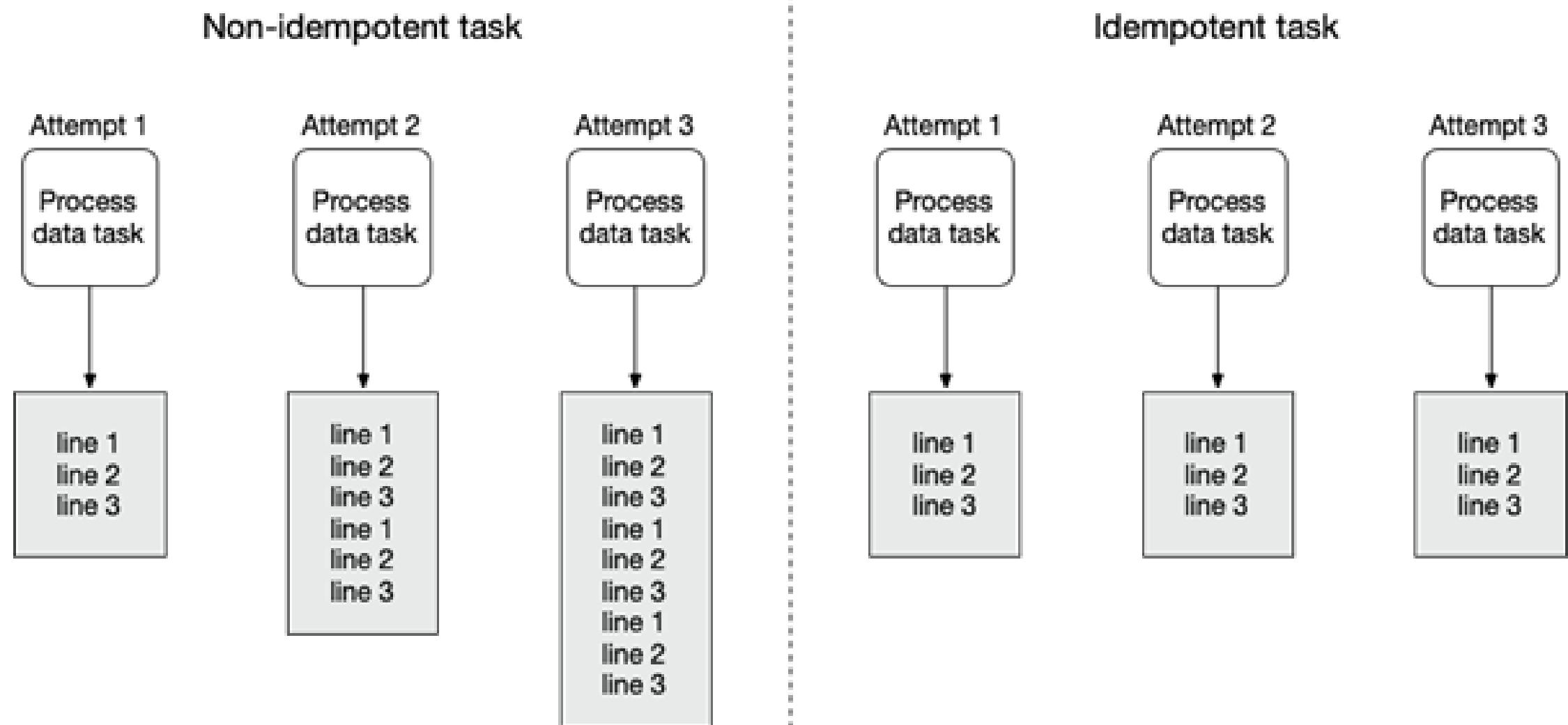
Field	Value
Conn Id *	postgres_bigishdata
Conn Type	Postgres
Host	localhost
Schema	bigishdata
Login	bigishdatauser
Password
Port	5432
Extra	(empty)

At the bottom of the page are four buttons: "Save" (dark blue), "Save and Add Another" (light blue), "Save and Continue Editing" (light blue), and "Cancel" (red).

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)

A screenshot of the Apache Airflow Web UI. At the top, there's a navigation bar with tabs for 'DAG', 'Run', and 'Task'. Below it, specific details are shown: 'test_ui_grid / 2022-06-06, 20:37:57 CEST / task_2'. A toolbar below the navigation bar includes buttons for 'Task Instance Details', 'Rendered Template', 'Log', 'XCom', 'List Instances, all runs', and 'Filter Upstream'. The 'Logs' tab is currently selected. On the left, there's a vertical timeline bar with a grid of colored squares representing task attempts. The main area shows log entries for task_2 attempt 1. A red box highlights the 'Full Logs' checkbox in the toolbar. The log output is as follows:

```
172ccbff4f08
*** Reading local file: /root/airflow/logs/dag_id=test_ui_grid/run_id=manual__2022-06-07T18:37:57.325148+00:00/
[2022-06-07 18:38:00,020] {taskinstance.py:1132} INFO - Dependencies all met for <TaskInstance: test_ui_grid.se
[2022-06-07 18:38:00,025] {taskinstance.py:1132} INFO - Dependencies all met for <TaskInstance: test_ui_grid.se
[2022-06-07 18:38:00,025] {taskinstance.py:1329} INFO -
-----
[2022-06-07 18:38:00,025] {taskinstance.py:1330} INFO - Starting attempt 1 of 1
[2022-06-07 18:38:00,025] {taskinstance.py:1331} INFO -

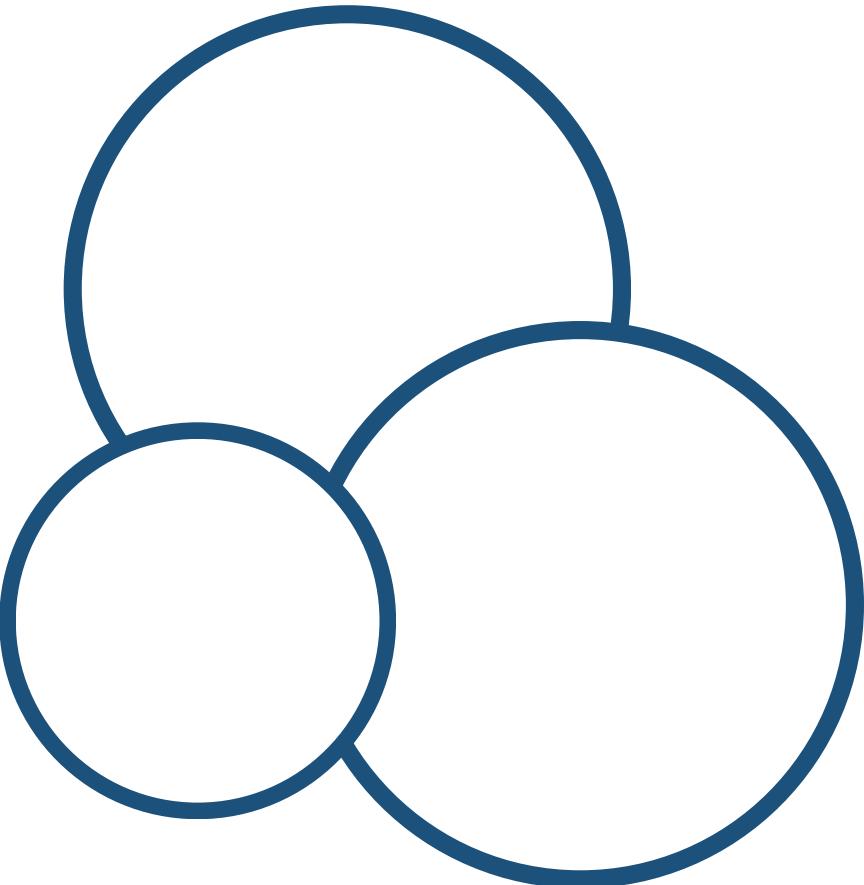
[2022-06-07 18:38:00,040] {taskinstance.py:1350} INFO - Executing <Task(BashOperator): section_1.task_2> on 202
[2022-06-07 18:38:00,043] {standard_task_runner.py:52} INFO - Started process 5434 to run task
[2022-06-07 18:38:00,044] {standard_task_runner.py:80} INFO - Running: ['****', 'tasks', 'run', 'test_ui_grid'],
[2022-06-07 18:38:00,045] {standard_task_runner.py:81} INFO - Job 1707: Subtask section_1.task_2
[2022-06-07 18:38:00,045] {dagbag.py:507} INFO - Filling up the DagBag from /files/dags/test_ui_grid.py
[2022-06-07 18:38:00,086] {task_command.py:377} INFO - Running <TaskInstance: test_ui_grid.section_1.task_2 man
[2022-06-07 18:38:00,129] {taskinstance.py:1548} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=****
AIRFLOW_CTX_DAG_ID=test_ui_grid
AIRFLOW_CTX_TASK_ID=section_1.task_2
AIRFLOW_CTX_EXECUTION_DATE=2022-06-07T18:37:57.325148+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_DAG_RUN_ID=manual_2022-06-07T18:37:57.325148+00:00
```

Hands-on:

Apache Airflow for Automation and Orchestration

Q&A and Discussion

Data Visualization and Reporting



Introduction to Data Visualization and Reporting

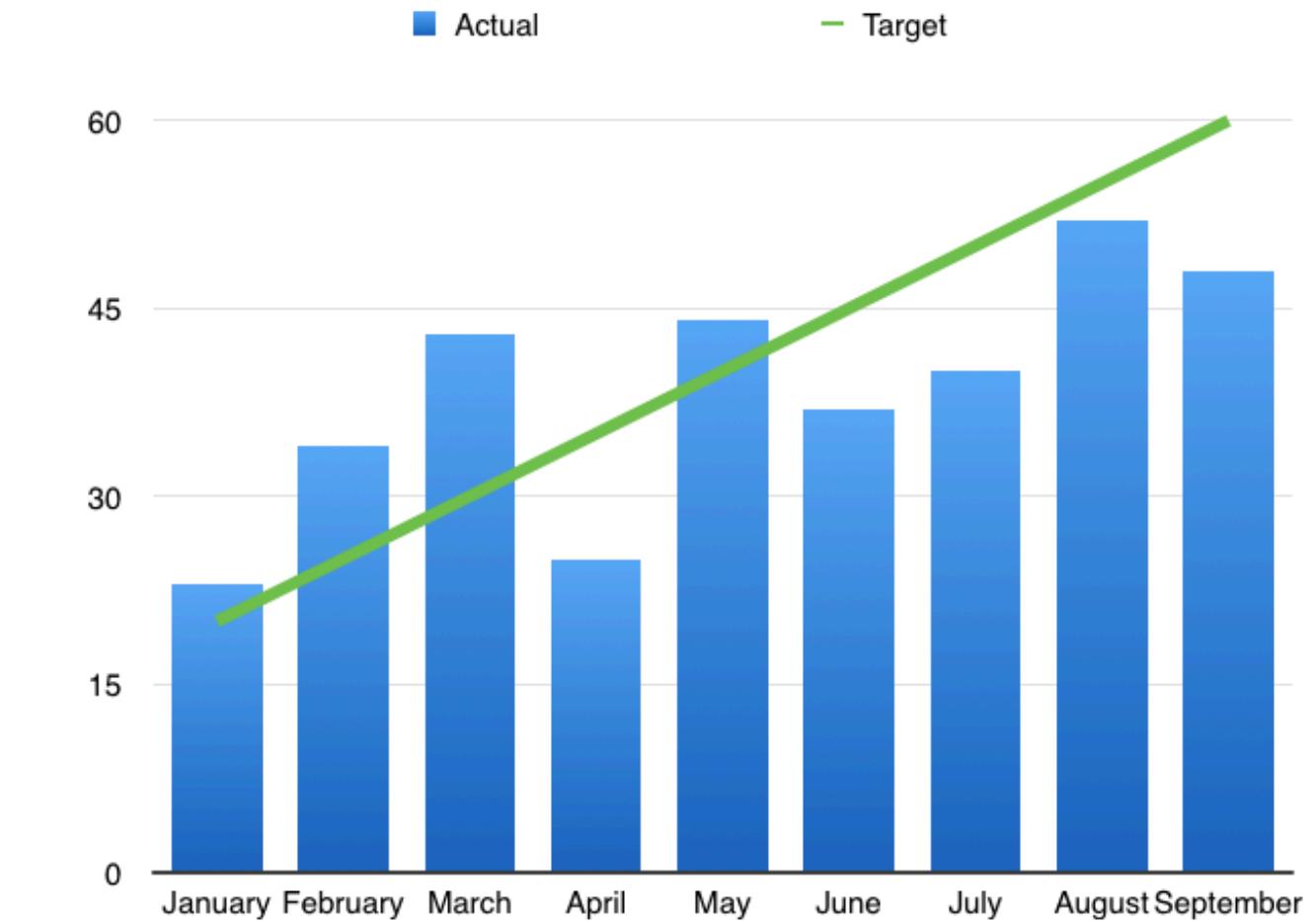
- **Definition:** The graphical representation of data and information
- **Importance** in data analysis and decision-making:
 - a. Allows business users to gain insight into their vast amounts of data.
 - b. Analyzing the Data in a Better Way
 - c. Faster Decision Making
 - d. Making Sense of Complicated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



The Power of Data Visualization

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

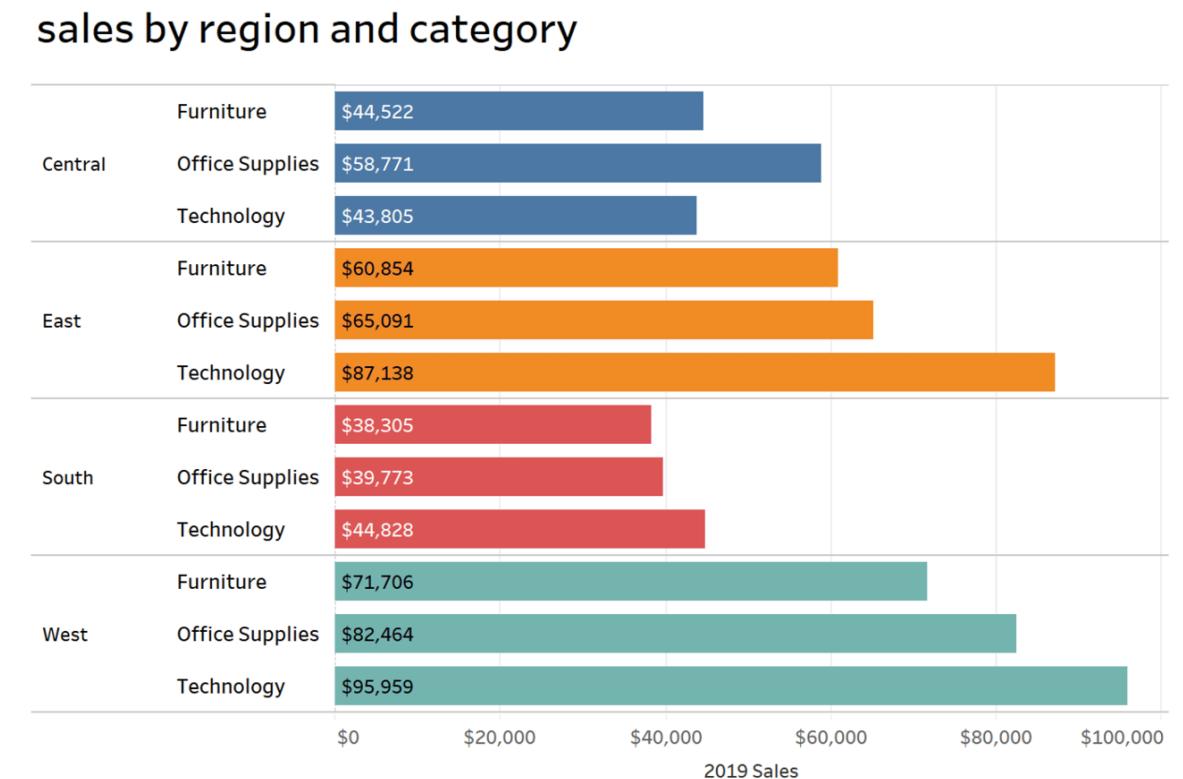
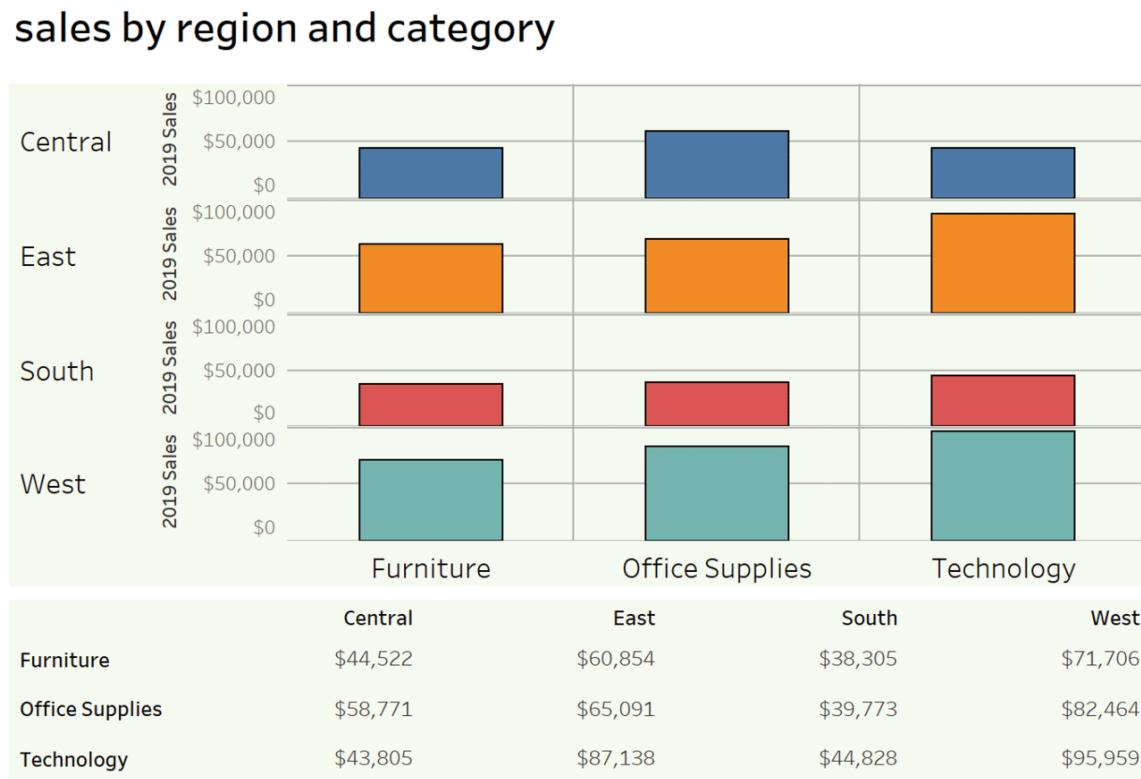
Month	Sales	
	Actual	Target
January	23	20
February	34	25
March	43	30
April	25	35
May	44	40
June	37	45
July	40	50
August	52	55
September	48	60



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

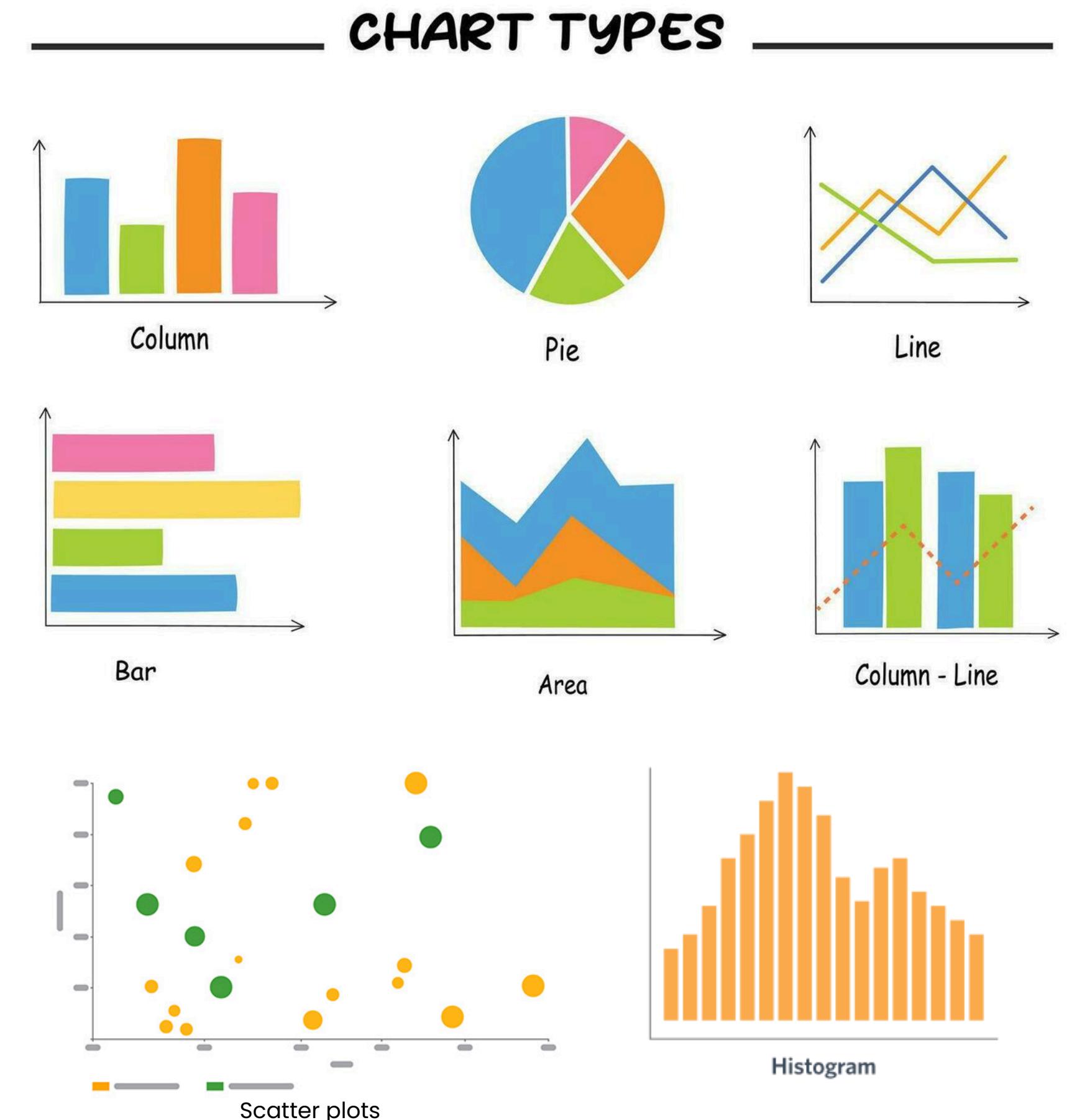


! ineffective

✓ 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

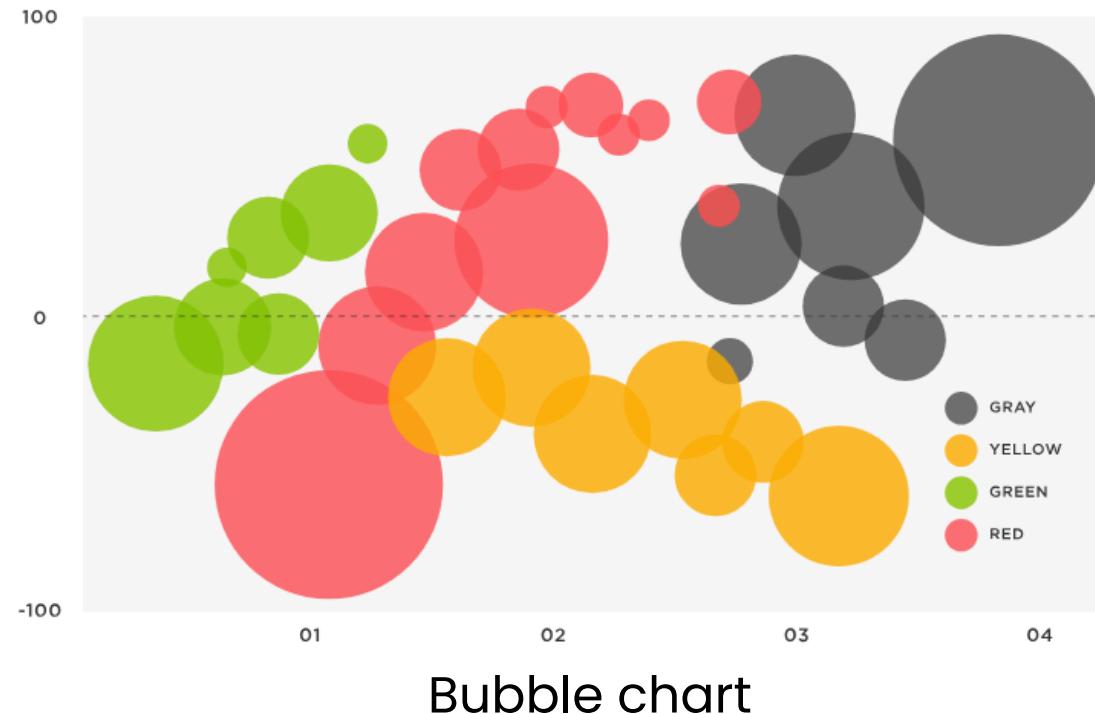


Advanced Chart Types

- 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

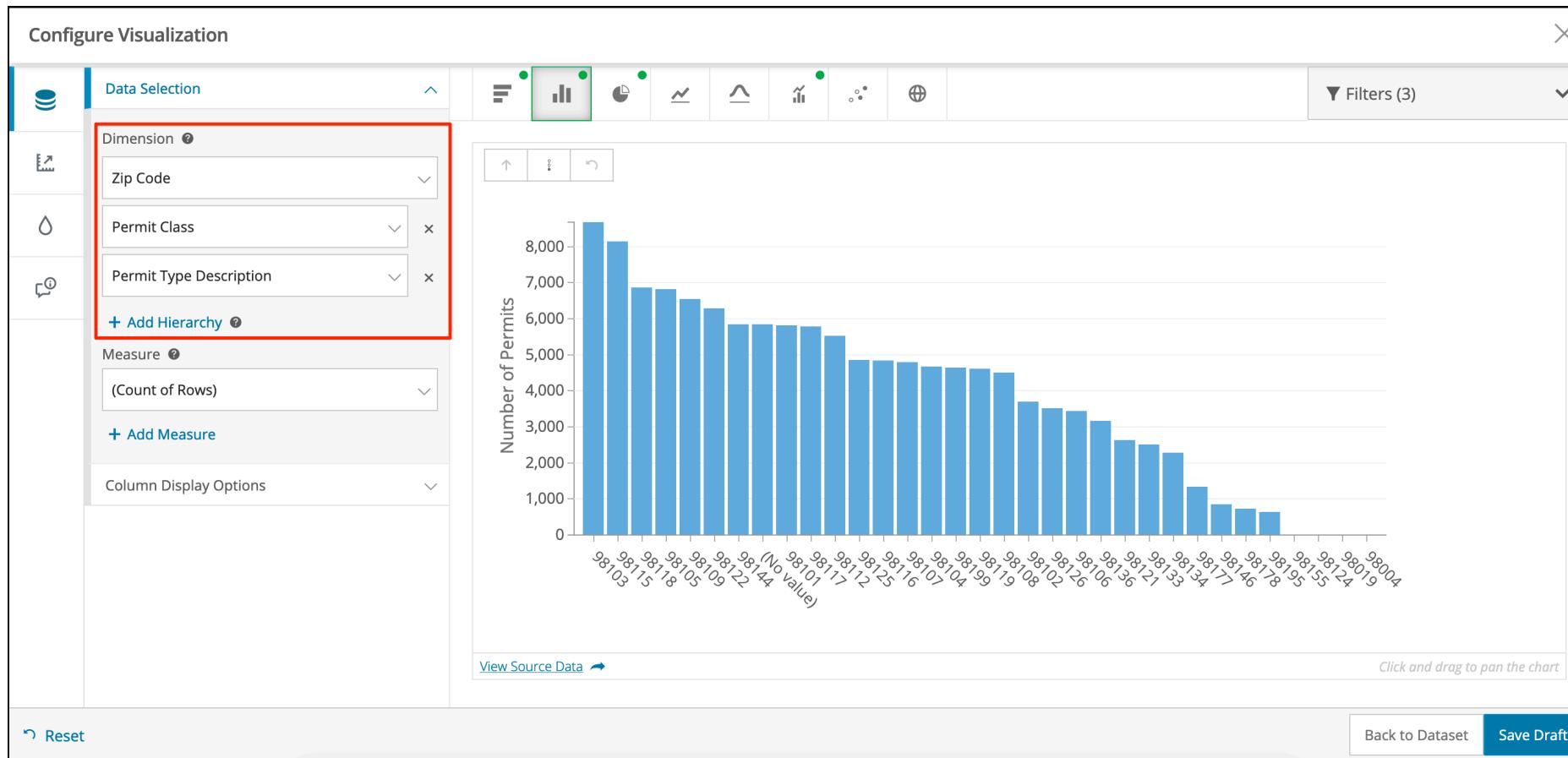
Based on:

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

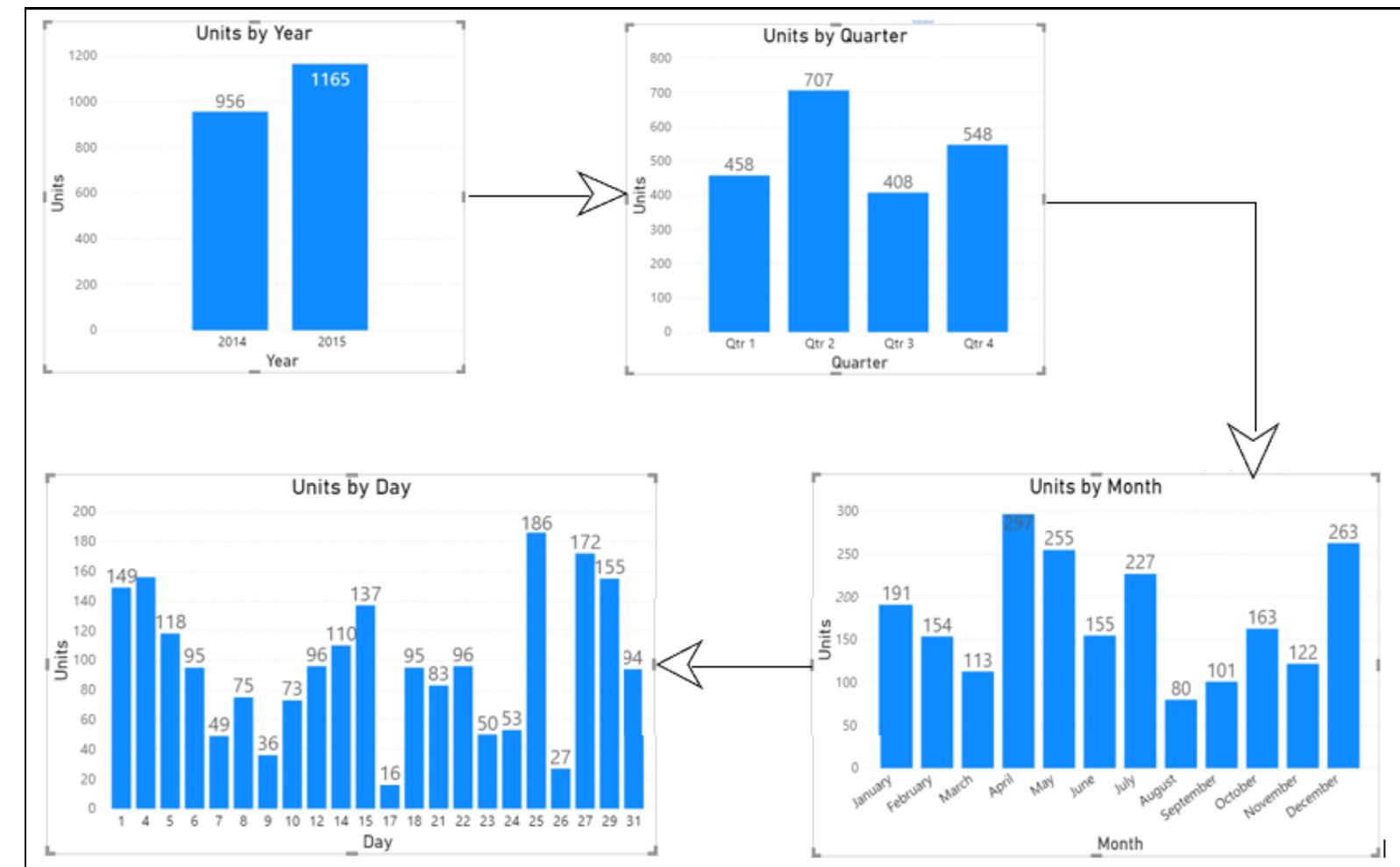
CHART TYPE	DATA TYPE
Line Chart 	Continuous or discrete
Bar Chart 	Discrete
Scatter Chart 	Continuous
Bubble Chart 	Relationship between three variables identifying trends
Histogram 	Distribution Analysis
Heatmap 	Density or distribution of data

Interactivity in Data Visualization

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



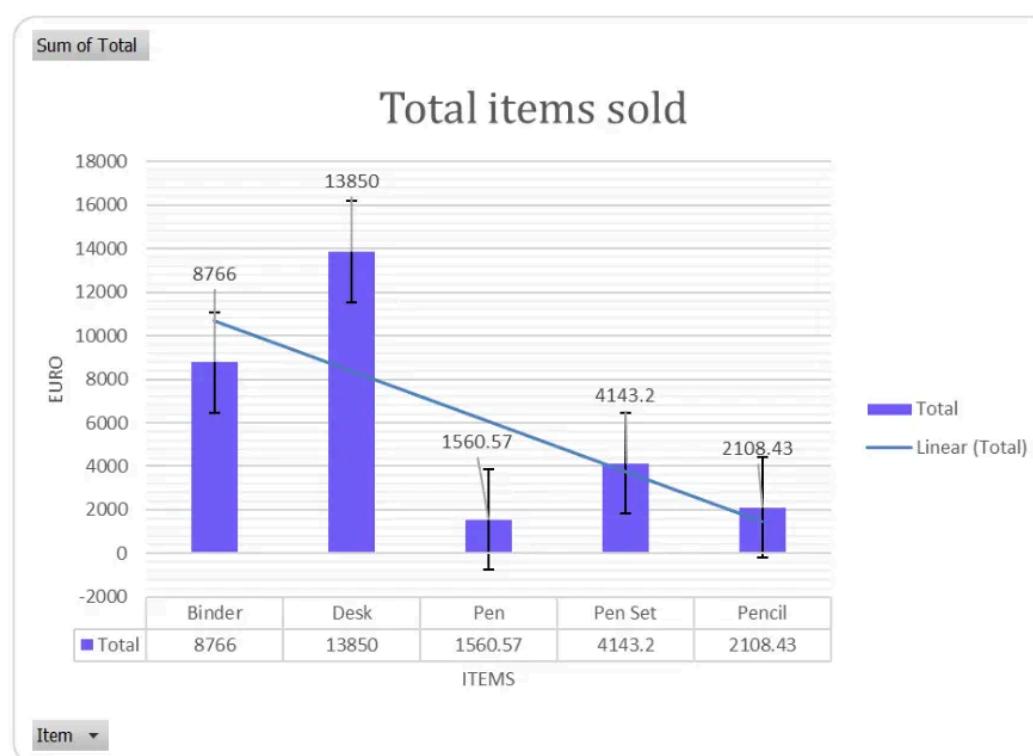
Filtering



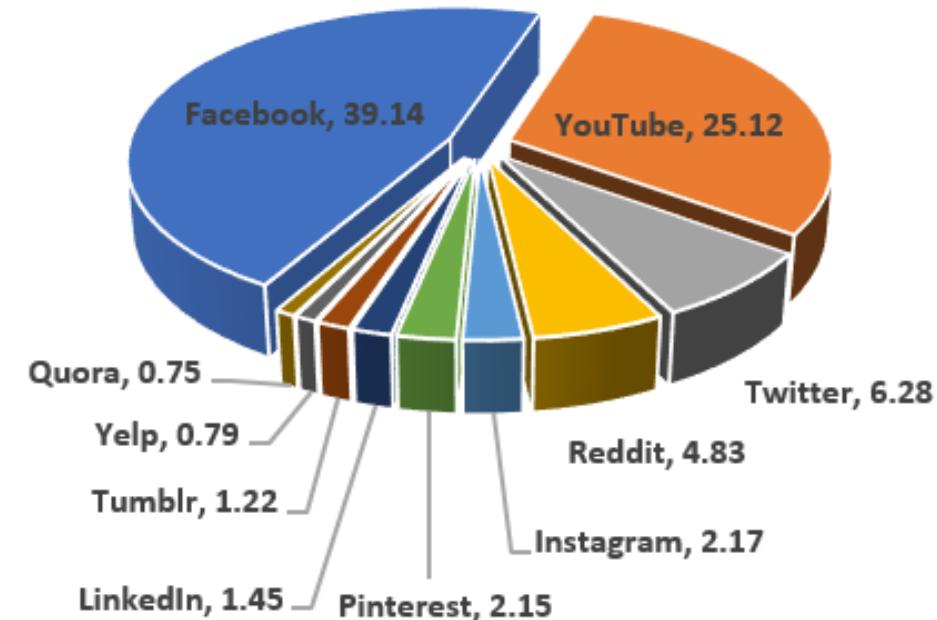
Drilling down

Common Pitfalls in Data Visualization

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



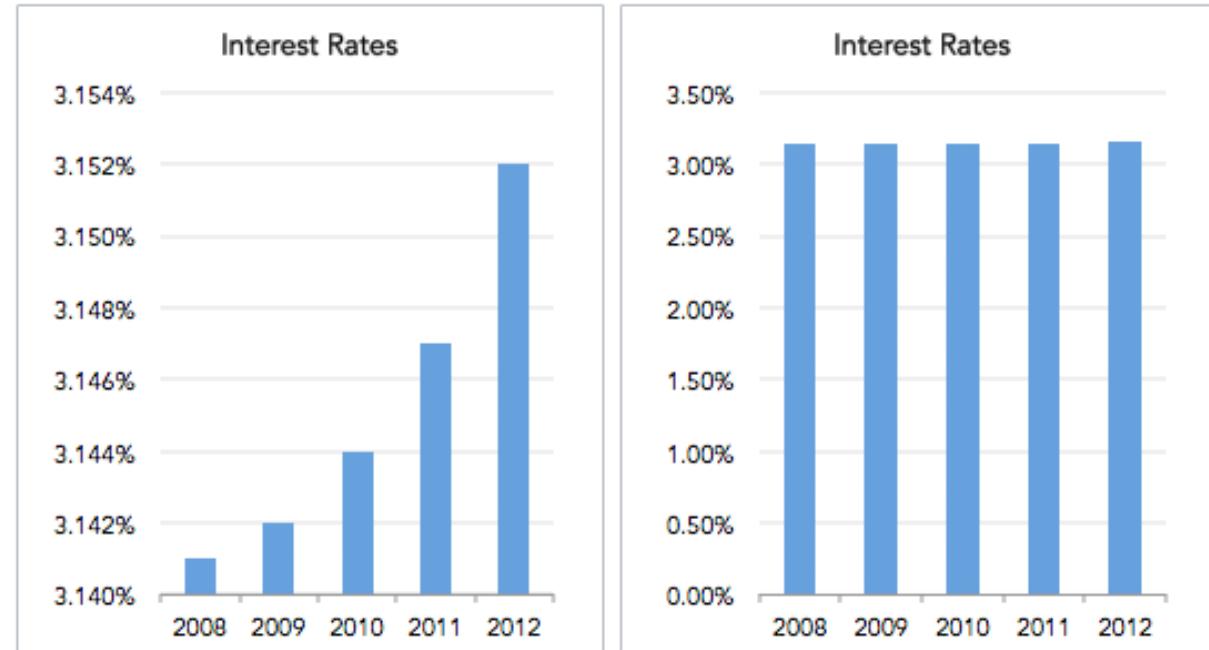
Cluttered designs



Overuse of pie charts

3D Charts

Same Data, Different Y-Axis



Misleading scales

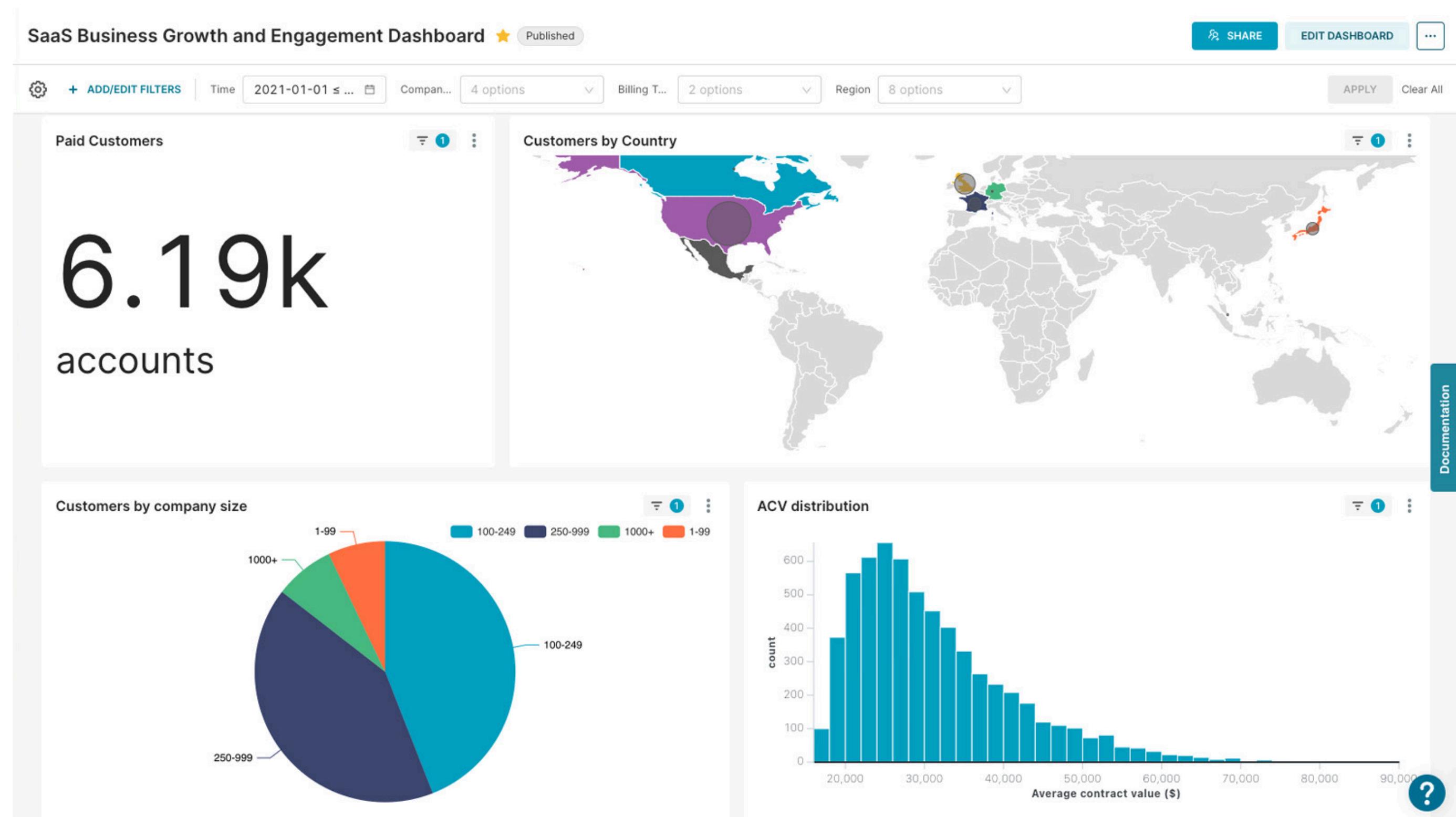
Data Visualization Tools: PowerBI & Superset

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.

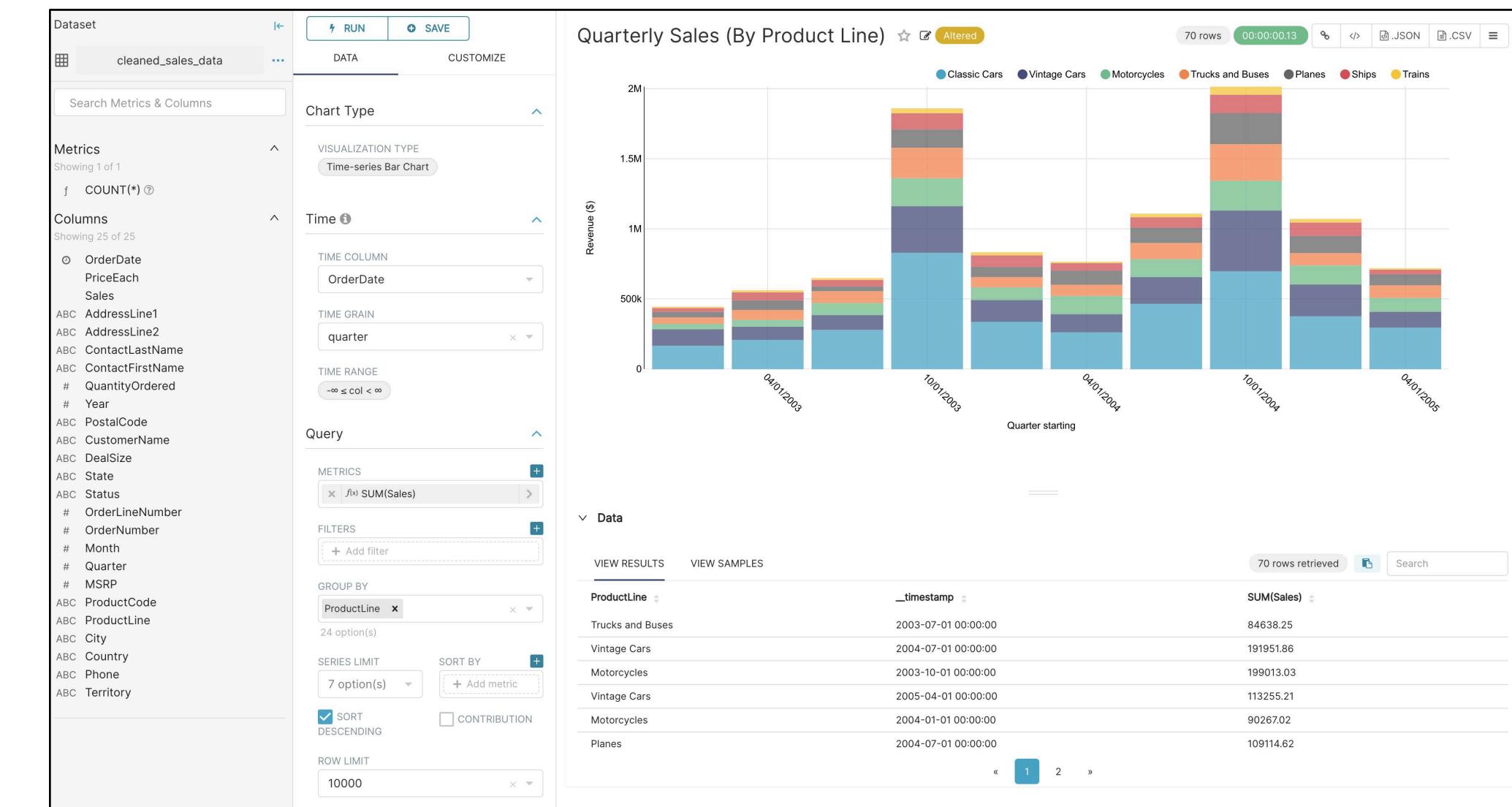
Introduction to Apache Superset



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.

The screenshot shows the Apache Superset SQL editor interface. On the left, there are dropdown menus for 'Schema' (set to 'community-data-bq'), 'Table' (empty), and 'View' (empty). The main area contains a code editor with the following SQL query:

```
1 with dates as (
2   SELECT cast(day as timestamp) as day
3     FROM UNNEST(
4       GENERATE_DATE_ARRAY(DATE('2016-01-01'), CURRENT_DATE(), INTERVAL 1 DAY)
5     ) as day
6   )
7
8   select dates.day, pr.user_login, pr.author_association, count(*) open_prs
9     from dates
10    inner join prod_core.github_pull_requests pr on
11      dates.day >= pr.created_dt and (dates.day <= pr.closed_dt or pr.closed_dt is null)
12    where pr.repository = 'apache/superset'
13    group by dates.day, pr.user_login, pr.author_association
14
```

Below the code editor are buttons for 'RUN' (highlighted in blue), 'LIMIT: 1 000', and a timer '00:00:02.87'. A tooltip 'Explore the result set in the data HISTORY exploration view' points to the results area. At the bottom, there are buttons for 'EXPLORE', 'DOWNLOAD TO CSV', 'COPY TO CLIPBOARD', and a 'Filter results' dropdown. A message box at the bottom indicates '1000 rows returned' with the note 'The number of rows displayed is limited to 1000 by the dropdown.' The results table header includes columns: day, user_login, author_association, and open_prs.

Data Exploration

Hands-on:

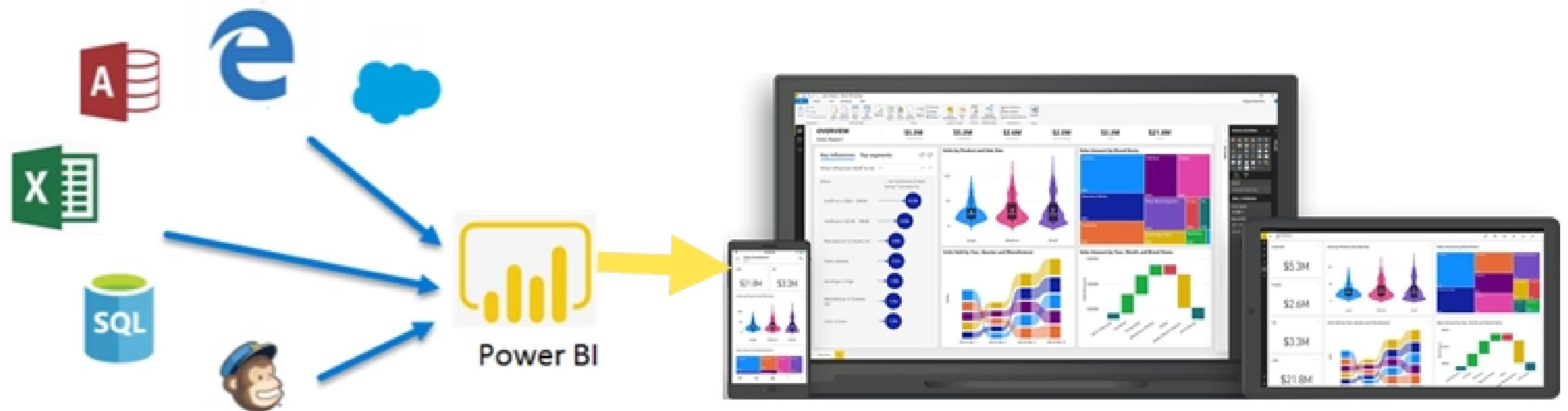
Exploring BI Tools: Apache Superset

Introduction to Power BI



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

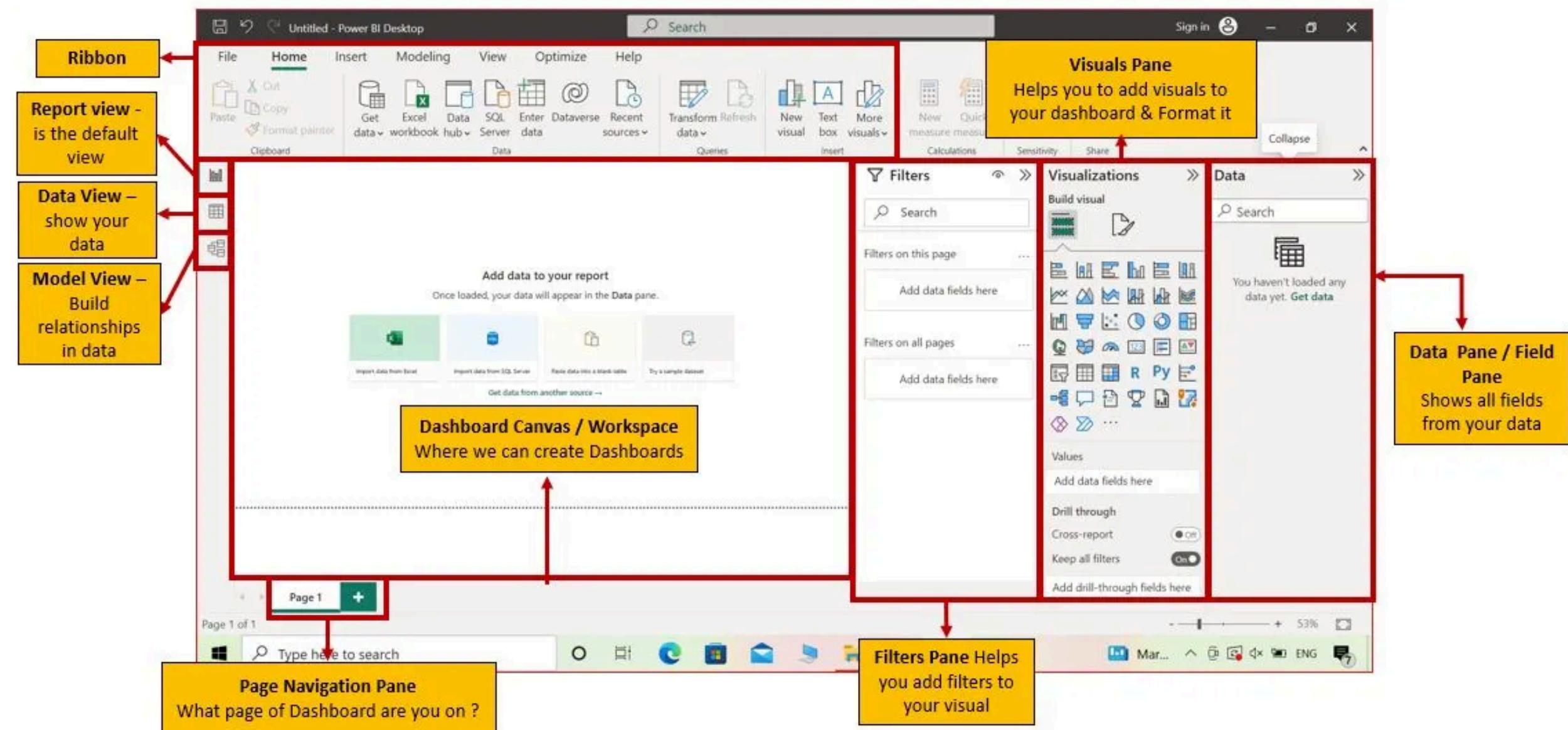


Power BI Interface



Power BI

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

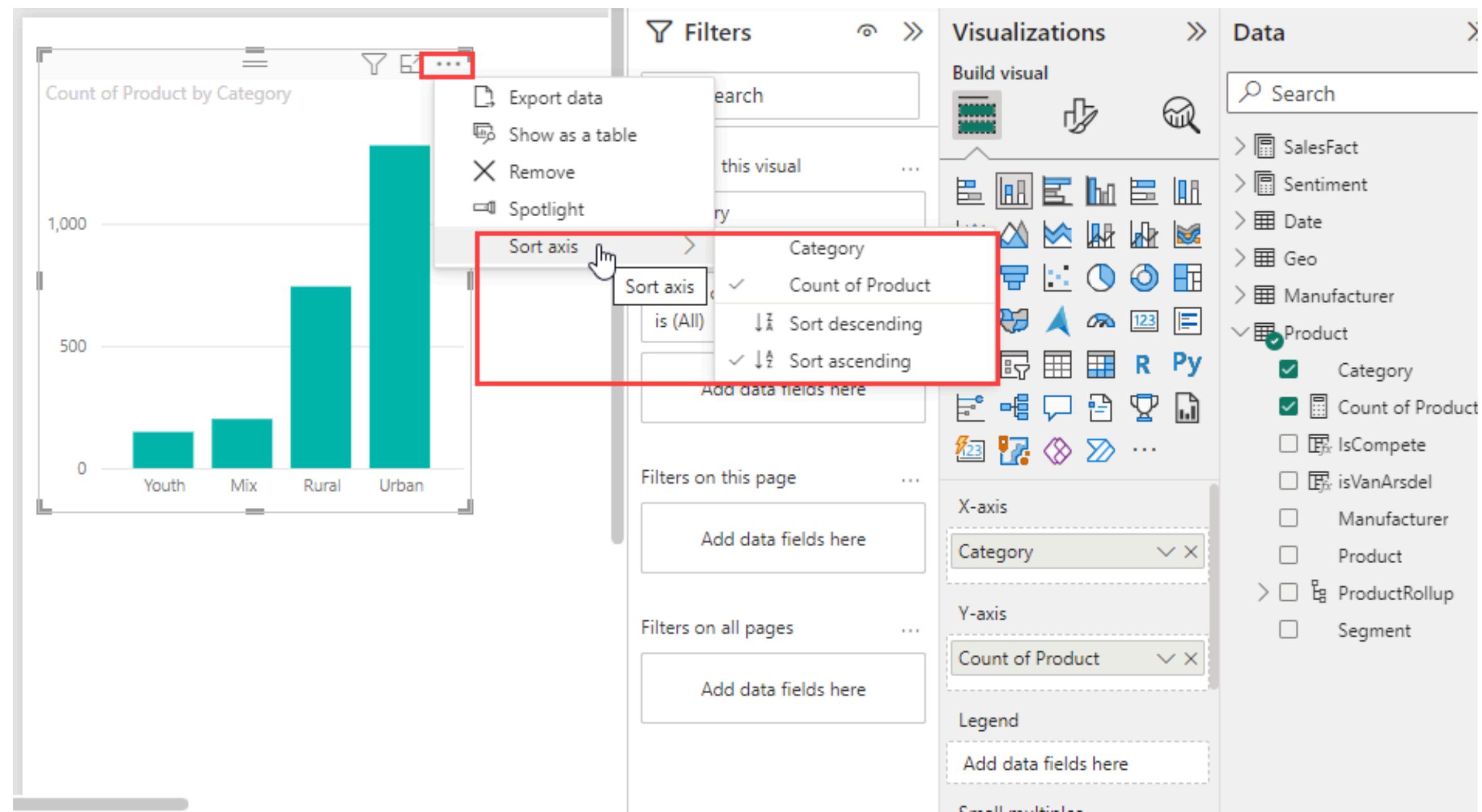


Creating Visualizations in Power BI



Power BI

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

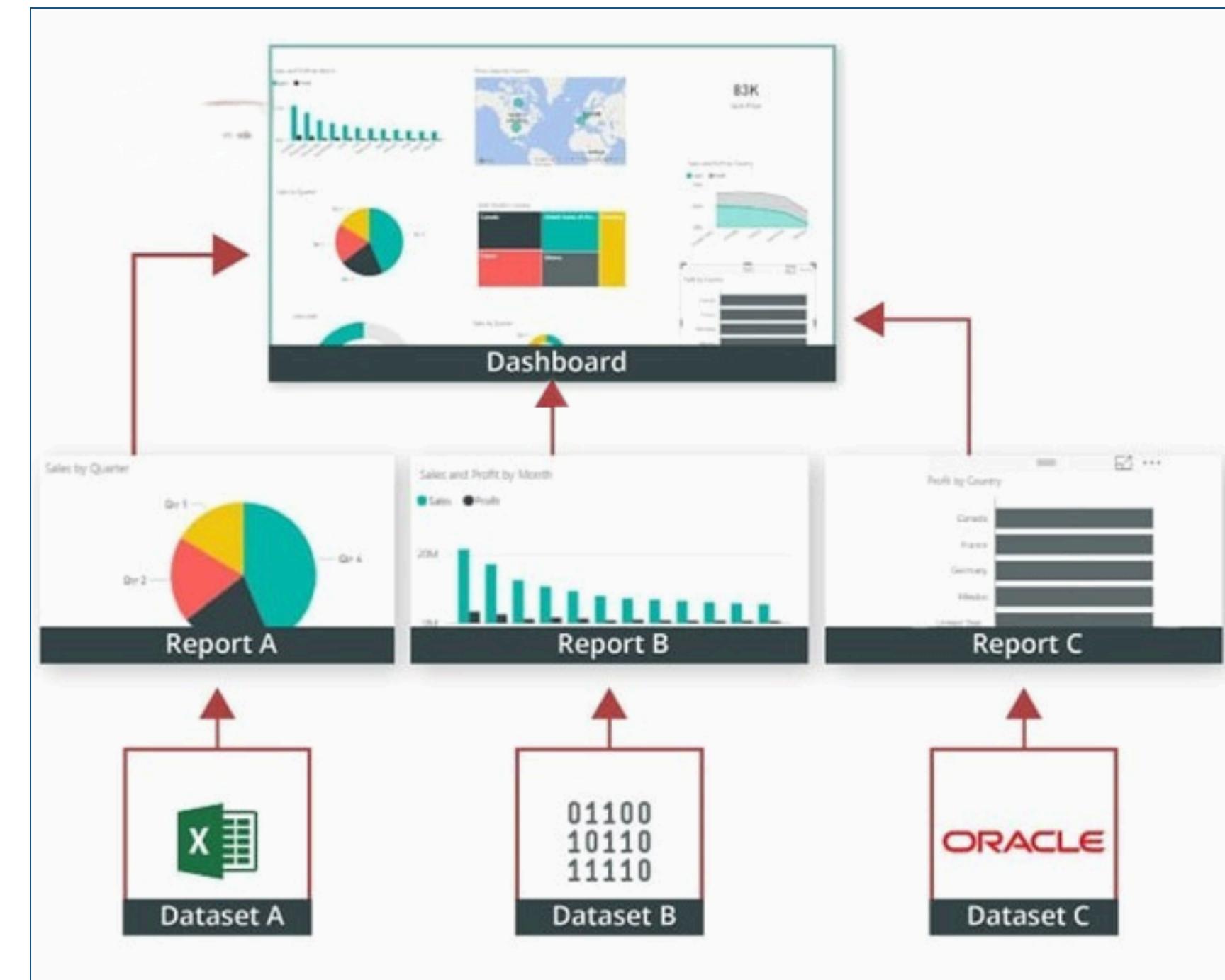
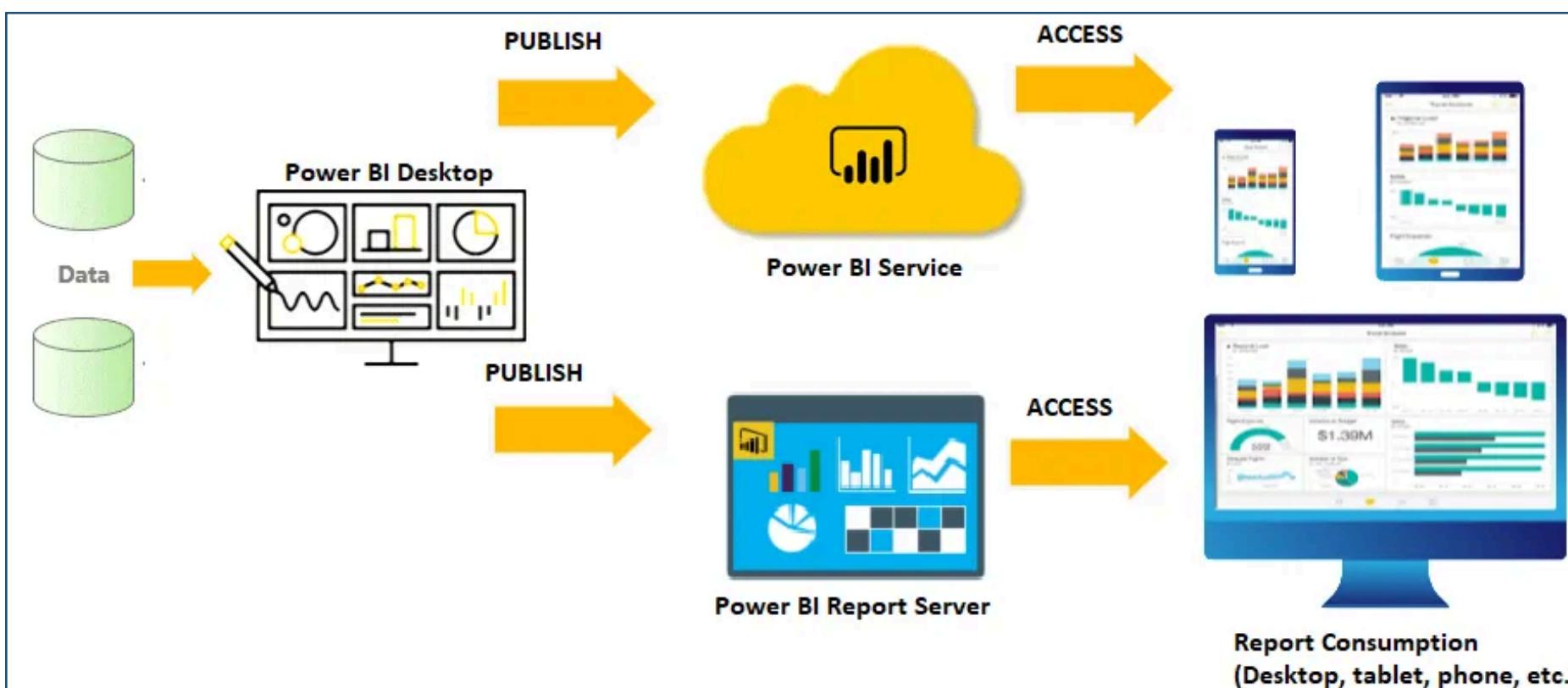


Power BI Dashboards and Reports



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



datechlabs.ai@gmail.com



Amman, Jordan