Mastering SQL for Data Science

Unlock the power of structured data

Duration: 1 hour

By

Ishaya, Jeremiah Ayock

Introduction to Data

Understanding the Foundation – Data

- **What is Data?**

  - Raw, unprocessed facts.
  - **Examples:** Numbers, text, images, videos.

- **Sources of Data**

  - Databases,
  - APIs,
  - IoT devices,
  - Social media,
  - Surveys.

- **Types of Data**

  - **Structured:** Databases (tables with rows and columns).
  - **Semi-Structured:** JSON, XML.
  - **Unstructured:** Images, videos, emails.

- **How is Data Generated?**

  - User interactions
  - Automated systems
  - Sensors.

From Data to Knowledge

The Journey : *Data* → *Information* → *Knowledge*

- **Data**: Raw, unprocessed facts.

- **Information**: Processed data that provides meaning.

  Example: Data = *Sales transactions*, **Information** = *Total revenue for a product.*

- **Knowledge**: Actionable insights derived from information.

  Example: Knowledge = **Increase inventory for a high-demand product**.

Databases

The core of Data Storage

- **What are Databases?**

- Organized collections of data for easy access, management, and updating.
- Used to store and retrieve data efficiently.

- **Relational Databases (RDBMS)**
- Data organized into rows and columns (tables).

Examples: MySQL, PostgreSQL, Oracle Database.

Features:

- Structured schema.
- ACID compliance for reliability.

Use Cases: Banking systems, e-commerce platforms.

- **Non-Relational Databases (NoSQL)**

- Flexible, unstructured data (documents, key-value, graph, or wide-column).

Examples: MongoDB, Cassandra, Redis.

Features:

- Scalability, flexible schema.

Use Cases: Social media data, IoT data, real-time analytics.

Notes:

- RDBMS to a traditional library (organized and structured) and NoSQL to a digital archive (flexible, adaptable).
- SQL’s evolution is also used toward querying non-relational databases using SQL-like tools.

Relational vs. Non-Relational Databases

Choosing the Right Database

- **Advantages of Relational Databases**
  - Data integrity, strong consistency.
  - Easy to use with SQL.
- **Disadvantages**
  - Limited scalability for big data.

- **Advantages of Non-Relational Databases**
  - Handles unstructured data.
  - High scalability.
- **Disadvantages**
  - Not ideal for transactions requiring ACID compliance.

OLAP Vs. OLTP

Understanding Database Systems

- **OLTP (Online Transaction Processing)**

- **Focus:** Operational, real-time transactions.

Example: Banking systems.

- **Advantages:** Real-time processing, low latency.
- **Disadvantages:** Not optimized for analytics.

- **OLAP (Online Analytical Processing)**

- **Focus:** Analytical queries, decision-making.

Example: Data warehousing for business intelligence.

- **Advantages:** Aggregates historical data for insights.
- **Disadvantages:** High latency for real-time transactions.

Technology Behind OLAP and OLTP

- **OLTP Technologies**: MySQL, PostgreSQL, SQL Server.
- **OLAP Technologies**: Snowflake, Amazon Redshift, Google Big Query.
-

Note: SQL’s versatility in both systems and its relevance to data science.

What is Data Science?

The Power Behind Modern Insights

- **Definition**:

The practice of extracting meaningful insights from data using statistical, programming, and machine learning techniques.

- **Importance**

- Driving business decisions.
- Automating processes.

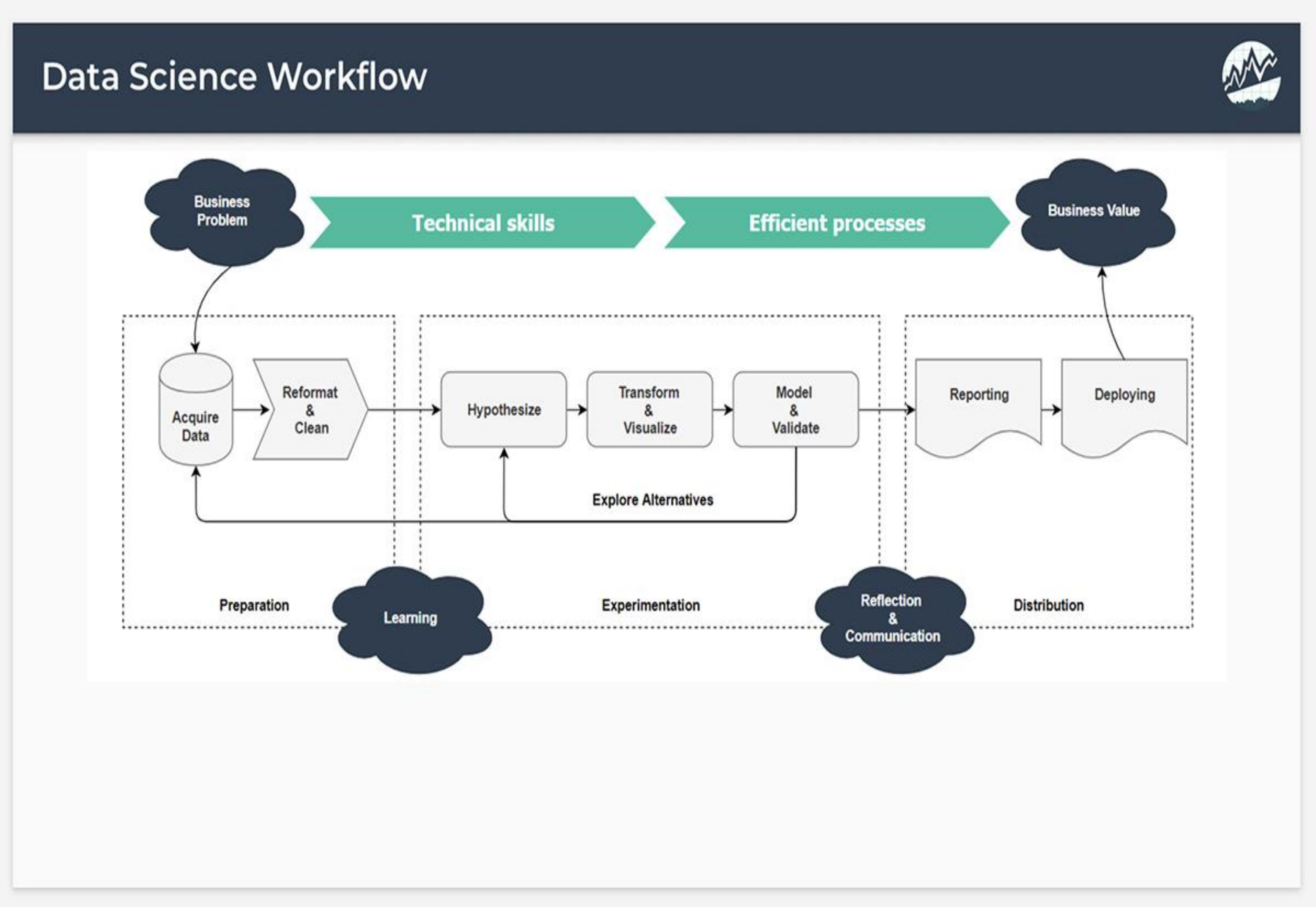
- **Applications**

- Predictive modelling in finance.
- Customer segmentation in retail.
- Fraud detection in banking.

Note: *Think about how Spotify suggests your favourite songs or how Uber predicts demand – "that' is Data Science in action".*

The Data Science Process

From Raw Data to Actionable Insights



- ❖ Problem Definition.
- ❖ Data Collection.
- ❖ Data Cleaning and Pre-processing.
- ❖ Exploratory Data Analysis (EDA).
- ❖ Modelling.
- ❖ Evaluation and Deployment.

Note: Focus on SQL’s role in the process: data extraction, cleaning, and integration.

SQl for Data Science

The Data Scientist’s Superpower

- **Why SQL for Data Science?**

- Most data resides in databases, making SQL essential for accessing and analysing it.

SQL is efficient for:

- Extract and Transform data for Analysis
- Summarize and aggregate data for Insights
- Connect to databases from analysis tools like Python, R, Tableau, or Power BI
- Clean and prepare data for Analysis

- **Use Cases of SQL in Data Science**

- Fetching data for machine learning models.
- Creating dashboards and reports.
- Exploratory Data Analysis (EDA).
- Joining datasets from multiple sources.

Notes: Over 80% of a data scientist's time is spent cleaning and preparing data?

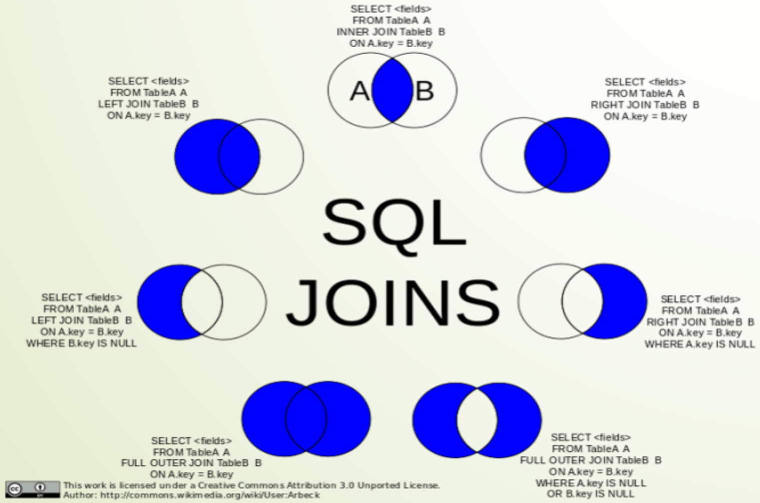
SQL makes this efficient and scalable.

Introduction to SQL for Data Science

The Basics that Matter

❖ **SQL Essentials**

- SELECT: Retrieve data.
- WHERE: Filter data.
- JOIN: Combine tables.
- GROUP BY: Aggregate data.
- ORDER BY: Sort data.



```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

❖ **Practical Examples**

- Fetch top-performing products.
- Summarize monthly sales revenue.

❖ SQL in Data Science Tools

- Integrated in tools like Python (via Pandas), R, Tableau, and Power BI.

Intermediate SQL for Data Science

Level Up Your SQL Skills

- **Advanced Concepts**

- **Subqueries:** Use queries within queries.
- **Window Functions:** Perform row-wise operations (e.g., rankings, running totals).
- **Common Table Expressions (CTEs):** Simplify complex queries.
- **Recursive Queries:** Solve hierarchical problems.

- **Use Cases**

- Identify top customers based on lifetime spending (Window Functions).
- Extract monthly growth trends (CTEs).

- **Best Practices**

- Optimize queries for performance.
- Avoid **SELECT *** in production queries

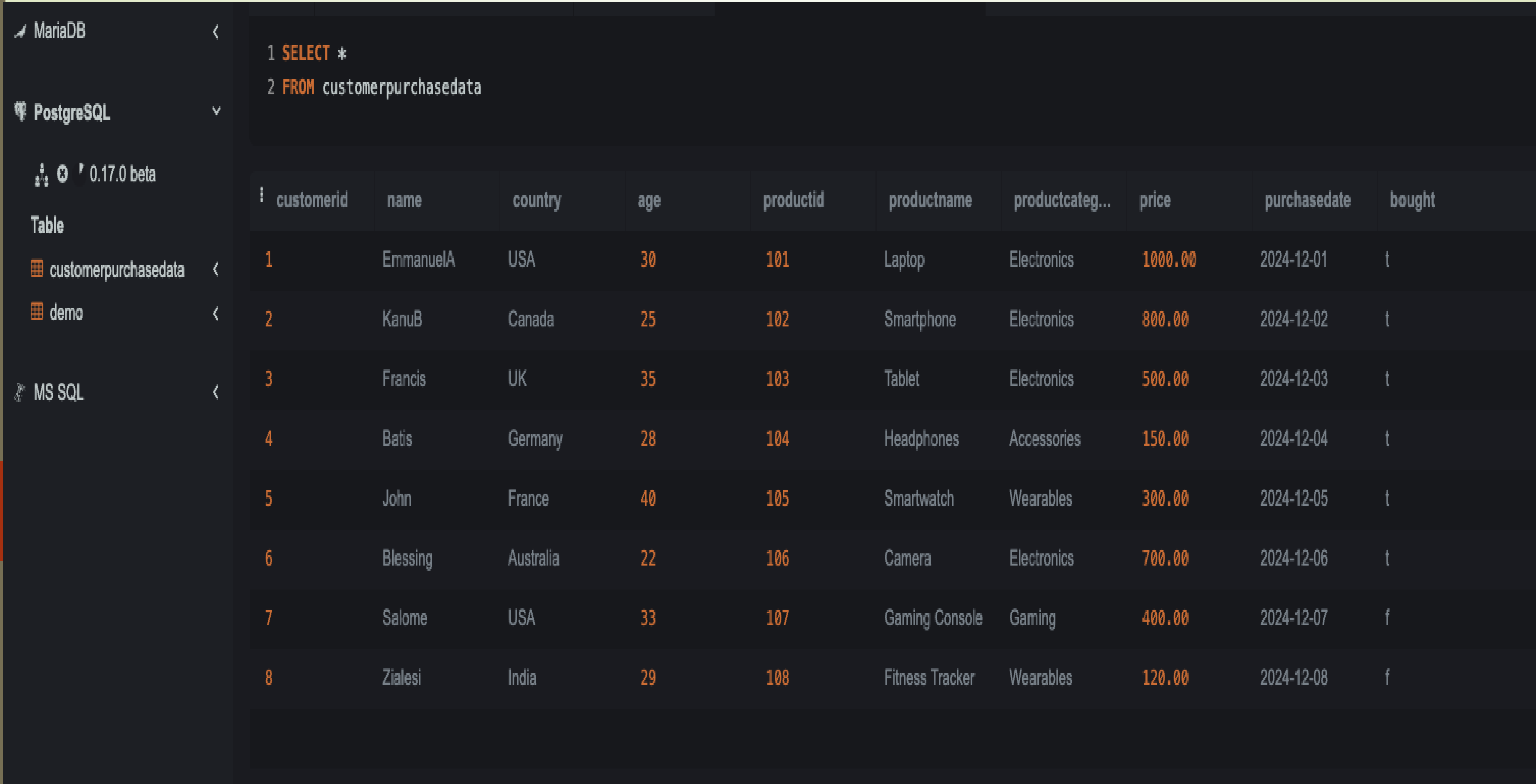
SQL Place in Data Science Project

Seamlessly Integrating SQL

- **Data Preparation:**
  - Cleaning datasets.
  - Aggregating data for insights.
- **Model Input:**
  - Joining and transforming datasets.
  - Exporting data for ML models.
- **Data Visualization:**
  - SQL as a backend for Tableau, Power BI, or custom dashboards.
- Real-Time Analytics
  - Streaming queries for live dashboards.

Note: From preprocessing to live dashboards, SQL powers every stage of data-driven decision-making.

DataSet



```
SELECT *
FROM customerpurchasedata
```

customerid	name	country	age	productid	productname	productcateg...	price	purchasedate	bought
1	EmmanuelA	USA	30	101	Laptop	Electronics	1000.00	2024-12-01	t
2	KanuB	Canada	25	102	Smartphone	Electronics	800.00	2024-12-02	t
3	Francis	UK	35	103	Tablet	Electronics	500.00	2024-12-03	t
4	Batis	Germany	28	104	Headphones	Accessories	150.00	2024-12-04	t
5	John	France	40	105	Smartwatch	Wearables	300.00	2024-12-05	t
6	Blessing	Australia	22	106	Camera	Electronics	700.00	2024-12-06	t
7	Salome	USA	33	107	Gaming Console	Gaming	400.00	2024-12-07	f
8	Zialesi	India	29	108	Fitness Tracker	Wearables	120.00	2024-12-08	f

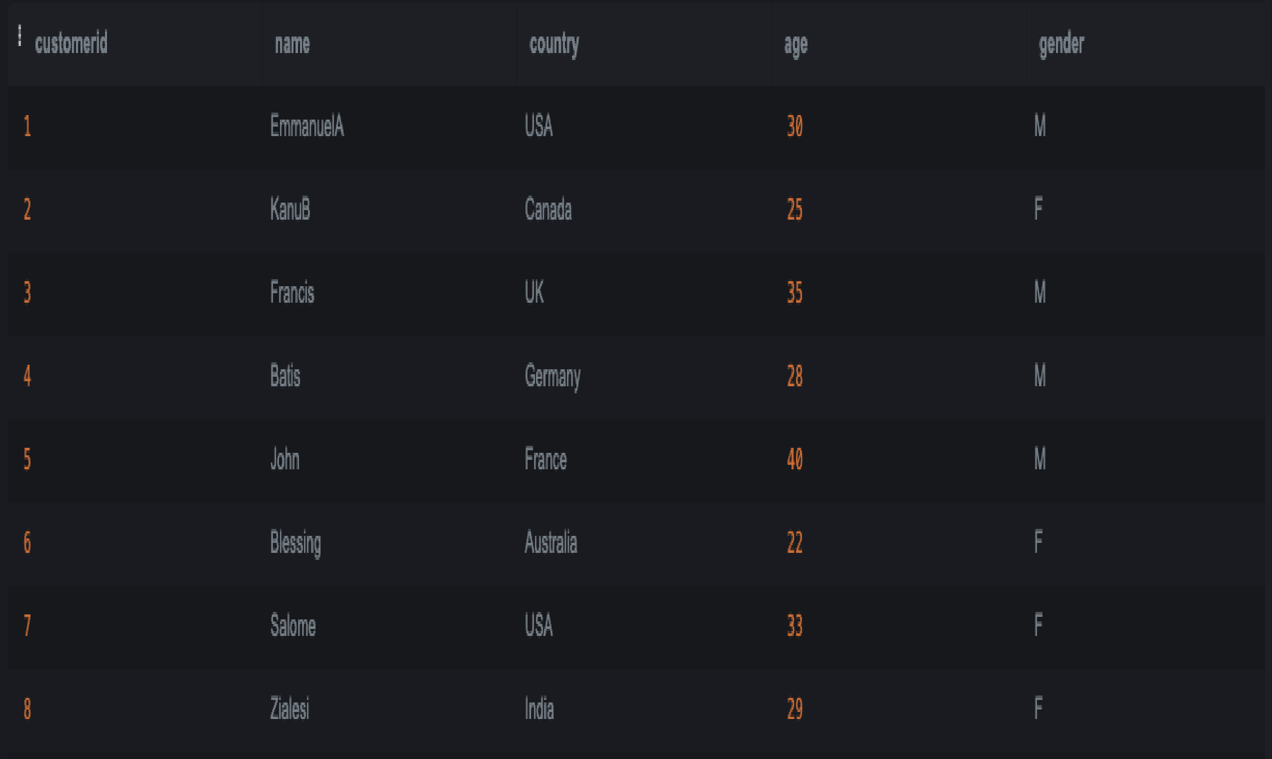
- **Benefits of Normalization**

- **Eliminates Redundancy**: Customer and product information are stored only once in their respective tables.
- **Improves Data Integrity**: Changes in customer or product data need to be updated in only one place.

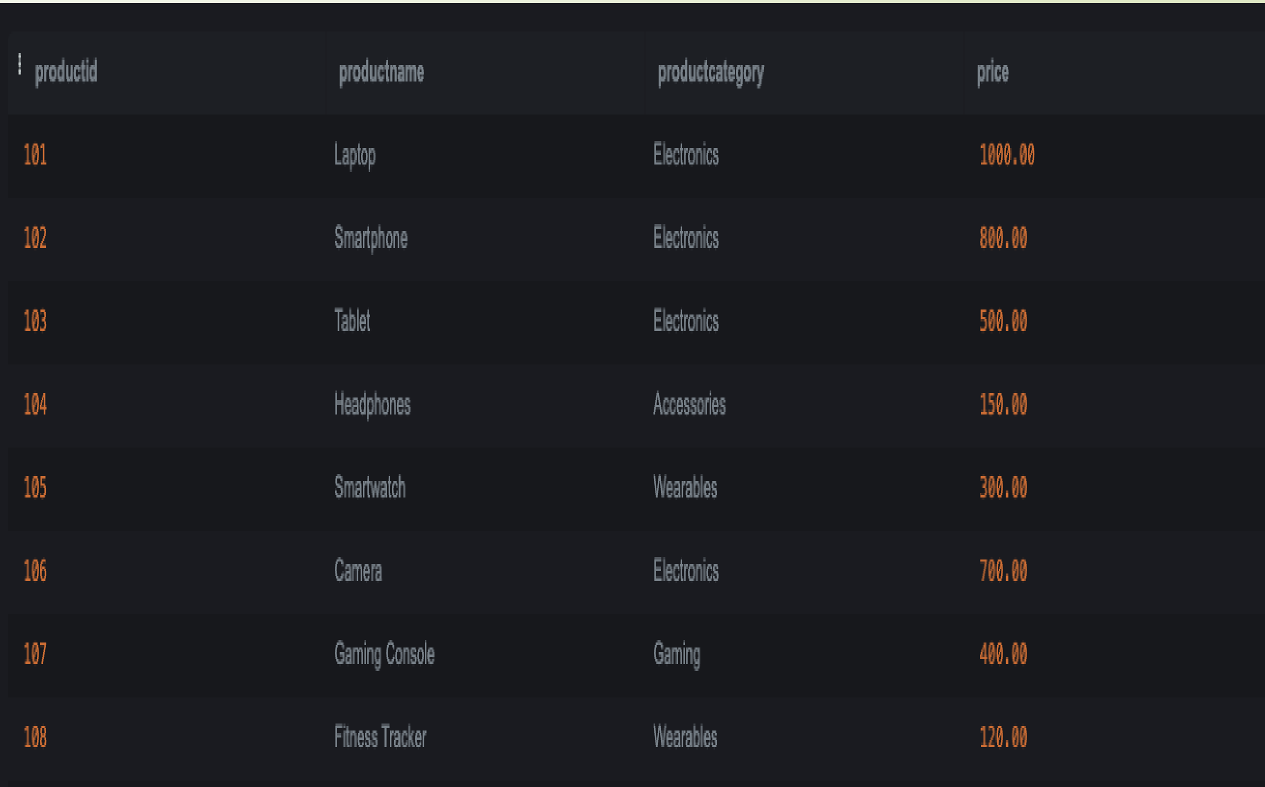
- **Enhances Query Performance**: Joining smaller tables on specific keys is often more efficient than querying a large flat file.

- **Approaches Followed**

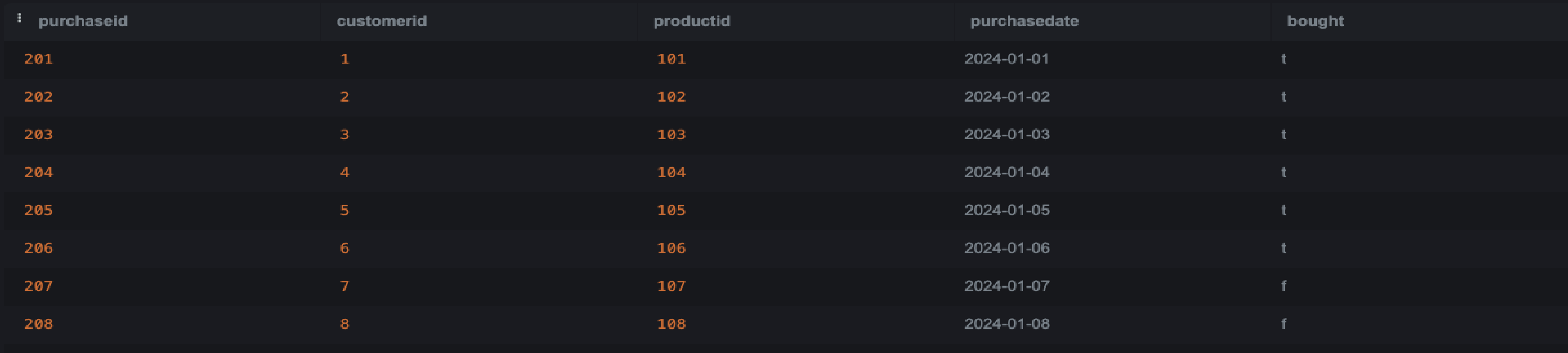
1. **Entity Identification**: Recognizing ***Customers, Products, and Purchases*** as distinct entities.
2. **Primary Keys**: Assigning unique identifiers (e.g., ***CustomerID, ProductID, PurchaseID***) for each table.
3. **Foreign Keys**: Establishing relationships (e.g., ***CustomerID and ProductID*** in the Purchases table).
4. **Data Integrity**: Ensuring consistency by linking the tables with well-defined keys.
5. **Query Optimization**: Making it easy to retrieve meaningful data using joins.



customerid	name	country	age	gender
1	EmmanuelA	USA	30	M
2	KanuB	Canada	25	F
3	Francis	UK	35	M
4	Batis	Germany	28	M
5	John	France	40	M
6	Blessing	Australia	22	F
7	Salome	USA	33	F
8	Zialesi	India	29	F



productid	productname	productcategory	price
101	Laptop	Electronics	1000.00
102	Smartphone	Electronics	800.00
103	Tablet	Electronics	500.00
104	Headphones	Accessories	150.00
105	Smartwatch	Wearables	300.00
106	Camera	Electronics	700.00
107	Gaming Console	Gaming	400.00
108	Fitness Tracker	Wearables	120.00



purchaseid	customerid	productid	purchasedate	bought
201	1	101	2024-01-01	t
202	2	102	2024-01-02	t
203	3	103	2024-01-03	t
204	4	104	2024-01-04	t
205	5	105	2024-01-05	t
206	6	106	2024-01-06	t
207	7	107	2024-01-07	f
208	8	108	2024-01-08	f

Summary and Key Takeaways

SQL for Data Science: Master the Essentials

- **Databases** are the **backbone** of **data storage**; understanding **relational** and **non-relational** systems is crucial.
- **SQL** is indispensable for **data extraction, cleaning, and preparation.**
- Start with basic SQL commands and progress to advanced features for complex data manipulations.
- SQL empowers data scientists to work efficiently with structured data and integrate it into data science workflows.

Note

- ✓ SQL is not just a tool for data engineers; it is a data scientist’s secret weapon.
- ✓ I encourage you all to practice writing queries on open datasets or platforms like Kaggle.