

SQL- Structured Query Language

Database - place where data is stored.

SQL- Query \rightarrow a question used to interact. that should be structured

RDDBMS - relational DataBase Management system

DBMS $\begin{cases} \text{structured} \rightarrow \text{tabular} & \text{Eg: MySQL} \\ \text{unstructured} \end{cases}$

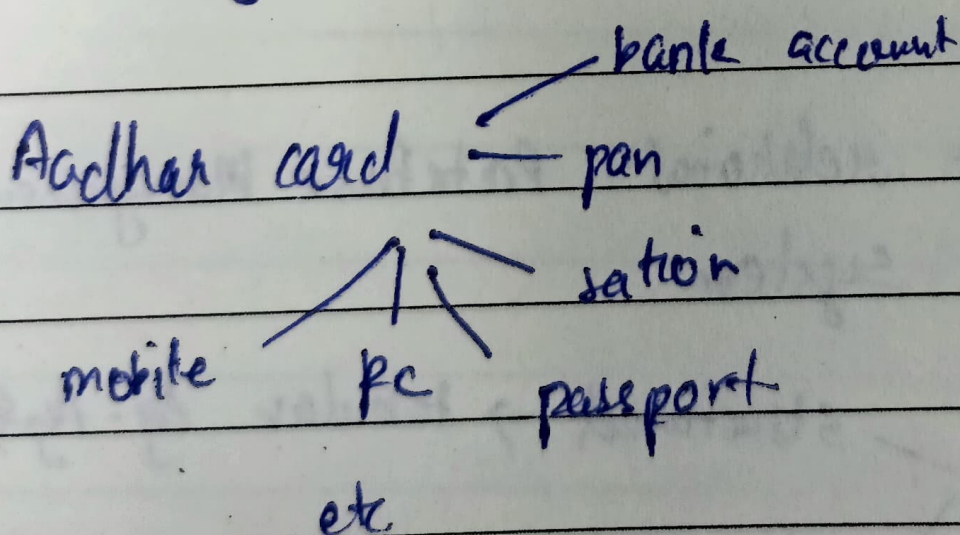
Key-value pair

↓
unstructured database

Eg. MongoDB

Why database over .csv?

- easy retrieval of data (using indexing)
- large amt of data in database
- security



why all this instead of one?

- restrict data usage and accessibility

'Schema' refers to the structure / blueprint of how a database is organised

Rapid changes → scalable; so unstructured databases are formed

Operations in SQL

→ Create
→ Read

→ Update
→ Delete

- create a new row
 - read a row
 - update a row
 - delete a row
- } or a whole table

Primary key → non-editable and mandatory data

- mandatory
 - unique for the table
- } table specific

Foreign key. → relation b/w 2 tables

Primary key of one table when brought in another table, it is a foreign key.



SQL - Domain specific language

Python - general purpose

- ML application
- web applications
- mobile applications

Commands

sql200.net

1) SELECT

- To get full table

SELECT * FROM "table name"

condition keyword

'WHERE'

LIKE - used for pattern matching

~~AL~~ AL% whether beginning with 'AL'

%a - ending with 'a'



Create → INSERT

Read → SELECT

Update → UPDATE

Delete → DELETE

order of certain commands

- GROUP BY
- HAVING
- ORDER BY
- LIMIT

SELECT * FROM world LIMIT 4



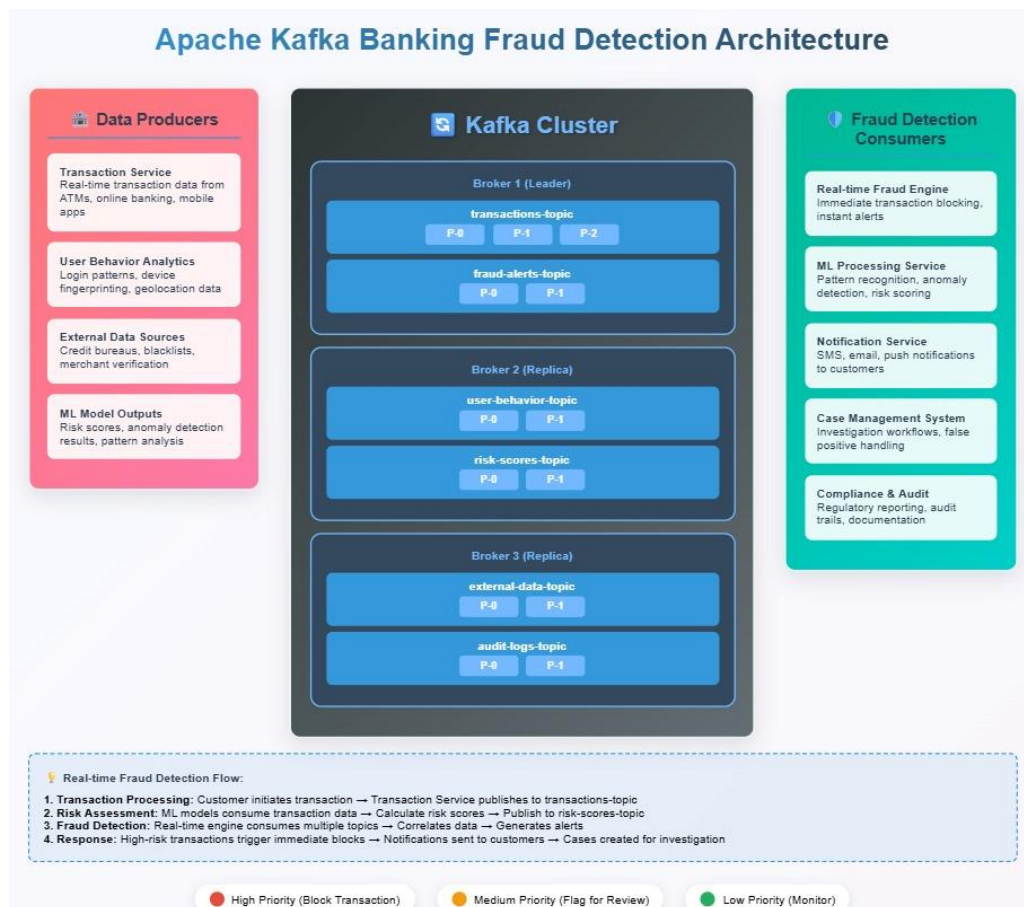
First 4 rows
shown

Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun ☐

Date. . .

Grouping functions can't
be used in WHERE

Notes: Kafka



Data Producers (Left Side of Diagram)

These are systems that generate and send real-time data to Kafka topics:

- Transaction Service

Captures real-time banking activity from mobile apps, online platforms, and ATMs.

- User Behaviour Analytics

Tracks login patterns, device fingerprints, and location to detect unusual behaviour.

- External Data Sources

Brings in external inputs like blacklists, credit bureaus, and merchant verification results.

- ML Model Outputs

Risk scoring, anomaly detection, and pattern recognition results are also pushed to Kafka.

All these producers send data to Kafka topics for centralized processing.

Kafka Cluster (Center of Diagram)

Kafka acts as the central data hub, collecting and distributing data using topics, which are divided into partitions for scalability and fault tolerance.

- Transactions Topic

Stores real-time transaction data from customers.

- Fraud Alerts Topic

Holds messages about suspicious transactions for alert systems.

- User Behaviour, Risk Scores, External Data Topics

Store respective data used in fraud detection logic.

- Audit Logs Topic

For compliance, stores event history and alerts for future reviews.

Each topic has multiple partitions (P-0, P-1, etc.) and is spread across Kafka brokers for performance and availability.

Fraud Detection Consumers (Right Side of Diagram)

These components consume data from Kafka to detect and respond to fraud:

- Real-time Fraud Engine

Core engine that analyzes incoming transactions, user behaviour, and risk scores. If a transaction seems fraudulent, it creates alerts or blocks it immediately.

- ML Processing Service

Continuously consumes transaction and behaviour data, applies ML models, and generates updated risk scores.

- Notification Service

Listens to the fraud-alerts-topic and sends alerts to customers via SMS, email, or push notifications.

- Case Management System

Used by internal fraud teams to investigate flagged transactions, especially false positives.

- Compliance & Audit

Subscribes to audit logs for regulatory reporting and internal documentation.

Example: Kafka in Banking App

User makes a payment → App pushes data to Kafka.

Kafka sends it to a fraud detection service.

If risky → Kafka triggers an alert service.

All of this happens in real time.

Overall Architecture

The producer-consumer-broker architecture follows a decoupled messaging pattern where:

Producers generate and send messages

Brokers receive, store, and route messages

Consumers receive and process messages

Producer Architecture and Flow

Architecture:

Producers are client applications that create and send messages to topics/queues

They maintain connections to the broker through network protocols (TCP, HTTP, etc.)

Often include serialization components to convert data into message format

May have buffering mechanisms for batching messages

Flow:

Application generates data/events that need to be shared

Producer serializes the data into message format

Producer sends message to specific topic/queue on the broker

Broker acknowledges receipt (depending on durability settings)

Producer continues with next message or terminates

Broker Architecture and Flow

Architecture:

Central messaging infrastructure that manages message storage and routing

Contains topic/queue management systems

Implements persistence layer (disk storage, memory, or hybrid)

Handles client connection management

Includes replication mechanisms for fault tolerance

Manages message ordering and partitioning

Flow:

Receives incoming messages from producers

Validates message format and authorization

Stores messages in appropriate topics/queues

Maintains message ordering (if required)

Tracks consumer subscriptions and offsets

Delivers messages to subscribed consumers

Handles acknowledgments and manages message lifecycle

Performs cleanup of processed messages (based on retention policies)

Consumer Architecture and Flow

Architecture:

Client applications that subscribe to topics/queues

Maintain persistent connections to brokers

Include deserialization components to process message content

Often implement message processing logic and error handling

May support consumer groups for load balancing

Flow:

Consumer subscribes to specific topics/queues

Broker pushes messages to consumer (push model) or consumer polls for messages (pull model)

Consumer deserializes message content

Processes the message according to business logic

Sends acknowledgment back to broker upon successful processing

Handles failures through retry mechanisms or dead letter queues

Updates consumer offset/position for next message

Key Interaction Patterns

At-Most-Once Delivery: Messages may be lost but never duplicated

At-Least-Once Delivery: Messages are never lost but may be duplicated

Exactly-Once Delivery: Messages are delivered exactly once (most complex to implement)

Consumer Groups: Multiple consumers can work together to process messages from the same topic in parallel, with the broker ensuring each message is processed by only one consumer in the group.

Message Ordering: Brokers can maintain message order within partitions/queues, ensuring sequential processing when required.

This architecture enables scalable, fault-tolerant, and loosely coupled distributed systems where components can operate independently while maintaining reliable communication.

1. Data Ingestion Techniques: Batch vs. Streaming Data (Kafka)

❖ Batch Data Ingestion

- **Definition:** Collects data over a period (e.g., hourly, daily) and processes it in chunks.
- **Use Cases:**
 - End-of-day sales reports
 - Historical data migrations
- **Tools:** Apache Sqoop, AWS Glue, Azure Data Factory
- **Pros:**
 - Efficient for large volumes
 - Easier to implement & test
- **Cons:**
 - Not real-time
 - Delay in data availability

❖ Streaming Data Ingestion

- **Definition:** Continuously processes data as it arrives in real-time or near real-time.
- **Use Cases:**
 - Fraud detection
 - Real-time user analytics (e.g., on websites, IoT)
- **Tools:** Apache Kafka, Apache Flink, AWS Kinesis
- **Kafka in Detail:**
 - Distributed event streaming platform
 - Producers push data → Kafka topics → Consumers process data
 - Guarantees durability, fault tolerance, and scalability
- **Pros:**
 - Real-time insights
 - Better for dynamic systems
- **Cons:**

- Complex to design
- Costlier infrastructure

Comparison Table

Feature	Batch Processing	Streaming Processing
Latency	High (minutes to hours)	Low (milliseconds to seconds)
Data Size	Large chunks	Continuous, small events
Complexity	Lower	Higher
Use Case Example	Daily revenue reports	Real-time traffic monitoring

2. ETL vs ELT: Building Robust Data Pipelines (Apache Airflow)

❖ ETL (Extract, Transform, Load)

- **Flow:** Extract → Transform → Load into warehouse
- **Best For:** On-premise and traditional data warehouses
- **Pros:**
 - Centralized control over transformation
 - Works well with structured data
- **Cons:**
 - Slower for large datasets
 - May duplicate work

❖ ELT (Extract, Load, Transform)

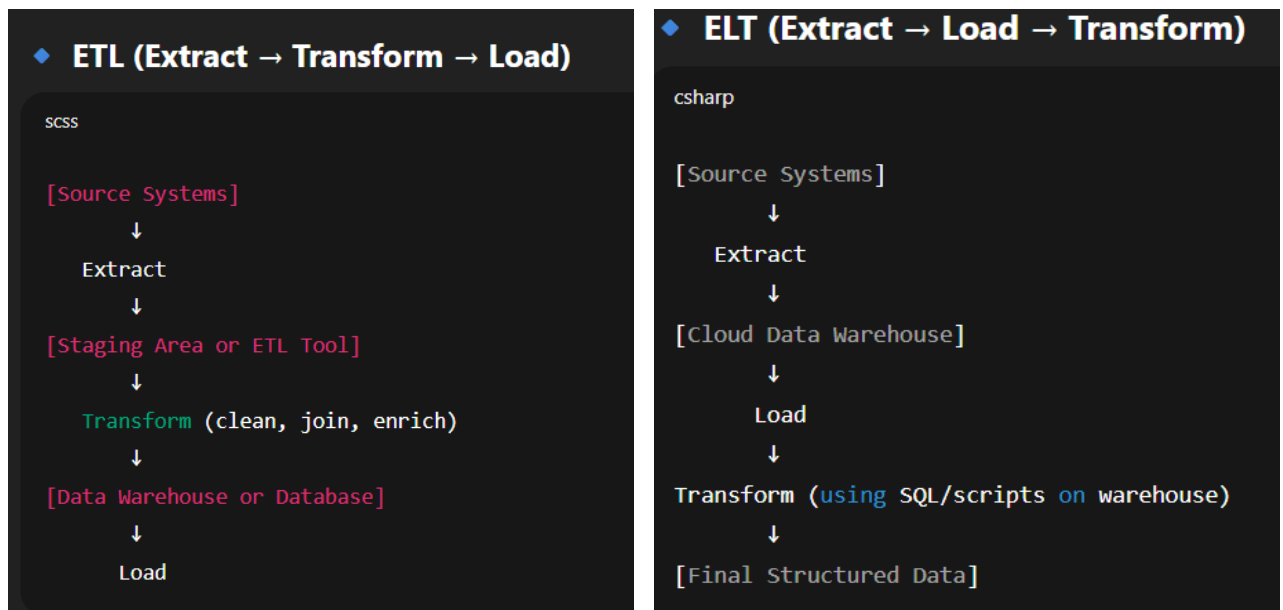
- **Flow:** Extract → Load into warehouse → Transform inside warehouse (e.g., BigQuery, Snowflake)
- **Best For:** Cloud-native systems with scalable compute
- **Pros:**
 - Takes advantage of cloud power
 - Faster load time
- **Cons:**

- Harder to debug transformations
- Can overload warehouse

❖ **Apache Airflow for Orchestration**

- **Purpose:** Automates, schedules, and monitors ETL/ELT workflows
- **How It Works:**
 - Uses DAGs (Directed Acyclic Graphs) to define workflow steps
 - Tasks can be Python, Bash, SQL, etc.
 - Includes retry logic, scheduling, monitoring, logging
- **Example Use Case:**
 - Fetching data from API every hour
 - Running transformations
 - Sending it to Redshift or BigQuery

Feature	ETL (Extract → Transform → Load)	ELT (Extract → Load → Transform)
Definition	Data is extracted, transformed (cleaned, enriched), then loaded into the target system.	Data is extracted and directly loaded into the target system where transformation happens.
Where Transform Happens	In a staging server or ETL tool	In the target system (typically a modern data warehouse like BigQuery, Redshift, Snowflake)
Best For	Traditional databases, on-premise systems	Cloud-native systems, big data pipelines
Speed	Slower for large datasets	Faster ingestion, scales better
Flexibility	High control over transformations	High flexibility with on-demand processing
Tools	Informatica, Talend, Apache NiFi, Pentaho	Apache Airflow, dbt, BigQuery, Spark SQL



3. Data Warehousing Concepts: OLAP, OLTP & Dimensional Modelling

Data Warehousing: A data warehouse is a large, centralized system that helps businesses store historical data and analyze it for decision-making.

❖ OLTP (Online Transaction Processing)

- **Purpose:** Handle real-time business transactions
- **Examples:** Bank systems, retail checkout, CRM
- **Characteristics:**
 - High number of short online transactions
 - Data is normalized (to reduce redundancy)
 - Fast reads/writes for individual rows

❖ OLAP (Online Analytical Processing)

- **Purpose:** Support decision making & complex queries
- **Examples:** Sales trend dashboards, performance reporting
- **Characteristics:**
 - Analytical queries (aggregations, pivots)
 - Historical data
 - Denormalized data (for fast querying)
 - Uses star/snowflake schemas

- **Dimensional Modelling**
- **Goal:** Make data understandable & fast for analysis
- **Two main types:**
 - **Fact Table:** Contains measurable events (e.g., sales)
 - **Dimension Table:** Descriptive context (e.g., product, customer)

1. Star Schema

- Fact table in center, joined directly to dimension tables
- Simple, fast querying

2. Snowflake Schema

- Dimensions are further normalized
- More complex, but less redundancy

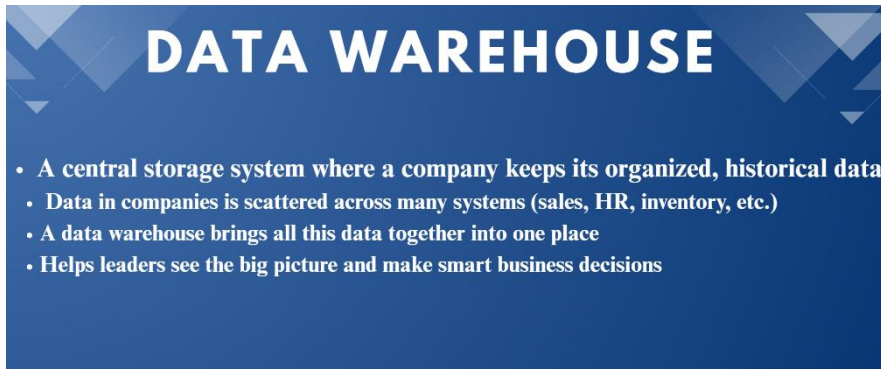
Step	Concept/Tool Used	Purpose/Description
1	OLTP (Online Transaction Processing)	Real-time transactional systems like CRMs, e-commerce databases, POS systems that collect day-to-day data.
2	Kafka	A distributed messaging platform that ingests streaming or batch data from source systems. Ideal for real-time data pipelines.
3	Airflow	Workflow orchestration tool that schedules and manages ETL/ELT pipelines using DAGs (Directed Acyclic Graphs).
4	ETL / ELT	Data is extracted, cleaned, transformed , and either loaded (ETL) or transformed after loading (ELT).
5	Data Warehouse	A centralized repository that stores structured data for long-term access, analytics, and historical tracking.

6	OLAP (Online Analytical Processing)	Provides multidimensional querying , aggregation, and slicing/dicing for fast analytics and reporting.
7	BI Tools	Tools like Tableau, Power BI, Looker used to create dashboards, reports, and insights for decision-making.

Google Colab Links:

1. ETL:
https://colab.research.google.com/drive/1xCqiwlzJOB_5PyHE3ss6WvOnJDMXEnKf?usp=sharing
2. Employee Details:
https://colab.research.google.com/drive/1QrF8tQdjaOVREUM0B-XPfDZOPPfbm_1?usp=sharing

Slides:



DATA WAREHOUSE

- A central storage system where a company keeps its organized, historical data
- Data in companies is scattered across many systems (sales, HR, inventory, etc.)
- A data warehouse brings all this data together into one place
- Helps leaders see the big picture and make smart business decisions

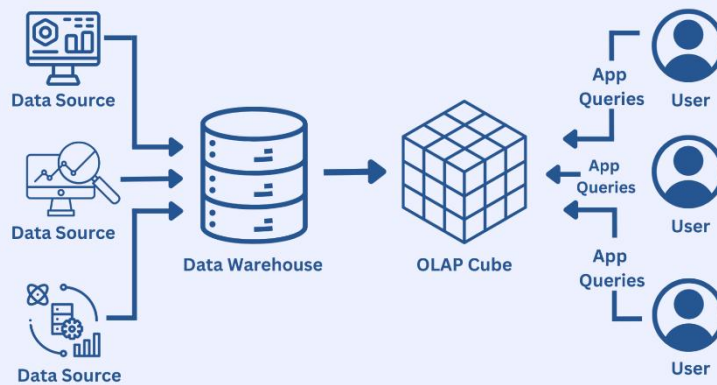
•OLTP (Online Transaction Processing)

- A system that handles day-to-day business transactions
- Handles lots of users and transactions simultaneously
- Data is current, not historical

Data Flow: OLTP to Warehouse to OLAP

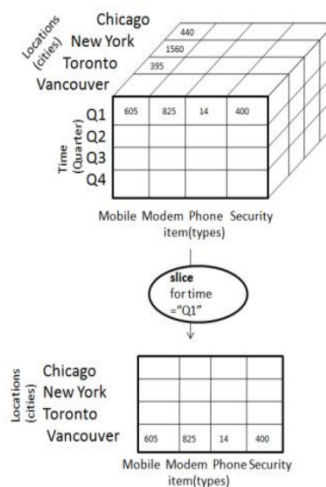
[OLTP System (MySQL, PostgreSQL)]
 ↓
 ETL Process (Extract, Transform, Load)
 ↓
 [Data Warehouse (Redshift, BigQuery)]
 ↓
 [OLAP Tools (Power BI, Tableau)]

ONLINE ANALYTICAL PROCESSING

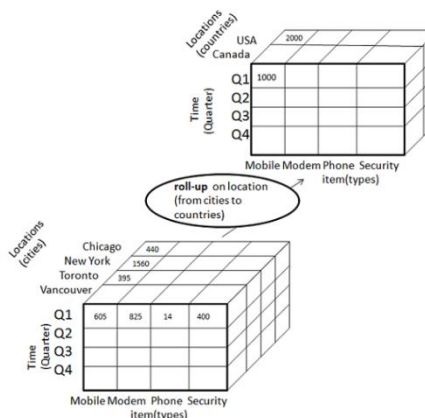


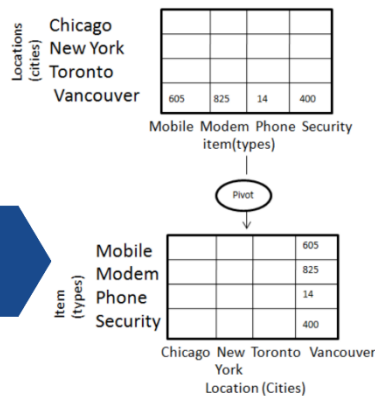
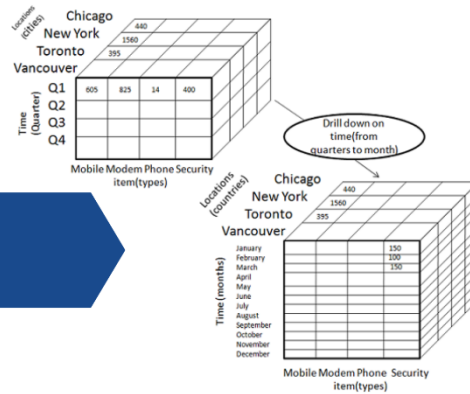
Multidimensional Operations

Slice Operation: Extracts a single layer of the cube



Roll up :
 Aggregates data by moving up the hierarchy





Types of OLAP

Relational OLAP (ROLAP)

- ROLAP stores data in relational databases and uses SQL for analysis.
- It is good for handling large amounts of detailed data.

Multidimensional OLAP (MOLAP)

- MOLAP stores data in multidimensional cubes for fast analysis.
- Not ideal for very large or sparse datasets.

Hybrid OLAP (HOLAP)

- HOLAP combines the strengths of ROLAP and MOLAP.
- Detailed data is stored in relational form, and summaries in cubes.

ORGANIZING WITH DIMENSIONAL MODELING



01

Facts = Numbers (e.g., sales, profit, quantity sold)

02

Dimensions = Context (Who, What, When, Where)

03

Data is arranged using a Star Schema

- Center: Facts
- Points: Dimensions

From Scattered Data to Smart Decisions

Step-by-step business data journey:

- Order placed (at store/website)
- OLTP: Fast daily system records the order
- ETL process: Data is extracted, cleaned, and loaded
- Data Warehouse: Stores organized, historical data
- OLAP: Business analysis — trends, insights, strategic decisions
- Better decisions: Accurate pricing, inventory, customer targeting