CSCI E-103: Reproducible Machine Learning
Lecture 06: Business Intelligence and Data Visualization

Harvard Extension School

Fall 2025

■ **Course:**	CSCI E-103: Reproducible Machine Learning
■ **Week:**	Lecture 06
■ **Instructors:**	Anindita Mahapatra & Eric Gieseke
■ **Objective:**	Understand Business Intelligence (BI) fundamentals, the Lakehouse architecture for BI, and practical Databricks SQL features including AI functions, Lakeview dashboards, and Genie

Contents

1 Lecture Overview

Key Summary

This lecture focuses on Business Intelligence (BI) analytics and data visualization within the context of modern data platforms:

Core Topics:

- **Data Lake vs. Data Warehouse** – Understanding the trade-offs
- **Lakehouse Architecture** – Combining the best of both worlds
- **Business Intelligence (BI)** – Definition, purpose, and comparison with BA
- **BI Personas** – The BI Analyst and their primary skill (SQL)
- **Data Modeling for BI** – Star Schema and dimensional modeling
- **ETL vs. Data Federation** – When to move data vs. query in place
- **Databricks SQL Features** – Serverless, Materialized Views, Streaming Tables
- **AI Functions in SQL** – Embedding LLMs directly in queries
- **Lakeview Dashboards and Genie** – Self-service BI and natural language analytics

Example:

The Crystal Ball Analogy Think of BI as the business executive's "crystal ball." Businesses want to use data to clearly see what's happening today (insight) and, more importantly, predict what will happen tomorrow (foresight) to gain competitive advantage.

2 Key Terminology Reference

Term	**Description**	**Full Name**	**Notes**
BI	Technology to collect, analyze, and visualize data for better business decisions (What happened? How?)	Business Intelligence	Reports, dashboards
BA	Uses past data to explain present and predict future (Why? What's next?)	Business Analytics	Statistics, predictive modeling
Data Warehouse	Fast, expensive "data library" storing only structured data with predefined schema	Data Warehouse (DW)	Schema-on-Write
Data Lake	Cheap, massive "data garage" storing all data types in raw format	Data Lake (DL)	Schema-on-Read
Lakehouse	Unified architecture: DW performance/reliability on DL's cheap storage	Lakehouse	DW + DL combined
Medallion Arch.	3-tier data organization: Bronze (raw) → Silver (cleaned) → Gold (aggregated)	Medallion Architecture	
Data Federation	Querying remote data sources without physically moving data	Data Federation	No ownership
View	Virtual table defined by a query; executed on access (no storage)	View	
Materialized View	Pre-computed query results stored physically for fast access	Materialized View (MV)	
Concurrency	How many simultaneous queries/operations the system can handle	Concurrency	KPI for BI
Latency	Time from query request to result delivery (delay)	Latency	KPI for BI

3 Evolution of Data Storage

3.1 Data Warehouse vs. Data Lake

Example:

Library vs. Garage Analogy **Data Warehouse (DW) = Well-organized library**

- Only accepts "books" (structured data)
- Librarian immediately catalogs and shelves everything (Schema-on-Write)
- Finding a specific book (BI query) is very fast and accurate
- Cannot store videotapes or photos (unstructured data)
- Expensive to build and maintain

Data Lake (DL) = Garage that stores everything

- Accepts books, photos, videos, broken bicycles... everything (all data types)
- Just throw things in raw (Schema-on-Read)
- Storage is very cheap
- Finding something later takes time (slower queries)
- Without management, becomes a **Data Swamp**

Dimension	**Data Lake**		**Data Warehouse**	
	Pros	**Cons**	**Pros**	**Cons**
Storage	All file types (open format)	Lower data quality; file-level access	High reliability; fine-grained access	Structured only; proprietary formats
Compute	Very economical (storage/compute separated)	Operational complexity	Easy to use; high concurrency, low latency	Expensive to scale (coupled)
Consumption	Rich tool ecosystem (ML, AI, DS)	Not optimized for BI	SQL-optimized (BI)	Limited ML/streaming support

3.2 The Two-Tier Problem and Lakehouse Solution

Historically, organizations used both systems together:

1. **First Generation (DW Only):** Only structured data via ETL into DW. BI only.
2. **Second Generation (Two-tier: Lake + DW):** All data stored in DL. Then ETL again to copy BI-relevant data into a separate DW. ML/DS uses DL; BI uses DW.

Caution

Problems with Two-Tier Architecture:

- **Data Duplication:** Same data stored in both Lake and Warehouse, wasting cost
- **Increased Complexity:** Managing and synchronizing two separate systems
- **Data Freshness Issues:** ETL from Lake to DW takes time; BI users may not see latest data

Definition:

Lakehouse A **Lakehouse** is a unified architecture that provides Data Warehouse capabilities (ACID transactions, governance, fast queries) on top of Data Lake's cheap, flexible, open storage (S3, ADLS). Example: Databricks with Delta Lake.

Lakehouse Advantages:

- **Single System:** No data duplication or separate system management
- **All Workloads:** BI, reporting, Data Science, ML from the **same single data source**
- **Cost Efficiency:** Warehouse performance at Lake costs
- **Freshness:** Data in one place = single source of truth; BI always queries latest data

3.3 Historical Evolution of Data Storage

1. **Spreadsheets:** Most primitive (CSV files)
2. **Data Warehouses:**
   - Bill Inmon: ER model-based, normalized (3NF) central DW
   - Ralph Kimball: Business-user focused, denormalized dimensional models (Star/Snowflake)
3. **MPP Databases:** Teradata, Greenplum – distributed data and compute across nodes; TB-scale but expensive
4. **NoSQL / BigTable:** Google's PB-scale columnar storage
5. **Hadoop / Data Lakes:** Horizontal scaling on commodity hardware; separated storage from compute
6. **Data Mesh / Fabric:** Decentralized ownership (Mesh) and unified access layers (Fabric)
7. **Lakehouse:** Current architecture combining DL + DW capabilities

Key Information

The Journey Continues: Technology keeps evolving. AI and LLMs are already transforming the field further. What comes after Lakehouse? Hard to say, but change is constant.

4 Business Intelligence (BI) Fundamentals

4.1 What is Business Intelligence?

Definition:

Business Intelligence (BI) **Business Intelligence** encompasses all technologies, applications, and processes used to collect, integrate, analyze, and present business information to support **better business decision-making**.

Core Philosophy of BI

> "Data is what you need to do analytics.
> Information is what you need to do business."

BI transforms raw data into actionable **information** that business leaders can use.

BI Components:

- **Data Analysis:** Data exploration and querying
- **Visual Analytics:** Charts, graphs, dashboards
- **Advanced Analytics:** Predictions, statistics (overlaps with BA)
- **Data Governance:** Quality, security, access control
- **Strategy Documentation:** Business mission and strategy

4.2 BI vs. BA: What's the Difference?

Business Intelligence (BI)	**Business Analytics (BA)**
Uses **past and present** data	Uses **past** data
Focuses on **"What"** and **"How"** happened (Descriptive)	Explains **"Why"** and predicts **"What will happen"** (Explanatory, Predictive)
Example Questions: • "What was last quarter's revenue?" (What) • "Which product sold most?" (Who) • "When did sales peak?" (When)	**Example Questions:** • "Why did that product sell well?" (Why) • "Will this trend continue?" (Prediction) • "What if we raise prices 10%?" (What-if)
Key Techniques: Reporting, Dashboards, OLAP, Ad-hoc queries	**Key Techniques:** Statistical analysis, Data mining, Predictive modeling, A/B testing

Key Information

BI Trend: Modern BI is evolving from purely "descriptive" analysis (past reporting) toward "prescriptive" analysis (recommending what actions to take).

4.3 The BI Process (5 Steps)

1. **Collect:** Integrate data from source systems (CRM, ERP, etc.) into warehouse/lakehouse via ETL
2. **Organize:** Structure data in analysis-friendly models (OLAP cubes, Star Schema)
3. **Analyze:** BI analysts query data using **SQL**
4. **Visualize:** Present results as charts, dashboards, reports
5. **Decide:** Executives and teams use visualized information for strategic decisions

5 BI Personas and Data Modeling

5.1 The BI Analyst Persona

While many roles exist in the BI workflow (Data Engineers, Data Scientists), the core **consumer** of BI is the **BI Analyst**.

- **Primary Skill: SQL** The BI Analyst's main tool is **SQL**. They use it to explore data, answer business questions, and extract data for dashboards.
- **Data Consumed: Curated Data** BI Analysts don't work with raw Bronze data. They primarily use data that Data Engineers have cleaned (Silver) and aggregated for business use (Gold).
- **Role: Analytics Engineering** Increasingly, BI Analysts are expected to model data and curate Gold tables themselves using SQL—this is called "Analytics Engineering."

5.2 Data Modeling for BI: Star Schema

BI queries must be very fast (low latency), so data must be structured optimally. The most popular approach is **Dimensional Modeling**, specifically the **Star Schema**.

Definition:

Star Schema A **Star Schema** looks like a "star":

- **Fact Table (Center):** Contains business event measurements (numeric data): sales_amount, quantity_sold
- **Dimension Tables (Points):** Surround the fact table, providing context: dim_customer, dim_product, dim_time

Example:

Star Schema for Online Store Sales

- **Fact_Sales:** {date_key, product_key, customer_key, sales_amount, quantity}
- **Dim_Time:** {date_key, date, month, year, quarter, day_of_week}
- **Dim_Product:** {product_key, product_name, category, brand}
- **Dim_Customer:** {customer_key, customer_name, city, country}

Query: "Q1 2025 sales by category for Seoul customers?" — Fast, simple JOINs!

Snowflake Schema: A variation where dimension tables are further normalized and linked to additional tables.

Caution

Data Vault Model: Data Vault uses Hubs (core business keys), Links (relationships), and Satellites (descriptive attributes). It's flexible for the Silver layer, but the **Gold layer for BI typically still uses Star Schema** for query performance.

6 ETL vs. Data Federation

Definition:

Data Federation **Data Federation** queries external data sources (Oracle, Redshift, Snowflake) **without physically moving the data**. It's **read-only** access where you don't "own" the data.

Very Important:

Key Difference: Ownership **ETL:** You extract, transform, and load data into your lakehouse. The lakehouse **owns** that data. Significant compute is spent curating through Bronze → Silver → Gold. **Federation:** You query data where it lives (external system). **No ownership**. Good for modest, reference/lookup data. Query pushdown translates your SQL to the remote system's native query.

When to Use Each:

- **Federation:** Small data volumes, reference/lookup data, one-off joins
- **ETL:** Large data volumes, performance-critical queries, when you need low latency

Example:

Federation Use Case Your core transactional data is in the lakehouse, but you need to join with a reference table from an on-prem Oracle system. Rather than ETL the entire Oracle table, you federate: write one SQL query that joins your lakehouse table with the Oracle table. The system translates the WHERE clause and does a "push down" to Oracle, bringing back only the needed rows.

7 Databricks SQL for BI

Databricks provides comprehensive BI and data warehousing capabilities through **Databricks SQL (DBSQL)**.

7.1 Platform Architecture

Source: All data types (structured, unstructured, streaming)

Ingest: ETL (data moves in) or Federation (query in place)

Transform: Medallion Architecture (Bronze → Silver → Gold)

Query and Process: DBSQL for BI; Spark/ML for Data Science

Governance: Unity Catalog for access control, lineage, audit

Engine: Photon (C++ rewrite of Spark for vectorized performance)

Serve/Analysis: Lakeview Dashboards, BI tool integrations, Lakehouse Apps

7.2 Key BI Features in DBSQL

1. Serverless Compute:

- Instant compute allocation—no waiting 3-6 minutes for VMs to spin up
- Auto-scaling based on workload
- Auto-termination after inactivity (saves cost)

2. Streaming Tables:

```sql
CREATE STREAMING TABLE my_streaming_table
AS SELECT * FROM cloud_files('/path/to/data', 'json');
```

Listing 1: Creating a Streaming Table from Cloud Storage

No complex code—just SQL to process streaming data automatically.

3. Materialized Views (MV):

```sql
CREATE MATERIALIZED VIEW revenue_by_route AS
SELECT route_id, SUM(fare_amount) as total_revenue
FROM trips
GROUP BY route_id;
```

Listing 2: Creating a Materialized View

Pre-computed results stored physically. Dashboards query MVs for instant response. MVs auto-update incrementally when source data changes.

4. Concurrency and Scaling:

- **Concurrency:** How many simultaneous queries can run without slowdown
- **Scaling:** Can the system handle growing data volumes?
- Ideal: Queries execute (green) without being queued (yellow)

5. Additional Features:

- SQL Editor with intelligent autocomplete
- Parameterized queries (different users see different data based on parameters)
- Query history and profiling
- Row-level security and column masking
- Geospatial support (H3)
- Workflow integration (dashboards as workflow tasks)

8 AI Functions in SQL

Databricks SQL allows embedding LLMs directly within SQL queries—a revolutionary capability.

8.1 ai_query(): External LLM Calls

```sql
SELECT
  sku_id,
  product_name,
  ai_query(
    "my-openai-endpoint",  -- Pre-registered model endpoint
    "You are a marketing expert. Generate a 30-word
     promotional text for product: " || product_name
  ) AS promotional_text
FROM retail_products;
```

Listing 3: Using ai_query() for Product Marketing

This embeds an LLM call within your SELECT statement, enriching your data with AI-generated content.

8.2 Built-in AI Functions

Databricks provides pre-built AI functions for common tasks—no external model setup needed:

```sql
-- Sentiment Analysis
SELECT ai_analyze_sentiment('I am happy');
-- Returns: 'positive'

-- Classification
SELECT ai_classify('My password is leaked',
                   ARRAY('urgent', 'not urgent'));
-- Returns: 'urgent'

-- Information Extraction
SELECT ai_extract('John Doe lives in New York',
                  ARRAY('person', 'location'));
-- Returns: {"person": "John Doe", "location": "New York"}

-- Grammar Correction
SELECT ai_fix_grammar('This sentence have some mistake');
-- Returns: 'This sentence has some mistakes'

-- Sensitive Data Masking
SELECT ai_mask('My email is john@example.com', ARRAY('email'));
-- Returns: 'My email is [MASKED]'
```

Listing 4: Built-in AI Functions

There are approximately 15-20 built-in AI functions covering sentiment, classification, extraction, summarization, masking, similarity, and more.

9 Lakeview Dashboards and Genie

9.1 Lakeview Dashboards

Definition:

Lakeview Databricks' built-in dashboard tool for data visualization directly within the platform.

Components:

- **Data Tab:** Define datasets that power the dashboard
- **Visualization Widgets:** Charts, graphs (bar, scatter, line, etc.)
- **Text Boxes:** Documentation and labels
- **Filters:** Interactive controls (dropdowns, date pickers)

Natural Language Creation: The built-in **Assistant** lets you create charts using natural language:

> "Show me number of trips by pickup zip code"

The assistant generates the visualization, which you can then customize.

Caution

When to Use Lakeview vs. PowerBI/Tableau:

- **Lakeview:** No additional licensing cost. Quick, operational dashboards for data engineers and analysts.
- **PowerBI/Tableau:** Sophisticated, polished dashboards for C-level executives with specific branding requirements.

9.2 Publishing and Sharing

Share (Internal): Grant view/edit access to other Databricks users in your organization.

Publish (External): Share with users who **don't have** Databricks accounts.

- Use "Embed credentials" option
- **Cost Warning:** When external users view a published dashboard and results aren't cached, compute costs are charged to the **publisher's account**
- Results are cached for 24 hours by default

9.3 Databricks Genie: Conversational AI Analytics

Definition:

Genie **Genie** is a conversational AI assistant available on published dashboards that allows users to ask questions in natural language.

Use Case Scenario: A marketing manager is viewing a dashboard created by the BI team. They wonder: "What if I filter to only male customers in their 20s?"

Previously, this would require a request to the BI team and weeks of waiting. Now, they simply ask Genie in natural language and get an instant answer.

How It Works:

1. **User Question:** "What is the average trip duration?"
2. **Genie Response:** "The average is 13.7 minutes."
3. **Transparency:** Genie shows the SQL query it generated:

```sql
SELECT AVG(dropoff_time - pickup_time)
FROM trips
WHERE pickup_time IS NOT NULL
  AND dropoff_time IS NOT NULL;
```

Very Important:

Why Genie Doesn't Hallucinate Unlike ChatGPT which tries to please users by answering any question (potentially making things up), Genie's scope is **strictly limited to the metadata of the specific tables** defined in the dashboard.

If you ask "What's the weather like?"—Genie will respond: "Sorry, I can only answer questions about the data available."

Genie generates **pure SQL** based on table schemas. No LLM creativity involved in the answer—just SQL execution.

Customizing Genie: You can provide instructions and example queries to improve Genie's understanding:

- "By trip duration, I mean duration in seconds, not minutes"
- Pre-written SQL for complex calculations

10 Databricks Platform Integration

10.1 SQL Warehouse Connectivity

Databricks SQL Warehouses can connect to many external tools:

- **BI Tools:** Tableau, PowerBI, Looker
- **Development:** dbt, Python, Java, NodeJS, Go
- **JDBC/ODBC:** Standard database connectivity

Connection details are available in the warehouse settings, including JDBC URLs.

10.2 Monitoring and Query History

Query History: Every query is logged with:

- Who ran it
- Duration (execution time vs. queue time)
- Status (success/failed)
- Query profile (memory usage, rows returned, scan details)

Warehouse Monitoring:

- Blue bars = queries running
- Orange bars = queries queued
- Ideal: mostly blue, minimal orange

Administrators use this to:

- Identify heavy users
- Debug failed queries
- Plan chargeback policies
- Optimize performance

10.3 Workflow Integration

Dashboards can be tasks in workflows:

- As new data arrives via pipeline, the dashboard auto-refreshes
- Combine Data Engineering, ML, and BI tasks in a single workflow
- Set up alerts if data volume drops (pipeline health monitoring)

11 One-Page Summary

Storage Comparison: Warehouse vs. Lake

- **Warehouse (DW):** Strict library (structured data, Schema-on-Write, BI/SQL optimized, high cost, high performance)
- **Lake (DL):** Everything garage (all data types, Schema-on-Read, ML/DS optimized, low cost, lower performance)

Lakehouse: Unified Architecture

Lake's cheap storage + Warehouse's performance/reliability/governance

- Single system for both BI and ML/DS workloads
- Eliminates data duplication and ETL complexity

BI vs. BA: Different Questions

- **BI:** "What happened?" (past/present reporting, dashboards)
- **BA:** "Why did it happen? What will happen?" (statistics, predictive modeling)

BI Analyst and Data Modeling

- **BI Persona:** Core skill is **SQL**
- **Data Modeling: Star Schema** (central Fact table + surrounding Dimension tables) is most efficient for BI queries (Kimball approach)

Databricks SQL Key Features

- **Serverless Compute:** No cluster boot wait time
- **Materialized Views:** Pre-computed queries for fast dashboards
- **Streaming Tables:** SQL-only streaming data processing
- **AI Functions:** LLMs embedded directly in SQL

SQL + AI: Intelligent Queries

- **ai_query():** Call external LLMs (GPT, etc.) from SQL
- **ai_analyze_sentiment(), ai_classify()...**: Built-in AI functions for instant sentiment, classification, extraction

Dashboards and AI Assistant: Lakeview and Genie

- **Lakeview:** Built-in Databricks dashboards (operational, free)
- **Genie:** Conversational AI on published dashboards. Natural language → SQL
- **Genie's Strength:** No hallucination—only references specified table metadata