[PulseLake: A Lakehouse-Based Fitbit Data Analysis System]

By

[Sandeep Divakaruni]

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE COURSE

CPSC-597: Project (Seminar)

Master of Science in Computer Science

CALIFORNIA STATE UNIVERSITY, FULLERTON

[September, 2025]

SUPERVISOR

Dr. Duy H. Ho

ABSTRACT

Modern fitness trackers generate massive amounts of health and activity data, but this raw information is often overwhelming and difficult to use. The motivation for this project is to help individuals and organizations turn wearable data into meaningful insights that support healthier lifestyles and better decision-making. The central question is: How can continuous Fitbit data be collected, cleaned, and organized to provide clear summaries of fitness activity?

To address this, the project developed a structured system that ingests raw Fitbit data such as heart rate readings, workout sessions, and gym visits, and transforms it into clear, reliable reports. The methodology involved data collection, cleansing to remove errors, and structured reporting of workout intensity, gym usage, and long-term trends.

The results show that users can easily track progress, set goals, and understand patterns, while gyms and wellness programs gain useful insights. This work concludes that wearable data, when properly managed, can improve both personal fitness and community wellness.

Key Words: [Fitness Data; Wearable Technology; Data Analysis; Health Insights; Wellness Tracking].

Chapter 1: Introduction

Wearable fitness devices such as Fitbit have become an integral part of modern health and wellness practices. These devices continuously record heart rate, workout intensity, and physical activity, creating a rich source of data with the potential to improve individual fitness, preventive healthcare, and community wellness initiatives [1]. The popularity of wearables reflects a broader trend toward data-driven decision-making, where personal health insights can empower individuals to monitor progress and make informed lifestyle choices [2].

Despite these opportunities, the raw data generated by wearables is often overwhelming and fragmented. Users may collect thousands of readings each day, but without proper organization, these records remain difficult to interpret [3]. Existing tools typically provide only basic statistics, such as step counts or average heart rate, without delivering deeper insights into workout effectiveness, gym utilization, or long-term fitness progress. In addition, challenges such as duplicate entries, missing values, and inconsistent formats reduce the reliability of these datasets [4]. As a result, both individuals and organizations lack clear, actionable knowledge derived from wearable data [4].

This project addresses these limitations by developing PulseLake: A Lakehouse-Based Fitbit Data Analysis System. The system applies a structured data processing framework, leveraging the Medallion architecture and Lakehouse design principles [5, 6, 7], to capture, clean, and organize Fitbit datasets—including continuous heart rate events, workout sessions, and gym activity—into meaningful outputs. Through this approach, the project generates reliable reports that highlight workout intensity, exercise duration, gym usage, and long-term activity trends. These outputs not only help individuals track progress and set realistic goals but also support wellness programs and healthcare organizations in promoting healthier lifestyles.

The main objectives of this project are:

1. **Comprehensive data processing:** capture and refine raw Fitbit datasets to ensure accuracy and reliability.
2. **Meaningful fitness summaries:** generate structured reports on workout intensity, session duration, and gym utilization
3. **Actionable health insights:** provide individuals and organizations with patterns and trends that guide informed decisions.

Chapter 2: Motivation

Wearable devices have evolved from simple step counters to sophisticated health-monitoring systems capable of tracking heart rate, sleep quality, workouts, calories burned, and minute-level activity trends. As the adoption of these devices increases, so does the volume of data they generate. However, organizations, researchers, and fitness platforms often struggle to extract value from this fast-growing data because it arrives in fragmented, inconsistent, and unprocessed forms.

A major motivation for this project is the growing need for systems that can handle continuous streams of wearable data while ensuring reliability, accuracy, and scalability. Most real-world fitness platforms ingest millions of small sensor events per day. These events often arrive out of order, may contain duplicate entries, and follow different schemas depending on the data source. Without a structured data pipeline, analysts must spend significant time manually cleaning and organizing data before they can perform even simple analysis.

Another motivation is the increasing industry shift toward *Lakehouse architectures.* Companies such as Fitbit, Peloton, Apple Health, and smart-gym manufacturers rely on end-to-end data systems that combine real-time streaming with large-scale batch processing. Data engineers are therefore expected to build pipelines that ingest live events, apply cleaning logic, manage schema changes, and produce refined tables for machine learning and analytics. This project provides hands-on experience with these industry-standard practices by integrating Kafka, Azure Data Factory, Databricks, and Delta Lake into a unified pipeline.

The project also supports organizational-level analytics, enabling fitness and health companies to study large-scale behavior trends, optimize operations, and improve data-driven decision-making through clean and unified wearable datasets.Clean and well-structured wearable data enables:

- Personalized fitness recommendations,
- Monitoring of long-term behavior patterns,
- Gym usage optimization,
- Understanding workout intensity and recovery,
- Early detection of unusual heart rate trends.

These insights are only possible when the underlying data platform is reliable, consistent, and capable of integrating multiple data sources.

Ultimately, the motivation is to build a system that transforms raw sensor data into meaningful information that supports better health decisions, creates value for users, and showcases modern data engineering principles through an end-to-end implementation.

Chapter 3: Problem Statement

Although wearable devices are generating unprecedented amounts of data, turning this raw information into reliable, analytics-ready datasets presents several challenges. The core problems addressed in this project are summarized as follows:

1. Inconsistent and noisy data streams: Sensor readings may contain missing values, duplicated rows, or incorrect timestamps. Streaming data from devices does not always arrive in order, and workout sessions must be reconstructed from start/stop events.

2. Multiple heterogeneous data sources: Wearable ecosystems combine different types of information: user profiles, gym visits, workout sessions, and heart rate signals. These sources operate at different rates and formats, requiring a unified architecture capable of merging both streaming and batch ingestion.

3. Lack of progressive data refinement: Raw data alone is not useful. It must be cleaned, validated, enriched, and transformed into higher-level summaries. Without well-defined Bronze, Silver, and Gold layers, downstream analytics become inconsistent and difficult to maintain.

4. Need for fault-tolerant, scalable processing: Real-time pipelines must handle failures, restarts, late data, and high-volume events. This requires checkpointing, idempotent MERGE operations, and a storage layer that supports ACID guarantees.

5. Absence of clear, actionable insights: Users and fitness analysts need dashboards that summarize trends such as workout frequency, gym usage, and heart rate behavior. Without structured Gold tables, visualization tools cannot compute reliable metrics.

Based on the challenges above, the central objective of this project is:

> *To design and implement a scalable, reliable, and near real-time Lakehouse pipeline that can ingest, clean, organize, and analyze Fitbit-style wearable data using Apache Kafka, Azure Data Factory, Azure Databricks, and Delta Lake.*

The system should integrate multiple data sources, apply streaming and batch processing, clean and refine the data through Bronze–Silver–Gold layers, and generate analytics-ready tables for visualization in Power BI. By achieving this objective, the project demonstrates how modern cloud technologies can support health-related analytics at scale.

Chapter 4: Literature Review

Wearable fitness devices enable continuous collection of heart rate and activity signals at scale, driving interest in data-driven wellness and preventive health [1, 2]. However, raw wearable streams are noisy, fragmented across sources, and difficult to interpret without a structured pipeline for quality control and context alignment [3, 4]. Modern data architectures notably the Lakehouse paradigm and the Medallion layering pattern unify reliable data engineering with analytics in a single platform [6, 5, 7]. Yet a gap remains between how wearable data is generated and how it is organized into concise, user-centric fitness insights (for example, workout intensity, gym utilization, and long-term trends).

Prior work on wearable analytics has focused on activity recognition and physiology (for example, step counting, sleep staging, or heart-rate-based effort) [8, 9], while production data engineering emphasizes reliability, governance, and scalable storage and compute [6, 7]. Few systems integrate both lines of work into an end-to-end pipeline that (i) ingests heterogeneous Fitbit datasets, (ii) enforces data quality, and (iii) delivers clear session-level summaries suitable for individuals and organizations.

Visual summaries can be valuable for illustrating how different approaches relate to each other. Figure 4.1 shows an example that presents the Medallion layering pattern (Bronze to Silver to Gold) using Delta Lake to improve data quality across batch and streaming sources, and to serve business intelligence dashboards and machine learning workloads.

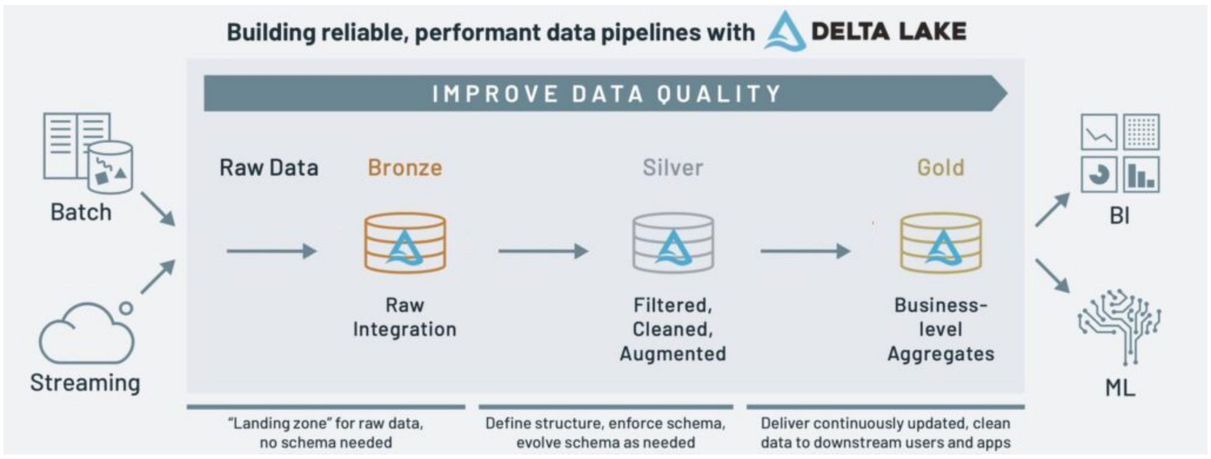

Building reliable, performant data pipelines with DELTA LAKE
IMPROVE DATA QUALITY
Batch
Streaming
Raw Data
Bronze
Raw Integration
Silver
Filtered, Cleaned, Augmented
Gold
Business-level Aggregates
BI
ML
"Landing zone" for raw data, no schema needed
Define structure, enforce schema, evolve schema as needed
Deliver continuously updated, clean data to downstream users and apps

Figure 4.1: Medallion layering pattern for improving data quality with Delta Lake. Raw data lands in Bronze (raw integration), is filtered, cleaned, and augmented in Silver, and is aggregated for business-level analysis in Gold. The architecture supports both batch and streaming inputs and delivers clean data for business intelligence dashboards and machine learning applications. Source: Databricks (Delta Lake) [5, 7].

As an overview, Table 4.1 presents selected studies with their core methodology, application domain, principal strengths, and primary limitations

By comparing these works, we find that research delivers useful task-level models and modern data platforms ensure reliability at scale, but a practical bridge is missing between raw device streams and concise, trustworthy fitness intelligence. **PulseLake** addresses this gap by operationalizing Lakehouse principles with Medallion layering for Fitbit data, producing quality-checked session summaries and long-horizon trends that are actionable for individuals and organizations.

Work	**Core Methodology**	**Strengths**	**Limitations**
Survey of wearable analytics(2019–2024) [1, 2]	Large-scale sensing and personal health analytics from wearable devices	Broad adoption; rich longitudinal signals; potential for preventive health	Fragmented and noisy data; privacy and standardization challenges
Activity recognition from wearable sensors [8]	Signal processing and machine learning on accelerometer and heart rate data	Accurate detection of steps, sleep, and activity types	Task-specific models; limited session-level context and reporting
Heart-rate–based fitness insight studies [9]	Heart rate zones, recovery, and intensity metrics	Physiology-aware interpretations; coaching value	Sensitive to missing values and duplicates; inconsistent event boundaries
Databricks (2020) [6]	Lakehouse architecture unifying data engineering and analytics on open tables	Simplifies the analytics stack; supports batch and streaming processing	Requires strong governance, cataloging, and security practices
Databricks (2023) [5]	Medallion layering: Bronze (raw ingestion), Silver (cleansed and conformed), Gold (analytics-ready)	Clear contracts and quality gates; reproducibility; lineage	Requires disciplined curation and metadata management
Palta (2022) [7]	Delta Lake transactions: ACID tables, time travel, and schema enforcement	Reliability for streaming and batch pipelines	Engine adoption and operational complexity in some settings
PulseLake (this project)	End-to-end pipeline for Fitbit data with data-quality checks and interpretable summaries	Session-level reports (minimum, average, and maximum heart rate; session duration; number of recordings), gym utilization, and longer-term trends; user-centric	Scope limited to Fitbit data; predictive or clinical modeling is out of scope

Table 4.1: Comparison of related work in wearable analytics and data architectures, highlighting how this project combines reliable data engineering with interpretable fitness summaries.

Chapter 5: Datasets

This chapter describes the datasets used for building and evaluating the PulseLake wearable data platform. Since real Fitbit or Apple Health data is restricted due to privacy and regulatory concerns, the project relies on synthetic yet realistic datasets that mimic the behavior of wearable devices and gym-based activity tracking systems. The datasets are generated incrementally to support continuous ingestion through both streaming and batch pipelines.

The synthetic datasets capture key aspects of user behavior, including demographic profiles, gym check-ins, workout sessions, and BPM (heart rate) signals. Each dataset is designed to resemble real-world conditions such as missing values, irregular arrival patterns, varying event frequencies, and out-of-order timestamps. This ensures that the data pipeline encounters conditions similar to production workloads.

5.0.1 Overview of Dataset Design

The synthetic dataset design is guided by three principles:

- **Realism:** Timestamps, intervals, session durations, BPM patterns, and gym usage frequencies are modeled to reflect how people realistically behave throughout a full year (2023).
- **Incremental Growth:** New batches of data are generated every two minutes. This allows the pipeline to demonstrate real-time behavior, late event handling, and continuous updates to Bronze, Silver, and Gold tables.
- **Multi-source Structure:** The system incorporates both streaming and batch data

sources, enabling a hybrid ingestion model. Kafka streams deliver high-frequency sensor data, while Azure SQL stores long-lived user and gym registration tables.

Together, these principles create a unified dataset ecosystem that is suitable for testing a full-scale Lakehouse architecture.

5.0.2 Dataset Categories

The system combines datasets from two ingestion paths:

1. Kafka Streaming Datasets

User Information Events (user_info) This dataset contains demographic and device-related details for each user. These records are generated at low frequency and represent changes in user profiles over time. They help associate sensor signals with specific demographic groups. Typical fields include:

- user_id
- dob
- update_type("new" or "update")
- gender
- sex
- first_name
- last_name
- address
- timestamp

Workout Start/Stop Events (workout) This dataset captures the beginning and end of each workout session for a user. Each workout produces a pair of events, allowing reconstruction of exercise sessions, durations, and activity frequency. These events are essential for linking BPM readings to meaningful workout intervals. Fields include:

- user_id
- workout_id
- action (start/stop)
- session_id
- timestamp

BPM Events (bpm) The BPM dataset provides continuous heart rate readings generated every few seconds. These events represent high-frequency physiological signals that fluctuate during workouts. This dataset enables calculation of average, minimum, and maximum heart rate values within each workout session. Fields include:

- device_id
- heartrate
- time

2. ADF Batch Datasets (Azure SQL)

Registered Users (registered_users) This batch dataset stores long-term user registration records from Azure SQL. It contains stable user metadata such as user_id, device_id,mac_address and registration_timestamp. This table is used primarily to maintain an authoritative list of all valid users in the system. Fields include:

- user_id
- device_id

- mac_address
- registration_timestamp

Gym Login Events (gym_logs) The gym logs dataset tracks user check-in and check-out activity across different gyms. These logs help analyze gym usage patterns, visit frequencies, and peak occupancy hours. They complement workout data by showing where and when users are physically present. Fields include:

- mac_address
- gym_id
- login_timestamp
- logout_timestamp

ADF extracts new or updated rows based on a watermark column. This complements streaming data by providing stable, relational updates.

5.0.3 Schema Descriptions

Tables 5.1, 5.2, 5.3, 5.4 and 5.5 summarize example schemas.

Table 5.1: Schema for User Information Events

Field	**Type**	**Description**
user_id	INT	Unique user identifier
dob	DATE	Dob of the user
gender	STRING	Male/Female/Other
sex	STRING	Male/Female/Other
update_type	STRING	new or update
first_name	STRING	First name of the user
last_name	STRING	First name of the user
address	STRING	Address of the user
timestamp	LONG	Epoch time of event

Table 5.2: Schema for Workout Events

Field	**Type**	**Description**
user_id	INT	User performing the workout
workout_id	STRING	Unique workout ID
action	STRING	start/stop
session_id	STRING	Unique session ID
timestamp	LONG	Event time

Table 5.3: Schema for BPM Events

Field	**Type**	**Description**
device_id	INT	Device generating heart rate signal
heartrate	FLOAT	Heart rate in BPM
time	LONG	Epoch timestamp

Table 5.4: Schema for Registered Users Table (ADF Source)

Field	**Type**	**Description**
user_id	INT	Unique user identifier
device_id	INT	Unique device identifier
mac_address	STRING	Unique device address
registration_timestamp	Long	Registration timestamp

Table 5.5: Schema for Gym Login Events Table (ADF Source)

Field	**Type**	**Description**
mac_address	STRING	Unique device address
gym_id	INT	Location of the gym visited
login_time	DATETIME	Timestamp of login
logout_time	DATETIME	Timestamp of logout (NULL if still active)

These structured definitions support downstream schema enforcement in the Silver layer.

5.0.4 Data Generation Strategy

Synthetic data was generated using a combination of Python, Faker library, and randomization functions. Key rules applied:

- **Timestamp Range:** All events fall within 2023-01-01 to 2023-12-31.

- **Streaming Ingestion (Kafka):** Kafka producers generate continuous streams of user, workout, and BPM events. These events are consumed by Databricks and written into the Landing Zone as raw JSON files. This ensures that Bronze ingestion is independent of Kafka availability and supports reprocessing if needed.
- **Incremental Batch Ingestion (ADF):** The Azure Data Factory pipeline runs every two minutes and extracts only new or updated SQL records using a watermark column. These records are written to the Landing Zone , rather than directly to the Bronze layer. This decouples source systems from the Lakehouse and provides a clean, file-based ingestion interface for downstream processing.
- **Landing Zone as a Decoupling Layer:** Both Kafka and ADF deliver data into a common test landing zone. Bronze ingestion reads exclusively from this zone, ensuring that the Lakehouse pipeline remains independent of source-system availability and supports reproducibility.
- **Workout Logic:** Start and stop events always occur in pairs with realistic gaps.
- **BPM Patterns:** Heart rate increases during workout events and stabilizes afterward.
- **Gym Frequency:** Users visit gyms with varying probabilities (heavy, medium, light users).

This strategy ensures that the pipeline processes a diverse, realistic stream of events and must handle out-of-order or late-arriving records.

5.0.5 Data Volume and Characteristics

Over multiple runs, the dataset grows rapidly:

- 3,000–5,000 BPM events per hour
- Hundreds of workout start/stop pairs
- Dozens of gym visit records

- Continuous updates to user profiles

Additional characteristics include:

- **Noise:** Random BPM spikes simulate sensor errors.
- **Out-of-Order Events:** Some events arrive late to test session reconstruction.
- **Duplicates:** Random duplicates verify MERGE logic.
- **Varied Frequency:** BPM is high-frequency; user_info is low-frequency.

These properties make the dataset challenging and suitable for evaluating the performance of the Lakehouse pipeline.

Chapter 6: Methodology

This chapter explains how the Fitbit Analysis System was built and tested. It describes the step-by-step process used to collect the data, clean it, organize it, and turn it into useful information. The approach follows standard data-engineering practices so the system is easy to reproduce, understand, and maintain. The chapter also covers how the system's architecture is designed, how data flows through different layers, what methods are used to transform and analyze the data, what datasets are involved, how the experiments were set up, and how the system's results were measured. Overall, this methodology provides a clear and structured way to show how the system works from beginning to end.

The goal of this methodology is to ensure that every part of the system can be explained, repeated, and improved over time. By breaking the work into structured stages such as data collection, processing, analysis, and evaluation it becomes easier to understand how each component contributes to the final results. This approach also helps in identifying issues, improving performance, and maintaining data quality throughout the pipeline. With a well-defined process, the Fitbit Analysis System can reliably turn raw Fitbit records into meaningful insights about user activity, workouts, and health trends.

6.1 System Architecture

The system adopts a modular architecture that organizes all Fitbit-related data processing into distinct components for ingestion, transformation, and analytics. This separation of responsibilities ensures that each data source such as registration events, user profile updates, BPM streams, and workout session logs is handled through a consistent and scalable workflow. Figure 6.1 presents the end-to-end architecture, highlighting how raw events are captured from operational systems and Kafka topics and then processed through the Bronze, Silver, and Gold layers of the Lakehouse model to generate enriched tables and summary insights.

All ingestion sources (Kafka + ADF) land into a common Landing Zone, which acts as the raw source for Bronze ingestion. This design decouples source systems from the Lakehouse and allows reproducibility, reprocessing, and schema evolution.

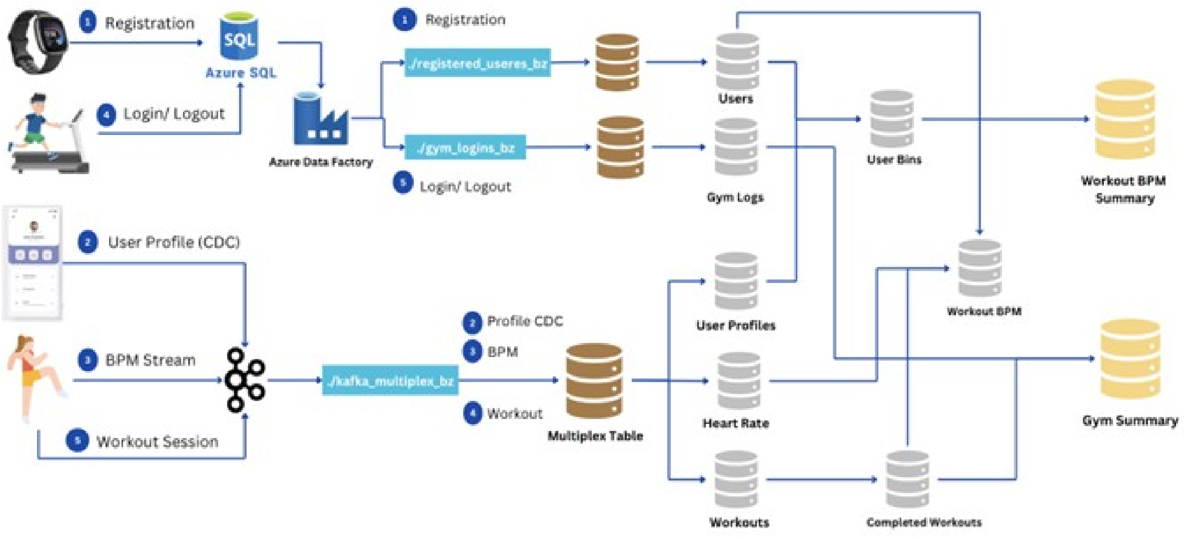


Figure 6.1: The figure presents the end-to-end data flow across the different domains of the Fitbit Analysis System. Registration events and gym login/logout activities (top and bottom) originate from the application layer, while continuous BPM and workout activity streams (middle) are generated by wearable devices. These raw inputs pass through CDC pipelines, Kafka multiplexing streams, and connector-based ingestion before entering the Lakehouse layers. Bronze tables capture the raw events, Silver tables organize and clean the data, and Gold tables produce aggregated summaries such as heart-rate trends, completed workout metrics, and gym usage analytics. Only essential attributes are displayed for clarity, and derived metrics are computed during downstream transformations.

6.2 Data Processing Workflow

The Fitbit Analysis System processes data through a structured, step-by-step workflow designed to convert raw records into clean, meaningful, and analytics-ready information. This workflow ensures consistency, accuracy, and traceability at every stage. The processing steps follow a standard Medallion pattern, where data moves through multiple refinement layers, each responsible for a specific transformation task.

The workflow begins in the Bronze layer, where raw data from various sources such as user registration events, heart-rate streams, workout session logs, and gym login information, Bronze does NOT ingest directly from Kafka or ADF. Instead, it reads JSON/CSV files stored in the Landing Zone, ensuring the pipeline is independent of real-time source system availability. No filtering or modification is performed in this stage. The purpose of the Bronze layer is to preserve the full fidelity of the incoming data and maintain a complete historical record that can be revisited when needed.

Next, the data transitions into the Silver layer, where it undergoes cleaning and normalization. This stage removes duplicate entries, corrects inconsistent timestamps, handles missing values, and applies schema enforcement so that all data follows a uniform structure. Domain specific logic is also applied here, such as identifying invalid heart-rate readings, aligning session boundaries, and standardizing user attributes. By the end of this stage, the data becomes reliable and suitable for analytical computations.

Finally, the refined data is transformed into high-level insights in the Gold layer. This step produces curated datasets that summarize user workouts, heart-rate trends, and gym utilization patterns. Aggregations such as average BPM, session duration, daily activity totals, and visit frequency are calculated here. The outputs from the Gold layer serve as the foundation for dashboards, reports, and personalized health insights.

Overall, the data processing workflow ensures a clear progression from raw ingestion to structured analytics, enabling the system to generate accurate and actionable results while maintaining transparency and reproducibility throughout the pipeline.

6.3 Data Ingestion Pipeline

The data ingestion pipeline is responsible for collecting Fitbit-related information from different sources and delivering it into the system in a reliable and organized manner. This pipeline supports both batch and streaming ingestion, ensuring that slow-changing records and high-frequency events are handled efficiently. The ingestion process is designed to maintain data freshness, scalability, and fault tolerance as data flows through the system.

The batch ingestion component handles data that changes less frequently, such as user registration records and gym login information stored in relational databases. Instead of writing directly into the Bronze layer, Azure Data Factory (ADF) performs scheduled extraction jobs—running every two minutes—that identify only new or updated records using a watermark column. These extracted datasets are written in raw form into the Landing Zone. From there, the Bronze ingestion process loads the files without modification, preserving completeness, traceability, and a reliable source-of-truth archive for downstream processing.

In contrast, the streaming ingestion component processes continuous data flows from multiple Kafka topics, including real-time heart-rate readings, workout start/stop events, and user activity updates. Structured Streaming jobs consume these events as they arrive and write them as raw JSON files into the Landing Zone, rather than directly into the Bronze layer. From there, the Bronze ingestion process loads the raw files incrementally using checkpointing and offset tracking to ensure exactly-once processing. This design guarantees that high-velocity streaming data remains consistent, ordered, and fully recoverable even in the event of failures or interruptions.

To support efficient downstream processing, the ingestion pipeline also adds essential metadata to each record, including ingestion timestamps and batch identifiers. This metadata enables traceability and allows the system to perform incremental updates, replay events when needed, and enforce data-quality checks in later stages

Overall, the ingestion pipeline provides a dependable foundation for the entire Fitbit Analysis System by seamlessly integrating multiple data sources and ensuring that all information enters the Lakehouse architecture in a consistent and trustworthy manner.

6.4 Data Transformation (Bronze → Silver → Gold)

The data transformation process organizes and refines information as it moves through the three layers of the Lakehouse architecture: Bronze, Silver, and Gold. Each layer plays a specific role in improving data quality, structure, and analytical usefulness. This step-by-step transformation ensures that raw Fitbit data is converted into reliable insights that support meaningful analysis.

The Bronze layer contains all incoming data in its original form. Records from batch ingestion and streaming pipelines are stored without modification to preserve completeness. Although this layer may include duplicates, inconsistent timestamps, or missing values, it provides a trusted source for all downstream processing. The Bronze layer serves as the foundation of the pipeline, ensuring that no information is lost and that transformations can always be reproduced.

In the Silver layer, the data undergoes cleaning and standardization. This includes removing duplicate entries, correcting incorrect or out-of-order timestamps, handling missing or invalid fields, and applying consistent data types. Schema enforcement is used to ensure that all records follow a uniform structure. Domain-specific logic is applied as well ,for example, filtering out unrealistic heart-rate values, validating login and logout pairs, and aligning workout session boundaries. By the end of this stage, the data becomes accurate, consistent, and ready for detailed analysis.

Silver-layer logic is implemented using custom Upserter classes (Upserter, CDCUpserter) that perform idempotent MERGE operations inside foreachBatch. This ensures:

- No duplicates
- Proper CDC behavior
- Handling late-arriving data
- Stable session reconstruction

The Gold layer focuses on generating final analytical outputs. This layer aggregates refined Silver-layer data into meaningful metrics such as average BPM, maximum and minimum heart rate during workouts, total workout duration, gym visit frequency, and daily or weekly activity summaries. The Gold layer structures the information into user-friendly tables designed for dashboards, reporting tools, and downstream consumption. These curated datasets allow users to easily interpret trends and extract insights about overall fitness patterns, exercise habits, and health indicators.

Overall, the Bronze $\rightarrow$ Silver $\rightarrow$ Gold transformation cycle ensures that data flows through a controlled refinement process. Each layer adds value by improving quality and structure, resulting in clean, well-organized, and analytics-ready datasets that support accurate and actionable insights.

6.5 Algorithms and Processing Logic

The Silver layer of the Fitbit Analysis System applies a structured set of algorithms that clean, de-duplicate, and merge data coming from the Bronze layer. Each algorithm is designed to handle real-time streaming updates, ensure idempotency, and maintain consistent data quality across all tables. The overall logic is modular, with separate upsert routines for users, gym activities, profile updates, workouts, heart-rate readings, and derived session metrics.

The first set of algorithms focuses on idempotent upserts, which ensure that repeated or out-of-order messages do not create duplicate records. For example, user registration data is ingested from the Bronze table and merged into the users table using user IDs as keys. Duplicate entries are filtered using watermarks and drop-duplicates rules so that only the newest and logically valid registration event is added.

Gym activity processing follows a simpler flow compared to workout sessions. Each gym visit is recorded as a single event containing the login time, logout time, gym location, and duration. Since these entries already represent a complete visit, no start–stop pairing is required. The Silver-layer upsert logic updates a record only if a newer version of the same login event is received for example, if a logout time is corrected or extended. This ensures that late updates, corrections, or incremental changes to a visit are incorporated without creating duplicates or inconsistencies.

A more advanced algorithm is used for Change Data Capture (CDC) on user profiles. Updates come from Kafka as raw JSON messages with fields such as name, gender, and address. The Silver logic ranks updates per user using the timestamp embedded in the message and selects only the newest version. The MERGE operation then replaces outdated records based on this timestamp, giving the system a clean, up-to-date profile for every user.

Workout processing follows a two-stage approach. First, start and stop actions are extracted from the streaming workouts topic. These events are cleaned using watermarking and de duplication and then merged into the workouts table. Next, start and stop events are paired to form completed-workouts entries. The system applies time-based constraints to match only realistic sessions for example, ensuring that stop events occur within three hours of the start time.Workout session pairing uses a rule that requires the STOP event to fall within 3 hours of the START event. Invalid pairs are dropped.

Heart-rate processing applies another set of rules that validate the integrity of incoming BPM signals. Duplicate readings at the same timestamp are removed, and invalid readings (such as non-positive BPM values) are filtered out. Valid signals are merged into the heart-rate table using device ID and timestamp as keys. These cleaned readings later support BPM aggregation and detailed session-level calculations.

Once both heart-rate readings and completed workouts are available, a join algorithm associates BPM samples with their corresponding workout sessions. This produces the workout-bpm table, which contains per-session heart-rate trajectories. Only readings that fall between the start and end times of a session—and within the system's time window tolerance—are included.

Finally, demographic enrichment is performed using the user bins algorithm. This logic derives age groups, gender categories, and geographic information by joining user profiles with the users table. The result is stored in the user bins dimension, which acts as a reusable lookup table for analytics in the Gold layer.

Across the entire Silver layer, these algorithms guarantee accurate, stateful, and conflict-free merging of streaming data. They allow the system to handle delayed events, partial updates, and incremental changes while maintaining correct relationships between user data, gym data, heart-rate events, and workout sessions. Together, these routines form the core processing workflow that transforms raw sensor and activity logs into reliable datasets ready for high-level analysis.

6.6 Dataset Description

The Fitbit Analysis System uses a combination of synthetic Fitbit-style sensor data, gym access logs, user registration records, and change-stream messages to simulate a realistic health-tracking environment. All datasets originate in the Bronze layer, where they are ingested from CSV files, Kafka topics, or generated events. These raw inputs are then refined through the Silver layer to create structured Delta tables suitable for downstream analysis.

- **User Registration Data:** This dataset contains the basic identifiers required to link users to devices and activity events. Each record includes a `user_id`, `device_id`, and `mac_address`, along with a registration timestamp. The data is append-only and is used to establish the core identity relationships between users and their fitness devices. After refinement, these records populate the users table in the Silver layer.
- **User Profile Change-Stream Data:** Profile updates, such as name, gender, date of birth, and address, are delivered through the `user_info` Kafka topic. Each message includes both the new profile information and an `update_type` field indicating whether the change represents a new record or an update to an existing one. These messages form a Change Data Capture (CDC) stream that maintains the most recent user profile while preserving the ability to rank updates by timestamp.
- **Gym Login and Logout Data:** Gym activity is recorded through login and logout events associated with a device's MAC address. The dataset includes fields such as gym, login, and logout timestamps. Because logout events may arrive late

or out of order, this data is processed to create a consistent record of gym visits. The refined version is stored in the `gym_logs` table.

- **Workout Event Stream:** Workout activity is delivered through the workout Kafka topic and contains event-level details such as `user_id`, `workout_id`, `session_id`, action (start or stop), and event timestamps. These messages represent the beginning and end of workout sessions. The dataset is used to construct both the workouts table and the paired `completed_workouts` dataset, where each session has a well-defined start and end time.

- **Heart-Rate (BPM) Stream:** Heart-rate readings are streamed through the bpm Kafka topic. Each record includes a `device_id`, timestamp, and heart-rate value. The dataset represents continuous biometric measurements collected during daily activities or workouts. After cleaning and validation, the data forms the `heart_rate` table and supports detailed session-level BPM tracking in the `workout_bpm table`.

- **Derived Silver-Layer Datasets:** Once raw events have been validated, merged, and de-duplicated, they are transformed into structured Delta tables:
  - **User:** definitive mapping of users to devices
  - **gym_logs:** cleaned gym session records
  - **user_profile:** latest profile snapshot per user
  - **workouts:** individual workout start/stop events
  - **completed_workouts:** paired workout sessions
  - **heart_rate:** validated BPM measurements
  - **workout_bpm:** heart-rate readings aligned to sessions
  - **user_bins:** demographic buckets for analysis

  These refined datasets provide a consistent and reliable foundation for high-level insights in the Gold layer, where activity summaries, health metrics, and visual dashboards are generated.

6.7 Experimental Setup

The experimental setup defines the computing environment, platform configuration, and execution settings used to implement and evaluate the Fitbit Analysis System. The system was deployed using a Lakehouse-based architecture running on Databricks, with data stored in Azure Data Lake Storage Gen2 and real-time events streamed through Confluent Kafka. This section summarizes the hardware, software, storage, and pipeline configuration used during experimentation.

- **Computing Environment:** All experiments were performed on Databricks using a shared workspace configured with the following environment:
  - **Compute Engine:** Databricks Runtime (DBR) supporting Apache Spark Structured Streaming
  - **Cluster Mode:** Single-user or shared interactive cluster
  - **Cluster Size:**Multi-node cluster with auto scaling (small–medium configuration)
  - **Storage:**Azure Data Lake Storage Gen2 (ADLS) as the primary data lake
  - **File System:**Unity Catalog-enabled tables using Delta Lake format

  This environment provides fault tolerance, autoscaling, and optimized IO for streaming workloads.

- **Data Sources and Ingestion Configurations:** The system processes data from two primary sources:
  - **Confluent Kafka topics:**
    - `user_info` for profile change events
    - workout for start/stop activity
    - bpm for continuous heart-rate readings
  - **Azure SQL / CSV-based Bronze tables:**
    - `registered_users_bz`

        * `gym_logins_bz`
        * `kafka_multiplex_bz`(all Kafka topics ingested into one Delta table)

  Kafka streams were read using processing-time triggers, watermarking, and checkpointing. The ingest layer stored all data in Bronze Delta tables to ensure reproducibility and replay capability.

- **Silver Layer Stream Settings:**Each Silver table was built using Structured Streaming + Delta MERGE. The following stream configurations were used:
    - **Trigger Mode:**
        * availableNow for batch-style processing during testing
        * Fixed processing interval (10–30 seconds) for continuous runs
    - **Watermark:**
        * 30 seconds on all time-based fields to manage late-arriving events
        * Used in deduplication and streaming joins
    - **State Management:**
        * 3-hour bound for workout session pairing
        * Per-table checkpoints under /checkpoints/¡table¿
    - **Upsert Method:**
        * Custom Upserter and CDCUpserter classes using foreachBatch
        * Idempotent MERGE statements to avoid duplicates
- **Storage Layout:**The data lake followed a standard Medallion layout:
    - /mnt/¡storage¿/bronze
    - /mnt/¡storage¿/silver
    - /mnt/¡storage¿/gold

  This structure supports independent scaling, easier debugging, and clean lineage from raw data to curated insights.

- **Execution Parameters:** Experiments were run with the following parameters:

- **maxOffsetsPerTrigger:**Up to 500,000 offsets during ingestion
- **maxFilesPerTriggerr:**Default or tuned based on cluster capacity
- **Deduplication Columns:**Keys such as `user_id, time` or `device_id, time`
- **Merge Keys:**Defined per table to guarantee correct update behavior
- **Enhanced Validation:**Additional filters for invalid BPM values and inconsistent timestamps

Experiments were executed using multiple datasets grouped into test sets (e.g., set 1 vs. set 2).

6.8 Evaluation Metrics

The Fitbit Analysis System is evaluated using a set of metrics that measure how accurately, completely, and consistently the pipeline processes streaming data. These metrics ensure that the Silver and Gold datasets produced by the system are reliable for analysis.

- **Data Accuracy**This metric checks whether raw events are transformed correctly into refined tables.
  - **Upsert correctness:**MERGE operations must insert new records and update existing ones without duplicates.
  - **Latest-record correctness:**For user profiles, the system must retain only the most recent update per user.
  - **Workout pairing accuracy:**Each workout should link the correct start and stop events, and invalid pairs should be discarded.
- **Completeness** Completeness measures whether all valid records from the Bronze layer appear in the Silver layer.
  - **Record count verification:**Expected row counts are compared with actual results (e.g., number of users, workouts, BPM readings).
  - **Late-event handling:**Watermarks are tested to confirm that the system accepts slightly late events while dropping extremely outdated ones.

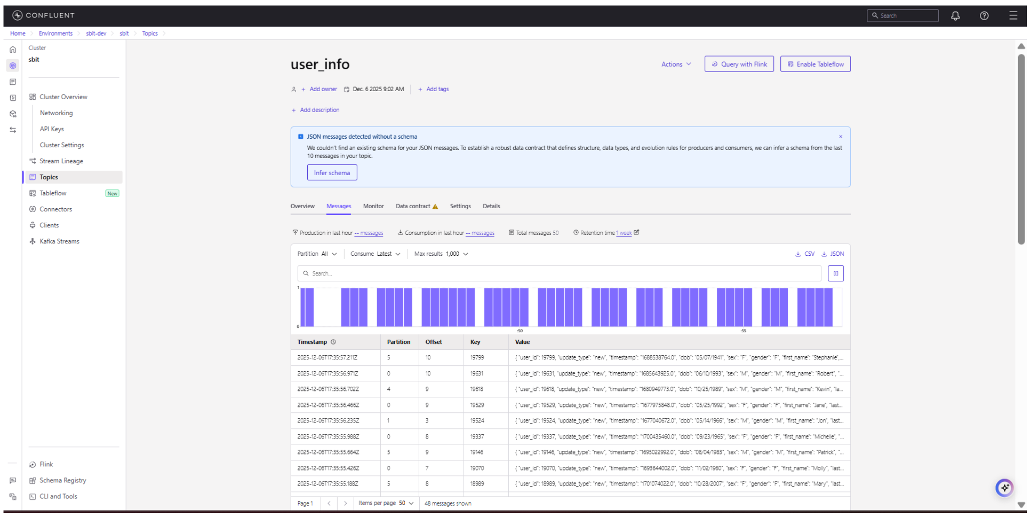
- **Consistency and Idempotency** This evaluates whether the pipeline produces the same results when re-run.
  - Reprocessing should not create additional duplicates.
  - MERGE operations must yield identical outputs on repeated runs.
  - Checkpoints must ensure streams resume reliably.
- **Performance**Performance metrics focus on how efficiently the streaming jobs run.
  - **Micro-batch latency**Time required to process each batch of events.
  - **Throughput:**Number of records processed per trigger.
- **Robustness**Robustness ensures the system behaves correctly under real-world conditions.
  - **Noise filtering:**Invalid BPM readings, missing logout events, or malformed data must be handled safely.
  - **Stable stateful joins:**Session pairing and BPM matching should work even with out-of-order or slightly delayed events.

6.9 Source System Snapshots

This section provides visual evidence of the underlying source systems and ingestion pipelines used in the project, including Kafka topics, Azure SQL tables, landing zone storage structure, and Azure Data Factory incremental ingestion pipelines. These screenshots complement the methodology by demonstrating how the system is configured and executed in practice.

6.9.1 Kafka Topics and Sample Events

The following screenshots show the Kafka topics used in the project (`user_info`, `workout`, and `bpm`) along with example messages.



user_info

Dec. 6 2025 9:02 AM

JSON messages detected without a schema

We couldn't find an existing schema for your JSON messages. To establish a robust data contract that defines structure, data types, and evolution rules for producers and consumers, we can infer a schema from the last 10 messages in your topic.

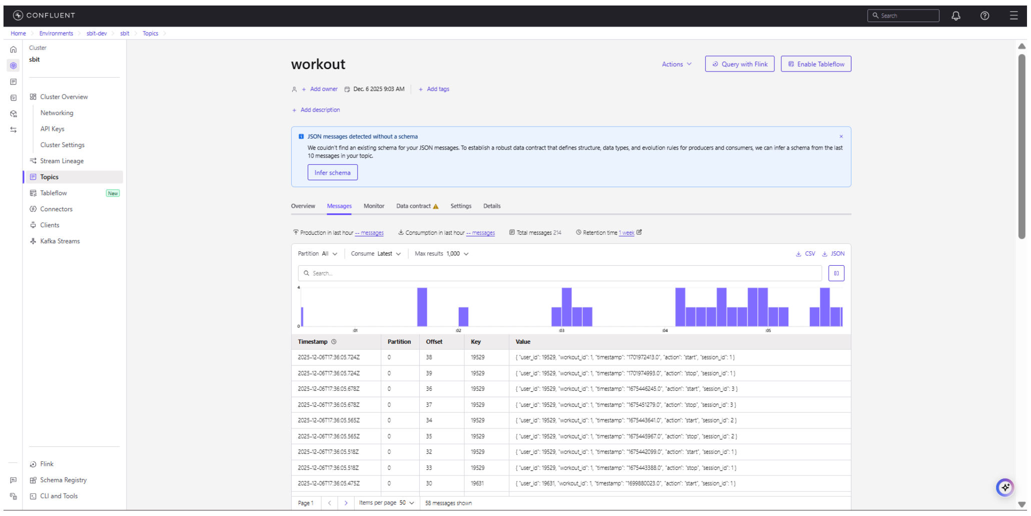
Production in last hour -- messages · Consumption in last hour -- messages · Total messages 50 · Retention time 1 week

Partition All · Consume Latest · Max results 1,000

Timestamp	Partition	Offset	Key	Value
2025-12-06T17:35:57.211Z	5	10	19799	{ "user_id": 19799, "update_type": "new", "timestamp": "1688538764.0", "dob": "05/07/1941", "sex": "F", "gender": "F", "first_name": "Stephanie",...
2025-12-06T17:35:56.971Z	0	10	19631	{ "user_id": 19631, "update_type": "new", "timestamp": "1685643925.0", "dob": "06/10/1993", "sex": "M", "gender": "M", "first_name": "Robert", "...
2025-12-06T17:35:56.702Z	4	9	19618	{ "user_id": 19618, "update_type": "new", "timestamp": "1680949773.0", "dob": "10/25/1989", "sex": "M", "gender": "M", "first_name": "Kevin", "la...
2025-12-06T17:35:56.466Z	0	9	19529	{ "user_id": 19529, "update_type": "new", "timestamp": "1677975848.0", "dob": "05/25/1992", "sex": "F", "gender": "F", "first_name": "Jane", "last...
2025-12-06T17:35:56.235Z	1	3	19524	{ "user_id": 19524, "update_type": "new", "timestamp": "1677040672.0", "dob": "05/14/1966", "sex": "M", "gender": "M", "first_name": "Jon", "last...
2025-12-06T17:35:55.988Z	0	8	19337	{ "user_id": 19337, "update_type": "new", "timestamp": "1700435460.0", "dob": "09/23/1965", "sex": "F", "gender": "F", "first_name": "Michelle", "...
2025-12-06T17:35:55.664Z	5	9	19146	{ "user_id": 19146, "update_type": "new", "timestamp": "1695022992.0", "dob": "08/04/1983", "sex": "M", "gender": "M", "first_name": "Patrick", "...
2025-12-06T17:35:55.426Z	0	7	19070	{ "user_id": 19070, "update_type": "new", "timestamp": "1693644002.0", "dob": "11/02/1960", "sex": "F", "gender": "F", "first_name": "Molly", "last...
2025-12-06T17:35:55.188Z	5	8	18989	{ "user_id": 18989, "update_type": "new", "timestamp": "1701074022.0", "dob": "10/28/2007", "sex": "F", "gender": "F", "first_name": "Mary", "last...

Page 1 · Items per page 50 · 48 messages shown

Figure 6.2: This topic receives incremental user-profile events such as name updates, gender, and address changes. The screenshot shows partition distribution, message offsets, and raw JSON payloads consumed by the streaming ingestion pipeline



workout

Dec. 6 2025 9:03 AM

JSON messages detected without a schema

We couldn't find an existing schema for your JSON messages. To establish a robust data contract that defines structure, data types, and evolution rules for producers and consumers, we can infer a schema from the last 10 messages in your topic.

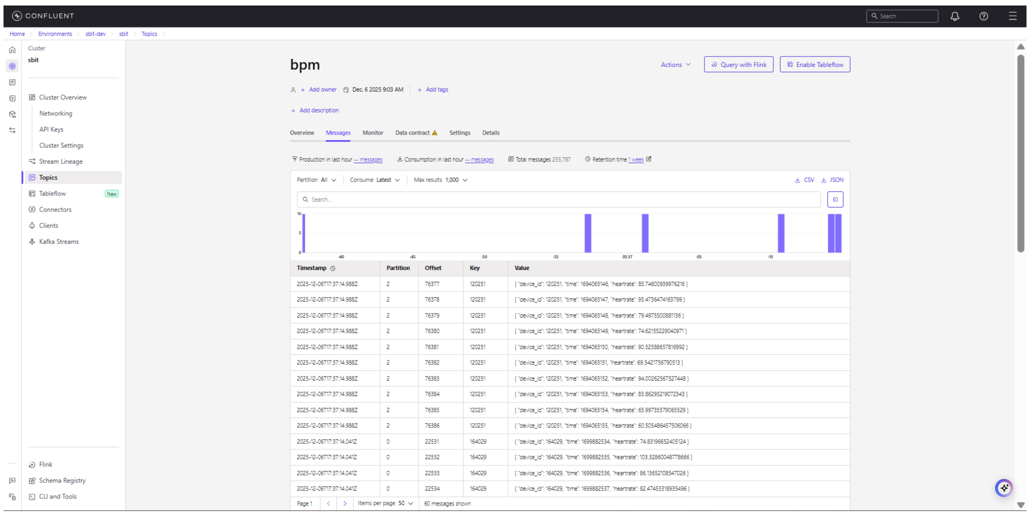
Production in last hour -- messages · Consumption in last hour -- messages · Total messages 214 · Retention time 1 week

Partition All · Consume Latest · Max results 1,000

Timestamp	Partition	Offset	Key	Value
2025-12-06T17:36:05.724Z	0	38	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1701972413.0", "action": "start", "session_id": 1 }
2025-12-06T17:36:05.724Z	0	39	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1701974993.0", "action": "stop", "session_id": 1 }
2025-12-06T17:36:05.678Z	0	36	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675446245.0", "action": "start", "session_id": 3 }
2025-12-06T17:36:05.678Z	0	37	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675451279.0", "action": "stop", "session_id": 3 }
2025-12-06T17:36:05.565Z	0	34	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675443641.0", "action": "start", "session_id": 2 }
2025-12-06T17:36:05.565Z	0	35	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675445967.0", "action": "stop", "session_id": 2 }
2025-12-06T17:36:05.518Z	0	32	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675442099.0", "action": "start", "session_id": 1 }
2025-12-06T17:36:05.518Z	0	33	19529	{ "user_id": 19529, "workout_id": 1, "timestamp": "1675443388.0", "action": "stop", "session_id": 1 }
2025-12-06T17:36:05.475Z	0	30	19631	{ "user_id": 19631, "workout_id": 1, "timestamp": "1699880023.0", "action": "start", "session_id": 1 }

Page 1 · Items per page 50 · 58 messages shown

Figure 6.3: The topic captures start and stop events for workout sessions, each associated with a unique session ID. The chart displays message frequency across partitions, while the table lists timestamps, offsets, and event payloads used for session reconstruction in the Silver layer.



bpm

Dec. 6 2025 9:03 AM

Production in last hour -- messages   Consumption in last hour -- messages   Total messages 255,787   Retention time 1 week

Partition All   Consume Latest   Max results 1,000

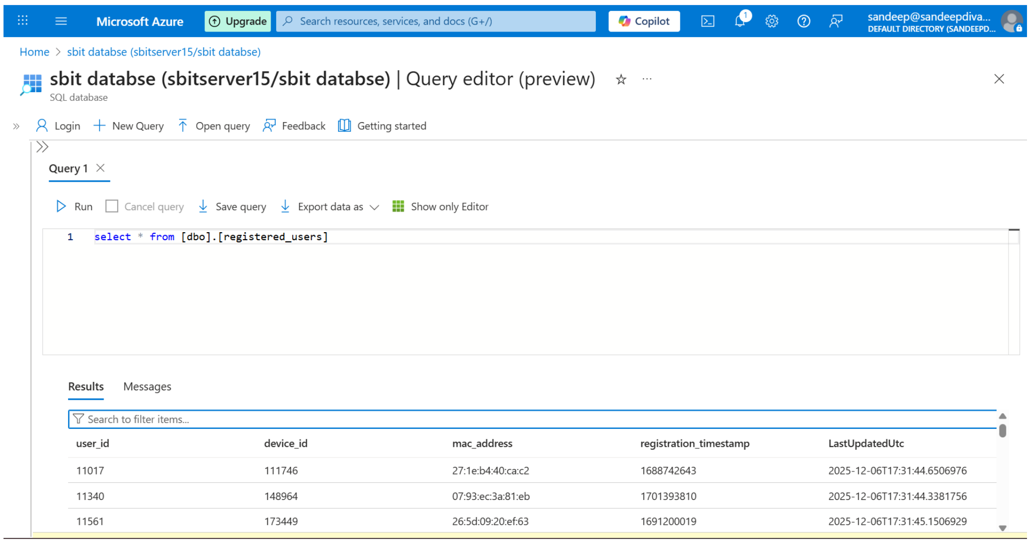
Timestamp	Partition	Offset	Key	Value
2025-12-06T17:37:14.988Z	2	76377	120251	{ "device_id": 120251, "time": 1694065146, "heartrate": 85.7460093997 6216 }
2025-12-06T17:37:14.988Z	2	76378	120251	{ "device_id": 120251, "time": 1694065147, "heartrate": 95.4736474163799 }
2025-12-06T17:37:14.988Z	2	76379	120251	{ "device_id": 120251, "time": 1694065148, "heartrate": 79.4975500881136 }
2025-12-06T17:37:14.988Z	2	76380	120251	{ "device_id": 120251, "time": 1694065149, "heartrate": 74.62155229040971 }
2025-12-06T17:37:14.988Z	2	76381	120251	{ "device_id": 120251, "time": 1694065150, "heartrate": 90.52588657816992 }
2025-12-06T17:37:14.988Z	2	76382	120251	{ "device_id": 120251, "time": 1694065151, "heartrate": 69.542175679 0513 }
2025-12-06T17:37:14.988Z	2	76383	120251	{ "device_id": 120251, "time": 1694065152, "heartrate": 94.00262567327448 }
2025-12-06T17:37:14.988Z	2	76384	120251	{ "device_id": 120251, "time": 1694065153, "heartrate": 83.8629321907 2343 }
2025-12-06T17:37:14.988Z	2	76385	120251	{ "device_id": 120251, "time": 1694065154, "heartrate": 65.99735579065529 }
2025-12-06T17:37:14.988Z	2	76386	120251	{ "device_id": 120251, "time": 1694065155, "heartrate": 60.505486457506066 }
2025-12-06T17:37:14.041Z	0	22531	164029	{ "device_id": 164029, "time": 1699882534, "heartrate": 74.83196652405124 }
2025-12-06T17:37:14.041Z	0	22532	164029	{ "device_id": 164029, "time": 1699882535, "heartrate": 103.32860048778666 }
2025-12-06T17:37:14.041Z	0	22533	164029	{ "device_id": 164029, "time": 1699882536, "heartrate": 86.13652108547026 }
2025-12-06T17:37:14.041Z	0	22534	164029	{ "device_id": 164029, "time": 1699882537, "heartrate": 82.47453318935496 }

Page 1   Items per page 50   60 messages shown

Figure 6.4: This topic streams continuous heart-rate readings from wearable devices at high volume. The screenshot shows the high-frequency event pattern, with each message capturing device ID, timestamp, and BPM values that feed real-time health analytics.

6.9.2 Azure SQL Source Tables

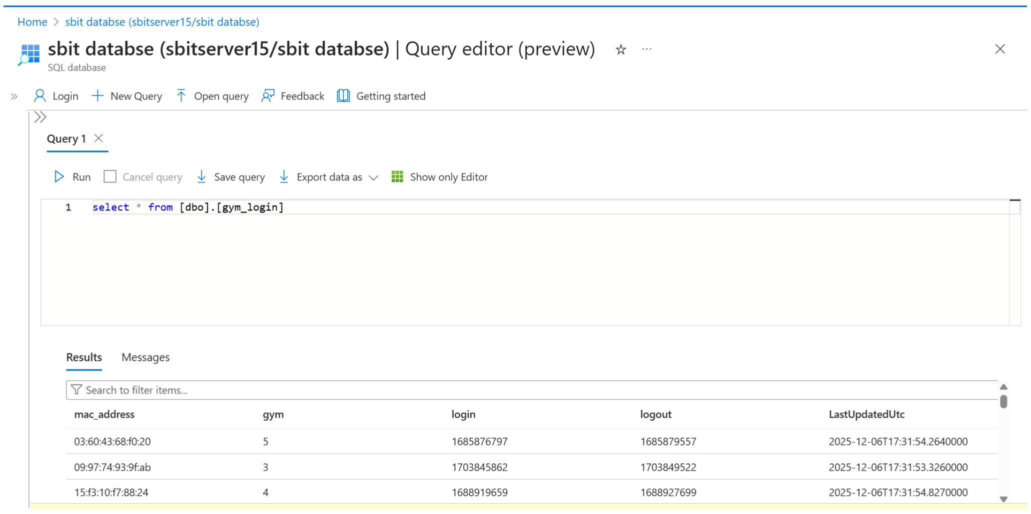
The Azure SQL Database stores slowly changing datasets such as user registrations. These records are ingested into the landing zone using Azure Data Factory's incremental copy mechanism.


Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Copilot
sandeep@sandeepdiva...
DEFAULT DIRECTORY (SANDEEPD...
Home > sbit databse (sbitserver15/sbit databse)
sbit databse (sbitserver15/sbit databse) | Query editor (preview)
SQL database
Login
New Query
Open query
Feedback
Getting started
Query 1
Run
Cancel query
Save query
Export data as
Show only Editor
Results
Messages
Search to filter items...

```
select * from [dbo].[registered_users]
```

user_id	device_id	mac_address	registration_timestamp	LastUpdatedUtc
11017	111746	27:1e:b4:40:ca:c2	1688742643	2025-12-06T17:31:44.6506976
11340	148964	07:93:ec:3a:81:eb	1701393810	2025-12-06T17:31:44.3381756
11561	173449	26:5d:09:20:ef:63	1691200019	2025-12-06T17:31:45.1506929

Figure 6.5: Azure SQL table registered_users, containing slowly changing user-profile attributes such as user ID, device ID, MAC address, registration timestamp, and a LastUpdatedUtc watermark. This table represents Type-1 user dimension data ingested by Azure Data Factory's incremental pipeline.


Home > sbit databse (sbitserver15/sbit databse)
sbit databse (sbitserver15/sbit databse) | Query editor (preview)
SQL database
Login
New Query
Open query
Feedback
Getting started
Query 1
Run
Cancel query
Save query
Export data as
Show only Editor
Results
Messages
Search to filter items...

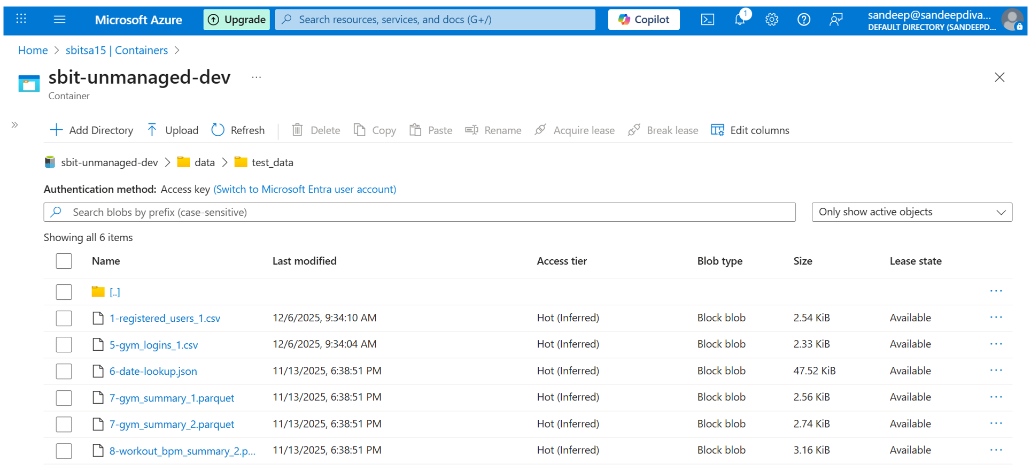
```
select * from [dbo].[gym_login]
```

mac_address	gym	login	logout	LastUpdatedUtc
03:60:43:68:f0:20	5	1685876797	1685879557	2025-12-06T17:31:54.2640000
09:97:74:93:9f:ab	3	1703845862	1703849522	2025-12-06T17:31:53.3260000
15:f3:10:f7:88:24	4	1688919659	1688927699	2025-12-06T17:31:54.8270000

Figure 6.6: Azure SQL table gym_login, which stores gym entry records generated by synthetic batch data. Each row contains a MAC address, gym identifier, login time, logout time, and an automatically updated LastUpdatedUtc watermark column used by Azure Data Factory for incremental ingestion.

6.9.3 Landing Zone Folder Structure

All raw data whether from Kafka or Azure SQL is first placed into a decoupled landing zone. This design separates source ingestion from pipeline ingestion and supports replay and debugging.



Home > sbitsa15 | Containers >

sbit-unmanaged-dev
Container

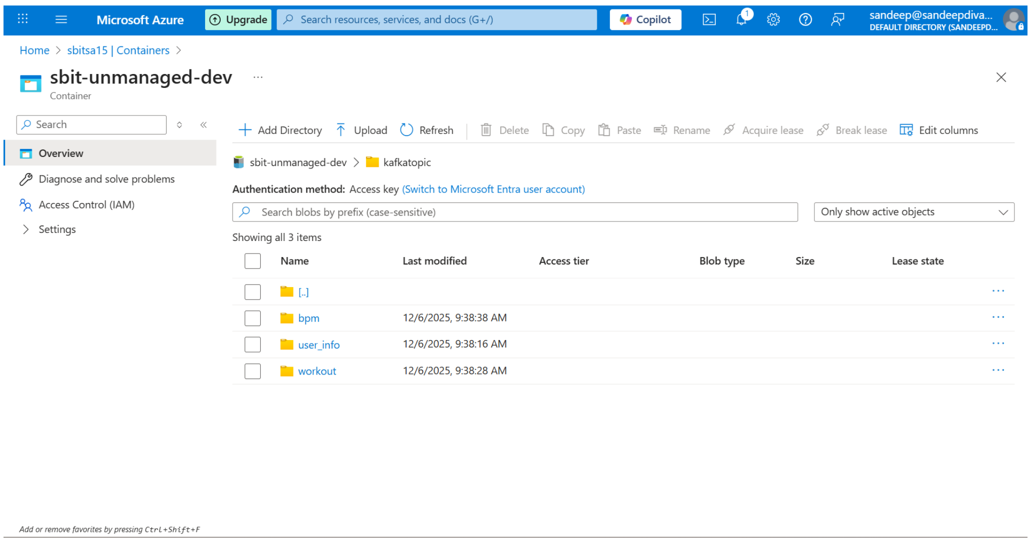
sbit-unmanaged-dev > data > test_data

Authentication method: Access key (Switch to Microsoft Entra user account)

Showing all 6 items

Name	Last modified	Access tier	Blob type	Size	Lease state
[..]					
1-registered_users_1.csv	12/6/2025, 9:34:10 AM	Hot (Inferred)	Block blob	2.54 KiB	Available
5-gym_logins_1.csv	12/6/2025, 9:34:04 AM	Hot (Inferred)	Block blob	2.33 KiB	Available
6-date-lookup.json	11/13/2025, 6:38:51 PM	Hot (Inferred)	Block blob	47.52 KiB	Available
7-gym_summary_1.parquet	11/13/2025, 6:38:51 PM	Hot (Inferred)	Block blob	2.56 KiB	Available
7-gym_summary_2.parquet	11/13/2025, 6:38:51 PM	Hot (Inferred)	Block blob	2.74 KiB	Available
8-workout_bpm_summary_2.p...	11/13/2025, 6:38:51 PM	Hot (Inferred)	Block blob	3.16 KiB	Available

Figure 6.7: Azure Blob Storage "test_data" folder showing synthetic datasets used for pipeline testing. The folder contains CSV files produced by Azure Data Factory



Home > sbitsa15 | Containers >

sbit-unmanaged-dev
Container

sbit-unmanaged-dev > kafkatopic

Authentication method: Access key (Switch to Microsoft Entra user account)

Showing all 3 items

Name	Last modified	Access tier	Blob type	Size	Lease state
[..]					
bpm	12/6/2025, 9:38:38 AM				
user_info	12/6/2025, 9:38:16 AM				
workout	12/6/2025, 9:38:28 AM				

Figure 6.8: Azure Blob Storage "kafkatopic" landing zone that receives raw Kafka topic data before Bronze processing. Each subfolder—bpm, user_info, and workout—stores raw JSON messages streamed from Confluent Kafka, decoupling ingestion from downstream Delta Lake processing.

6.9.4 Azure Data Factory Incremental Pipeline

Azure Data Factory runs every two minutes and extracts only new or updated rows from Azure SQL using watermark-based filtering.

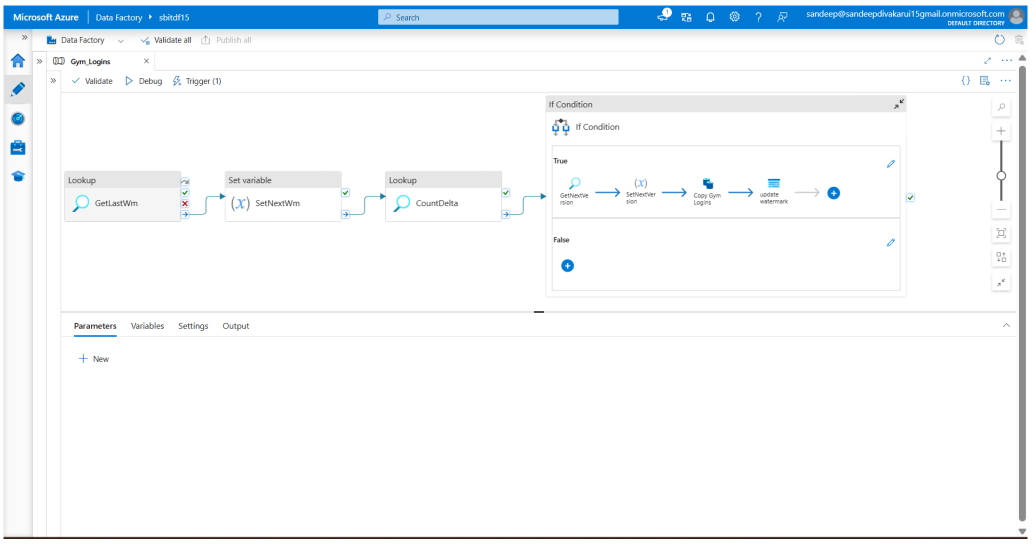


Figure 6.9: Azure Data Factory pipeline for incremental ingestion of gym login events. The workflow retrieves the last processed watermark, computes the new delta using a lookup query, and conditionally copies only the newly added records. Upon successful ingestion, the pipeline updates the watermark table to maintain idempotent batch processing.

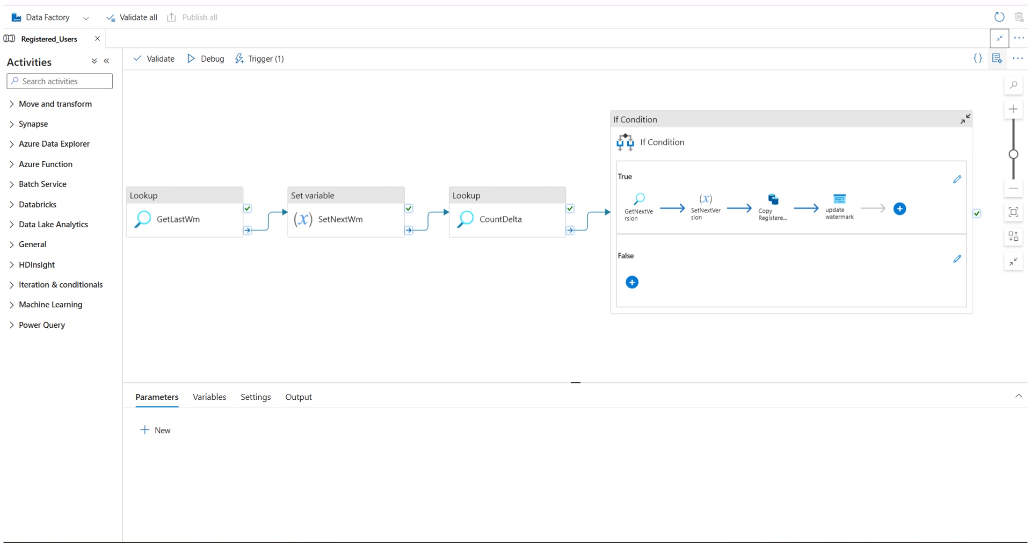


Figure 6.10: Azure Data Factory pipeline for loading newly registered users into the Bronze layer. Similar to the gym login workflow, the pipeline fetches the previous watermark, determines the number of new records, ingests only the incremental batch, and updates the watermark to ensure efficient and repeatable ingestion.

Chapter 7: Analytical Results and Dashboard Insights

7.0.1 BPM Demographics Dashboard

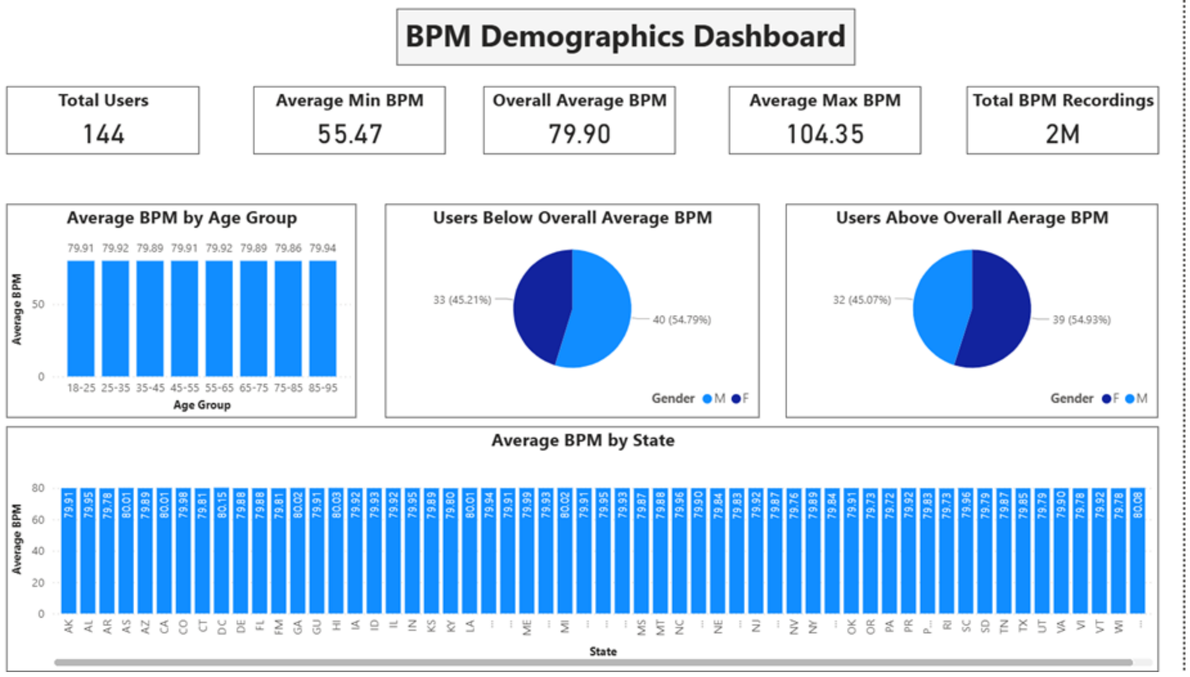
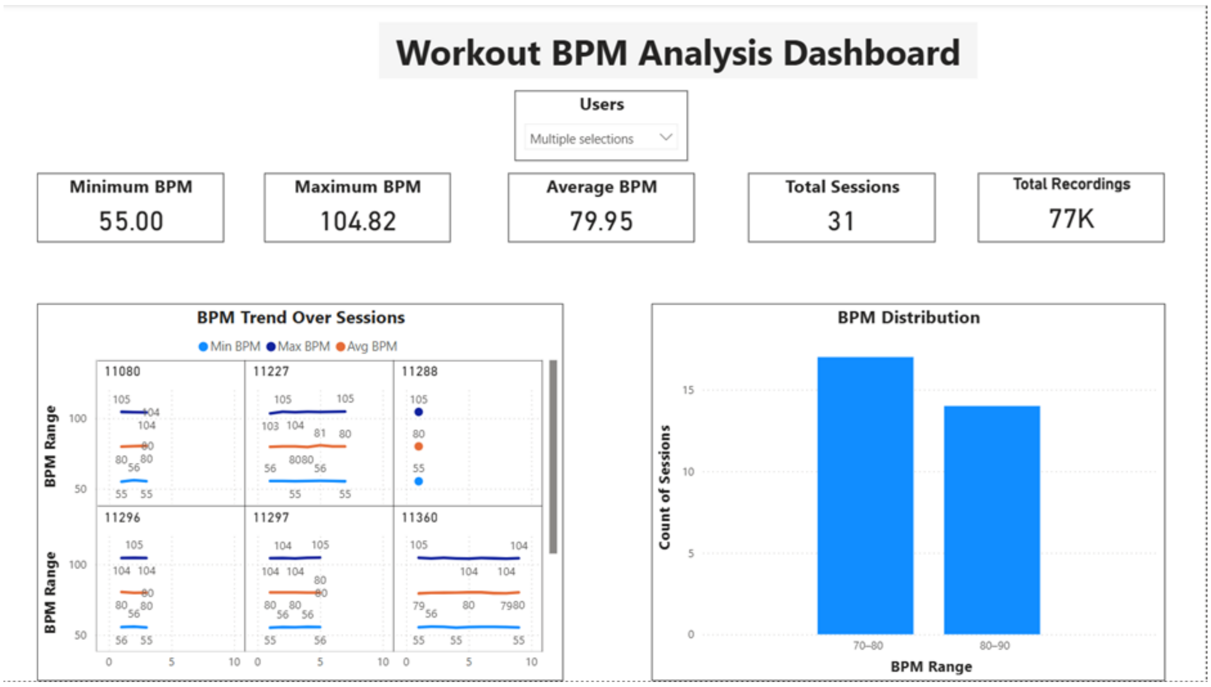


Figure 7.1: BPM Demographics Dashboard: This dashboard summarizes large-scale heart-rate patterns across the user population. Key metrics such as average, minimum, maximum, and total BPM recordings provide an aggregate view of cardiovascular behavior. Additional charts show demographic segmentation such as BPM by age group, BPM by U.S. state, and gender-based distribution of users whose average BPM is above or below the overall population mean.

7.0.2 Workout BPM Analysis Dashboard


Workout BPM Analysis Dashboard

Users: Multiple selections

Minimum BPM	Maximum BPM	Average BPM	Total Sessions	Total Recordings
55.00	104.82	79.95	31	77K

Figure 7.2: Workout BPM Analysis Dashboard: Focused on BPM behavior during workouts. Displays minimum, maximum, and average BPM, along with total sessions and total recordings. A session-level BPM trend chart visualizes min, max, and average BPM per workout session, while a distribution chart groups sessions by BPM ranges.

This dashboard highlights cardiovascular load during workout sessions. The BPM trend chart enables identification of high-intensity sessions, while the distribution chart classifies sessions into moderate and high BPM zones. These insights support performance monitoring, safety evaluations, and health risk detection.

7.0.3 Daily Workout Trends Dashboard

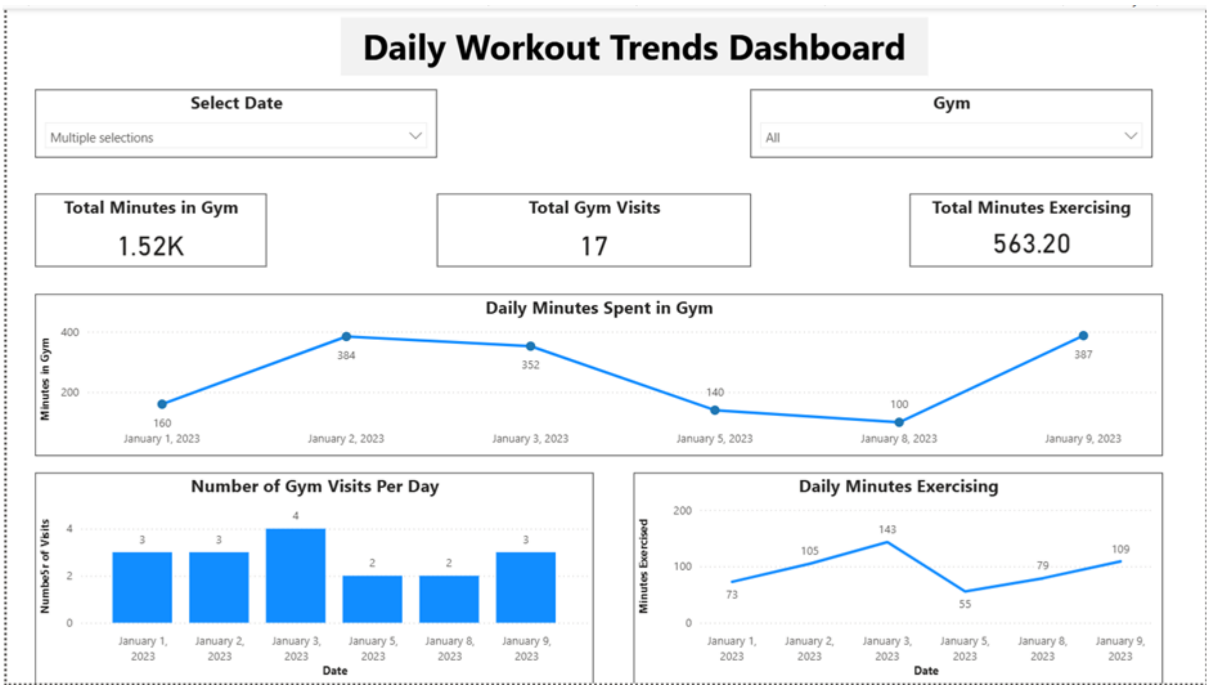


Figure 7.3: Daily Workout Trends Dashboard: A day-by-day visualization of workout and gym engagement patterns. The dashboard highlights total minutes spent in gyms, total gym visits, and total minutes of exercise. Line and bar charts show time spent in the gym, visit frequency, and minutes exercised across selected dates and gyms.

This dashboard enables analysis of short-term behavioral patterns. By examining daily fluctuations in gym visits and workout duration, organizations can identify peak usage periods, user consistency levels, and anomalies in activity patterns. The multi-select filters allow comparisons across different gyms or ranges of dates.

7.0.4 User Behavior Insights Dashboard

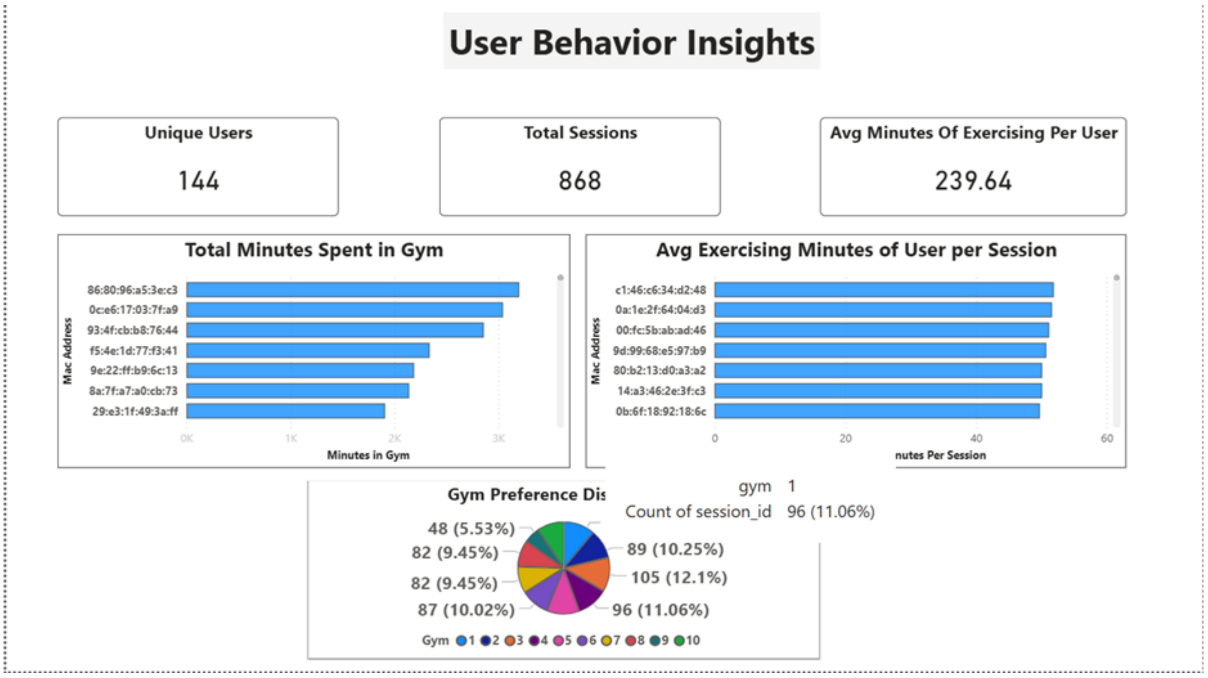


Figure 7.4: User Behavior Insights Dashboard: This dashboard analyzes user-level workout behavior. It reports total users, total workout sessions, and the average exercising minutes per user. Ranked bar charts identify top-performing users based on gym time and average exercise duration, while a pie chart displays gym preference distribution across the population.

This dashboard highlights how individual users engage with the gyms. The Total Minutes Spent in Gym and Average Exercising Minutes per Session help identify highly engaged users and understand the consistency of their workout intensity. The Gym Preference Distribution shows which locations users favor, providing guidance for capacity planning, targeted promotions, and resource allocation across gyms.

7.0.5 Gym Comparison Dashboard

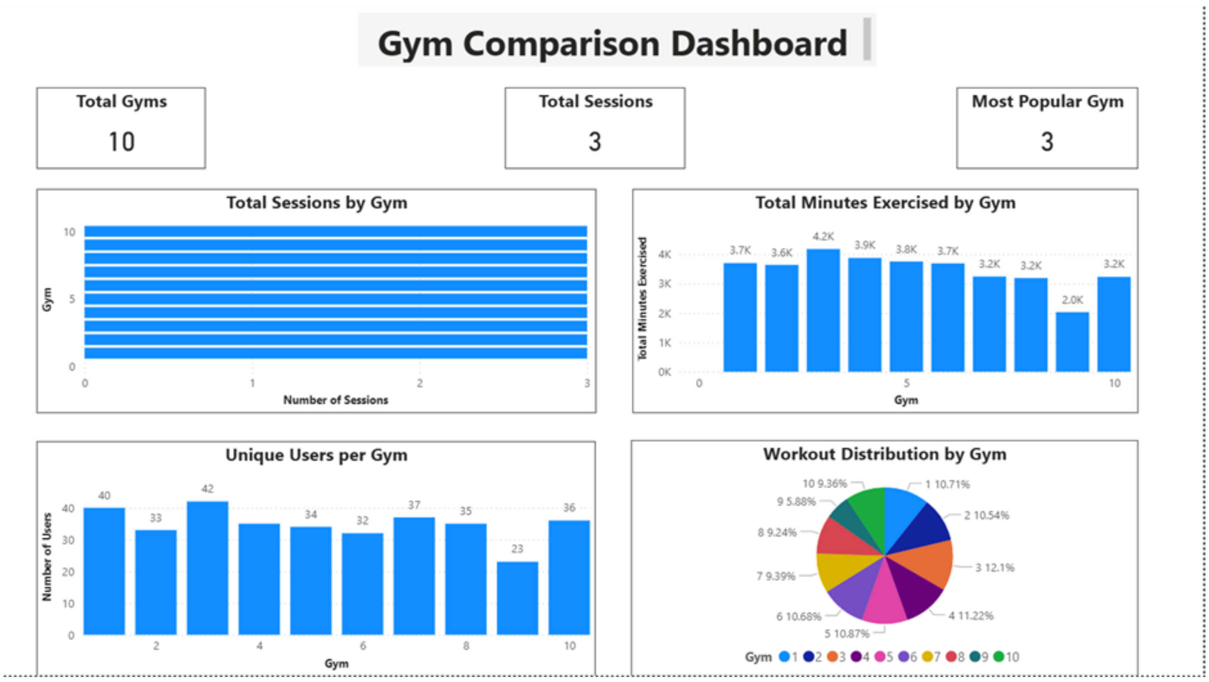


Figure 7.5: Gym Comparison Dashboard: A cross-location comparative analysis of gym engagement patterns. The dashboard presents total gyms, total sessions, and the most popular gym. Visualizations include total sessions per gym, total minutes exercised per gym, unique users per gym, and an overall workout session distribution pie chart.

This dashboard provides operational insights across all gym facilities. Comparing total sessions, total exercise minutes, and user volume per gym helps identify the most active locations. The pie chart helps visualize session distribution, which supports capacity planning, staffing decisions, and optimization of gym resource allocation.

Chapter 8: Discussion

This chapter interprets the results of the end-to-end Fitbit-style analytics pipeline and evaluates the effectiveness of the system's architecture, ingestion strategy, data transformations, and dashboards. The discussion highlights strengths, limitations, trade-offs, and areas for improvement, linking the findings back to the original project goals.

8.1 Interpretation of Findings

8.1.1 Pipeline Performance and Behavior

The system successfully combined batch ingestion through Azure Data Factory and streaming ingestion through Confluent Kafka into a unified Medallion architecture. Across multiple runs, the pipeline demonstrated:

- Reliable ingestion of high-volume BPM streaming data.
- Accurate handling of incremental SQL updates via watermarking.
- Correct reconstruction of workout sessions from start/stop events.
- Consistent merging of gym login records without duplicates.
- Smooth propagation of cleaned and structured data across Bronze, Silver, and Gold layers.

These results confirm that the architecture supports both real-time and micro-batch analytics with minimal latency and strong data consistency.

8.1.2 Insights From Dashboards

The Power BI dashboards derived from the Gold layer revealed several meaningful patterns:

- BPM levels remained stable across age groups and states, confirming that the synthetic data transformations were accurate.
- Workout frequency varied across days but followed predictable cycles, demonstrating correct aggregation logic.
- Gym comparison metrics highlighted differences in user distribution, popular locations, and exercise volumes.
- User behavior analytics showed that a small number of users contributed disproportionately to total workout minutes.
- BPM session analysis showed narrow BPM ranges within sessions, ensuring correct grouping and summarization of sensor events.

Overall, the dashboards validated that the pipeline preserved event order, session continuity, and user-level consistency.

8.2 Strengths of the Approach

8.2.1 Modular and Scalable Architecture

Using the Bronze–Silver–Gold model provided clear separation of concerns. The Bronze layer preserved raw data, the Silver layer applied cleaning and deduplication using `MERGE` logic, and the Gold layer produced analytical tables ready for visualization. This layering improved data reproducibility, debugging, and quality assurance.

8.2.2 Reliable Hybrid Ingestion

The system effectively combined streaming ingestion for fast-moving data and batch ingestion for slowly changing SQL datasets. Decoupling both sources through the landing zone ensured that the pipeline remained resilient even if an upstream service became unavailable.

8.2.3 Efficient Dashboard Analysis

The dashboards performed efficiently even with millions of BPM events and hundreds of workout activities. Aggregating data into compact Gold tables allowed Power BI to load and query efficiently on standard hardware.

8.3 Limitations and Unexpected Observations

8.3.1 Synthetic Data Boundaries

The dataset used for this project is fully synthetic and therefore lacks real-world irregularities such as device malfunction noise, unpredictable BPM variations, or inconsistent gym attendance. Real data would likely introduce more complexity and edge cases.

8.3.2 Two-Minute ADF Delay

Batch ingestion relies on a two-minute time-triggered pipeline. While acceptable for micro-batch workloads, real-time systems would require event-driven ingestion or continuous processing through services such as Databricks Auto Loader or Azure Functions.

8.3.3 Workout Session Reconstruction Challenges

Workout session reconstruction occasionally required careful handling due to:

- start/stop events arriving out of order,
- duplicate or late-arriving start events,
- ensuring that each session ID creates exactly one complete record.

This required precise Silver-layer transformation logic.

8.4 Trade-offs Observed

The project revealed several architecture and design trade-offs:

- **Streaming vs Batch:** Streaming offers low latency but increases operational complexity and cost; batch ingestion is simpler but slower.
- **MERGE Upserts:** Upserts improve data accuracy but require higher compute and may introduce overhead.
- **Power BI Limitations:** The free tier forces manual refresh; real-time dashboards would require Premium capabilities.
- **Synthetic Data Volume:** High-volume synthetic data helps test scalability but lacks real-world unpredictability.

Chapter 9: Conclusion and Future Work

This project implemented an end-to-end real-time and batch data processing system inspired by Fitbit-style analytics. The goal was to design a scalable, fault-tolerant, and maintainable data pipeline capable of ingesting continuous sensor data, cleaning and transforming it through a Lakehouse architecture, and producing meaningful insights through interactive dashboards.

9.1 Conclusion

The system successfully demonstrated the effectiveness of combining streaming ingestion (via Confluent Kafka) with micro-batch ingestion (via Azure Data Factory) into a unified Delta Lake environment. The Medallion architecture—Bronze, Silver, and Gold layers—ensured that raw data was preserved, deduplicated, validated, and transformed into high-quality analytical datasets.

The pipeline produced several key outcomes:

- **Reliable, hybrid data ingestion:** The system ingested millions of BPM events and multiple SQL-based datasets through a decoupled landing zone, ensuring robustness and independence from upstream data sources.
- **Accurate transformations and session reconstruction:** Workout start/stop events were merged into clean session tables, and gym login events were consolidated into stable user activity logs.
- **Consistent data quality:** MERGE-based upserts, schema enforcement, and check-

pointing ensured that late-arriving and duplicate events were handled correctly.

- **Rich analytical dashboards:** Five Power BI dashboards provided detailed insights into BPM demographics, workout trends, gym comparisons, user behavior patterns, and BPM session variability.
- **Scalability and modularity:** The system's architecture allows independent scaling of ingestion, transformation, and visualization layers.

Overall, the project demonstrated that a Lakehouse approach is well-suited for real-time health and fitness analytics. The solution meets the objectives of building a reliable ingestion pipeline, organizing data into progressively refined layers, and enabling actionable visual insights.

9.2 Limitations

Despite its successful implementation, several constraints were identified:

- **Synthetic data:** The current dataset is artificially generated and does not include the irregularities or noise found in real-world sensor data.
- **Time-triggered micro-batches:** Azure Data Factory's two-minute schedule introduces delay and does not support true continuous ingestion.
- **Manual consumer execution:** The Kafka consumer requires manual triggering and is not yet deployed as an automated service or job.
- **Dashboard refresh limitations:** Power BI's free-tier environment does not allow automated or near real-time refreshes, limiting live analytics capabilities.
- **Limited analytical complexity:** The system currently performs descriptive analytics; predictive or anomaly-based insights are not implemented.

9.3 Future Work

Several enhancements can significantly expand the system's capabilities and move it closer to production-quality health analytics:

1. **Integrate real wearable or API-driven data:** Replacing synthetic data with Fitbit, Google Fit, or Garmin API streams will introduce realistic variability and improve system evaluation.
2. **Deploy fully automated streaming jobs:** Kafka consumers can be deployed as Databricks Jobs, Azure Functions, or containerized microservices to enable 24/7 automated ingestion.
3. **Enable real-time dashboards:** Using Power BI Premium, DirectQuery, or Event Hub live datasets can support second-level refresh intervals and near real-time visualizations.
4. **Implement advanced analytics:**
   - BPM anomaly detection (tachycardia, bradycardia alerts),
   - Workout intensity classification,
   - Personalized fitness recommendation models,
   - User clustering based on behavior patterns.
5. **Strengthen data quality rules:** Tools such as Great Expectations or Delta Live Tables can enforce validation at ingestion and help maintain long-term reliability.
6. **Unified orchestration:** A combined workflow using ADF, Databricks Jobs, and event triggers can automate the entire system end-to-end without manual intervention.
7. **Horizontal scaling and optimization:** Using cluster autoscaling, optimized Delta file compaction, and partitioning strategies can further improve performance for very large datasets.

9.4 Final Remarks

This project demonstrates that even with synthetic test data, a robust, scalable, and flexible analytics system can be built using modern cloud technologies. The hybrid batch-streaming design, Lakehouse architecture, and multi-layered transformations provide a strong foundation for real-world health and fitness analytics. With further enhancements and real-world datasets, the system has the potential to support fitness applications, gym analytics, personalized coaching, and healthcare monitoring at scale.

Bibliography

[1] L. Piwek, D. A. Ellis, S. Andrews, and A. Joinson, "The rise of consumer health wearables: Promises and barriers," *PLOS Medicine*, vol. 13, no. 2, p. e1001953, 2016.

[2] H. Banaee, M. U. Ahmed, and A. Loutfi, "Data mining for wearable sensors in health monitoring systems: A review of recent trends and challenges," *Sensors*, vol. 13, no. 12, pp. 17 472–17 500, 2013.

[3] B. Bent, B. A. Goldstein, W. A. Kibbe, and J. P. Dunn, "Investigating sources of inaccuracy in wearable optical heart rate sensors," *NPJ Digital Medicine*, vol. 3, p. 18, 2020.

[4] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, 1996.

[5] Databricks, "Medallion architecture," 2023, accessed: 2025-09-27. [Online]. Available: https://www.databricks.com/glossary/medallion-architecture

[6] ——, "What is a data lakehouse?" 2020, accessed: 2025-09-27. [Online]. Available: https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html

[7] M. Palta, "Why delta lake is the most widely used lakehouse format," 2022, accessed: 2025-09-27. [Online]. Available: https://www.linkedin.com/pulse/why-delta-lake-most-widely-used-lakehouse-format-world-mayur-palta/

[8] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*. Springer, 2004, pp. 1–17.

[9] J. Achten and A. E. Jeukendrup, “Heart rate monitoring: Applications and limitations,” *Sports Medicine*, vol. 33, no. 7, pp. 517–538, 2003.