week01: Intro to Data Engineering & Data Lake

Introduction: Objective

- **Empowering Smart Manufacturing:** Revolutionizing precision with data platforms and AI-driven solutions.
- **From Data to Decisions:** Leveraging SECS/GEM, MES, and ERP for actionable insights.
- **JamAI & LLM Integration:** Transforming workflows with real-time intelligence and innovation.
- **Collaborative Excellence:** Partnering to enhance efficiency, quality, and scalability in manufacturing.

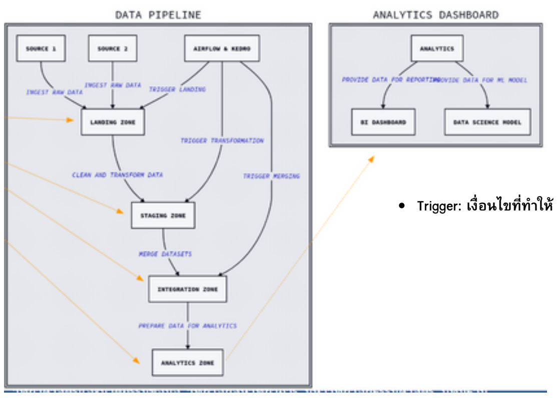
Importance of data platforms and AI in manufacturing

- **Enhanced Decision-Making:** Real-time analytics turn raw data into actionable insights for efficiency and quality improvements.
- **Predictive Maintenance:** Minimize downtime with AI-driven anomaly detection and machine performance forecasting.
- **Integrated Workflows:** Data platforms unify SECS/GEM and MES for seamless production monitoring.
- **Success Story:** WestRock leverages AI to develop smart packaging, reducing waste and enhancing sustainability.

Architecture and components of a modern data platform

- **Centralized Data Integration:** Combines SECS/GEM, MES, and ERP Systems into a unified repository for real-time insights.
- **Advanced Analytics Tools:** Supports DuckDB, Dask, and Ibis for scalable processing and predictive modeling.
- **Real-Time Monitoring:** Tracks sensor data and alerts for immediate issue resolution.
- **Example:** StoreMesh powers actionable insights with real-time anomaly detection and visual querying.

Data Lake Zone

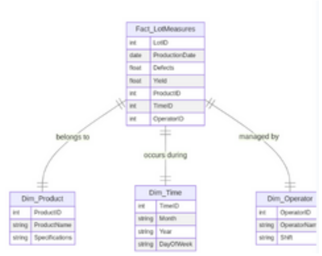


- **Trigger: เงื่อนไขที่ทำให้ Pipeline เริ่มทำงาน**

- **Landing Zone (Raw Zone):**
  - พื้นที่แรกที่ขข้อมูลถูกนำเข้ามา ดิบ 100% ยังไม่ถูกปรับแต่ง มักเก็บในรูป original format เพื่อให้แน่ใจว่ามี raw backup
- **Staging Zone (Cleansing Zone / Processing Zone)**
  - พื้นที่สำหรับ เตรียมข้อมูล (Data Preparation) ทำความสะอาดข้อมูลและแปลงข้อมูล (Cleaning and Transformation)
- **Integration Zone (Curated Zone / Conformed Zone)**
  - เป็นโซนที่ข้อมูลจากหลายแหล่งถูก รวม (integrate) และจัดให้อยู่ในรูปมาตรฐานเดียวกัน มีการออกแบบ schema แบบ conformed dimensions เพื่อให้ข้อมูล cross-system ใช้ร่วมกันได้
- **Analytics ZOne (Consumption Zone / Applicatioon Zone)**
  - เป็นพื้นที่ที่ข้อมูลถูก ปรับแต่งเพื่อการใช้งานเชิงวิเคราะห์จริง
    - Data Mart: subset ของ Data warehouse ที่โฟกัสเฉพาะเรื่อง (domain)
    - BI (Business Intelligence): การสร้าง Dashboard, Reports, KPI Tracking
    - ML (Machine Learning / Data Science): โซนสำหรับนักวิทยาศาสตร์ข้อมูลที่ใช้ในการ Train / Test / De[loy Models

Fact tables and dimensions

- **Fact Tables:** Capture transactional data, e.g., defect rates, sensor readings, and production logs.
- **Dimension Tables:** Provide context, such as machines, operators, time, and components.
- **Optimized Analytics:** Enable roll-ups and drill-downs for insights into operations and performance.



- **Fact Table:** เก็บ เหตุการณ์หรือธุรกรรมที่เกิดขึ้นจริงและมีการวัดได้ (measurable data) เช่น จำนวน, ยอดขาย
  - เก็บ metricsc หรือ measures
  - มักมี foreign key เชื่อมไปที่ Dim tables เพื่ออธิบาย context ของ fact
  - มักมี record มากที่สุดในระบบ
- **Dimension Table:** เก็บ รายละเอียดหรือคุณลักษณะ ของ entity ที่ใช้ในการอธิบาย fact เช่น ลูกค้า, สินค้า, เวลา
  - ให้ context กับข้อมูล fact table
  - เก็บข้อมูลที่เป็น ข้อความ/คำอธิบาย (descriptive data)
  - มักที่ primary key ที่ fact จะอ้างอิงมา

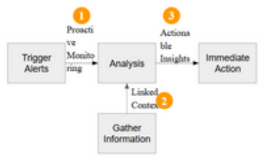
Roll-up vs Drill-down

- **Roll-up (การสรุปขั้นสูง / การย่อระดับ)**
  - การรวม (aggregate) ข้อมูลจาก ระดับรายละเอียด (detailed level) ไปยัง ระดับที่สูงขึ้น (summarized level) ตามมิติที่สนใจ
  - เช่น จากยอดขายรายวัน → สรุปเป็นยอดขายรายเดือน → ยอดขายรายปี
- **Drill-down (การเจาะลึก / การขยายรายละเอียด)**
  - การเจาะลึกจากระดับสูง (summary level) ลงไปยัง ระดับรายละเอียด (detailed level) เพื่อดูข้อมูลเชิงลึก
  - เช่น จากยอดนายรวมทั้งประเทศ → Drill-down ดูเป็นยอดขายรายภูมิภาค → รายจังหวัด → รายสาขา

Session 2: Real-Time Data Insights

Understanding real-time data ingestion and alerting

- **Seamless Data Flow:** Capture live sensor readings from machines into Fact_SensorData.
- **Trigger Alerts:** Automatically detect anomalies with predefined thresholds in Fact_Alerts.
- **Immediate Action:** Link alerts to Dim_Machine and Dim_AlertType for rapid troubleshooting.



- **Proactive Monitoring (การเฝ้าระวังเชิงรุก)**
  - การติดตามและตรวจสอบ/ข้อมูลอย่างต่อเนื่อง ก่อนที่ปัญหาจะเกิดขึ้นหรือส่งผลกระทบ โดยไม่รอให้เกิดขึ้นก่อน
- **Linked Context (การเชื่อมโยงบริบท)**
  - การเชื่อมโยงข้อมูลหรือเหตุการณ์ที่เกี่ยวข้องกัน เพื่อให้เห็น ภาพรวมและความสัมพันธ์ของปัญหา/ข้อมูล แทนที่จะดูเหตุการณ์แบบแยกส่วน
- **Actionable Insights (ข้อมูลเชิงลึกที่นำไปปฏิบัติได้)**
  - การวิเคราะห์ข้อมูลเพื่อให้ได้ ข้อสรุปที่สามารถลงมือทำได้จริง (ชี้แนวทาง action)

Performing Basic Exploratory Data Analysis (EDA)

The goal of EDA is summarize and understand the main characteristics of the data, ofter through visual methids and statistics

Initial Data Structure and Type Inspection

- .shape: Returns a tuple representing the number of rows and columns, giving a sense of dataset's scale.
- .columns: Lists the names of all columns in the DataFrame.
- .dtypes: Returns a Series with the data type of each column
- .info(): Provides a concise summary of DataFrame, columns names, data types, the number of non-null values, abd memory usage.
- .unique(): This method returns an array of all distinct
- .nunique(): This method returns a single integer count of the nuber of unique values. (faster and more memory-efficient)
- .isnull(): method returns a DataFrame of the same shape, boolean values indicating True for nulls and False for valid data.

Week02: SQL Data Sources & Ingestion

Introduction: The Foundation of Data Ingestion

The ability to effectively access and utilize data is the cornerstone of any data science project. in most professional environments, data is not stored in simple flat files like CSVs but rather in sophisticated relational databases. Necessary to bridge the gap between these databases and analytical power of Python's data ecosystem.

Data Ingestion: คือกระบวนการนำข้อมูลจากแหล่งต่างๆ (data soruces) เข้ามาเก็บในระบบศูนย์กลาง เช่น Data Lake, DW หรือ Database เพื่อสามารถให้สามารถนำไปใช้งาน วิเคราะห์ หรือทำ ML ต่อได้

A Toolkit for Structured Data: Python, Pandas, and SQL Alchemy

- **The pandas library** is the central workhorse for data manipulation and analysis, providing the DataFrame data structure that is ideal for working with tabular data
- **sqlite3 library** is Python's built-in connector for SQLite databases, which are lightweight, file-based databases excellent for local development and tutorials.
- **SQLAlchemy** is its ability to handle database sessions and connections efficiently, which prevents common errors and resource leaks. Furthermore, SQLAlchemy standardizes the connection process, allowing a practitioner to switch between different database systems like SQLite, PostgreSQL, or MySQL, simplt by changing a single connection string. This portability is a key advantage, making the code more adaptable and maintainable for a wide range of projects.

An Overview of the Northwind and Chinook Schemas

- **Northwind and Chinook Schemas**
  - offer a rich set of tables with distinct business contexts. The Northwind database mdels a wholesale food company, featuring sales and logistic-related tables like Customers, Orders, and Products. The Chinook database is designed to model a digital music store, with tables for artists, albums, and tracks

- **Table 1: Northwind Database Schema Overview**

Table Name	Purpose	Key Columns
Categories	Stores product categories (e.g., Beverages, Seafood)	CategoryID, CategoryName
Customers	Contains information about the company's customers	CustomerID, CustomerName, Country
Employees	Holds employee data	EmployeeID, LastName, FirstName, BirthDate
OrderDetails	Line items for each order, linking orders and products	OrderID, ProductID, Quantity
Orders	Main sales orders table	OrderID, CustomerID, EmployeeID, OrderDate, ShipperID
Products	Lists all products sold	ProductID, ProductName, SupplierID, CategoryID, Price
Shippers	Details about the shipping companies	ShipperID, ShipperName
Suppliers	Information on product suppliers	SupplierID, SupplierName

- **Categories:** Record 8, Features 4
- **Customers:** Record 91, Features 10
- **Employees:** Record 9, Features 13
- **OrderDetails:** Record 2,000+, Features 5
- **Orders:** Record 830, Features 14
- **Products:** Record 77, Features 10
- **Suppliers:** Record 29, Features 10
- **Shippers:** Record 3, Features 3

- **Table 2: Chinook Database Schema Overview**

Table Name	Purpose	Key Columns
artists	Stores information about artists	ArtistId, Name
albums	Lists all albums, with a foreign key to the artists table	AlbumId, Title, ArtistId
tracks	Details for each track, including length and file size	TrackId, Name, AlbumId, Milliseconds
genres	Lists music genres	GenreId, Name
customers	Information on the store's customers	CustomerId, FirstName, LastName, Email
invoices	Invoice header data	InvoiceId, CustomerId, InvoiceDate, Total
invoice_items	Individual items on each invoice	InvoiceLineId, InvoiceId, TrackId
playlists	User-defined playlists	PlaylistId, Name

- **Artists:** Record 275, Feature 2
- **albums:** Record 347, Feature 3
- **tracks:** Record 3,000+, Feature 9
- **genres:** Record 25, Feature 2
- **customers:** Record 59, Feature 11
- **Invoices:** Record 414, Feature 9
- **InvoiceLine:** Record 2,000+, Feature 5
- **playlists/MT:** Record 5, Feature 2

The Role of SQLAlchemy's create_engine()

- **The create_engine()** function is the primary entry point for connecting to a database with SQLAlchemy. It takes a single argument: a connecting string that specifies the database dialect, driver, and file path. For an SQLite database, the connection string follows a specific format
- the create_engine() can also accept an optional **echo=True** parameter. When enabled, this feature logs every SQL command that SQLAlchemy executes, allowing a practitioner to see the underlying queries in real-time. This is invaluable fool for debugging and for gaining a deeper understanding of how the Python cpde translate to SQL

```
chinook_engine = create_engine("sqlite:///chinook.db", echo=True)
```

- **dialect:** ชนิดฐานข้อมูล เช่น Postgresql, mysql, sqlite
- **driver:** ตัวเชื่อมต่อ Python เช่น psycopg2 (Postgres), pymysql (MySQL)
- **echo=True:** เป็นพารามิเตอร์ของ create_engine()
  - ถ้า echo=True → SQLAlchemy จะพิมพ์ทุก SQL statement ที่ส่งไปยัง DB ลง console
  - ใช้สำหรับ debug / ตรวจสอบ query

```
chinook_conn = chinook_engine.connect()

# Or, as a best practice, use a 'with' statement for automatic resource management
with chinook_engine.connect() as conn:
    print("Connection to Chinook database established successfully.")
    # All database operations would happen here
    pass

chinook_conn.close() # Close the explicit connection
```

Using a with statement with the database connection ensures that the connection is automatically closed when the block of code is finished, even if error occur. it prevents resource leaks that could defrade system performance over time.

The read_sql() Family: A Crucial Distinction

The **pd.read_sql()** function acts as a convenient, high-level wrapper. It intelligently delegates to either **pd.read_sql_table()** or **pd.read_sql_query()** based on whether its first argument is a table name or a SQL query string.

While **pd.read_sql()** is convenient, should be aware of a critical nuance, especially when working with SQLite databases. *When a connection object from the standard sqlite3 library is used (as opposed to an SQLAlchemy engine), **pd.read_sql()** will not accept a table name as valid argument and will fail.*

read_sql_table() when loading an entire table and **read_sql_query()** when executing a custom query.

```
# Load the entire Employees table from Northwind
northwind_employees_df = pd.read_sql_table('Employees', con=northwind_engine)
print("Northwind Employees DataFrame loaded successfully.")
print(northwind_employees_df.head())
```

Ingesting Data with Custom SQL Queries: read_sql_query()

that require filtering, joining, or aggregating data before it is loaded into a DataFrame.

```
products_query = "SELECT ProductName, Price FROM Products WHERE Price > 50;"
expensive_products_df = pd.read_sql_query(products_query, con=northwind_engine)
```

week03: Star Schema

Data Type

- **Logic/boolean:** data can be one of two values: True or False.
- **Alphanumeric:** text refer to all the letters and numbers and other characters such as /, @, etc. Text is just letter AtoZ
- **Number:** (real and integer): Float and Int
- **Date:** common date format are:
  - dd/mm/yy
  - yyyy-mm-dd

Database structure

- **Flat-file database**
  - Data is stored in files. There is no relation between elements of data, ex. a table in spread sheet (Excel)
- **Relational database**
  - All data is stored in several tables with links between the tables to enable the data in the separate tables to be combined together if needed.

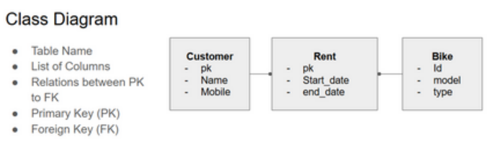
Keywords

- **Files** is a groups of data or records
- **Record** all data relating to a single thing
- **Field** is a item of data about a fact of thing(s)
- **Data** these are facts or value of thing(s)
- **Information** is data with meaning
  - 12/05/15 is
  - 12/05/15 with "field name: ExamDate" and "format: dd/mm/yy"
- **Table** is a collection of data elements arranging in row and column. If database consists of a table, we called it a flat-file database.
- **Key field** is a piece of data in table that is unique to allow linking between different tables

Problems with flat-file system

- **Data redundancy.** A lot of duplicate data (ข้อมูลเดียวกันถูกเก็บซ้ำหลายที่)
- When a recore is deleted, **some useful data may be deleted**

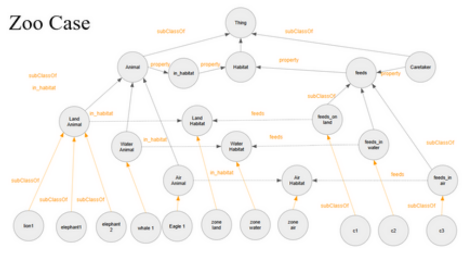
Class Diagram



RDF File

- The Resource Description Framework is a World Wide Web Consortium standard originally designed as a data model for metadata.
- It has come to be used as a general method for description and exchange of graph data
  - Triple (ข้อมูล 3 ส่วน) ex. Alice – knows – Bob

Zoo Case

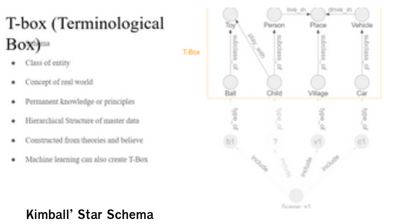


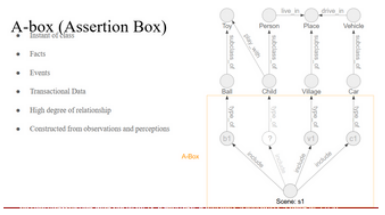
Multi-Hop Reasoning

- Multi-hop reasoning is when we use a series of inferential steps to derive new knowledge

Knowledge Base

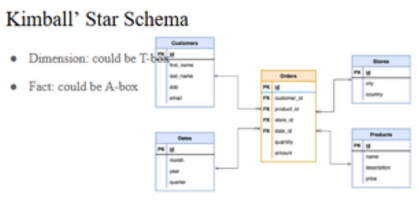
- **T-Box (Terminological)**
  - เป็นส่วนหนึ่งของ Knowledge Base ที่ใช้นิยามโครงสร้างความรู้ (schema / ontology)
  - ระบุว่า Concept / Class มีคุณสมบัติอะไร และสัมพันธ์กับ Concept อื่นอย่างไร
  - T-Box จะบอกว่า อะไรเป็นอะไร และมีความสัมพันธ์อย่างไร แต่ยังไม่ระบุ instance จริง
    - ex. Classes: Animal, mammal, Bird (mammal เป็น subclass ของ Animal)
    - Concept, Hierarchical
- **A-Box (Assertional Box)**
  - เป็นส่วนหนึ่งของ Knowledge Base ที่บรรจุข้อมูลเชิงข้อเท็จจริง (instance / assertion)
  - A-Box = "Assertions about individuals"
  - A-Box จะบอกว่า instance ไหนมีคุณสมบัติอะไร ตาม schema ที่กำหนดใน T-Box
    - ex. Leo is a Lion (instance)
    - About facts, includes many entity, Unique identity


T-box (Terminological Box)
Class of entity
Concept of real world
Permanent knowledge or principles
Hierarchical Structure of master data
Constructed from theories and believe
Machine learning can also create T-Box
T-Box
Toy
Person
Place
Vehicle
live_in
drive_in
subclass_of
play_with
Ball
Child
Village
Car
type_of
b1
?
v1
c1
include
Scene: s1

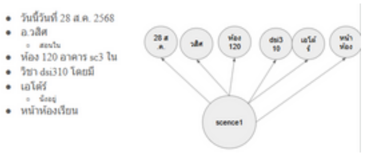


Kimball' Star Schema

- **Dimension:** could be T-box
- **Fact:** Could be A-Box



- **Fact Table:**
  - เป็นตารางหลักของ Star Schema
  - เก็บ measure / metric / transactional data
  - มี foreign key เชื่อมกับ dimension table
  - ลักษณะข้อมูล เป็นตัวเลข / quantitative
- **Dimension Table**
  - เป็นตารางสนับสนุน / Descriptive data
  - ใช้ foreign key จาก fact table
  - ลักษณะข้อมูล: ตัวอักษร / descriptive / categorical



Chapter 1: Introduction to Data Eng & Data Lake

The Strategic Role of a Data Engineer

- **Defining the Role and Responsibilities:** Data engineers are responsible for designing, constructing, and maintaining, architectures, including databases and systems for large-scale processing. Their work includes the development and maintenance of data management systems.
- **Essential Skills for Success:** The role demands a diverse skill set, blending technical expertise with a strong business acumen
  - **Technical Skills:** Proficiency in programming languages like Python or Java is crucial for autoomating processes and building data pipelines. A deep understanding of both relational (SQL) and non-relational (NoSQL) databases is essential. Big data engineers must talso be familiar with Big data technologies like Apache Hadoop and Apache Spark for distributetd systems, as well as cloud services (AWS or Azure)
  - **Non-Technical Skills:** Effective communication and collaboration are vital. A data engineer must be able to understand the goals and needs of both technical and non-technical colleagues and then translate complex technical concepts into clear,
- **Enabling Intelligence:** The work of a data engineer directly supports the application of advanced technologies like Artificial Intelligence (AI) and Large Language Models (LLMs). Without a clean, well-structured, and continuously updated data foundation, these sophisticated models cannot function effectively.

Understanding the Modern Data Platform Architecture

A modern data platform is unified, end-to-end ecosystem for managing all of an organization's data assets. While the specific componentst can vary, a common and highly effective architectuural patterns is the data lake, which is often segmented into distinct zones based on the maturity and usability of the data

Modern Data Platform คือสถาปัตยกรรมและเครื่องมือชุดใหม่ที่ใช้ในการจัดการข้อมูลแบบครบวงจร การนำเข้าข้อมูล (Ingestion), การจัดเก็บ (Storage), การประมวลผล (Processing), การวิเคราะห์ (Analytics/BI/ML) ออกแบบตอบโจทย์ในยุค Big Data และ Variety รวมถึง Velocity (เปลี่ยนแปลงอย่างรวดเร็ว) based-on cloud

The Data Lake Paradigm

A data lake is a centralized reposiroty that allows you to store all of your structured and unstructured data at any scale. Unlike a traditional data warehouse, which requires a pre-defined schema, a data lake stores in its raw, native format. This flexibility is a significant advantage, but it also necessitates a disciplined approach to organization to prevent the data lake from becoming a "data swamp" This is where the concept of "Data Lake Zones" becomes critical. The data lake serves as a foundation and landing zone for raw data, ensuring that no information is lost by prematurely transforming or aggregating it.

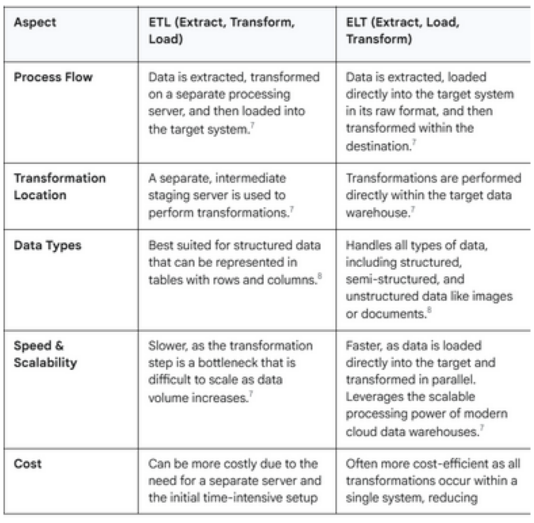
- "data swamp": (แหล่งข้อมูลที่ไร้ระเบียบและใช้งานยาก)

The Layered Architecture: Data Lake Zones

- **Landing Zone (Raw Zone):**
  - **Purpose:** This is the initial entry point for all data. Data is ingested "as-is" from its source systems (e.g., relational databases, APIs, IOT sensors) and stored without any transformations.
  - **Key Characteristics:**
    - **Immutability:** The raw data in this zone is never changed. This provides a complete historical record and allows for reprocessing if a new business requirement or a data quality issue is discovered downstream. The only exception is the removal of personally identifiable information (PII) to ensure privacy and regulatory compliance before the data is saved.
    - **Schema-on-Read:** The schema is not enforced at the time of writing; instead, the data's structure is inferred when it is read by a downstran process
- **Staging Zone (Cleansed Zone)**
  - **Purpose:** The Staging Zone is where the inital data cleaning and preparation takes place. Data is read from the landing Zone, and preliminary transformations are applied to address data quality issues.
  - **Key Characteristics:**
    - **Data Cleansing:** Common tasks include handling missing values, removing duplicates, and correcting inconsistent data types. This is where data profiling is performed to understand the data's characteristics and potential anomalies
    - **Standardization:** Data is standardized into a consistent format. For instance, dates are converted to a standard format (e.g., YYYY-MM-DD), and Text fields are cleaned
- **Integration Zone (Curated/Analytics Zone):**
  - **Purpose:** The Integration Zone is the final destination for data before it is made available for analysis. This is where data from different sources is combined and transformed into a structured format that is optimized for analytical querying. This zone often takes the form of a data warehouse or data mart.
  - **Key Characteristics:**
    - **Aggregation and Denormalization:** This zone often involves complex transformations, such as aggregating data and creating denormalized tables (e.g., star schemas) to improve query performance.
    - **Business Logic:** Business rules and logic are applied here. For example, calculating a new metric like
      - SalesAmount from UnitPrice and Quantity.
    - **Optimized for Performance:** Data in this zone is typically stored in a columnar format (e.g., Parquet), which is highly efficient for analytical queries.

ETL vs. ELT: A Fundamental Choice in Data Integration

- as ETL (Extract, Transform, Load) or its modern alternative, ELT (Extract, Load, Transform). While both processes aim to move data from a source to a destination



Aspect	ETL (Extract, Transform, Load)	ELT (Extract, Load, Transform)
Process Flow	Data is extracted, transformed on a separate processing server, and then loaded into the target system.[7]	Data is extracted, loaded directly into the target system in its raw format, and then transformed within the destination.[7]
Transformation Location	A separate, intermediate staging server is used to perform transformations.[7]	Transformations are performed directly within the target data warehouse.[7]
Data Types	Best suited for structured data that can be represented in tables with rows and columns.[8]	Handles all types of data, including structured, semi-structured, and unstructured data like images or documents.[8]
Speed & Scalability	Slower, as the transformation step is a bottleneck that is difficult to scale as data volume increases.[7]	Faster, as data is loaded directly into the target and transformed in parallel. Leverages the scalable processing power of modern cloud data warehouses.[7]
Cost	Can be more costly due to the need for a separate server and the initial time-intensive setup	Often more cost-efficient as all transformations occur within a single system, reducing

- Process Flow:
  - ETL: ดึงข้อมูลจากแหล่ง → แปลง → โหลดเข้า Data Warewhouse
  - ELT: ดึงข้อมูลจากแหล่ง → โหลดเข้า Data Lake/Data Warehouse → แปลงทีหลัง
- Transformation Location:
  - ETL: Transform เกิดขึ้นก่อน เข้าคลังข้อมูล
  - ELT: Tramsform เกิดขึ้นภายใน Data Warehouse/Lake (เช่น BigQuery, Snowflake)
- Data Types:
  - ETL: เหมาะกับ Structured Data (RDBMS, CSV)
  - ELT: รองรับ Structured, Semi-structured, unstructured Data
- Speed & Scalability:
  - ETL: อาจช้ากว่า เนื่องจาก Transform เกิดก่อนโหลด และต้องใช้ ETL server
  - ELT: เร็วกว่าและ scalable เพราะใช้ power ของ Data Warehouse/Lake ในการ transform
- Cost:
  - ETL: อาจมี ต้นทุนสูง เนื่องจากต้องใช้ ETL tool/servers แยก
  - ELT: โดยทั่วไป คุ้มค่า กว่า เพราะ leverage power ของ cloud warehouse/lake (pay-as-you-go)

- ETL = เหมาะกับระบบเก่า/ข้อมูลเชิงโครงสร้างที่ต้อง clean ก่อนโหลด
- ELT = เหมาะกับยุค Cloud/Big Data ที่เน้น scale และความเร็ว

Chapter 1: Introduction to Data Eng & Data Lake (2)

Real-World Applications of Data Platforms

Smart Manufacturing and Predictive Maintenance

- **Predictive Maintenance:** By applying AI to analyze machine performance and sensor data, companies can forecast potential equipment failure and perform proactive maintenance, thereby minimizing costly downtime. This strategy can cut unplanned downtime by up to 50% and slash maintenance costs by 10-40%.
- **How it works:** IoT devices with sensors continuously monitor metrics like temperature, pressure, and vibration levels in real time. This data is streamed to a central system where machine learning algorithms analyze it to detect subtle changes or anomalies that may signal wear and tear.
- **Examples in Action:** A cloud-based data platform allowed an automobile manufacturer to analyze its global fleet performance without significant infrastructure costs.9 In the manufacturing sector, data platforms help identify production bottlenecks and allow for real-time adjustments to optimize inventory and efficiency. The **Hitachi Lumada** platform is a prime example of this, using data and AI to optimize industrial operations.1

Data Platforms in Other Industries

- **Healthcare and Life Sciences:** In healthcare, data analytics tools can be used to detect fraudulent billing patterns, such as "phantom bills" or "upcoding". They also play a crucial role in patient data security by monitoring network traffic and assigning real-time risk scores to transactions to prevent cyberattacks. These platforms enable the creation of a "patient 360" view by unifying fragmented data from various sources like clinical, claims, and consumer data.
- **Finance:** Data platforms are critical for fraud detection and risk management. Real-time analytics can be used to detect stock market manipulation, for example, by using technologies like Generative Adversarial Networks (GANs) to distinguish between real and manipulated stock prices.
- **International Trade:** The course materials highlight a successful project with the Department of International Trade Promotion (DITP). A data warehouse was built to support small and medium-sized exporters, enabling the tracking of trade achievements, assessment of SMEs, and the provision of targeted trade recommendations.

From Data Models to Knowledge Graphs: A Deeper Connection

The Principles of dimensional modeling are not just practical; they are also grounded in a formal theoretical framework for knowledge representation. draw a direct parallel between the components of a star schema and the ontological concepts of a knowledge graph.

- **The Terminological Box (T-Box):** The Ti-Box represents the schema, concepts, and hierarchical structure of permanent knowledge. This aligns perfectly with the function of **dimension tables.** The T-Box is a master list of entities and their properties, mirroring the role of a dimension table.
- **The Assertion Box (A-Box):** The A-Box represents the transactional, factual data, or "events". This is the conceptual equivalent of a **fact table.** The A-Box records these individual facts and the high degree of relationships between them, which is percisely the purpose of a fact table.

This conceptual alignment demonstrates that a star schema is not just a performance hack for databases; it is a logically sound and robust method for representing business knowledge. By separating the static, descriptive master data (dimensions/T-Box) from the dynamic, event-driven transactional data (facts/A-Box), the model provides a clear, powerful, and theoretically consistent framework for analysis.

Key Theories and Keywords

- <u>**Data Platform**</u>: A centralized, end-to-end system that encompasses the infrastructure, tools, and processes for the collection, processing, and management of data to support business intelligence and analytics.
- <u>Data Engineering:</u> he professional discipline focused on the design, construction, and maintenance of the data platforms and pipelines that enable organizations to turn data into strategic insights.
- <u>**Data Lake**</u>: A massive, centralized repository that holds all forms of data in its native format, regardless of its structure. It is a key component of a modern data platform.
- <u>Landing Zone:</u> The first layer of a data lake, where raw, immutable data is stored exactly as it was ingested from the source system.
- <u>Staging Zone:</u> The second layer of a data lake, where preliminary data cleansing, standardization, and quality checks are performed to prepare the data for further transformations.
- <u>**Integration Zone**</u>: The third layer of a data lake, where data from various sources is combined, aggregated, and transformed into a structured, analytics-ready format, such as a star schema.
- <u>ETL (Extract, Transform, Load):</u> A traditional data integration process where data is first extracted from a source, then transformed in a separate staging area, and finally loaded into a target data store.
- <u>ELT (Extract, Load, Transform):</u> A modern data integration process where data is first extracted and loaded into a target data store, with transformations occurring within the target system itself.
- <u>**Predictive Maintenance**</u>: An AI-driven strategy that uses real-time data from machinery to predict potential failures, allowing for proactive maintenance and minimizing downtime.
- <u>Dimensional Modeling:</u> A data design philosophy that organizes data into a central fact table and surrounding dimension tables to optimize for analytical queries.
- <u>**Start Schema**</u>: A simple but highly effective dimensional model consisting of a central fact table surrounded by multiple dimension tables, resembling a star.
- <u>**Fact Table**</u>: A central table in a star schema that contains the quantitative metrics, or "facts," of a business process (e.g., SalesAmount, SalesQuantity). It is the conceptual equivalent of a Knowledge Graph's Assertion Box (A-Box).
- <u>Dimension Table:</u> A table in a star schema that provides the descriptive context for the facts (the "who, what, where, when, and how"). It is the conceptual equivalent of a Knowledge Graph's Terminological Box (T-Box).

Chapter 2: SQL Data Sources & Data Ingestion

The Relational Database: The Heartbeat of Business Operations

The vast majority of transactional data—from sales orders to customer records—is stored in relational databases. These databases are the core of Online Transaction Processing (OLTP) systems that support day-to-day business operations. They are optimized for fast, reliable, and consistent data modifications, which is why they are the primary choice for applications like e-commerce, banking, and inventory management

From Flat-Files to a Relational Model

- **Flat-File Database:** A flat-file system stores all data in a single table, like a spreadsheet. While simple, this approach leads to significant problems:
  - **Data Redundancy:** The same information, such as a customer's address, is duplicated across many records, wasting storage and creating a risk of inconsistency.
  - **Data Anomalies:** If a customer's address changes, it must be updated in multiple places, making the process prone to error. Deleting a record might inadvertently delete useful data that is not stored anywhere else. Imagine a flat file containing car rental data. Deleting a record for a specific car rental would also delete the customer's name and contact information.
  - **Lack Scalability:** Managing a single, massive file for a large business becomes inefficient and difficult to query, as the entire file must be scanned to find information.
- **Relational Database:** A relational database solves these problems by organizing data into multiple, distinct tables with predefined relationships between them.
  - **Data is segmented:** into logical groups (e,g,, a Customer table, an Order table)
  - **Relationships are defined:** using Key Fields to link the tables togethher. This is the essence of a relational model.
  - The key to this system is that data is stored once, eliminating redundancy and ensuring data integrity. This design is also known as a normalized schema, which is highly efficient for transactional systems.

Key Concepts in Relational Databases

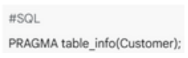
- **Table**: A collection of data elements arranged in rows and columns. In our OmniCorp project, Customers, Employees, and Orders are all tables in the Northwind database.
- **Field (or Column)**: A single item of data about a thing. For example, CustomerID or CompanyName are fields within the Customers table. The course materials also define various data types that a field can hold, such as Alphanumeric/text, Number (real or integer), Date, and Boolean values.
- **Record (or Row):** A collection of all data relating to a single item or entity. Each row in the Customers table represents a single customer.
- **Primary Key (PK):** A field (or combination of fields) in a table that uniquely identifies each record. For example, CustomerID is the primary key in the Northwind Customers table. It ensures that every record is unique and can be referenced by other tables.
- **Foreign Key (FK):** A field in one table that links to the primary key of another table. It establishes a relationship between the tables. For example, a CustomerID field in the Orders table would be a foreign key that links to the CustomerID in the Customers table, showing which customer placed the order. These keys are the glue that holds the relational model together.

SQL: The Language of Data Exploration

SQL (Structured Query Language) is the standard language for managing and manipulating relational databases. is often through SQL to perform Exploratory Data Analysis (EDA). EDA is the process of examining a dataset to summarize its main characteristics, often with visual methods, to better understand its structure, content, and potential quality issues before it is used for analysis.

Step 1: Discovering the Schema and Data Types:

- **Finding Tables:** The first step is to list all tables in the database. A common SQL query for this is SELECT name FROM sqlite_master WHERE type='table';.
  - This initial discovery is crucial for identifying the common entities we'll need to unify for the project, such as Customers, Employees, and Products.
- **Understanding Data Types:** In SQLite, you can use the PRAGMA table_into() statement to get information about the columns in a table, including their data type.



```
#SQL
PRAGMA table_info(Customer);
```

- This command returns a result set with details about each columns, including its name, data type(Text, Integer, etc.), and wheter it can be null.

Step 2: Initial Data Profilling with SQL Queries

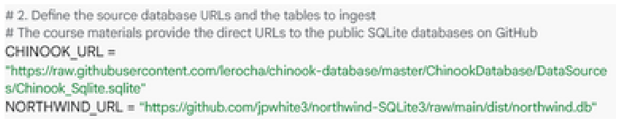
- **Counting Records:** One of the simplest yet most informative queries is to count the number of records in a table to understand its size
- SELECT COUNT(*) FROM Customers;
- **Finding Unique Values:** To understand the diversity of data in a column, you can find the unique values. This is particularly useful for categorical data.
- SELECT DISTINCT Country FROM Customer;
- **Aggregating Data:** SQL is powerful for aggregation. This query provides a summary that can reveal important business insights
- SELECT Country, COUNT(*) AS NumberOfCustomers FROM Customer GROUP BY Country ORDER BY NumberOfCustomers DESC;
- **Filtering Data:** You can use a WHERE clause to filter data based on specific conditions. This is essential for identifying potential data quality issues, such as missing values.
- SELECT * FROM Customers WHERE CompanyName IS NULL;
- **Joining Tables:** While we want to extract raw, un-transformed data for the Landing Zone, a quick join can help us understand relationships between tables.
- SELECT T.Name AS TrackName, I.Quantity, I.UnitPrice FROM InvoiceLine AS I JOIN Track AS T ON I.TrackId = T.TrackId LiIMIT 5;

Data Ingestion: Moving Data to the Landing Zone

the next step is to programmatically extract the data and save it to the Landing Zone of our data lake. This process is known as Data Ingestion. The Landing Zone is the destination for raw, unmodified data. The objective is to capture the data as-is, preserving its original structure, data types, and integrity. This allows us to re-process the data later if needed, a crucial principle of a modern data platform.

Python and SQLAlchemy for Database Connectivity

SQLAlchemy is a powerful Python library that provides a consistent way to interact with many different types of databases



```
# 2. Define the source database URLs and the tables to ingest
# The course materials provide the direct URLs to the public SQLite databases on GitHub
CHINOOK_URL =
"https://raw.githubusercontent.com/lerocha/chinook-database/master/ChinookDatabase/DataSource
s/Chinook_Sqlite.sqlite"
NORTHWIND_URL = "https://github.com/jpwhite3/northwind-SQLite3/raw/main/dist/northwind.db"
```

Explaining the Ingestion Process and File Format

The script above is a simplified representation of a data pipeline. It demonstrates the ETL process in its most basic form: Extract (from the SQL database) and Load (to the Landing Zone).

- **The Role of Python Libraries:** The requests library is used to get the database file from a web server. This simulates ingesting data from an external source. The sqlalchemy provides the create_engine function, which acts as a powerful abstraction layer, allowing us to connect to different database types (e.g., SQLite, PostgreSQL, MySQL) using a consistent API. Finally, pandas provides a high-level, intuitive way to read the data from the database into a DataFrame and then write it to a file.
- **The SELECT * Principle:** The use of SELECT * is a deliberate choice that strictly adheres to the principle of a Landing Zone—store data as-is, with no transformations. This ensures that we have a permanent, raw copy of the data from the source, which can be re-processed in the future if new requirements or data quality issues are discovered.
- **Why Parquet?:** We chose to save the data to the Parquet format. Parquet is not just a file; it is a columnar storage format This means that instead of storing data row-by-row, it stores data by column. This is a significant advantage for analytical queries because most analytical tools only need to read a few columns at a time.By only reading the columns they need, Parquet files dramatically reduce I/O (input/output) operations and improve query performance. Parquet also preserves the data types and schema, making it a reliable format for a Landing Zone where data integrity is paramount.

Parquet Format: Apache Parquet คือ Columnar Storage Format ที่ถูกออกแบบมาให้เก็บข้อมูลอย่าง มีประสิทธิภาพทั้งด้านการบีบอัด (Compression) และการอ่านข้อมูล (Read Performance), ไฟล์ format แบบคอลัมน์ เหมาะสำหรับ Big Data และงาน Analytics

Key Theories and Keywords

- <u>Relational Database:</u> A database system that organizes data into a set of tables with predefined relationships between them, using a structured approach to ensure data integrity and minimize redundancy. This is the foundation of most Online Transaction Processing (OLTP) systems.
- <u>Flat-File Database:</u> A simple database that stores all data in a single, un-related table, often leading to data redundancy and inconsistencies. This model is generally not scalable for complex business operations.
- <u>SQL (Structured Query Language):</u> The standard programming language used for managing, retrieving, and manipulating data in a relational database. It is a declarative language, meaning you describe
- <u>Exploratory Data Analysis (EDA):</u> The process of using statistical and visual methods to understand the main characteristics of a dataset, identify patterns, spot anomalies, and prepare for further analysis. It is a crucial first step for any data project.
- <u>Data Ingestion:</u> The process of collecting, importing, and transferring data from various sources into a storage system, such as a data lake, for subsequent processing and analysis. This is the initial step of the ETL or ELT data pipeline.
- <u>Landing Zone:</u> The first layer of a data lake, which serves as the raw, immutable destination for data as it is ingested from its source systems. The data here is stored in its native format and is not transformed, with the exception of stripping out sensitive personally identifiable information (PII).
- <u>SQLAlchemy:</u> A popular Python library that provides a flexible and powerful way to connect to and interact with a wide range of relational databases. It allows data engineers to write code that is independent of the specific database system being used.
- <u>Primary Key (PK):</u> A unique identifier for a record in a table (e.g., CustomerID). Its purpose is to uniquely identify a record and ensure data integrity.
- <u>Foreign Key (FK):</u> A field in a table that links to the primary key of another table, establishing a relationship (e.g., the CustomerID in an Orders table). Foreign keys are essential for relational database design.
- <u>Data Types:</u> The classification of the kind of data a field can hold, such as Alphanumeric/text, Number (real or integer), Date, and Boolean. Correctly identifying and handling these types is a fundamental data engineering task, especially when integrating data from different sources.
- <u>Parquet:</u> A columnar storage file format optimized for big data analytics. It stores data by column rather than by row, which significantly improves query performance by allowing analytical engines to read only the data they need. It is also highly efficient in compression and storage.